Business Process Security Specification
Automatic Extraction

David Alexandre Silva Marques

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor(s): Prof. Dr. Carlos Nuno da Cruz Ribeiro
Eng. Nelson Escravana

Examination Committee

Chairperson: Prof.ª Dra. Teresa Maria Sá Ferreira Vazão Vasques

Supervisor: Prof. Dr. Carlos Nuno da Cruz Ribeiro

Members of the Committee: Prof.ª Dra. Maria Dulce Pedroso Domingos

July 2019
Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.
Acknowledgments

First and foremost, I would like to thank my supervisor, Prof. Carlos Ribeiro, for giving me the opportunity to work on this project, for the professional teachings provided and all the support throughout my Master Thesis. Also, would like to thank my co-supervisor, Eng. Nelson Escravana and Eng. Filipe Apolinário, from INOV, for all the patience, availability, motivation by pushing myself to perform the best work I could and for their most useful insights. Without their contribution, this work would not be possible. Additionally, a special mention to all the wonderful people at INOV who welcomed me with open arms and contributed to this success.

I am grateful to all my colleagues and friends, which were my second family, providing me with the motivation and strength, but also for sticking with me through all these years. Their ongoing support and friendship were crucial to reach another milestone in my life.

Last but certainly not least, I would like to thank my family, in particular my parents, who forged the person I am today, for the unconditional support and for providing me the best education possible allowing me to pursue my professional goals. I also would like to thank my girlfriend for all the support, sacrificed hours, for all the caring and sharing and for making me see the brightest side of life, every time.
Abstract

Traditional intrusion detection systems that have been used suffer from effectiveness problems, either because their ineffectiveness on detecting new unidentified threats or because they produce a high level of false positives and negatives alarms. These systems had limited efficiency due to being data-oriented and for that reason there is a need to change this approach. Process-Aware Information Systems (PAIS), in contrast with the traditional systems, are process-oriented and these processes can involve applications, people and/or information sources on the basis of process models. Business Process Intrusion Detection System (BP-IDS) is a specification-based IDS that has proven to be successful in reducing the false positive and false negative errors. However, in order for BP-IDS to monitor the network in search for anomalies, a specification model of the business processes needs to be produced a priori. This specification model is constructed manually by the system administrator and, consequently, labor-intensive since the administrator needs to manually extract activities based on the data, i.e. find patterns within the log and specify which activities belong to each process. This dissertation is focused on providing system administrators with a module that is easy to use, versatile and produce good results. Throughout this document different approaches will be explained and compared in order to see which better fits the overall goal for the module. It was concluded that the module built throughout this dissertation, is able to, after an input of log files describing business processes, produce Business Process Model and Notation (BPMN) that are identical to specification models, produced by hand after analyzing the same log files used as input. Additionally, this work concluded that, among the different process mining algorithms considered, the Inductive Miner provided the best results, outputting BPMN models that are identical to a specification. This led to the conclusion that the Inductive Miner performed better than the other algorithms, on providing better models, and responded better to noise in the log. This work achieved its goal of providing system administrators with a module that can ease the work of producing specification by hand.

Keywords

Resumo

Os sistemas tradicionais de deteção de intrusos que são usados sofrem de problemas de eficácia, tanto devido à sua ineficácia em detetar novas ameaças não identificadas, como por produzirem um nível elevado de alarmes falsos, sejam falsos positivos ou falsos negativos. Estes sistemas eram possuíam eficiência limitada dado que eram orientados a dados e por essa razão existe uma necessidade de mudar de abordagem. Sistemas de Informação Conscientes de Processos (PAIS), em contraste com os sistemas tradicionais, são orientados aos processos e estes processos podem envolver aplicações, pessoas e/ou fontes de informação com base em processos de negócio. Business Process Intrusion Detection System (BP-IDS) é um sistema de deteção de intrusos com base numa especificação e que já provou que ser bem-sucedido em reduzir os erros de falsos positivos e falsos negativos. Contudo, para que o BP-IDS possa monitorizar a rede em busca de anomalias, um modelo de especificação dos processos de negócio tem de ser produzido a priori. Este modelo de especificação é construído manualmente pelo administrador de sistemas e, consequentemente, torna-se um trabalho intenso dado que o administrador precisa de manualmente extrair atividades com base nos dados, i.e., encontrar padrões dentro do log e especificar que atividades pertencem a cada processo. Esta dissertação está focada em providenciar ao administrador de sistemas um módulo fácil de utilizar, versátil e produz bons resultados. No decorrer deste documento, diferentes abordagens são explicadas e comparadas para que seja possível determinar qual será a mais adequada para cumprir com os objetivos do módulo. Foi concluído que o módulo produzido durante a dissertação tem a possibilidade de, depois de ser providenciado com ficheiro de log que descreve os processos de negócio, produz um modelo BPMN que é idêntico ao modelo de especificação (modelo do processo de negócio de uma companhia feito à mão), após análise ao ficheiro de log providenciado. Adicionalmente, este trabalho permitiu concluir que, entre os diferentes algoritmos de process mining considerados, o Inductive Miner providenciou os melhores resultados fabricando modelos que são idênticos a uma especificação. Isto levou à conclusão que o Inductive Miner teve uma melhor performance que os outros algoritmos, providenciando melhores modelos, e respondeu melhor ao ruído nos ficheiros de log. Este trabalho atingiu os seus objetivos providenciando um módulo aos administradores de sistemas que lhes alivia o trabalho de produzir uma especificação à mão.

Palavras Chave

Mineração de Processos, Sistemas de Deteção de Intrusos, Sistemas de Informação Conscientes de Processos, Descobrimento de Processos, Clustering, BP-IDS.
4.3.1 Combinatorial and Redundancy Checker Algorithms .................................................. 27
4.4 MINING ALGORITHM ........................................................................................................ 27
4.5 SUMMARY .......................................................................................................................... 28

5 EVALUATION ......................................................................................................................... 29

5.1 LIMITATIONS ....................................................................................................................... 29
  5.1.1 Input Handler – How much log quality influence the end result? .................................. 29
  5.1.2 Mining Stage ..................................................................................................................... 31
5.2 INDUCED MINER VS HEURISTICS MINER ........................................................................ 31
  5.2.1 Noise Handling between algorithms .............................................................................. 31
  5.2.2 Experimental Work ......................................................................................................... 32
    5.2.2.1 1st Experiment ......................................................................................................... 34
    5.2.2.1.1 Setup ................................................................................................................... 34
    5.2.2.1.2 Comparison Results ............................................................................................ 35
    5.2.2.2 2nd Experiment ....................................................................................................... 37
    5.2.2.2.1 Setup ................................................................................................................... 37
    5.2.2.2.2 Comparison Results ............................................................................................ 38
    5.2.2.3 3rd Experiment ......................................................................................................... 39
    5.2.2.3.1 Setup ................................................................................................................... 41
    5.2.2.3.2 Comparison Results ............................................................................................ 41
    5.2.2.4 4th Experiment ......................................................................................................... 43
    5.2.2.4.1 Setup ................................................................................................................... 43
    5.2.2.4.2 Comparison Results ............................................................................................ 43
5.3 HOW DOES THIS MODULE HELP SYSTEM ADMINISTRATORS? .................................... 45
5.4 SUMMARY .......................................................................................................................... 46

6 CONCLUSION .......................................................................................................................... 48
  6.1 ACHIEVEMENTS ................................................................................................................. 49
  6.2 FUTURE WORK .................................................................................................................... 49

7 REFERENCES .......................................................................................................................... 50

APPENDIX A – SELECTION STAGE ......................................................................................... 53

APPENDIX B – PSEUDO-CODE ............................................................................................... 54
List of Figures

FIGURE 1 - EXAMPLE OF AN INFRASTRUCTURE MONITORED BY BP-IDS. ................................................................. 2
FIGURE 2 - EXAMPLE OF A BP-IDS DEPLOYMENT IN A WATER TEMPERATURE MONITORING CONTROL SYSTEM. ................................................................. 3
FIGURE 3 - TEMPORAL MESSAGE EXCHANGE. ........................................................................................................ 4
FIGURE 4 - EXAMPLE OF THE BPMN MODEL FOR THE WATER TEMPERATURE MONITORING SYSTEM. ................................................................. 5
FIGURE 5 - PROCESS MODEL DISCOVERY. ............................................................................................................. 7
FIGURE 6 - MINING RESULTS FROM THE PROCESS PERSPECTIVE (A) AND ORGANIZATIONAL PERSPECTIVES (B AND C)[16]. ........................................ 9
FIGURE 7 - MODELS THAT CANNOT BE DISCOVERED WITH THE ALPHA ALGORITHM. ........................................... 10
FIGURE 8 - D/F GRAPH RESULTING FROM TABLE 2. ............................................................................................. 11
FIGURE 9 - COMMON STRUCTURE OF A MULTI-LAYER PERCEPTRON...................................................................... 14
FIGURE 10 - STAGES THAT COMPRIZE THE DISCOVERY MODULE ARCHITECTURE. .................................................. 20
FIGURE 11 - EXAMPLE OF THE INTENDED FUNCTIONING OF THE MODULE. ...................................................... 21
FIGURE 12 - HOME PAGE OF THE PROCESS DISCOVERY MODULE ......................................................................... 26
FIGURE 13 - MODULE’S COMPOSITION............................................................................................................... 28
FIGURE 14 - BPMN OBTAINED WITHOUT NOISE................................................................................................. 30
FIGURE 15 - BPMN OBTAINED WITH THE ADDING OF NOISE. .............................................................................. 30
FIGURE 16 - BPMN OBTAINED WITHOUT NOISE USING THE INDUCTIVE MINER. ..................................................... 32
FIGURE 17 - BPMN OBTAINED WITH THE ADDING OF NOISE USING THE INDUCTIVE MINER. .................................... 32
FIGURE 18 - SPECIFICATION FOR THE LOG FILE OF THE 1ST AND 2ND EXPERIMENT .............................................. 33
FIGURE 19 - INDIVIDUAL COMPARISON RESULTS – 1ST EXPERIMENT ................................................................. 36
FIGURE 20 - INDIVIDUAL COMPARISON RESULTS – 2ND EXPERIMENT ............................................................... 38
FIGURE 21 - SPECIFICATION USED IN THE 3RD AND 4TH EXPERIMENT ............................................................. 40
FIGURE 22 - INDIVIDUAL COMPARISON RESULTS – 3RD EXPERIMENT ............................................................... 42
FIGURE 23 - INDIVIDUAL COMPARISON RESULTS – 4TH EXPERIMENT ............................................................... 44
List of Tables

TABLE 1 - EXAMPLE OF EVENT LOG [16]. ........................................................................................................ 8
TABLE 2 - EXAMPLE OF A D/F-TABLE.......................................................................................................... 11
TABLE 3 - COMPARISON BETWEEN CLUSTERING ALGORITHMS. ......................................................... 15
TABLE 4 - REDUNDANCY EXAMPLE ............................................................................................................... 23
TABLE 5 - PREVIEW OF THE LOG FILE FOR THE 1ST EXPERIMENT. ................................................... 34
TABLE 6 - OPTIONS CHOSEN FOR EACH COLUMN FOR THE 1ST EXPERIMENT. ........................................ 35
TABLE 7 - COMPARISON STATISTICS – 1ST EXPERIMENT ........................................................................... 36
TABLE 8 - OPTIONS CHOSEN FOR EACH COLUMN FOR THE 2ND EXPERIMENT. ...................................... 37
TABLE 9 - COMPARISON STATISTICS – 2ND EXPERIMENT ........................................................................ 38
TABLE 10 - PREVIEW OF THE LOG FILE USED IN THE 3RD AND 4TH EXPERIMENT. ............................... 40
TABLE 11 - OPTIONS CHOSEN FOR EACH COLUMN FOR THE 3RD EXPERIMENT. ................................. 41
TABLE 12 - COMPARISON STATISTICS – 3RD EXPERIMENT. ...................................................................... 41
TABLE 13 - OPTIONS CHOSEN FOR EACH COLUMN FOR THE 4TH EXPERIMENT. ................................. 43
TABLE 14 - COMPARISON STATISTICS – 4TH EXPERIMENT. ...................................................................... 43
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACK</td>
<td>Acknowledge</td>
</tr>
<tr>
<td>AHC</td>
<td>Agglomerative Hierarchical Clustering</td>
</tr>
<tr>
<td>BP-IDS</td>
<td>Business Process Intrusion Detection System</td>
</tr>
<tr>
<td>BPMN</td>
<td>Business Process Model and Notation</td>
</tr>
<tr>
<td>CLI</td>
<td>Command Line Interface</td>
</tr>
<tr>
<td>EPC</td>
<td>Event-driven Process Chain</td>
</tr>
<tr>
<td>EU</td>
<td>European Union</td>
</tr>
<tr>
<td>GDPR</td>
<td>General Data Protection Regulation</td>
</tr>
<tr>
<td>HIPAA</td>
<td>Health Insurance Portability and Accountability Act</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>IDS</td>
<td>Intrusion Detection System</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>PAIS</td>
<td>Process-Aware Information System</td>
</tr>
<tr>
<td>PDV</td>
<td>Packet Delay Variation</td>
</tr>
<tr>
<td>PLC</td>
<td>Programmable Logic Controller</td>
</tr>
<tr>
<td>QTC</td>
<td>Quality Threshold Clustering</td>
</tr>
<tr>
<td>SOM</td>
<td>Self-Organizing Map</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>WMS</td>
<td>Workflow Management System</td>
</tr>
</tbody>
</table>
1 Introduction

In the early 1990’s organizations started to adopt software systems that manage and execute operational processes [1]. These systems are named Process-Aware Information Systems (PAIS), which are software systems that manage and execute operational processes involving people, applications, and/or information sources based on a process model [2]. The adoption of these systems illustrates a shift from data-oriented systems to process-oriented systems. This management trend shift clearly separates business process logic from application programs, which facilitates the redesign and extension of process models. The critical aspect of these systems is that the people who execute the activities have full control on deciding how actions unfold in order to achieve the desired outcome for the case [2]. As a consequence of having full access in the unfolding of actions, these systems allow for more control by whoever is executing the processes and so, administrators now have the ability to evaluate and provide the proper outcome for each action and how each action is connected to each other. This provides a better flexibility of workflow systems [3]. One of this kind of systems is the Workflow Management System (WMS) which typically store the start and completion of activities [4] and offer generic modeling and enactment capabilities for structured business processes [5].

