Continuously optimizing business process using process mining

Maxim Bos
University of Twente
MSc Business Information Technology
m.bos-4@student.utwente.nl

ABSTRACT – Businesses perform business processes to deliver their product or service to the customer. Analyzing data about the execution of the business process with process mining allows organizations to improve their operations. Applying the PM^2 methodology, several analysis iterations are performed based on the chosen objective. A pre-processed event log as input for process discovery algorithms and performing a root cause analysis allows bottlenecks to be detected in the process. A developed dashboard allows these process mining algorithms to be combined in one tool. Using this tool allows single activities or process characteristics to be identified as bottlenecks. In the future, research can be done on how process mining can support the business process in an online setting. So that during execution, bottlenecks can be avoided or solved.

KEYWORDS – Bottleneck Detection, Continuous Improvement, Event Log, Petri net, Pre-Processing, Process Discovery, Process Mining, Root Cause Analysis

1. Introduction

In 2020, three workgroups within the community Accounttech, part of the NBA (Koninklijke Nederlands Beroepsorganisatie van Accountants), started researching forthcoming technologies that are relevant for accountants (Patrick Konniger & Dirk Niestadt, 2021). Despite the advantages it could bring accountants, the research showed that existing technologies such as data analytics and machine learning are not yet employed by accountants. SoliTrust is an organization which provides data analytics and IT audit services to bridge this gap for small to mid-cap auditor firms. The services offered by SoliTrust can be leveraged over multiple audit firms to support their business process. Thereby, the service of IT auditing and data analytics results in a higher quality audit. Since the legitimacy of a company's total financial administration can be checked at once, reducing the workload and duration of the audit.

To deliver the services, SoliTrust has a Petri net to execute workflows. The Petri net is a static structure which models the process in terms of activities. During a workflow, tokens flow through the Petri net triggering activities to be executed. These workflows result in a report requested by an auditor firm. The Petri net can handle diverse workflows for different outputs and inputs, resulting in heterogeneous activities during workflow executions. Furthermore, as the Petri net handles different kinds of workflows there are numerous activities and the process becomes complex. In general, the process can be described as loading, conversing, analyzing and reporting the data delivered by a company. The execution of events within the Petri net is logged in an event log storing information on all activities performed for a workflow.

The objective of the graduation project is to continuously improve the Petri net to improve workflow handling. By means of detecting bottlenecks in the Petri net by analyzing the event log. Solving these bottlenecks allows SoliTrust to perform the workflows more efficiently while requiring fewer human interventions.

2. Research Design

This section discusses the design of the research. It explains what methodology is used to achieve the objective. First, the methodology of the project is discussed. Thereafter, the research questions together with their methodology are explained. As well as the structure of the paper.

2.1 Project Methodology

The project aims to improve the business process performance by finding bottlenecks in the Petri net by utilizing the event log as input for process mining. Hence, a methodology is chosen which is suitable for process mining projects. The PM^2 methodology is chosen as a guideline during the project. As PM^2 is iterative and specifically designed for process mining projects tailoring for specific process mining activities. Other data-science project methodologies such as CRISP-DM are of a higher level and not made for process mining activities. While other process mining methodologies as the L* life-cycle model and process diagnostics method are not iterative and not suited for large complex process models.

The PM^2 methodology consists of six stages. The first two stages are (1) planning and (2) extraction, in these stages the initial research questions are defined and the data required for the project is extracted. After the first two stages, one or more analysis iterations are performed. An analysis iteration consists of the stages (3) data processing, (4) mining and analysis and (5) evaluation. Depending on whether the outcomes are satisfactory, the project moves to the next stage (6) operational support (Van Eck et al., 2015).

Figure 1 plots the methodology in the project based on the PM^2 methodology visually. It shows what steps during the project are performed and how this relates to the PM^2 methodology. The dotted lines are steps which could be performed.

2.2 Research Questions

To find improvement points with the event log the following main research question is formulated:

How can SoliTrust continuously improve the Petri net performance utilizing the event log as input for process mining algorithms to detect bottlenecks?

To answer the main research question, the following sub-questions are defined:

1. *What is a Petri net and how to influence its performance?*
    a. *What is a Petri net?*
    b. *What is an event log?*
2. *What is process mining and what models/techniques are available for (continuous) improvement?*
3. *How to prepare the event log as input for process mining algorithms?*
4. *How can the bottlenecks of the (constructed) process (model) be detected and what is the outcome of this?*
5. *Is the method of detecting bottlenecks valid?*
6. *What improvement points can be detected based on the results of process mining algorithms?*

2.3 Methodology Research Questions

Research Questions 1 & 2

For RQ1 & RQ2, a systematic literature review has been conducted according to the guidelines set by Kitchemham (Kitchenham, 2004).

The digital libraries Scopus and Web of Science are used to search for published studies. The search in Scopus and Web of Science are performed on 22 December 2022 for RQ1 and 9 January 2023 for RQ2. The search concerns studies published until the date of the search performed. To filter for relevant results, the following keywords are used:

- Petri Network

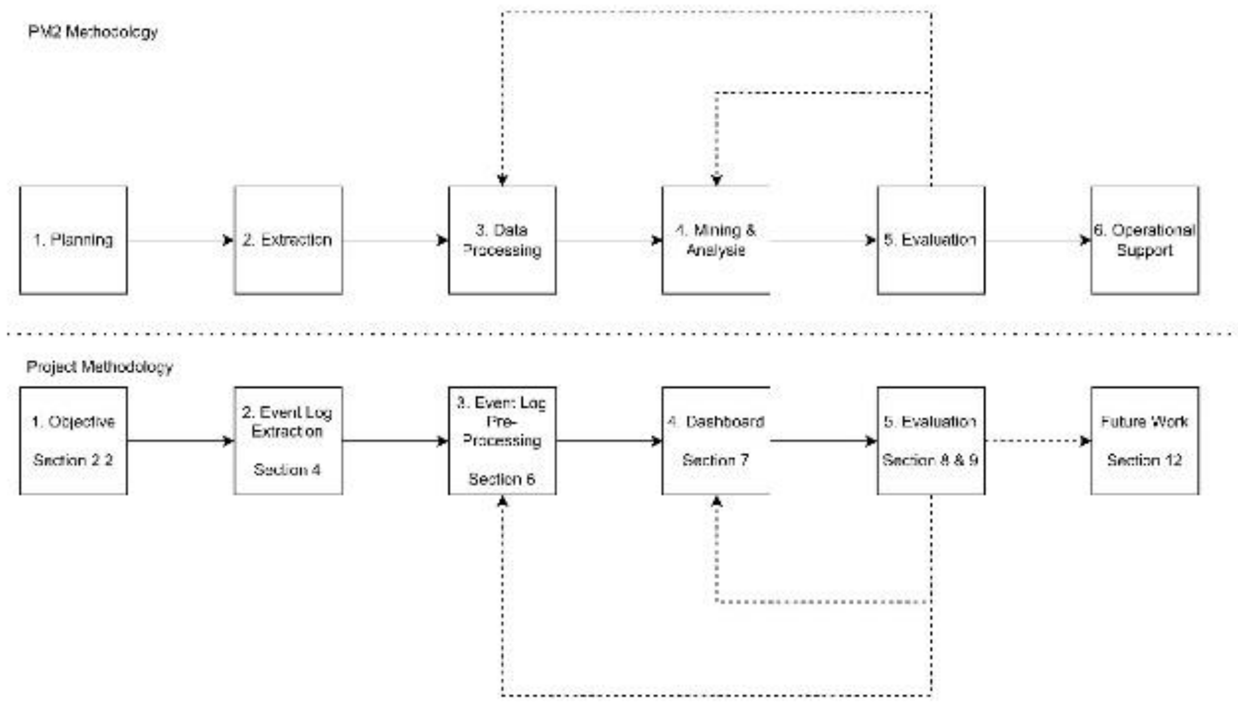


Figure 1: Project Methodology

- Performance
- Bottleneck
- Process Mining
- Discovery
- Performance Analysis
- Algorithm
- Continuous