Although this flexibility and control comes with advantages, it may also hinder detecting frauds and errors, due to the fact that these systems tend to be monitoring diverse infrastructures and can enable critical business operations [6] such as, for example, monitoring a reactor’s temperature. This means that processes are not only beneficial to organizations, but also to attackers to exploit these processes.

As previously stated, these process-oriented systems not only have great benefits for human-driven scenarios, but also for automatically driven scenarios. Automatic scenarios are very prone to attacks because the automation makes users more oblivious to how the system operates, since there is no interaction with them. For that reason, these systems are less strictly monitored and controlled (by users), making them more security vulnerable [6], given that attackers know that these automated systems have less monitoring. Therefore, there is a need to have systems that can detect errors, whether they are intentional or not, acts of fraud or other security incidents, which will be named anomalies throughout this dissertation. A good example of an intentional anomaly is the nuclear programs high-speed centrifuges spinning out of control due to malware being injected into the machine that controls these centrifuges. This was what a worm named Stuxnet did when it infected Iran’s nuclear program [7], so it goes to show that these cases are not just hypothetical. An unintentional anomaly is, for example, the earthquake and tsunami combination that affected several Japanese nuclear power plants [8].

Overall, the common property of the above scenarios is the lack of monitoring. One solution to solve this gap is the use of Intrusion Detection Systems (IDS) to detect anomalies. As these events are detected, the IDS dispatch an alert to the system administrator of the occurrence, so the administrator can fix the issue before any major problem arises. Historically, the effectiveness of IDSs has been put into question, due to the fact that these systems generate a huge number of false positive and false negative alerts, making them unreliable and
untrustworthy [9] since they adopt the “one size fits all” strategy, making these systems too generic. Modern approaches try to overcome those issues by abandoning the previously mentioned approach employed by many systems and proposing detection models that are more adapted/adaptable to the target environments.

One of these approaches is named BP-IDS [10]. BP-IDS is a specification-based IDS that was designed to successfully reduce the false positive and false negative alerts. BP-IDS monitors the business process executions and identifies non-compliance (anomalies) based on a specification, which is a model describing the exact business processes of an organization. This specification is determined before BP-IDS deployment. However, the cost of creating, adapting and validating the model specification is still quite high, not only in BP-IDS but in all PAIS. This is due to the fact that pointing out which activities belong in a process is manual, and consequently labor-intensive, since the administrator needs to manually insert the processes and specify which activities belong to each process. Whenever there is a trace of a process violating the pre-determined specifications, BP-IDS will notify the organization of this anomaly, so it can be further investigated. As illustrated in Figure 1, which is an example of BP-IDS monitoring infrastructure, the solution is implemented to analyze and identify non-compliant activities to the pre-specified processes. Network traffic which ingresses the switch is captured by the sensor and sent to the monitoring core. Although Figure 1 only depicts one sensor, this monitoring core can have multiple sensors connected to it.

If an anomaly is detected, a notification will be issued to this application describing the anomaly. In other words, BP-IDS will examine if said activities are in compliance with the business process specification.

![Figure 1 - Example of an infrastructure monitored by BP-IDS.](image)
1.1 Problem Description

Although BP-IDS is a modern approach of the “classical” IDS, there is an inherent problem that needs to be solved and which will be the foundation for this thesis. Before BP-IDS can monitor the organization’s network, a specification for the company’s business processes must be specified and then, and only then, can BP-IDS start to monitor the network. This task is performed manually by the system administrator and, for large networks, is labor-intensive. Therefore, there is a need to automatize as much as possible the business process discovery task. To tackle the issue of this task being performed manually is where the contribution of this work will help BP-IDS, and system administrators, by implementing a module that does automatic process discovery.

But even with this module and after all the traffic being analyzed, BP-IDS will produce a possible specification. Then it is the administrator who has the final word on, after examining the possible specification provided by BP-IDS, inspect if the specification is correct or not. If it is, then all the traffic is going to be compared to this specification to detect anomalies. If not, the administrator has the ability to properly correct BP-IDS’s specification and then create the specification.

To better describe the problem, let’s suppose there is a need for a certain organization to control water temperature at 50°C. A thermometer that is in contact with the water is then connected to a sensor that reads the temperature of the water and is connected to a Programmable Logic Controller (PLC) using Modbus TCP protocol, which is a serial communication protocol (over TCP) used to connect industrial devices. Also connected to the PLC is an actuator that is connected to a resistor that, when the water’s temperature falls below 50°C will, due to Joule’s first law, heat the water. Figure 2 presents this scenario and shows where BP-IDS would be monitoring the network.

![Figure 2 - Example of a BP-IDS deployment in a water temperature monitoring control system.](image)

As can be seen in Figure 3, the message exchange is as follows: (1) As the temperature is being read, the sensor sends messages to the PLC to inform if the temperature complies with the specification. (2) Then, the PLC acknowledges the communication and, if the temperature is below 50°C Celsius, (3) it sends a command to the
actuator to heat the water. (4) The sensor keeps sending the temperature of the water to the PLC ((5) with all the acknowledgments being sent by the PLC) and when it complies with the specification, (6) the PLC commands the actuator to stop heating the water.

The discovery module will receive an event log containing the behavior described above and will map the different activities (messages) into business processes automatically, which means perform business process discovery. Then, it will export a Business Process Model and Notation (BPMN) model, similar to the one shown in Figure 4, to be used by BP-IDS.
After the model is exported, BP-IDS will start to monitor the network in search for anomalies. It will compare the activities it is capturing with the BPMN model generated by the discovery module and, if there is an anomaly, BP-IDS will issue an alert. An example of an anomaly in this scenario, would be the actuator continuing to heat the water when the water temperature is above 50° Celsius.

1.2 Motivation

After understanding the problem, it is evident that the construction of the specification is labor-intensive and on one hand this task constrains the system administrator time by not allowing him to focus on other critical tasks. On the other hand, systems like BP-IDS cannot monitor the network and do compliance checks without a specification model of the business processes. In this sense, a semi-automatic module was designed to assist system administrators by reducing the time spent on producing a specification model. This module takes a log file or network capture as input and is able to produce various models that the system administrator can choose which is more accurate and make necessary changes if needed to make the model as accurate as possible.

To evaluate this module, some tests were performed in order to check the accuracy of the outputs and the disparity between the various models. Additionally, experiments were conducted to check which algorithms were better suited to be integrated on the module.

1.3 Contributions

The main contributions of this thesis are: the design and implementation of a semi-automatic module that will assist system administrators on one of the most time-consuming tasks, which is inferring business activities and processes from log files. The module produces a BPMN output (or multiple BPMNs, depending on the knowledge
that the system administrator has on the log) based on the log file to be used by, in this case BP-IDS, as specification. This module is a standalone module and can be used by any PAIS. It is designed to be as versatile as possible, not only on the execution of the module itself but also on the outputs that it produces as those outputs can be changed by the administrator if they do not depict with total accuracy the business processes of the company. This thesis will also contribute in identifying the main struggles for obtaining models from logs using process mining and process instance discover.

1.4 Structure of the document

This chapter served as an initial exposure to the problem and the motivation that led to the theme of this thesis, as well as providing an overview of the contents presented in the dissertation. The main motivation for this work is to provide system administrators with a module that is versatile and capable of producing accurate modules and save them time to focus on more critical tasks. Chapter 2 covers the analysis and comparison of different process discovery techniques in order to see which one will be better suited to be implemented on any system that requires a process discovery module. Chapter 2 will also cover the different approaches suitable to automatize the process instance discovery since log mining algorithms cannot operate on logs without every activity being categorized as well as every process instance. Chapter 3 and 4 will present the design and architecture of the designed module as well as the in-depth details of its implementation. Finally, chapter 5 and 6 will present the evaluation of the performed work and the conclusion of the dissertation, respectively.
2 Related Work

As stated before, the inherent problem with BP-IDS is that it needs a specification model of the business processes a priori to do compliance checks. To produce such specification model is a time-consuming task and, therefore, there is a need to automatize this task. This chapter will be focused on different process discovery techniques in order to see which will be better suited to solve the specification generation problem.

2.1 Process Model Discovery

Process Mining techniques are used to generate a specification model, or for other words a reference process model, based on the behavior of process execution. This specification is then used as a label to what normal behavior should be [6]. This means that these techniques can be used in situations that a need for a specification is presented, to be served as training set or as ground-truth, as previously stated. To generate a specification an algorithm that can analyze the different activities and discover which are related to other activities must be used. The usage of these techniques is named process model discovery. Figure 5 provides an overview of what process model discovery is in a simplified way.

![Process Model Discovery Diagram](image)

This dissertation will focus on several approaches for process discovery in process mining techniques. Each approach has its own algorithms to perform process discovery.

In process mining it is often distinguished three different perspectives [11]: (1) the process perspective ("How?") , (2) the organizational perspective ("Who?") , and (3) the case perspective ("What?"). The process perspective is based on control-flow, i.e., ordering tasks and the objective is to discover a good characterization of all paths, for example, extracted in terms of a Petri net [12] (also known as place/transition net). It is a net that offers a graphical notation for stepwise processes that includes choice, iteration and concurrent execution, or an Event-driven Process Chain (EPC [13]), which is a diagram to lay out business process workflows. The organizational perspective is centered on the originator and how they relate to each other. In table 1, the originator column details the person who performed the activity, but it does not necessarily need to be a person performing an activity, for example, it can be a light switch or a gas valve turning on and off. With all the relations determined, a social network [18,19] which means a structural view of the organization is built. The case
perspective is centered on the properties of cases. These cases can be described by their path in the process or by the person working on the case or even by the values of corresponding data elements.

All of these perspectives have their own discovery approaches, which means to do process discovery without \textit{a priori} knowledge, and conformance checking approaches, but this thesis is focused on the first. For a better understanding of these concepts, Table 1 shows an example of an event log that will be mined (in this case the technique is process mining in logs) with the first two perspectives referenced.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
Case ID & Activity ID & Originator & Timestamp  \\
\hline
Case 1 & Activity A & John & 9-3-2004:15.01  \\
Case 2 & Activity A & John & 9-3-2004:15.12  \\
Case 3 & Activity A & Sue & 9-3-2004:16.03  \\
Case 3 & Activity B & Carol & 9-3-2004:16.07  \\
Case 1 & Activity B & Mike & 9-3-2004:18.25  \\
Case 1 & Activity C & John & 10-3-2004:9.23  \\
Case 2 & Activity C & Mike & 10-3-2004:10.34  \\
Case 4 & Activity A & Sue & 10-3-2004:10.35  \\
Case 2 & Activity B & John & 10-3-2004:12.34  \\
Case 2 & Activity D & Pete & 10-3-2004:12.50  \\
Case 5 & Activity A & Sue & 10-3-2004:13.05  \\
Case 4 & Activity C & Carol & 11-3-2004:10.12  \\
Case 1 & Activity D & Pete & 11-3-2004:10.14  \\
Case 3 & Activity C & Sue & 11-3-2004:10.44  \\
Case 3 & Activity D & Pete & 11-3-2004:11.03  \\
Case 4 & Activity B & Sue & 14-3-2004:11.18  \\
Case 5 & Activity E & Clare & 17-3-2004:12.22  \\
Case 5 & Activity D & Clare & 18-3-2004:14.34  \\
Case 4 & Activity D & Pete & 19-3-2004:15.56  \\
\hline
\end{tabular}
\caption{Example of Event Log [16].}
\end{table}

In Figure 6 is shown the result of the mining, clearly illustrating the difference in results from the two approaches. (a) shows the control-flow demonstrated in terms of a Petri net. (b) shows the organizational structure in terms of an activity-role performer diagram and (c) shows a sociogram connecting the transfer of work.
2.1.1 Process Mining Techniques - Using Logs

Process mining using logs, in business process, has a clear goal of extracting information about processes from logged events [17]. As referred, this technique can generate a specification to be used as a training set of what normal behavior is.

2.1.1.1 Alpha / Alpha ++ Miner

One of the algorithms available to do process discovery in the process perspective is the alpha miner algorithm [18]. This algorithm has a clear goal of constructing a particular type of Petri net called workflow net from event logs and it was one of the first algorithms to correctly handle concurrency, or in other words, explicit casual dependencies and parallel tasks [19]. The algorithm starts by analyzing event logs and, based on that, build various dependency relations. Based on these relations, the α algorithm constructs the correspondent Petri net. This algorithm, although it can deal with multiple forms of concurrency, it has problems in correctly discovering implicit dependencies and this is due to a particular use of non-free choice construct in Petri nets. This is a crucial fault, due to the fact that these behaviors occur in multiple real-life processes. Figure 7 shows the incorrect models that are generated by this algorithm (on the right) from the process models (on the left) that contain non-free-choice constructs and thus, cannot be discovered by this algorithm.
For this reason Lijie Wen et al. [19] propose a newer algorithm, the α++ which, when applying two rules to reduce the redundant implicit dependencies, it can produce process models that possess non-free-choice constructs. However, the models produced with this algorithm are difficult to visualize and interpret by analysts. Also, the algorithm has problems with noise on the logs, which mean exceptions of incorrectly logged events or other non-recurrent events and non-complete logs which, this one particularly, the algorithm heavily relies on.

2.1.1.2 Dependency graph construction

Weijters et al. [5] have another type of approach for process mining using logs. This approach tackles flexibility problems, noise problems and “also be used to validate workflow processes by uncovering and measuring the discrepancies between prescriptive models and actual process executions.”. This approach is based on three assumptions regarding the recording of events: (i) each event refers to a task, which means a well-defined step in the workflow), (ii) each event refers to a case (i.e., a workflow instance) and (iii) events are totally ordered. In this approach, three mining steps can be distinguished: The first step is the construction of a dependency/frequency table (D/F-table presented in table 2), which is a table that contains:

- The overall frequency of a task (for example task A) noted as #A;
- The frequency of a task (A) preceded by another task (for example task B) noted as B<A;
- The frequency of A directly followed by another task B noted as A>B;
- The frequency of A directly or indirectly preceded by another task B but before the previous appearance of A noted as B<<<A
• The frequency of A directly or indirectly followed by another task B but before the next appearance of A noted as $A \gg \gg B$

• Finally, it contains a metric that indicates the strength of the causal relation between task A and another task B noted as $A \rightarrow B$.

Table 2 - Example of a D/F-table.

<table>
<thead>
<tr>
<th>B</th>
<th>#B</th>
<th>B&lt;A</th>
<th>A&gt;B</th>
<th>B&lt;&lt;&lt;A</th>
<th>A&gt;&gt;&gt;B</th>
<th>A-&gt;B</th>
</tr>
</thead>
<tbody>
<tr>
<td>T10</td>
<td>1035</td>
<td>0</td>
<td>581</td>
<td>348</td>
<td>1035</td>
<td>0.803</td>
</tr>
<tr>
<td>T5</td>
<td>3949</td>
<td>80</td>
<td>168</td>
<td>677</td>
<td>897</td>
<td>0.267</td>
</tr>
<tr>
<td>T11</td>
<td>1994</td>
<td>0</td>
<td>0</td>
<td>528</td>
<td>1035</td>
<td>0.193</td>
</tr>
<tr>
<td>T13</td>
<td>1000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>687</td>
<td>0.162</td>
</tr>
<tr>
<td>T9</td>
<td>1955</td>
<td>30</td>
<td>46</td>
<td>366</td>
<td>338</td>
<td>0.161</td>
</tr>
<tr>
<td>T8</td>
<td>1994</td>
<td>68</td>
<td>31</td>
<td>560</td>
<td>925</td>
<td>0.119</td>
</tr>
<tr>
<td>T3</td>
<td>3849</td>
<td>146</td>
<td>206</td>
<td>831</td>
<td>808</td>
<td>0.019</td>
</tr>
<tr>
<td>T6</td>
<td>1035</td>
<td>0</td>
<td>0</td>
<td>348</td>
<td>248</td>
<td>0.000</td>
</tr>
<tr>
<td>T7</td>
<td>959</td>
<td>0</td>
<td>0</td>
<td>264</td>
<td>241</td>
<td>-0.011</td>
</tr>
<tr>
<td>T12</td>
<td>994</td>
<td>0</td>
<td>0</td>
<td>528</td>
<td>505</td>
<td>-0.093</td>
</tr>
<tr>
<td>T1</td>
<td>1000</td>
<td>0</td>
<td>0</td>
<td>687</td>
<td>0</td>
<td>-0.246</td>
</tr>
<tr>
<td>T2</td>
<td>1994</td>
<td>0</td>
<td>0</td>
<td>1035</td>
<td>505</td>
<td>-0.487</td>
</tr>
<tr>
<td>T4</td>
<td>1994</td>
<td>691</td>
<td>0</td>
<td>1035</td>
<td>505</td>
<td>-0.823</td>
</tr>
</tbody>
</table>

The second step is the induction of a D/F-graph (presented in Figure 8) out of a D/F-table which consists of applying heuristic rules in order to filter noise, using a noise factor that can be changed with the noise level that a log may present and to prevent the non-recognition of recursion.

Figure 8 - D/F graph resulting from Table 2.

And lastly the third step is to reconstruct the process model from the D/F-table and D/F-graph by applying algorithms. This approach, although it can tackle the problems mentioned above, has some limitations. The report was based on a limited number of experiments and it is also based on assumptions, presented above, that may not be applicable in real-world contexts. Also, the heuristic rules were not proven to work in every scenario and not every process model can be mined [5].
2.1.1.3 Heuristics Miner

Another algorithm that can be used to mine logs is the Heuristics Miner [16]. It tackles the issue of noise on logs and can express the main behavior, which means it discards some details and exceptions that rarely occur, registered in event logs [16]. This algorithm mines the control-flow perspective of a process model and to do that it only considers the order of the events in a case. The Heuristics Miner algorithm can be defined in three steps of execution:

- **Mining of the dependency graph** – This concept, introduced previously, has the goal to indicate the degree of certainty of a dependency relation between events.

- **Construction of the input-output expression for each activity** – Activities are represented by transitions, which means that after executing the first activity, there is another one to follow or there is a choice between different activities, based on the output of the first activity. To obtain the expressions for these transitions is the goal of this step.

- **Mining long-distance dependency relations** – There are process models that have choices that do not depend on a particular activity, but instead they depend on other choices made previously in other parts of the process model. This step has a clear goal of mining these long-distance dependencies.

The time complexity for this algorithm is, for the first step, linear to the number of traces and \(k^2\) to the number of different activities. Then, for the construction of the process model, the time complexity is again \(k^2\). Weijters et al. [16] concluded, based on the experiments of their work, that the most significant result was the robustness of the algorithm for noise, which is relevant in real-world scenarios.

2.1.1.4 Inductive Miner

The final mining algorithm explored in this dissertation is the \(B'\) algorithm [20] or as known in the ProM framework [21], the Inductive Miner [22]. This algorithm aims to construct a Process Tree for a given log and it works recursively using the **divide and conquer** strategy. It starts by dividing the activities in the log into sets, constructs part of the process tree and then proceed to splitting the log over those sets and handling them separately to construct the rest of the process tree. When the process tree is fully determined, it can be converted directly into a Petri net [12]. Leemans et. al [20] have proven that that if the model underlying the log is representable as a process tree that has no duplicate activities, contains no silent activities and does not contain too-short loops, then the Inductive Miner rediscovers this model. The main advantages of this algorithm are:

- **Guarantees soundness** – which means that the model is free of deadlocks and other anomalies;

- **Guarantees fitness** – which means that the model represents fully the behavior expressed on the log file;

- **Guarantees termination for any input log in finite time**;
• All discovered models correspond to sound, block-structured workflow (WF) net systems.

As for time complexity, Leemans et. al [20], argued that the algorithm runs in a time polynomial to the number of activities and the size of the log.

2.1.2 Process Mining Techniques- Hybrid approach using Clustering

Existing log mining techniques are unable to deal with over-complexed models without the excessive over-generalization, if a single model for every case is constructed [23]. Medeiros et al. [23] approach this problem using clustering based techniques. Clustering can be defined as the process of partitioning a set of data objects/observations into subsets [24]. Using clustering techniques, instead of constructing one big model by just mining event logs that may produce a process model that is too complicated to understand and over-generalized, it is possible to construct a much simpler and specific model for every cluster [25]. This can be accomplished by using a clustering algorithm, such as, *k*-means [26] first and then a process discovery algorithm. K-means is based on Euclidean distance in vectorial spaces and it functions by finding central points (or centroids) over which a set of vectors are clustered. The k parameter determines the number of central points, which means the number of clusters presented. This algorithm is popularly used, due to being easy to implement and because the problem grows linearly as it grows, so it has a time complexity of $O(n)$ [26].

K-means algorithm (and clustering techniques) have a clear function, which is to analyze the event logs and detect points of over-generalization and then use a mining algorithm, for instance, the *Heuristics Miner* [16] to actually do the process discovering. Clustering techniques are not going to do process discovery all by themselves, but they help group activities into clusters. Then extracting the process models for each of the clusters is easier to interpret and does not have over-generalization, making it more flexible in scenarios that deal with changing circumstances.

Although K-means is a popular algorithm, due to the reasons presented above, it has a flaw of needing to provide the desired number of clusters in advance. For this reason, M. Song et al. [27] tested some clustering algorithms, namely K-means, *Quality Threshold Clustering* (QTC), *Agglomerative Hierarchical Clustering* (AHC) and *Self-Organizing Map* (SOM).

QTC algorithm was developed for the clustering of coexpressed genes in the field of bioinformatics [27]. This algorithm, although more computationally expensive than the K-means, since its complexity can be as much as $O(n^3)$, where $n$ is the number of objects, it does not need to specify the number of clusters in advance and it guarantees to return the same set of clusters over multiple runs. Clustering is guided by a quality threshold, which determined the maximum diameter of the clusters.

The AHC algorithm [28], unlike the previous, gradually agglomerates smaller clusters into large ones. It works from the dissimilarities between the objects to be grouped together. A type of dissimilarity can be suited to the
subject studied and the nature of the data. The result of the algorithm is a dendrogram which shows the progressive grouping of the data which makes it possible to have an overview of the suitable number of classes into which the data can be grouped, i.e. the hierarchy of the clusters [27]. This is an advantage of using this algorithm but it also has some significant disadvantages, such as [29]:

- The AHC algorithm is very sensitive to a good initialization, which means that the algorithm needs necessary and sufficient data to produce meaningful clusters;
- The AHC algorithm may produce coincident clusters.

SOM algorithm [30] is a neural network technique used to map high dimensional data onto low dimensional spaces. Neural network learning methods, for certain type of problems such as learning to interpret complex real-world data, are one of the most effective. In a rough analogy, these networks are built out of a densely interconnected set of simple units, where each unit takes a number of real-valued inputs, which are possibly the outputs of other units, and produces a single real-valued output, that may become other unit’s input [31]. According to [32], Figure 9 shows the most common structure of a neural network, which is named multi-layer perceptron. The input layer will be given the input values to process, within the individual neurons, and then pass those values (output from the input layer) to the hidden layer. Then this layer processes the values and feeds its output to another hidden layer or, in this case to the output layer.

![Figure 9 - Common structure of a multi-layer perceptron.](image)

The aim of the algorithm is to group similar cases close together in certain areas of the value range, which will be mapped onto the same node (or neighboring nodes) in the SOM. Although this algorithm is easy to understand, simple to use, produces good intuitive results and the computational cost is comparable to K-means but it has some drawbacks. One of those is that it requires necessary and sufficient data in order to develop meaningful clusters. The quality of the data set is crucial in whether the algorithm is going to be used or not.
Based on the work of M. Song et al. [27], it was concluded that SOM was a better approach than the other clustering algorithms stated above. Table 3 compares the algorithms stated above presenting the respective advantages and disadvantages for each algorithm.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-means</td>
<td>• Easy to implement.</td>
<td>• Number of clusters needed to be provided in advance.</td>
</tr>
<tr>
<td></td>
<td>• Complexity of $O(n)$.</td>
<td></td>
</tr>
<tr>
<td>Quality Threshold Clustering (QTC)</td>
<td>• No need to provide the number of clusters a priori.</td>
<td>• Has time complexity as much as $O(n^3)$, where $n$ is the number of objects.</td>
</tr>
<tr>
<td></td>
<td>• Guarantees the return of the same set of clusters over multiple runs.</td>
<td></td>
</tr>
<tr>
<td>Agglomerative Hierarchical Clustering</td>
<td>• The output is a dendrogram showing the hierarchy of the clusters.</td>
<td>• Very sensitive to good initialization</td>
</tr>
<tr>
<td>(AHC)</td>
<td>• No need to provide the number of clusters a priori.</td>
<td>• Coincident clusters may result</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Self-Organizing Map (SOM)</td>
<td>• Easy to understand algorithm.</td>
<td>• It requires necessary and sufficient data to develop meaningful clusters.</td>
</tr>
<tr>
<td></td>
<td>• Produces intuitive results.</td>
<td>• Finding a good data set is critical to whether use a SOM or not.</td>
</tr>
<tr>
<td></td>
<td>• Computational cost similar to K-means.</td>
<td></td>
</tr>
</tbody>
</table>

2.1.3 Process Instance Discovery

Even though all of the process mining techniques mentioned before deal with process discovery issues and are able to produce a BPMN which correctly translates the behavior seen on the logs to a process model, the event logs used by these approaches are constructed in a way that every process instance and every activity are
identified by a unique ID (referred as case ID and activity ID, respectively, from now on). This case ID and activity ID are able to distinguish all the different activities and are able to point out which of those activities belong in a process instance, as shown in Table 1. It is a requirement to have this case ID and activity ID explicit on the log in order for the mining algorithms to mine the individual behavior of each process instance [33]. Therefore, there is a need to have those IDs explicitly in the log, but systems that support the execution of a business process often do not record in their logs such explicit information [34]. In order to solve this, there is a requirement to have a manual preprocessing of the logs, which is again a time-consuming task for the system administrator. Some automatic approaches exist and will be presented in the next section.