For RQ1, the following search string out of the keywords is determined:

- “Petri net*” AND “performance” AND “bottleneck”

For RQ2, the following search strings deducted from the keywords are used:

- “process mining” AND “discovery” AND “petri net*”
- “process mining” AND “performance analysis”
- “process mining” AND “discovery” AND “algorithm*”
- “process mining” AND “contin*”

To filter for relevant results, inclusion and exclusion criteria are set. For the SLR, the following inclusion criteria are set:

1. The paper directly relates to Petri networks or process mining in the field of accounting and software systems.
2. The paper addresses the research question.
3. The paper is accessible online or via university resources.
4. The paper is in English.
5. The paper discusses process mining techniques for large processes and data or the technique seems scalable.

The following excluding criteria are set for this SLR:

1. The paper discusses constructs such as process mining and Petri networks to other specific domains such as cellular biology or digital twins.
2. The paper discusses analysis made in software not available for the research.
3. The reference comprises papers about a conference.
4. The paper discusses process mining with specific characteristics for data such as exogenous or incomplete.
5. The paper discusses perspectives of process mining not relevant to the project or not contained in the available log such as social network mining.

The table showing the number of papers found and what keywords are used is presented in Appendix 13.1.

Research Questions 3

For research question 3 a method is developed to pre-process the event log as input for process mining algorithms. Where a stepwise refinement (Wirth, 1971) methodology is used. Here the top goal, a ready event log for process mining, is split up into multiple steps. Specifically, the limitations of the event log are addressed by the steps below the top goal.

Research Questions 4 and 5

Research questions 4 and 5 are related to the development of the dashboard. Applying a process-oriented dashboard design methodology proposed by Maximilian Kintz (Kintz, 2012). However, for the project, a simpler version is needed as only one data source is used; the event log. Furthermore, there is no online setting so alerting and controlling the process is also not part of the graduation project.

Research Question 6

Research question 6 utilizes the developed dashboard to find improvement points in the Petri net by using the algorithms in the dashboard. For the discovery algorithm and root cause analysis, the settings can be adjusted.

Adjusting the settings is done with trial and error. Depending on the outcome, the settings can be adjusted. Setting the right interval is important, otherwise, the outcome becomes large and complex. Depending on the outcome and what the outcome shows, new settings can be chosen to make the model larger or incorporate other activities. For the discovery algorithm, the outcomes are judged visually by the user.

For the root cause analysis, the outcome is a decision tree. Here, the settings are set so the outcome underfits the data at first. So that only one root node is presented which shows the best split of the data. Then, the settings are changed so that the decision tree grows in depth and width. This stops when the decision tree starts to overfit the data or no changes can be made in the resulting tree with the new settings. The outcomes are judged based on the number of traces in the leaf nodes and the Gini-Index.

2.4 Paper Structure

Chapter 3, 4 and 5 address research question 1 and 2. Chapter 6 concerns research chapter 3. Question 4 and 5 are discussed in chapter 7 and 8. Chapter 9 concerns the results of the dashboard and addresses research question 6. The conclusions are drawn in Chapter 11.

3. Petri net

The project is about improving the business process model performance by finding bottlenecks. The business process of SoliTrust is executed through a Petri net. To understand what the business process looks like and what kind of data could be logged in the event log, an explanation of a Petri net is provided in this section. To understand event attributes in the event log, an understanding of Petri nets is needed. For that reason, Petri nets are discussed before the event log in Chapter 4.

3.1 Petri net definition

Petri nets are an abstract and formal model of information flow. Petri nets are subject to many prior investigations for process modelling languages and allow for concurrency and hierarchy. Petri networks model systems that are constructed by places and transitions, called nodes, as well as the relationship between them. The structure of the network is static. However, tokens can flow through the network. A Petri net can have different states, which are determined by the distribution of the tokens over places. The distribution of tokens over places is referred to as the marking of a Petri network. An example of a Petri net is provided in the bipartite graph of Figure 2. Here, the circles are places and the squares are transitions for an insurance claim request.