2.1.3.1 Automatic approaches for case ID discovery

Ferreira and Gillblad [35] tackle this issue using a probabilistic approach based on an iterative Expectation-Maximization procedure. After providing an estimate to the transition matrix $M$ (which is also a first-order Markov model which specifies the conditional probabilities for the transition between any two activities) and having the symbol sequence (each activity is labelled with a symbol letter), it is possible to obtain the unknown source sequence, where each element of the sequence, specifies which source produced which activity.

Although it tackles the issue of not having the case ID explicitly on the log, this approach has some workflow patterns that is not able to deal with, mainly patterns with loops or parallelism. These patterns are often encountered in real life scenarios and cannot be ruled out and due to this, the approach was discarded.

Bayomie et al. [36] have an approach named Deduced Case ID (DCI) which automates the preprocessing step of discovering the case id in the log by deducing said ID from an unlabeled log. Then, a set of labelled event logs is generated, and those logs are listed according to a ranking score used to indicate the degree of trust. The DCI approach is based on three major inputs:

- The unlabeled event log;
- Heuristic data, obtained from domain experts, simulation results, or aggregation of the events in the original;
- The process model from the original documentation of the business process at design time.

These inputs are not always available and also this approach is not able to deal with cyclic process models (also the problem with the approach mentioned above), therefore this approach was excluded for these reasons.

Andaloussi et al. [33] developed a plug-in from ProM Tools framework [21] named Infer Case ID (ICI) that has the purpose of inferring the case ID attribute from the log files. This plug-in generates different process models and evaluates them from four different quality dimensions called Control-flow Quality Dimensions. These dimensions are:
• Fitness - represents the ability of a process model to reproduce the control-flow of the traces recorded in the event log;

• Precision - used to quantify all possible behaviors allowed by a process model that are not recorded in the log;

• Generalization - used to quantify the extent to which the process model can reproduce correctly log traces not yet recorded in the event log;

• Simplicity - which is used to quantify the extent to which a process model is simple i.e. longer models that have the same score as the simpler one in the other dimensions are discarded.

Based on the score of those dimensions the model with better score is produced. This plug-in is built upon an assumption that cannot be guaranteed for all log files in a real-life scenario. The assumption is that the case ID must be explicitly mentioned in the log file. This means that one attribute (column on a CSV file, for example) needs to contain the case ID values. This attribute doesn’t need to be labelled but needs to contain the values which will be used to infer the case ID. This assumption cannot be considered due to the fact that there are a lot of PAIS that do not record such information in just one attribute. In fact, more than one attribute may be required in order to correctly identify a process instance and therefore, providing our case ID.

2.2 Discussion

After analyzing all the different approaches presented in this chapter, it can be concluded that two requirements are needed in order to architect a module to achieve the intended result. Those requirements are:

• To choose a process instance (which describes all activities participating in a case) discovery technique to infer the process instance identifier (Case ID) and the activity identifier (Event Name);

• To choose a process discovery algorithm so a log file can be mined.

For the process discovery problem, it was presented an approach relying solely on process mining techniques using logs, such as the alpha miner algorithm [18]. Unfortunately, this algorithm has problems in correctly discovering implicit dependencies, which poses a problem as in real-life problems behaviors like these occur. As an alternative Lijie Wen et al. [19] proposes the α++ algorithm, which, when applying two rules to reduce the redundant implicit dependencies, it can produce process models that have non-free-choice, but it still has issues with noisy logs and non-complete logs. Real-life environments have a lot of noise introduced in the logs, due to systems not being totally designed to provide full parsed logs and/or are environments which have a lot of exceptions (hospitals for example). These situations might present issues with noisy models and/or non-complete logged events. For that reason, one alternative needs to be presented. Weijters et al. [5] present an approach that as a goal to reconstruct a process model from a D/F-table and a D/F-graph, but is not proven to work in every
scenario and not every process model can be discovered. Also Weijters et al. [16] present an algorithm named Heuristics Miner that showed, based on their experiments, resiliency when presented logs with noise and behaviors that occur in low frequency, which is important in real-life scenarios and is very suitable for what the proposed module is trying to achieve. Another promising mining algorithm is the Inductive Miner (or $B'$) proposed by Leemans et al. [20] which demonstrated that it returns the most-precise, most-general model adhering to the models restrictions. Also, this algorithm is able to guarantee soundness, fitness and it was proven to terminate execution for any input log, i.e. the framework doesn’t endlessly loop when trying to produce a model for a given input. This mining algorithm is also a suitable candidate for the module’s mining.

Although process models can be constructed only using the mining algorithms presented above, there is an inherent problem. The available algorithms are unable to deal with over-complexed models without the excessive over-generalization, if a single model for every case is constructed [23]. To overcome this problem a hybrid approach can be implemented using clustering algorithms. After the data is partitioned in smaller clusters, a process mining algorithm can then discover the process model for each cluster. Models mined with this approach are smaller, simpler to understand and more specific than models using only process mining techniques. In this dissertation, it was highlighted some clustering algorithms: k-means algorithm, QTC, AHC and the SOM algorithm. Medeiros et al. [23] concluded in their work, that the SOM algorithm, presented the best results when compared to the other clustering algorithms.

All these approaches are able to deal with process discovery, but they all work under the same assumption which is to have the case ID explicitly discriminated in the event log. Due to the fact that real life logs are rarely originating from a centrally orchestrated process execution, the case ID might be missing from the log, classified as unlabeled log [35]. In section 2.1.3.1 it was presented some approaches to try to overcome this problem, but not one approach was able to deal with the level of abstraction and generalization that is required by the Process Discovery module.

To summarize, the approaches presented on section 2.1.1 can deal with the initial problem of performing process discovery, however they are unable to deal with the over-complexity and over-generalization problems. The hybrid approach however seems to be a better approach to deliver a simpler, more specific process models and it offers flexibility in scenarios with changing circumstances. However, they are not able to deal with the inherent lack of the process instance ID problem. Since all the approaches presented are unable to deal with this problem, this thesis will present an alternative solution that will be discussed in the following chapter.
3 Process Discovery Module

The main goal of this thesis is to design and develop a module to assist administrators to have a global view of the processes and activities employed in the organization and even inferring some that may not be strictly defined by the organization, saving precious time from administrators. This is done by automatically extracting processes from log files and network traffic. The module was designed following the strategy presented below:

- Receive log files from the systems that are being monitored and participating in the business processes – These log files are obtained during the normal functioning of the business processes;
- Assist the administrator in getting an overview of the log file – This helps in the sense that the administrator doesn’t need to study the log file to know which column represent what, thus saving time;
- The ability to produce accurate results – Using the appropriated algorithms to produce the most possible accurate representation of the business process model, in case that the process instance ID and activity ID are known or produce good quality approximations when they are not known;
- Being as versatile and independent as possible – The module is envisioned to be a plug-and-play, which means that everything that the module needs to perform a task is provided and no other third-party software or specific hardware is needed.

One aspect that is important to emphasize is that this strategy can be applied not only in a closed environment, which means in an environment that is not under attack in the moment that the business processes are being categorized, but also can be applied in an open environment where attacks may be occurring at the same time the log file is being generated. This is possible because it is the administrator who has the final word if a BPMN translates the correct behavior of the normal functioning of the business processes. If it is not correct and the administrator detects any flaws in the model, he can then correct it making it fully compliant with the business processes.

After having these business processes fully determined, reviewed and categorized, every activity/action that differs from the established “truth environment” are considered anomalies. Those anomalies may be considered faults in the system or potential attacks to the organization’s infrastructure.

3.1 Module’s Architecture

As mentioned previously, the module was designed to be semi-automatic in contrast with a fully automatic approach. But why was it designed that way? Is this a disadvantage? How is the module supposed to work with a PAIS? This chapter will answer all these questions.

To answer the design choice of opting for a semi-automatic approach and not a fully automated one, it is important to point out all the automated approaches in chapter 2.1.3.1. Almost all the approaches presented fail
to provide valid results when dealing with some workflow patterns, mainly patterns with loops or parallelism, which are characteristics presented in many real-life scenarios. Other approaches fail to work due to the fact that the process instance ID and activity ID must be explicitly described in one column of the log file. But if said IDs are spread to one or more columns? In that case these approaches fail to infer these IDs and will not produce good quality results. Finally, there are other approaches that require inputs that are not always available and therefore cannot be used in all scenarios. With this in mind, the decision to opt for a semi-automatic approach sounded appealing, since the automated approaches had such lackluster limitations and may require information that will not be always available. Additionally, another problem that the automatic approaches have it is in the training stage where, in most of the time, the output is incomprehensible and impossible to validate, introducing wrong data/patterns into the models. The semi-automatic approach assures that the output can be validated and, therefore, more reliable.

3.1.1 How does the module work?

The Process Discovery module was architected as illustrated in Figure 10: An input handler will receive an event log or network capture, then a labeling algorithm will create option labels for each column of the input provided (log, network traffic). Those option labels can refer to the attributes that identify process instances and activities or can be other options that will be discussed later. The administrator decides which option each column will have in the selection stage. Also, in this stage a filter algorithm will suggest to the administrator an action to be performed on every column, based on the type of values that said column contains. Afterwards, a mining algorithm will take all the selected options and mine a BPMN (or multiple) using the event log and use as process instance ID and activity ID the columns selected in the selection stage. After all that, the module will output all BPMN files mined.

![Figure 10 - Stages that comprise the Discovery Module architecture.](image)

Only the first two parts (the log input and choosing the options for each column) of the functioning of this module require feedback from the administrator, as the rest of the functioning is fully automatic. By doing this, this approach eliminates the need to have additional info that most of the time is not available, have the ability to not be restricted to simpler workflows, and the process instance ID and activity ID are not restricted to have one column to describe them.
Figure 11 shows an example of how the Process Discovery module is intended to function with a PAIS-IDS. One thing that is important to notice, and that is mentioned throughout this dissertation, is that the module is independent from any PAIS. The module will produce a BPMN, which is a graphical representation for specifying business processes, which the PAIS-IDS will use as a specification to perform conformance checking. The model produced by the module will only be used as a specification after the administrator checks if it is accurate enough or if it needs changes, which the administrator can perform.

The following sections will explain all the approaches used to conceive the Process Discovery module.

### 3.2 Handlers

The first and last stage on the module’s execution are the input handler (depicted on Figure 12) and the output handler, respectively and these handlers have one function. The input handler will receive a log file or network capture to be labeled and processed by the module and the output handler has the task of sending a zip file with all the resulting BPMNs for the administrator to download and inspect.

### 3.3 Selection Stage

The selection stage is comprised of two different algorithms: the labelling algorithm and the filter algorithm. Both algorithms will output two types of feedback to the administrator after analyzing the logfile/network capture. The next sections will describe the conceptual idea behind those algorithms.

#### 3.3.1 Labelling algorithm

The first algorithm, the labelling algorithm, will inspect the file and provide option labels for each column of the log file for the administrator to choose. There are four distinct actions to be chosen:

- **CaseID** - This action instructs the module to use the correspondent column as a one that contain values that unequivocally characterize all different process instances (referred to as Case ID in most literatures) when generating a BPMN;
- **Event** - This action instructs the module to use the correspondent column as a one that contain values that unequivocally characterize all different activities (referred to as Event Name in most literatures) when generating a BPMN;
• Don’t Know - If the administrator has no idea or is not sure of what certain column represents this option should be selected. This will instruct the module to use this column on all possible combinations of Case ID and/or Event Name;

• Delete - This option should be selected if the administrator is sure that a certain column has no meaningful information and should not be considered for the generation of the BPMN.

### 3.3.2 Filter Algorithm

As stated previously, when the administrator submits his log file or network capture, a filter algorithm is executed in order to try to filter which action should the administrator choose for each column. The filtered action can be changed by the administrator in the page shown in Appendix A – Selection Stage.

The algorithm works as follow:

- When a detection of a timestamp occurs, the corresponding column is assigned with the option “Delete”, as the ProM framework has a certain intolerance for timestamps and, therefore, cannot be used to serve as Case ID or Event Name;

- When there is an incomplete column (column that contains at least one blank element) in the log, the corresponding column is filtered as a “Delete”, as this column cannot be used by the mining algorithm as it throws an exception;

- If the label of the column (the first value of a column) is either Case ID or Event Name, the algorithm filters that the corresponding columns should be used as “Case ID” and “Event Name”, respectively;

- If none of the conditions stated previously are verified, then the algorithm is not able to filter anything, so it sets the value of the corresponding column as “Don’t Know”, so that each column can be used as “Case ID” and/or “Event Name” by the combinatorial algorithm that will be presented in the next section.

After the administrator has chosen the options for each column, the mining algorithm will inspect those options and will start to operate. The following figure depicts the selection stage page:

### 3.4 Mining Stage

Entering this stage indicates that the administrator has selected and submitted all actions that will be performed for each column and this is where the module will spend most of its execution time. As previously mentioned, from this point onwards, no more user interaction is needed, as the tasks are fully automated. There are two algorithms in this stage: The Combinatorial Algorithm and the Mining Algorithm and both work together inside a loop. The next sections will describe in detail how these algorithms work.
3.4.1 Combinatorial Algorithm

When submitting the desired actions for each column, the module will then proceed to analyze the options that the administrator chose and create a new labelled log file, based on the log file/network capture received in the input handler, including all columns except those that the administrator chose to delete. For each column chosen with the option “Don’t Know”, a BPMN must be mined for each combination of Case ID and/or Event Name, due to the fact that the administrator has no idea of what the column represents, and each combination can provide a completely unique BPMN. However, there are cases of redundancy, i.e. a combination that outputs the same model as a previously mined combination, when applying all the combinations for all the columns with the option “Don’t Know”. Table 4 demonstrates some cases of redundancy:

<table>
<thead>
<tr>
<th>Activity Sequence</th>
<th>Case ID</th>
<th>Event Name</th>
<th>Redundant?</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-B</td>
<td>A</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>A-B-C</td>
<td>A</td>
<td>B/C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A/B</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>C/B</td>
<td>✔️</td>
</tr>
<tr>
<td></td>
<td>A/C</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B/A</td>
<td>C</td>
<td>✔️</td>
</tr>
</tbody>
</table>

With this in mind, it is crucial to detect and ignore cases of redundancy due to the fact that the framework takes some time to initialize and mine a log file. An algorithm to detect these redundancies was implemented and when the algorithm detects a redundancy case, it skips that combination and the next combination is then checked. One important thing to emphasize is that if there are no columns with the “Don’t Know” option, the user needs to select a Case ID column and an Event Name column, due to the fact that both are a requirement for the mining algorithm to work.
This algorithm decreases the computational dramatically as the cost is no longer \( n! \), where \( n \) is the number of activities that belong in a sequence.

### 3.4.2 Mining Algorithm

When a combination is validated, i.e. non-redundant, a mining algorithm will use the check which columns will be used as process instance ID and which will be used as activity ID and proceed to mine a BPMN. After the BPMN is obtained, it is then stored in a folder. When all non-redundant combinations are used and all the BPMNs have been obtained, the algorithm will zip the folder and send it for the administrator to download.

For the process mining algorithm and based on the research presented in chapter two, two different algorithms were used to check the quality and accuracy of the BPMNs produced. Those algorithms are: The Heuristics Miner and the Inductive Miner.

### 3.5 Summary

In this section it was presented the structure and the basic concepts of the various functioning stages of the Process Discovery module. This module receives a log file and after the administrator submits the options for every column, the module will analyze the selected options and will produce the corresponding BPMNs.

This module helps system administrators in the sense that they don’t need to analyze the logs to find every activity and every process instance manually, which takes a lot of time. The module assists them due to the fact that they don’t even need to know column represent the Case ID or Event Name, they can just select the “Don’t Know” and let the module do every possible of combination of Case ID/Event Name from those options. When the module is done, a zip file containing all the BPMNs is available for download and, after that, is just a matter of analyzing them and see which better corresponds to the real scenario. If none corresponds to the real scenario, the administrator can adjust the BPMN as they are XML based.

It is also important to emphasize that the decision to opt for a semi-automatic approach was not considered in the design as a handicap for the module, as it was in fact the opposite. The purpose of the semi-automatic approach was to avoid the lackluster limitations and the possibility of needing certain types of data or “a priori” knowledge of the system that the log or capture was extracted, in order to provide an accurate result. In fact, the log file/network capture input and the option selection are the only components that require user feedback, as the rest of the module is fully automated.

In conclusion, this module saves time from the time-consuming task that is to manually generate a BPMN model from a log file which has no explicit column to be used as Case ID or Event Name. Also, the module can be deployed very easily since it is built in a way that it doesn’t require any specific operation system or program to be deployed.

The next chapter will explain in more detail how the Process Discovery module is implemented and which technologies that it uses to make it as versatile and easy to use as possible.
4 Implementation

In this chapter it will be discussed which technologies the Process Discovery module uses and how it’s built to be easy to use and versatile. When building this module, it was considered various approaches to build it with as little assumptions about the data as possible. Due to this, the module was designed to be semi-automatic so there is no need to assume or filter anything about the data and also there is no need to perform any type of probabilistic approach which can induce errors and/or not being able to produce a valid output.

4.1 Assumptions

The Process Discovery was envisioned and built as a semi-automatic module (as explained in chapter 3.1), which means that it is designed to operate alongside with the system administrator. Due to this there are some assumptions made in regards of the input data, more specifically it is assumed that: (a) Every line in the log corresponds to an activity. (b) There are no lines that do not contain meaningful info (containing noise or do not have any relevance for the purpose of obtaining a business process model). (c) There are no incomplete lines, leaving columns without elements and (d) there are no grammatical disparities between the same activity description, for example the activity “Lock Door A” appearing multiple times written in different forms, such as “lock door A” or “Lock door A”.

In contrast with the solutions presented in chapter 2, in this situation there is no need to assume that the Case ID and the Event Name are explicitly mentioned in the log, labeled or unlabeled, or that they are given by only one column. However, it is assumed that they can be obtained using one or more columns from the log file.

4.2 Webservice & Containerization

The Process Discovery module has two major components: a front-end and a back-end. The front-end which contains all the web pages was developed using HTML and JavaScript and the back-end was developed in Python using the Flask library to build the webserver. The communication between front-end and the back-end is done using JSON, which is very versatile and simple to use. There’re only two supported requests in the moment of writing this dissertation which are the GET and POST methods. The GET method is used to fetch something from the server whilst the POST method is used to submit values for the server to process. The following sections will show the implementation of the front-end.

4.2.1 Input Handler

When the administrator opens the link to the Process Discovery module, the following page will appear, prompting to the upload of a log file:
After the log file submission, the labelling algorithm and the filter algorithm will provide the options and filters, respectively and the user will be redirected to the Selection Stage, where all those values are exhibited.

4.2.2 Output Handler

After all the valid combinations have been mined, a zip file is created containing all the BPMN files that were mined using the option chosen in the selection stage. After that, the zip file is sent for the administrator to download and analyze.

4.2.3 Module’s container

As stated previously the module was built with the intention of running without any type of pre-requisites, which means that anyone who wants to use the module, just need to run it without worrying about any kind of dependencies and operating system’s restrictions. To achieve this, the module is entirely built inside a Docker Container [37] which is a software that performs operating-system-level virtualization (or containerization). Inside this container we define an operating system that said container will use to run the scripts which, in this case, is Ubuntu. Inside this container are also installed all the libraries, framework and scripts that the module needs to operate. Also, by using containerization, it is possible to use the module independently or in conjunction with other containers, making this solution a perfect candidate to perform automatized chains of tasks. Additionally, the module was built in a web service approach, which means that the only thing needed to run the module is to initialize the Docker container and open a web browser. This combination of technologies makes this module plug-and-play, low maintenance and versatile.

4.3 Mining Stage

After the log file or network capture is uploaded, two different algorithms will perform two different tasks. The first algorithm will create a dictionary, indexed by the column number, with the first ten distinct values for each column with the purpose of letting the system administrator know what values were found in each column. When this is done, the second algorithm will start to filter which kind of action each column will be used for.
4.3.1 Combinatorial and Redundancy Checker Algorithms

As explained in the previous section, when the administrator doesn’t know what the columns from the log represent the option “Don’t know” can be selected. When the back-end receives the selected options, it will check if any columns have the “Don’t know” option selected and if they don’t, only one BPMN will be produced, since there’s no uncertainties to deal with. If there is any number of columns with the “Don’t know” option however, all possible combinations for the Case ID and Event Name must be determined (pseudo-code available in: Appendix B – Pseudo-Code). For that, the module uses the permutations method from the itertools python library. After getting all combinations, the algorithm presented below will check if a combination is redundant, i.e. a combination that generates a BPMN identical from a previously used combination (Table 4 shows an example of this), and if it doesn’t detect a redundancy case, it will write the parameter file with all the fields needed, i.e. which column(s) represent the Case ID and which represent the Event Name, for the mining algorithm to produce an output. It is also important to emphasize that if two or more columns are selected to determine the Case ID and/or Event Name, a concatenation of all columns involved is performed and added to the log that will be used to mine the BPMNs. Finally, it starts the mining script which will produce a BPMN file.

When the combinatorial algorithm finishes, i.e. after all combinations were analyzed and, if not redundant, used to mine a BPMN file, it returns the name of the zip file containing all the mined BPMN files which is then redirected to the webpage for the system administrator to download.

4.4 Mining Algorithm

One of the strong components of this module is that it contains the ProM framework which is an extensible framework, developed in Java, that supports a wide variety of process mining techniques in the form of plug-ins [21] developed by the Process Mining Group, Eindhoven Technical University. Researchers and developers can contribute to the growth of the framework by implementing algorithms in the form of plug-ins.

If the algorithm described above doesn’t detect a redundancy case for a certain combination, a parameter file containing the name of the log file created after the selection stage, the labels of the columns used for the Case ID and Event Name and the name of for the mined BPMN is created and then, a bash script is executed. This script reads initializes the ProM framework [21] in Command Line Interface (CLI) mode and instructs the framework to run a Java script. This Java script reads the parameter file, sets all needed variables to use the necessary plug-ins and instructs the framework to mine a BPMN using the Heuristics Miner algorithm, described in section 2, using the columns in the parameter file. When the BPMN generated, it is exported to a folder created after the administrator the script ends and the combinatorial algorithm will provide a new combination for a given sequence until all sequences have been mined.

The mining script is a simple Java script (since the ProM framework is built in Java) which reads the parameter file and instructs the framework to perform three distinct tasks:
1. Generate a XES [38] file using as Case ID and Event Name the columns in the parameter file. The reason why XES format was opted over other formats is because it is the IEEE Standard for eXtensible Event Stream for process mining (IEEE standard 1849-2016);

2. Mine the XES file using the Heuristics Miner algorithm to obtain a Heuristics Net;

3. Convert the Heuristics Net into a Petri Net [12], and from a Petri Net into a BPMN file. This step is needed to be performed due to the fact that there is no plugin (using the CLI) that converts directly a Heuristics Net into a BPMN file;

4. Finally, the BPMN is exported to a specific folder (determined by the Python back-end and passed in the parameter file).

After the mining script finishes, it returns to the combinatorial algorithm for the next iteration.

4.5 Summary

To conclude this chapter is important to emphasize two aspects of the developed module. The first aspect is that it is packed with a powerful framework (ProM) developed by the Process Mining Group and it is constantly being updated and new plug-ins are constantly being built for it. The second aspect is that it is built inside a Docker container which allows the module to be independent of any kind of operating system or libraries, since the plug-in is built with all those libraries inside the container, making it a plug-and-play module with the capabilities to be put on a chain of automated tasks. Figure 13 presents a diagram illustrating the module’s composition.
5 Evaluation

During this thesis the main question that was always present was “At to what extent does this solution help system administrators?” and from this question many other sub questions begin to appear, for example “how can the module output accurate models?”. The answers to these sub questions led to the way that the module is designed and implemented. This chapter will be focusing on answering all these questions and to determine how much this module helps, to semi-automatically generate a BPMN to be used as specification, and what are its limitations.

5.1 Limitations

Before discussing how the module helps, it is important to first take look on what are the limitations of the module regarding all the stages of the module. Each stage has its own limitation not only due to the implementation of the module itself, but also from the framework that this module uses. The next sections will explain what these limitations are for each stage.

5.1.1 Input Handler – How much log quality influence the end result?

In this stage the main focus is the log file that the administrator uploads to be processed and the limitations that every stage of the module (and this stage in particular) has is regarding the quality of those logs. As explained in chapter 4.1 there are a few assumptions (which work as limitations) that were made during the development stage of this module. One of these assumptions is that the module is not able to function if there are columns that have blank elements (due to line incompleteness). For example, if the column that contains the Event Name has blank elements and the administrator chooses that column to be processed, an exception will occur when the framework tries to use it on the mining algorithm. Due to this fact the filter algorithm will filter that such columns should be eliminated to prevent this exception to occur. Eliminating the lines with blank values could be an option, however this may introduce incorrectness in the models.