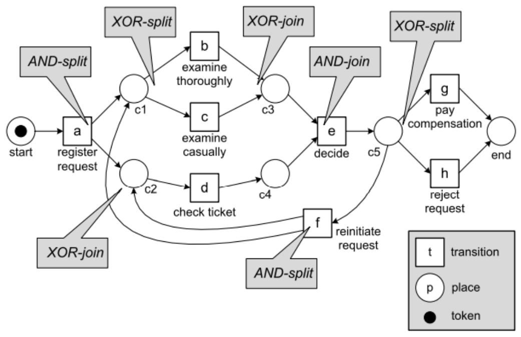


Figure 2: Example Petri net (I. Mukhlash, 2018; Peterson, 1977; Salimifard & Wright, 2001; W. Van der Aalst, 2016)

3.2 Petri net notation

A Petri net can be defined by the triplet $N = (P, T, F)$. Here P is the finite set of places, T marks the finite set of transitions and F is called the flow relation with a set of directed arcs. In the Petri net example of Figure 2, the triplet is defined as;

$P = \{start, c1, c2, c3, c4, c5, end\}$

$T = \{a, b, c, d, e, f, g, h\}$

$F = \{(start, a), (a, c1), (a, c2), (c1, b), (c1, c), (c2, d), (b, c3), (c, c3), (d, c4), (c3, e), (c4, e), (e, c5), (c5, f), (f, c1), (f, c2), (c5, g), (c5, h), (g, end), (h, end)\}$

A Petri net has to adhere to the following conditions;

- $P \cap T = \emptyset$

Meaning that a node in the Petri net can either be a place or a transition but not both.

- $F \subseteq (P \times T) \cup (T \times P)$

Meaning that a flow must always go from a place to a transition or a transition to a place (W. Van der Aalst, 2016).

3.3 Marked Petri nets

In the example Petri net of Figure 2, a token is placed on start, therefore the marking shown in Figure 2 is $\{start\}$. To go on towards a new marking, a transition has to be enabled according to the firing rule, which is explained in section 3.5. A transition can fire if each of the input places of the transition contains a token. In the example Petri net, *a* is enabled since the input place of transition *a* possesses a token. Firing *a* leads to the marking [*c1*, *c2*]. When *a* is fired, one token is consumed but two tokens are produced. At marking [*c1*, *c2*], *a* is not able to fire, since the input place of transition *a* no longer holds a token. However, at marking [*c1*, *c2*], transitions *b* ,*c* and *d* are enabled. When *b* is fired, it results in marking [*c2*, *c3*}]. Now, *b* and *c* are not enabled anymore. Contrary to transition *d*, which still is enabled (W. Van der Aalst, 2016).

A marked Petri net is denoted by a pair (N, M), where $N = (P, T, F\})$ and $M \in B\ (P)$ is a multi-set over P denoting the marking of the Petri net. The set where all Petri nets are marked is denoted as N.

A multi-set is a set with elements where each element may occur multiple times. Let $[a, b^2, c^3, d^2, e]$ be an example of a multiset. In this multiset, there are 9 elements: one *a*, two *b*, three *c*, two *d* and one *e*. Then the three multi-sets $[a, b, b, c^3, d, d, e]$, $[e, d^2, c^3, b^2, a]$, and $[a, b^2, c^3, d^2, e]$ are identical. Since only the number of occurrences matters, not the sequence (W. Van der Aalst, 2016).

3.4 Input and output nodes

To formulate the notation for input and output places or transitions, the elements in $P \cup T$ are seen as nodes. Node x can serve as an input node of another node y if and only if there is a directed arc from x to y meaning $(x, y) \in F$. Otherwise, x is an output node of another node y if and only if $(y, x) \in F$. Meaning that the input and output nodes can be denoted as:

$\bullet x = \{y \mid (y,x) \in F\}$, notation for node x being an input node of node y

$x\bullet=\{y \mid (x,y) \in F\}$, notation for node x being an output node of node y

In the example Petri net of Figure 2, $\bullet c2 = \{a, f\}$ and $c2\bullet=\{d\}$ (W. Van der Aalst, 2016).