Another assumption is that logs may not contain any kind of noise such as, logs that contain grammatical errors and/or lines that do not contain any information that can introduce noise to the log, also as explained in chapter 4.1. This kind of noise will show on the BPMN producing inaccurate results. Figure 14 and Figure 15 show an example of this where it was added one line at the end of the log containing the event “Induced Error”. The BPMN obtained and shown in those figures were obtained with the Heuristics Miner and using the same log file, obtained by capturing the packets from a connection to the first ever website\(^1\), depicted on Appendix A –

\(^1\) Available at: http://info.cern.ch/hypertext/WWW/TheProject.html
Selection Stage by selecting the column “Case ID” as the Case ID option and the “Event” column as the Event Name.

This is an example of adding an intentional line in the log that does not contain any relevant info, hence being named noise, and the module is not able to detect it. This behavior can also be the result of a known exception and, in which case, must be on the log if it happens often enough (and consequently be in the BPMN). However, in this case it only happens once and doesn’t hold any useful information, thus it is considered an inaccurate result. Also, to note that Figure 14, although showing the correct behavior from the log file, it presents some redundant paths making it not the most concise and simpler model.

Log quality is not only the limitation for this stage, but it also is for all stages making this limitation the most impactful one and the one that the administrator needs to ensure that it complies with the module’s restrictions in order for it to be able to produce BPMN. When facing the log quality issue, another issue that comes to mind is the “Ground Truth” problem. When the quality of the log, that will generate the specification is put into
question, so is the ground truth basis. IDSs that utilize specifications as a learning basis need to have the most accurate data provided to them, otherwise they will not be able to differentiate an attack from a human error, which is going to contribute to false alarms or the non-detection of attacks. As a result, the chosen algorithm for the module needs to be able to assess what is noise in the log and what is not, so it can provide models that reflect the business processes, as accurately as possible, in order for the IDSs to have a reliable training set.

### 5.1.2 Mining Stage

As mentioned in section 3.4, this stage is where most of the time is spent during execution, due to the fact that each time a BPMN is produced, the ProM framework needs to initiate which takes some time to do so, making it a bit slow when the administrator selects the “Don’t Know” action for many columns. Rather than this, the only true limitations are the ones involving the framework itself, as that the CLI mode doesn’t get the same development time and attention as the Graphical User Interface (GUI) gets, so a lot of plugins are not available without a GUI. One example of this limitations is the inability of the plugin that converts the log file into the XES format (which is a standard format and it is required for the mining algorithms to work) doesn’t support more than one column to be used as Case ID or Event Name. This limitation was overcome by concatenating every column that was selected to represent the Case ID and Event Name into a single column with all the values concatenated and then add that column to the log file to be mined. This, again, is an issue with the plugins developed for the framework and a clear example that the CLI and GUI version diverge in terms of functionalities, since that using more than one column is possible in the GUI version.

### 5.2 Induced Miner vs Heuristics Miner

In chapter 2.1.1 it was introduced two different approaches to mine BPMN from log files, but throughout this report only the Heuristics Miner was discussed. This chapter discusses the differences between the outputs of both mining algorithms and will assess which algorithm is more suitable for the module.

#### 5.2.1 Noise Handling between algorithms

As previously mentioned, the Heuristics Miner has a certain amount of intolerance to noise. In the example shown in Figure 15, only one line was added containing an error message. This created an inaccurate result since the behavior depicted, although translating the correct behavior registered in the log, it doesn’t do it with total accuracy. The last line (the error line) was depicted in the model as a state that always needs to occur when, in fact, was just a onetime occurrence. Instead of the model presenting one alternative path to the end, it translates that this “Induced error” state must occur every time when a new process is executed. As stated throughout this dissertation, the system administrator is able to check these results and change them accordingly to make them reproduce the correct behavior.

With all of this in mind, can the induced miner reproduce the correct behavior? Figure 16 and Figure 17 show the output of the Induced Miner produced using the same log file and the columns for CaseID and Event Name.
as the BPMN produced by the Heuristics Miner shown in Figure 15, with and without the added noise in the end of the log.

![BPMN diagram](image1)

*Figure 16 - BPMN obtained without noise using the Inductive Miner.*

![BPMN diagram](image2)

*Figure 17 - BPMN obtained with the adding of noise using the Inductive Miner.*

As depicted in Figure 16, the BPMN obtained using the Inductive Miner correctly translates the behavior (when compared to the specification depicted in Figure 18) and, in contrast with Figure 14, the model obtained from the Inductive Miner is free of redundant paths which make it the more concise and simpler model. Also, the BPMN obtained from the log which was added noise now shows the correct behavior with the added path to the end and the added exclusive gateway. It is also crucial to emphasize that the model in Figure 17, 100% corresponds to the specification depicted on Figure 16 which demonstrates that the Inductive Miner can produce very accurate results.

### 5.2.2 Experimental Work

In the previous chapter it was presented how the two algorithms differ in behavior in the presence of noise in the log file, but which of the two algorithms present better results when comparing to a specification, like the
one illustrated in Figure 18? To obtain a quantitative measure for the comparison between models, a need for a BPMN comparison tool was presented. From the research conducted it was assessed that the tool BPMNDiffViz\(^2\) [39] was the most appropriate to be selected due to being a web based application, which makes a versatile tool, it is maintained by the IEEE Task Force for Process Mining and gets regular updates (as of the writing of this dissertation, the last update was on the 2019-04-26). This tool has the purpose of finding structural differences between two business processes represented in BPMN format. The differences between models can be assessed visually by launching the tool but in this case, they were determined by utilizing a script that runs the tool’s comparison modules. This makes it possible to automatize the comparison process and log the results for further analysis.

![Figure 18 - Specification for the log file of the 1st and 2nd experiment.](image)

The further sections will be comprised of experiments made to assess which algorithm is better suited to integrate the Process Discovery Module in order to enrich its potential and to analyze the extent of the usage and usefulness of the Process Discovery Module.

---

\(^2\) Available at: https://pais.hse.ru/en/research/projects/CompBPMN
5.2.2.1 1st Experiment

The following section will describe the setup, results and conclusions for the first experiment, which has the specification in the figure above.

5.2.2.1.1 Setup

An experiment to evaluate the performance for each algorithm was conducted using the log file previewed in Table 5 and described in chapter 5.1.1. The chosen option for each column of the log file are shown in Table 6. The “Protocol” column was discarded since the column only has one value (“HTTP”) and does not present a valid contribution to the experiment. The “Time” column was also discarded since, as the moment of writing of this dissertation, the plugins used to mine the BPMNs were intolerant to timestamps.

Table 5 - Preview of the log file for the 1st experiment.

<table>
<thead>
<tr>
<th>CaseID</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>18:07.5</td>
<td>192.168.5.4</td>
<td>188.184.64.53</td>
<td>HTTP</td>
<td>485</td>
<td>Get Project</td>
</tr>
<tr>
<td>1</td>
<td>18:14.4</td>
<td>192.168.5.4</td>
<td>188.184.64.53</td>
<td>HTTP</td>
<td>516</td>
<td>Get Bibliography</td>
</tr>
<tr>
<td>1</td>
<td>18:14.5</td>
<td>188.184.64.53</td>
<td>192.168.5.4</td>
<td>HTTP</td>
<td>1480</td>
<td>Successful Operation</td>
</tr>
<tr>
<td>2</td>
<td>18:24.5</td>
<td>192.168.5.4</td>
<td>188.184.64.53</td>
<td>HTTP</td>
<td>485</td>
<td>Get Project</td>
</tr>
<tr>
<td>2</td>
<td>18:30.2</td>
<td>192.168.5.4</td>
<td>188.184.64.53</td>
<td>HTTP</td>
<td>510</td>
<td>Get What is</td>
</tr>
<tr>
<td>2</td>
<td>18:33.0</td>
<td>192.168.5.4</td>
<td>188.184.64.53</td>
<td>HTTP</td>
<td>505</td>
<td>Get Terms</td>
</tr>
<tr>
<td>2</td>
<td>18:33.1</td>
<td>188.184.64.53</td>
<td>192.168.5.4</td>
<td>HTTP</td>
<td>1339</td>
<td>Successful Operation</td>
</tr>
<tr>
<td>3</td>
<td>18:39.3</td>
<td>192.168.5.4</td>
<td>188.184.64.53</td>
<td>HTTP</td>
<td>485</td>
<td>Get Project</td>
</tr>
<tr>
<td>3</td>
<td>18:43.7</td>
<td>192.168.5.4</td>
<td>188.184.64.53</td>
<td>HTTP</td>
<td>511</td>
<td>Get History</td>
</tr>
<tr>
<td>3</td>
<td>18:45.9</td>
<td>192.168.5.4</td>
<td>188.184.64.53</td>
<td>HTTP</td>
<td>509</td>
<td>Get Proposal</td>
</tr>
<tr>
<td>3</td>
<td>18:46.1</td>
<td>188.184.64.53</td>
<td>192.168.5.4</td>
<td>HTTP</td>
<td>880</td>
<td>Successful Operation</td>
</tr>
<tr>
<td>4</td>
<td>18:52.8</td>
<td>192.168.5.4</td>
<td>188.184.64.53</td>
<td>HTTP</td>
<td>485</td>
<td>Get Project</td>
</tr>
<tr>
<td>4</td>
<td>18:54.8</td>
<td>192.168.5.4</td>
<td>188.184.64.53</td>
<td>HTTP</td>
<td>510</td>
<td>Get People</td>
</tr>
<tr>
<td>4</td>
<td>18:54.9</td>
<td>188.184.64.53</td>
<td>192.168.5.4</td>
<td>HTTP</td>
<td>385</td>
<td>Successful Operation</td>
</tr>
<tr>
<td>5</td>
<td>19:02.5</td>
<td>192.168.5.4</td>
<td>188.184.64.53</td>
<td>HTTP</td>
<td>485</td>
<td>Get Project</td>
</tr>
<tr>
<td>5</td>
<td>19:10.1</td>
<td>192.168.5.4</td>
<td>188.184.64.53</td>
<td>HTTP</td>
<td>516</td>
<td>Get Bibliography</td>
</tr>
<tr>
<td>5</td>
<td>19:10.5</td>
<td>188.184.64.53</td>
<td>192.168.5.4</td>
<td>HTTP</td>
<td>1053</td>
<td>Successful Operation</td>
</tr>
</tbody>
</table>
Table 6 – Options chosen for each column for the 1st experiment.

<table>
<thead>
<tr>
<th>Column</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case ID</td>
<td>Don’t Know</td>
</tr>
<tr>
<td>Timestamp</td>
<td>Delete</td>
</tr>
<tr>
<td>Source</td>
<td>Don’t Know</td>
</tr>
<tr>
<td>Destination</td>
<td>Don’t Know</td>
</tr>
<tr>
<td>Protocol</td>
<td>Delete</td>
</tr>
<tr>
<td>Length</td>
<td>Don’t Know</td>
</tr>
<tr>
<td>Event</td>
<td>Don’t Know</td>
</tr>
</tbody>
</table>

It is important to emphasize that any model that took more than 5 minutes (which was considered to be a reasonable threshold since more disperse models may take a lot of time to compare between each other), to be processed by BPMNDiffViz was discarded from the results (this is true for all experiments) since it was so disperse and complex, compared with the specification depicted in Figure 18, that retrieving valid results for those models was not possible.

### 5.2.2.1.2 Comparison Results

To understand the comparison results that are going to be presented in this chapter it is important to explain the measures that BPMNDiffViz [39] uses to give a final comparison result, i.e. how close/disperse is the evaluated model compared to other model, which in this case is always the specification model. When the tool receives two models it assessed the following information:

- How many elements in common have the two elements (which are designated as "Matched Elements")?
- How many elements are present in the first model and not the second (which are designated as "Deleted Elements")?
- How many elements are present in the second model and not the first (which are designated as "Added Elements")?

After this assessment the tool assigns a "Final Score" based on how many "Added" and "Deleted" elements are present in the comparison. "Matched" elements do not contribute to the "Final Score". With this in mind it can be concluded that the higher number of "Added" and "Deleted" elements presented in the comparison, the more disperse the two models are and higher the "Final Score" will be. It is also important to emphasize that, as this are comparisons, the lower the final score, the closer the two models are, and this means that a final score of 0 translate to identical models. These criteria and metrics will be used throughout all the experiments.
performed in this dissertation. Table 7 shows statistics from the performed comparisons using the two algorithms being evaluated:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Total BPMNs Available</th>
<th>Num. of Valid Results (%)</th>
<th>Lowest Total Score</th>
<th>Highest Total Score</th>
<th>Mean Total Score</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heuristics Miner</td>
<td>130</td>
<td>108 (83%)</td>
<td>11</td>
<td>908</td>
<td>205.6019</td>
<td>137.2980</td>
</tr>
<tr>
<td>Inductive Miner</td>
<td>130</td>
<td>74 (57%)</td>
<td>0</td>
<td>969</td>
<td>219.6081</td>
<td>158.8884</td>
</tr>
</tbody>
</table>

Figure 19 portrays the individual results for each comparison using the two different algorithms:

After observing the graph above and the statistics presented in Table 7, the following conclusions can be made:

- The Heuristics Miner produced more valid comparisons, i.e. comparisons that took less than 5 minutes to conclude than the Inductive Miner, which only about half are valid;
• The Heuristics Miner produced the lower mean value of the “Final Score”;
• The Heuristics Miner obtained the most concise results due to lower standard deviation;
• Finally, the most important conclusion is that the Inductive Miner was able to produce a BPMN model that is identical to the specification, i.e. when the tool compared the two models, the “Final Score” was equal to 0. This model was obtained using the column “CaseID” and the Process Instance ID and the “Event” column has the Event Name.