3.5 Transition and firing

In a marked Petri net, transition $t \in T$ is enabled, notated by $(N, M)[t>$, if and only if $\bullet t \leq M$.

$(N, M)[t>(N, M')$ denotes that firing enabled transition t results in marking M'. In the Petri net of Figure 2, examples are $(N, [start])[a>(N, [c1, c2])$ and $(N, [c3, c4])[e>(N, [c5])$.

From a marking, a sequence can be determined in which the firings can take place. A firing sequence of (N, M) is denoted as $\sigma \in T$, if and only if there exists a natural number for which there are markings $M_1,\ldots,M_n$ and transitions $t_1,\ldots,t_n \in T$ such that $\sigma = <t_1,...,t_n>$ and, for all i with $0 \leq i<n$,

$(N, M_i)[t_i+1>$ and $(N, M_i)[t_i+1>(N, M_i+1)$.

Not all markings can be reached from an initial marking. In the Petri net of Figure 2, the initial marking $M_0 = [start]$ enables the empty sequence $\sigma = < >$. The sequence $\sigma = <a, b>$ is also enabled resulting in the marking $[c2, c3]$. The firing sequence $<a, c, d, e, f, b, d, e, g>$ is also enabled. A marking M is reachable from the initial marking if there exists a sequence of transitions whose firing leads to M from M_0. The set of reachable markings of (N, M_0) is denoted by $[N, M_0>$ (M. Leemans, 2018; W. Van der Aalst, 2016).

3.6 Labeled Petri net

In Figure 2 the transitions have been regarded with a single letter. However, in the Petri net also larger description with more detailed information about the activities could be provided. A labeled Petri net is denoted by $N = (P, T, F, A, l)$. Here P, T and F have the same definition as in the normal definition of a Petri net. The set of activity labels is defined by $A \subseteq A$ and $l \in T \rightarrow A$ is a labeling function.

It could be that multiple transitions have the same label. The label of the transition can also be seen as the observable action. However, sometimes a transition is not observable. With the label τ, it is denoted that the transition is not observable, referred to as a silent or invisible transition. A Petri net can be transformed into a labeled Petri net by taking $A = T$ and $l(t) = t$ for any $t \in T$. Reversing a labeled Petri net is not always possible, as transitions may have the same label (W. Van der Aalst, 2016).

3.7 Generic properties

The Petri net is k-bounded if no node in the Petri net contains more than k tokens. The Petri net is called safe if and only if it is a 1-bounded net. The Petri net in Figure 2 is an example of a safe Petri net. As no node could contain more than 1 token.

A Petri net is deadlock-free if at every marking a transition is enabled. When a transition is enabled in every possible marking, the transition is deemed as live. The entire Petri net is live when every transition in the net is always enabled. A deadlock-free Petri net does not require a live transition, as long as one transition is enabled, the Petri net is deadlock-free (W. Van der Aalst, 2016).

3.8 Colored Petri nets

When the Petri net should capture data-related and time-related aspects, a colored Petri net (CPN) could be used. Tokens in a CPN can be assigned a data value and a time stamp. This data value is often referred to as color and describes the properties of the object which is modelled by the token. The timestamp of the token can be used to track the time in the Petri net. Transitions can assign delays to the tokens, making a CPN suitable for modelling waiting and service times as an example (W. Van der Aalst, 2016).

3.9 Conclusion on (SoliTrust) Petri net

Petri nets are a static form of modelling processes. Through the net flows tokens which triggers the activities in the net. Key characteristics of Petri nets are hierarchy and concurrency. There can be multiple tokens on one place and several statistics can be gathered on a token. This is important for the graduation project to know to gain an understanding on the Petri net of SoliTrust. It explains how the Petri net operates which is needed for understanding the event log.