These results suggest that the Inductive Miner although having less valid results, a higher mean total score and the highest final score, it provided the most concise values and, most importantly, an exact replica of the specification model. This suggests that, in these conditions, the Inductive Miner would be the most suitable algorithm for the Process Discovery Module.

5.2.2.2 2nd Experiment

The following section will describe the setup, results and conclusions for the second experiment, which, just like the first experiment, has the specification illustrated in Figure 18, but in this case the CaseID column was deleted from the experiment.

5.2.2.2.1 Setup

Even though the specification for the two experiments are the same, the options for this experiment are slightly different as the one presented in Table 5 in the sense that the columns containing the “CaseID” and the “Length” were deleted. The “Length” column was deleted since it contains very few values and, therefore, deemed irrelevant. This experiment serves to assess if the algorithms can produce a BPMN similar to the specification without the explicit mention of the process instance ID.

<table>
<thead>
<tr>
<th>Column</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case ID</td>
<td>Delete</td>
</tr>
<tr>
<td>Timestamp</td>
<td>Delete</td>
</tr>
<tr>
<td>Source</td>
<td>Don’t Know</td>
</tr>
<tr>
<td>Destination</td>
<td>Don’t Know</td>
</tr>
<tr>
<td>Protocol</td>
<td>Delete</td>
</tr>
<tr>
<td>Length</td>
<td>Delete</td>
</tr>
<tr>
<td>Event</td>
<td>Don’t Know</td>
</tr>
</tbody>
</table>

Table 8 - Options chosen for each column for the 2nd experiment.
5.2.2.2 Comparison Results

Similarly, to section 5.2.2.1.2, Table 9 shows statistics from the performed comparisons using the same criteria:

Table 9 - Comparison Statistics – 2nd experiment.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Total BPMNs Available</th>
<th>Num. of Valid Results (%)</th>
<th>Lowest Total Score</th>
<th>Highest Total Score</th>
<th>Mean Total Score</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heuristics Miner</td>
<td>15</td>
<td>14 (93%)</td>
<td>14</td>
<td>235</td>
<td>145.2857</td>
<td>76.7625</td>
</tr>
<tr>
<td>Inductive Miner</td>
<td>15</td>
<td>8 (53%)</td>
<td>49</td>
<td>219</td>
<td>141.0000</td>
<td>60.5103</td>
</tr>
</tbody>
</table>

Figure 20 portrays the individual results for each comparison using the two different algorithms:

![Comparison Results](image)

Figure 20 - Individual Comparison Results – 2nd experiment.

Even though the sample size for this experiment is very small, with only 8 valid results for the Inductive Miner and 14 for the Heuristics Miner, after observing the graph above and the statistics presented in Table 9 some conclusions can still be made:
• The Heuristics Miner produced more valid comparisons, i.e. comparisons that took less than 5 minutes to conclude, than the Inductive Miner which only about half are valid;

• The Inductive Miner produced the lower mean value and standard deviation of the “Final Score”, because the sample size was short;

• In the conditions where the Process Instance ID was not present in the log, the Heuristics Miner was able to produce a BPMN model that was very similar to the specification, presenting a “Final Score” of just 14. On the other hand, the lowest “Final Score” that the Inductive Miner produced was 49 revealing that the Heuristics Miner could handle, in this example, the lack of the Process Instance ID better than the Inductive Miner.

These results suggest that the Heuristics Miner although having a higher mean total score and the highest final score, it provided a BPMN that is very similar to the specification. This suggests that, in these conditions, the Heuristics Miner would be the most suitable algorithm for the Process Discovery Module.

5.2.2.3 3rd Experiment

For the 3rd and 4th experiment a new log file, previewed in Table 10, was used to assess the performance of both algorithms. This new log file is an example available on ProM’s3 website and it is based on the business processes of a telephone repair company.

---

3 Available at: http://www.promtools.org/prom6/downloads/example-logs.zip
Table 10 - Preview of the log file used in the 3rd and 4th experiment.

<table>
<thead>
<tr>
<th>CaseID</th>
<th>EventID</th>
<th>dd-MM-yyyy:HH.mm</th>
<th>Activity</th>
<th>Resource</th>
<th>Costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>35654423</td>
<td>30-12-2010:11.02</td>
<td>register request</td>
<td>Pete</td>
<td>50</td>
</tr>
<tr>
<td>1</td>
<td>35654424</td>
<td>31-12-2010:10.06</td>
<td>examine thoroughly</td>
<td>Sue</td>
<td>400</td>
</tr>
<tr>
<td>1</td>
<td>35654425</td>
<td>05-01-2011:15.12</td>
<td>check ticket</td>
<td>Mike</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>35654426</td>
<td>06-01-2011:11.18</td>
<td>decide</td>
<td>Sara</td>
<td>200</td>
</tr>
<tr>
<td>1</td>
<td>35654427</td>
<td>07-01-2011:14.24</td>
<td>reject request</td>
<td>Pete</td>
<td>200</td>
</tr>
<tr>
<td>2</td>
<td>35654483</td>
<td>30-12-2010:11.32</td>
<td>register request</td>
<td>Mike</td>
<td>50</td>
</tr>
<tr>
<td>2</td>
<td>35654485</td>
<td>30-12-2010:12.12</td>
<td>check ticket</td>
<td>Mike</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>35654487</td>
<td>30-12-2010:14.16</td>
<td>examine casually</td>
<td>Sean</td>
<td>400</td>
</tr>
<tr>
<td>2</td>
<td>35654488</td>
<td>05-01-2011:11.22</td>
<td>decide</td>
<td>Sara</td>
<td>200</td>
</tr>
<tr>
<td>2</td>
<td>35654489</td>
<td>08-01-2011:12.05</td>
<td>pay compensation</td>
<td>Ellen</td>
<td>200</td>
</tr>
<tr>
<td>3</td>
<td>35654521</td>
<td>30-12-2010:14.32</td>
<td>register request</td>
<td>Pete</td>
<td>50</td>
</tr>
<tr>
<td>3</td>
<td>35654522</td>
<td>30-12-2010:15.06</td>
<td>examine casually</td>
<td>Mike</td>
<td>400</td>
</tr>
<tr>
<td>3</td>
<td>35654524</td>
<td>30-12-2010:16.34</td>
<td>check ticket</td>
<td>Ellen</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>35654525</td>
<td>06-01-2011:09.18</td>
<td>decide</td>
<td>Sara</td>
<td>200</td>
</tr>
<tr>
<td>3</td>
<td>35654526</td>
<td>06-01-2011:12.18</td>
<td>reactivate request</td>
<td>Sara</td>
<td>200</td>
</tr>
<tr>
<td>3</td>
<td>35654527</td>
<td>06-01-2011:13.06</td>
<td>examine thoroughly</td>
<td>Sean</td>
<td>400</td>
</tr>
<tr>
<td>3</td>
<td>35654530</td>
<td>08-01-2011:11.43</td>
<td>check ticket</td>
<td>Pete</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>35654531</td>
<td>09-01-2011:09.55</td>
<td>decide</td>
<td>Sara</td>
<td>200</td>
</tr>
<tr>
<td>3</td>
<td>35654533</td>
<td>15-01-2011:10.45</td>
<td>pay compensation</td>
<td>Ellen</td>
<td>200</td>
</tr>
<tr>
<td>4</td>
<td>35654641</td>
<td>06-01-2011:15.02</td>
<td>register request</td>
<td>Pete</td>
<td>50</td>
</tr>
<tr>
<td>4</td>
<td>35654643</td>
<td>07-01-2011:12.06</td>
<td>check ticket</td>
<td>Mike</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>35654644</td>
<td>08-01-2011:14.43</td>
<td>examine thoroughly</td>
<td>Sean</td>
<td>400</td>
</tr>
<tr>
<td>4</td>
<td>35654645</td>
<td>09-01-2011:12.02</td>
<td>decide</td>
<td>Sara</td>
<td>200</td>
</tr>
<tr>
<td>4</td>
<td>35654647</td>
<td>12-01-2011:15.44</td>
<td>reject request</td>
<td>Ellen</td>
<td>200</td>
</tr>
</tbody>
</table>

Figure 21 - Specification used in the 3rd and 4th experiment.

The following section will describe the setup, results and conclusions for the third experiment, which has the specification illustrated above.
5.2.2.3.1  Setup

The options to be used by both algorithms to mine the BPMNs for this experiment are presented in Table 11. As previously mentioned, the timestamps were not taken into consideration, due to the intolerance of the ProM framework to timestamps. The options translate a total lack of knowledge of the BPMN file, meaning that all non-redundant combinations have been mined.

Table 11 - Options chosen for each column for the 3rd experiment.

<table>
<thead>
<tr>
<th>Column</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case ID</td>
<td>Don’t Know</td>
</tr>
<tr>
<td>Event ID</td>
<td>Don’t Know</td>
</tr>
<tr>
<td>Timestamp</td>
<td>Delete</td>
</tr>
<tr>
<td>Activity</td>
<td>Don’t Know</td>
</tr>
<tr>
<td>Resource</td>
<td>Don’t Know</td>
</tr>
<tr>
<td>Cost</td>
<td>Don’t Know</td>
</tr>
</tbody>
</table>

5.2.2.3.2  Comparison Results

Just like the previous experiments, Table 12 show the statistics obtained for this experiment and Figure 22 shows the individual results of the comparisons.

Table 12 - Comparison Statistics – 3rd experiment.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Total BPMNs Available</th>
<th>Num. of Valid Results (%)</th>
<th>Lowest Total Score</th>
<th>Highest Total Score</th>
<th>Mean Total Score</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heuristics Miner</td>
<td>160</td>
<td>148 (92,5%)</td>
<td>22</td>
<td>351</td>
<td>148.7973</td>
<td>53.3707</td>
</tr>
<tr>
<td>Inductive Miner</td>
<td>160</td>
<td>67 (41,9%)</td>
<td>0</td>
<td>600</td>
<td>319.6716</td>
<td>203.9368</td>
</tr>
</tbody>
</table>
After observing Figure 22, containing the individual comparison results and the statistics presented in Table 12, the conclusions can be made:

- The Heuristics Miner produced more valid comparisons, i.e. comparisons that took less than 5 minutes to conclude than the Inductive Miner, which less than half are valid;
- The Heuristics Miner produced the lowest mean value of the “Final Score”;
- The Heuristics Miner obtained the most concise results due to much lower standard deviation than the Inductive Miner;
- Finally, even though the Heuristics Miner surpassed the Inductive Miner in most of the aspects evaluated, the Inductive Miner, once again, was able to produce a BPMN model that is identical to the specification, i.e. when the tool compared the two models, the “Final Score” was equal to 0. This model was obtained using the column “CaseID” and the Process Instance ID and the “Activity” column has the Event Name.

These results suggest that the Inductive Miner although having less valid results, a higher mean total score and the highest final score, it provided the most concise values and, most importantly, an exact replica of the specification model. This suggests that, in these conditions, the Inductive Miner would be the most suitable algorithm for the Process Discovery Module.
5.2.2.4 4th Experiment

The following section will describe the setup, results and conclusions for the forth experiment which, just like the third experiment, has the specification illustrated in Figure 21.

5.2.2.4.1 Setup

Even though the specification for the third and fourth experiments is the same, the options for this experiment are slightly different as the one presented in Table 11 in the sense that the column containing the “CaseID” was deleted. Just like with the second experiment it is important to assess if the Process Discovery Module can produce BPMNs that are close to the specification without the Process Instance ID being explicitly in the log file.

Table 13 - Options chosen for each column for the 4th experiment.

<table>
<thead>
<tr>
<th>Column</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case ID</td>
<td>Delete</td>
</tr>
<tr>
<td>Event ID</td>
<td>Don’t Know</td>
</tr>
<tr>
<td>Timestamp</td>
<td>Delete</td>
</tr>
<tr>
<td>Activity</td>
<td>Don’t Know</td>
</tr>
<tr>
<td>Resource</td>
<td>Don’t Know</td>
</tr>
<tr>
<td>Cost</td>
<td>Don’t Know</td>
</tr>
</tbody>
</table>

5.2.2.4.2 Comparison Results

Just like the previous experiments, Table 14 show the statistics obtained for this experiment and Figure 23 shows the individual results of the comparisons.

Table 14 - Comparison Statistics – 4th experiment.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Total BPMNs</th>
<th>Num. of Valid Results (%)</th>
<th>Lowest Total Score</th>
<th>Highest Total Score</th>
<th>Mean Total Score</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heuristics Miner</td>
<td>70</td>
<td>60 (85,7%)</td>
<td>31</td>
<td>276</td>
<td>147.2500</td>
<td>56.7983</td>
</tr>
<tr>
<td>Inductive Miner</td>
<td>70</td>
<td>34 (48,6%)</td>
<td>31</td>
<td>586</td>
<td>301.2059</td>
<td>213.4886</td>
</tr>
</tbody>
</table>
After observing Figure 23, containing the individual comparison results and the statistics presented in Table 14, the conclusions can be made:

- The Heuristics Miner produced more valid comparisons, i.e. comparisons that took less than 5 minutes to conclude than the Inductive Miner, which less than half are valid;
- The Heuristics Miner produced the lowest mean value and lowest “Highest Total Score” value of the “Final Score”;
- The Heuristics Miner obtained the most concise results due to much lower standard deviation than the Inductive Miner;
- Finally, even though the Heuristics Miner surpassed the Inductive Miner in most of the aspects evaluated, both algorithms have the same lowest “Final Score”.

These results suggest that the Heuristics Miner although having the same lowest value for the “Final Score” as the Inductive Miner, it surpassed the Inductive Miner in every other aspect evaluated in this experiment. This suggests that, in these conditions, the Heuristics Miner would be the most suitable algorithm for the Process Discovery Module.
5.3 How does this module help System Administrators?

Since the beginning, the main focus and problem to solve on this dissertation is to help system administrators to semi-automatically generate Business Process models without having to spend a lot of time understanding the flow of the log, identify all the different activities and all the different process instances. Administrators who spend countless hours trying to make the model 100% accurate and simple are bound to not have time to perform other crucial tasks and this poses a problem.

With this module, not only one can produce models by only requiring the administrator to parse the logs, according to the restrictions imposed in section 5.1.1, but also it is a versatile module which can operate without the need to install any other third-party software or libraries. Also, if the model produced by the Process Discovery module is not 100% accurate or has redundant paths, the system administrator can change the model to be in conformance with the business process of the company. The advantage of the module being semi-automatic (and for this case is a requirement, due to the need of validation of the model), is that it doesn’t consume much time to validate a model and there is a guarantee that the final model, after revision, will be 100% in conformance.

Another aspect that is important to emphasize is that the administrator doesn’t need to know in full detail the behavior of the log file. This is possible due to two distinct features:

- When the administrator submits a log file it is presented the first ten unique values for each column of the log (as presented in Appendix A – Selection Stage), making it possible for the administrator to have an insight of which columns contain which type of values and if those values are relevant to the generation of the BPMN;

- If there is a column, that contains certain values that the administrator doesn’t know if they are relevant or not, by selecting the option “Don’t Know” for that column, it is generated all non-redundant combinations for the selected columns. A BPMN is produced for each one of the combinations and the administrator only needs to check which of the combination is the most accurate and simpler and make the necessary changes if any is required.

Additionally, as seen in the results of the experiments above, the Process Discovery module can, in certain conditions, produce a BPMN model that is an exact replica of the specification model.

To summarize, this module not only offers versatility to work as independently as possible and a powerful process mining framework, ProM, also offers the administrators to spend as little time as possible when generating a business process model compliant with the business processes for the company due to the fact that they don’t need to have in-depth knowledge of the behavior of the log nor do they need to categorize every aspect of the said log to produce an accurate BPMN.
5.4 Summary

In this chapter it was discussed the limitations that the module has in each of its stages, comparison between two process mining techniques and as well as on answering the question “How does this module assists system administrators?” which is the main question and problem of this thesis.

As limitation go, the most crucial ones revolve around the quality of the input logs, meaning that logs that contain grammatical errors and/or lines that do not contain any meaningful information may produce inaccurate ore event incorrect BPMNs. Also, it is assumed that the log contains a column (or multiple columns) that serve as Case ID and the same is mandatory for the Event Name. If the log file doesn’t meet this condition, the module won’t be able to produce a correct BPMN. The administrator must ensure that the log file meets these requirements in order for the module to produce correct and the most accurate BPMNs possible.

Another set of limitations come from the framework used to generate the BPMN models, the ProM framework [21]. Its CLI component doesn’t have the same number of features that its GUI component have, making it more difficult to perform automated tasks. In fact, a simple design choice in the implementation of the module could meant a rework on the developed plugins that the framework already contains in its release state, so it is important to assess the difference between the two interfaces and adjust the design accordingly.

In this chapter it was also compared two mining algorithms that initially fit the requirements to be good candidates for the mining stage of the module. Those algorithms, Heuristics Miner and Induced Miner, were tested to determine which could produce the most sound, fit and simpler models by having those algorithms mine the same log with the same options, chosen in the Selection Stage, and compare those result with the specification. To perform the comparisons the tool BPMNDiffViz [39] was used since it was built and is maintained by the IEEE Task Force for Process Mining. From the four experiments made in this chapter, four main conclusions can be made:

- The Heuristics Miner performed better in almost every aspect than the Inductive Miner, when no Case ID was present in the log. It performed better in having most valid results (i.e. comparisons that took less than 5 minutes to conclude), more concise results (by having a much lower standard deviation) and lower mean values in almost every test;
- However, the Heuristics Miner could not generate, under any circumstance, a BPMN that was an identical match to the specification. The Inductive Miner however succeeded in that task under the condition that the Process Instance ID was explicitly present in the log. This leads to the conclusion that the Inductive Miner produces simpler results (without redundancy, as previously shown) and it is also more resistant to noise in logs;
- If the Process Instance ID is explicitly on the log file, then the Inductive Miner is the more suitable algorithm. If it is not, then it is more or less a tie between both algorithms since the objective is to be closer to the specification as possible and not to assess which algorithm present a lower average or the furthest model compared to the specification. With this in mind, it was concluded that the Inductive Miner is the more suitable algorithm for the Process Discovery module;
- It is possible for the Process Discovery module to produce an identical BPMN to the specification when the Process Instance ID is explicitly present in the log file. However, it was also assessed that, even though the Process Instance ID is not explicitly present, the module can produce a close-enough
BPMN to the specification, which can be refined by the system administrator to be used as specification for the business process.

To summarize, this module helps system administrators in two major ways: Not having the necessity to know the contents of the log file in-depth and to not have the necessity to categorize each activity and each process instance. If an administrator doesn’t know if a column could represent a Process Instance ID or an Activity Name, by selecting the “Don’t Know” option, all non-redundant combinations will be mined and a BPMN for each combination will be produced. Afterwards, the administrator only needs to inspect those BPMNs who are more accurate and perform changes, if needed, to make the BPMN correctly portray the business processes of the company. By its ability to be able to generate an exact replica of the specification model, the added value of the administrator not having to study the logs in-depth and for producing multiple possible results that can be changed accordingly, this module saves a lot of time for administrators to focus on more security critical tasks. The examples of log files used in the experiments translate to simple models, but the module is able to generate possible specifications with logs that translate to more complex models, due to the fact that the framework and algorithms used are capable of handling these more complex logs files. Even though a specification will not always be there to compare to the outputs, it was determined that, if the Case ID is present in the log, a model that, accurately, translates the behavior of the log file can be outputted.
6 Conclusion

PAIS-IDS are systems which use a process model specification to detect anomalies. But the formulation of the specification is performed manually, and, for large organizations, this task is one of the most time-consuming and labor-intensive in process monitoring. Therefore, there is a need to automatize the task of inferring business activities and processes, that may not be strictly defined by the company as much as possible. Also, a specification needs to be produced, as BP-IDS needs to have one prior to its deployment. In this work, a process discovery module was created based on the strategy described in chapter 3. After feeding a log file as input, the module will output one or more possible specifications (depending on the administrator’s knowledge on the log). Then the administrator is able to choose the most accurate model produced and if said model correctly translates the behavior presented in the log, the IDS (in this case BP-IDS) is ready to be deployed and start monitoring the network, in search for anomalies. If not, the system administrator can make changes to this model and correct it.

The module as a few limitations in particular the quality of logs which, it is assumed, to not contain any kind of noise such as, logs that contain grammatical errors and/or lines that do not contain any meaningful information, incomplete columns and it is assumed that is a combination of one or more columns that can unequivocally identify process instances and each activity, i.e. serve as Process Instance ID (Case ID) and Activity Name (Event Name) respectively. Also, the framework itself presents limitations since the CLI has fewer functionalities than the GUI, which makes design choices to be more calculated.

As for comparing the Heuristics Miner with the Inductive miner it was concluded that the Heuristics Miner proved to be better at outputting more concise and overall accurate BPMN models. However, it fails at portraying an exact replica of the specification under any circumstance, which the Inductive Miner is able to do when the Process Instance ID is explicitly present in the log file. This suggests that if the Process Instance ID is present, the better algorithm would be the Inductive Miner. If it is not present, then it is more or less a tie between both algorithms since the objective is to be closer to the specification as possible and not to assess which algorithm present a lower average or the furthest model compared to the specification. With this said it can be concluded that the Inductive Miner fits more criteria for what the Process Discovery module is trying to achieve and therefore, is the more suitable algorithm to be integrated.

This module helps system administrators in two major ways: Not having the necessity to know the contents of the log file in-depth and to not have the necessity to categorize each activity and each process instance, which saves a lot of time due to the fact that this task is done manually. As stated throughout this dissertation, if the administrator doesn’t know what a column represents, by selecting the “Don’t Know” option all non-redundant combinations will be mined and multiple BPMNs will be generated. Afterwards, the administrator only needs to inspect those BPMNs who are more accurate and perform changes, if needed, to make the BPMN correctly portray the business processes of the company.
When approaching a process discovery problem, it is of prime importance to understand which process discovery techniques are available and which produce a satisfiable specification model and in a more automated way. Future work will attempt to determine which technique is better suited for BP-IDS to ensure that the most reliable, correct, flexible and simpler business process model is determined.

6.1 Achievements

This dissertation reached an important achievement, which was to semi-automate the task of formulating the specification module for a PAIS to use as a ground truth, to perform conformance checking. This task was performed manually and was labor intensive, as described throughout this dissertation. As this task can never be a truly automated one, since the system administrator must validate that the specification is correctly showing the expected behavior of the workflow, it is now a less time-consuming time since that the only action that the system administrator needs to do is to validate the outputted models and see if they match with the business processes of the company. This was achieved by building a module that was able to correctly output BPMN models that corresponded with the specification for the business process or output a close-enough model to be inspected, modified and to be used as specification afterwards. The module was also built in a way that it is totally independent from the system that it is running on. This was possible due to it being built inside a container with all the required dependencies and libraries inside.

6.2 Future Work

As future work this dissertation could be continued by addressing the following:

- Implementation and testing of possible new techniques to correctly, more efficiently and automatically discover the process instance identification (Case ID) and the activity name (Event Name) column in the logs which, as explained, is mandatory to have for the mining algorithms to produce the resulting BPMN. This will save more time from the system administrator as he doesn’t need to spend time to understand and to choose which columns are more suitable for the process instance and activity discovery problem;

- Implementation of a network traffic sniffing function that will capture packets in real-time and, when finished capturing will generate a BPMN of the captured traffic.

- Implementation of better ways to filter the correct option based on the attribute values for each column. This will also increase the range of types of values that can be successfully filtered with this algorithm and will improve the ability to make this module more useful, as the administrator doesn’t need to label the log file in order for the filter algorithm to work.

- BP-IDS needs a set of handlers that recognize if an activity as occurred or not. These handlers are generated by fetching information from the log file. Implementation of an automated extraction of these
handlers need to be performed in order to better optimize the process of BP-IDS monitoring the network.

7 References


# Appendix A – Selection Stage

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>2021-01-01</td>
<td>Event1</td>
<td>Description1</td>
</tr>
<tr>
<td>2021-01-02</td>
<td>Event2</td>
<td>Description2</td>
</tr>
<tr>
<td>2021-01-03</td>
<td>Event3</td>
<td>Description3</td>
</tr>
</tbody>
</table>

## CSV Airline Selection

```csv
Date, Event, Details
2021-01-01, Event1, Description1
2021-01-02, Event2, Description2
2021-01-03, Event3, Description3
```
Appendix B – Pseudo-Code

*Pseudo-code for the combinatorial algorithm.*

def combinatorialFunction(possibleCombinations):
    for combination in possibleCombinations:
        # There are no redundancy cases in these situations
        if len(combination)==1 or len(combination)==2:
            write params file
            mine BPMN
            continue
        # There can be redundancies in these situations
        else:
            # Two situations are verified here. For the activity sequence A-B-C
            # Situation 1: Case ID = A and Event Name = B,C
            # Situation 2: Case ID = A,B and Event Name = C
            # Situation 1
            if len(combination%2) != 0:
                if !redundancyChecker(Used_Combination_List,combination1):
                    write params file;
                    mine BPMN;
                    if !redundancyChecker(Used_Combination_List,combination2):
                        write params file
                        mine BPMN
                        continue
                else:
                    if !redundancyChecker(Used_Combination_List,combination):
                        write params file
                        mine BPMN
                        continue
            return ZipFileName
Pseudo-code for the redundancy checker algorithm.

def redundancyChecker(Used_Combination_Dictionary, combinationCaseID, combinationEventID):
    if combinationCaseID in Used_Combination_Dictionary['Case ID']:
        if combinationEventID in Used_Combination_Dictionary['Event Name']:
            return True
    return False