The Petri net of SoliTrust consists of 2816 activities of which 339 activities are silent activities. There are 2928 input places and 2605 output places. The Petri net can handle concurrency and also has hierarchy in the structure. The places in the Petri net are allowed to have multiple tokens on one place. Information about time are not stored on the token but the event log holds a starting and ending time of activities. The tokens do hold information about the command executed in the activity.

4. Event Log

The project aims to find bottlenecks in the Petri net using process mining. The input for process mining algorithms is an event log. This section discusses the definition of an event log as well as the characteristics and quality of the event log. Having an understanding of the event log is important for the success of the project and the quality of the outcomes.

4.1 Definition

The execution of the Petri net is logged in an event log. An event log captures activities performed in the execution of a workflow. The activities stored in the event log are referred to as an event. The workflow for which the event is performed is seen as a case in process mining. Each event in a log corresponds to one single case and relates to an activity or task. Furthermore a case or event can have attributes which contain information about the case or event. For example; for which customer the case is performed or at what time the event started. All activities together capture the execution of a process as an event log.

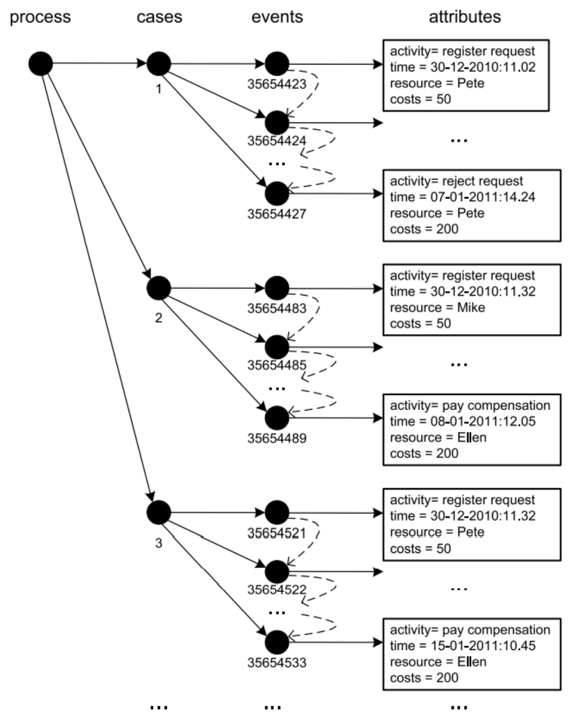


Figure 3: Structure Event Log

Figure 3 visually presents the structure of an event log, it shows that the process consists of cases which are composed of events. These events relate to only one case and the events within a case are ordered. ttributes which hold information about the event. These attributes can store information about the event (W. Van der Aalst, 2016; W. Premchaiswadi and P. Porouhan, 2015; Y. Caesarita, 2018).

4.2 Event Log SoliTrust

The event log of SoliTrust contains activities executed for the workflows in the Petri net. As the entire event log would be too large with 4,458,549 events. The event log is reduced to the workflows executed for the financial book year 2022. Meaning that the event log used in the project holds 1,226,714 events with 2816 unique activities for 758 cases. The event log is stored as a table in a database of SoliTrust's SQL server. The table in Appendix 13.2 shows what original attributes are present in the Event Log.

4.2.1 Characteristics Event Log SoliTrust

An event log contains the events executed for the business process. These business processes have different characteristics, which have an impact on the events contained in the event log. Depending on the characteristics of the process, challenges may arise to using the event log as input for process mining. In the case of SoliTrust, process characteristics of the Petri net are also represented in the event log creating some challenges.

To start the event log of SoliTrust stores a large number of events. The total event log with over 4 million events is around 30GB, preventing the total event log to be loaded into the memory of the computer. Furthermore a larger number of events also have implications on what process mining algorithms are applicable.

Also there is case heterogeneity in the event log of SoliTrust. Inferring that cases have different orders in which activities are executed which is known as a trace. The event log contains 2 traces which are executed in the same manner, all other cases are executed differently. Which is caused because the Petri net is flexible. The same Petri net can deal with different inputs and outputs by executing a different part of the Petri net. Which is represented in the event log by heterogeneous traces.

Furthermore the events stored in the event log are fine-granular, meaning that there is a large number of distinct activities in the event log. Process mining algorithms have difficulties with fine-granular events as this results in complex models. However a higher-level view of activities might discard relevant information from the log (Bose et al., 2013).

4.2.2 Quality Event Log SoliTrust

Regarding the quality of the data in an event log four broad categories are distinguished; missing, incorrect, imprecise and irrelevant. The event log of SoliTrust deals with irrelevant data for the project. All activities of the Petri net are logged even though the activity does not perform anything. Resulting in many events contained in the event log not being required for the project. Thereby there are activities in the Petri net only for layout options with no functioning, these activities are also represented within the event log. In addition there are also cases which are irrelevant in the event log such as test cases.

The ordering of events in the event log is important. As the event is logged when it is fully executed it can arise in the wrong location in the log. The timestamps are recorded in an Epoch Unix timestamp. To be used as process mining input the timestamps should be transformed into a timestamp in the form of YYYY-MM-DD HH-MM-SS. The activities are recorded with a precision to a second, which prevents calculating the duration of activities with a duration smaller than 1. However this is not seen as a problem for the project as these are not seen as activities of interest (Bose et al., 2013).

4.3 Conclusion on event log

An event log stores information about the execution of a process. It stores cases which are made out of events. Both cases and events can have attributes that hold information. Event logs capture the process and therefore also the behavior of the process. In case of SoliTrust, the event log is voluminous, fine-granular and heterogeneous. Furthermore, data in the event log is irrelevant for the project.

5. Process mining

Process mining is a research discipline that can be seen as a combination of machine learning and data mining on one hand and process modelling as well as analysis on the other hand. Process mining intends to discover, monitor and improve processes by analyzing available information from event logs. This section discusses process mining in terms of its role in the business process lifecycle as well as what types and perspectives process mining can take.

5.1 Process mining in the BPM life cycle

Figure 4 represents the business process management (BPM) life cycle. The BPM describes how processes are designed and how the process is adapted through the life cycle. The process starts at the design phase, where a process is designed. The process is transformed into a functioning system during the configuration and implementation phase. Once the system configured supports the designed process, the phase of enactment/monitoring starts to trace improvement points. When improvement points are detected, changes in the process can be made. In case no new design of the process or new software is needed, therefor only adapting predefined controls to reconfigure the process. The changes can be acted on in the adjustment phase. While the diagnosis/requirements phase monitors for larger emerging changes in the process and whether the system meets the requirements. Such changes may trigger the (re)design phase starting the cycle again. Process models have a significant role in the design and configuration/implementation phases of a system. Contrary, data plays a significant role in the enactment/monitoring and diagnosis/requirements phases. The data can be used as documentation to provide insights and analyze the performance of the process. While the process models have a role in verifying the system and determining its specification and configuration. In practice, many organizations do not continuously or systematically support the diagnosis/requirements phase. Only major problems or external changes trigger an iteration of the lifecycle. Thereby, once the (re)design phase is activated, the factual available information about the process is not actively taken into account. Here, process mining can help to connect all phases of the BPM life cycle. The available information can be applied to provide valuable insights into the process, detect abnormalities and support improving the quality of the models (Garcia et al., 2019; Leemans et al., 2015; Manoj Kumar et al., 2018; Milani et al., 2022; W. Van der Aalst, 2016; W. M. P. van der Aalst, 2011).

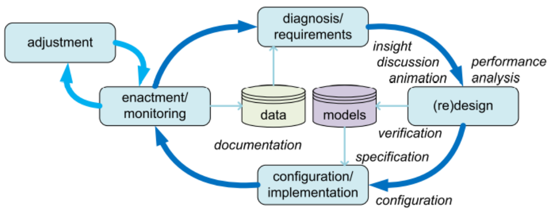


Figure 4: Representation of the business process management life cycle

5.2 Types of process mining

Process mining links the process and the generated data to the process model. This link can be made in different ways. Therefore, in general, three main types of process mining can be identified.

1. *Discovery.* Techniques for discovery process mining can produce a model out of an event log. Such a technique can transform an event log into a model without deductive reasoning (Batista et al.,

2019; Leemans et al., 2015; W. Van der Aalst, 2016; W. M. P. van der Aalst, 2010).

2. *Conformance.* Here, the event log is used to compare the process with the event log. The event log is used to check whether the actual process in the log corresponds to the designed process. For example, it could be that an activity with a certain condition requires a check. With conformance process mining, it can be seen whether this check is actually executed if the requirements are met. Conformance checking can therefore be used to detect, locate and explain deviations. Furthermore, the gravity of the situation can be assessed (Gyunam Park and Wil M.P. van der Aalst, 2021; Leemans et al., 2018).
3. *Enhancement.* With enhancement, the information available in the event log serves as a basis to extend or improve an existing process model. Enhancing the process is contradictory to conformance, as it aims for changing and extending the process model. Process mining that uses enhancement can use different types:
    a. *Repair.* Here the model is adapted to better reflect the reality it should represent. Which can occur when activities in the model are modelled sequentially. While in reality, the activities can occur in any order. Correcting the model to better reflect reality is an example of enhancement.
    b. *Extension.* An example of an extension is to add performance data within the model. Adding timestamps allows for bottlenecks, service levels, throughput times, and frequencies to be shown (Van der Aalst, 2016).

5.3 Perspectives of process mining

When the process model is extended with additional information rather than just the flow, process mining can take on different perspectives. Perspectives which are commonly used on process mining are:

- *The control flow perspective* focuses on the flow of the process. In other words, the ordering of the activities. When using this perspective in process mining, the goal is to find the behaviour of the process of all possible paths.
- *The organizational perspective* focuses on the resources that are hidden in the log. Depending on what information is available, the organizational perspective searches for the involvement and relationships of actors in the process. The aim is to structure the organizational structure by classifying actors or constructing the social network.
- *The case perspective* focuses on the properties of cases in the event log. The cases can be represented by the paths it takes in the process or by the activities performed. Nevertheless, cases can also be characterized by the values of their data elements. With delivery, it might be useful to know the supplier or the number of products ordered.
- *The timing perspective* is concerned with the timing and frequency of events in the process. Including timestamps in the event, allows for the detection of bottlenecks, measuring service levels, monitoring the utilization of resources and making a prediction about the remaining processing time (Kaouni et al., 2021; W. Van der Aalst, 2016).

5.4 Online and offline process mining

Many process mining techniques are done offline. Meaning that processes are analyzed afterwards to see how they can be improved. However, many techniques can also be used in an online setting, where improvement is made during the process. Which is also referred to as operational support. An example of operational support is predicting the remaining processing time or intercepting the process when a deviation in conformance takes place (W. Van der Aalst, 2016).

5.5 Play-In, Play-Out and Replay

An important aspect of process mining is the relationship between a process model and the reality represented by the event log. The terms Play-In, Play-Out and Replay are used to reflect on this relationship. Figure 5 visually shows Play-In, Play-Out and Replay.

Play-Out refers to the traditional use of process models, where the events in the process model are logged. In the case of a Petri net, Play-Out traces the token in the network. Play-Out can be used to analyze and enact business processes. Simulation tools and workflow engines are suitable for the Play-Out relationship.

Play-In is in contrast with Play-Out, it uses example behaviour to construct a process model. The Play-In relationship is also called inference. With the usage of process mining, process models can be discovered from event logs.

Replay uses both an event log and a process model. Where the event log is rehearsed on the process model. Replay may be used to extend the model, construct predictive models, conformance checking or operational support (W. Van der Aalst, 2016).

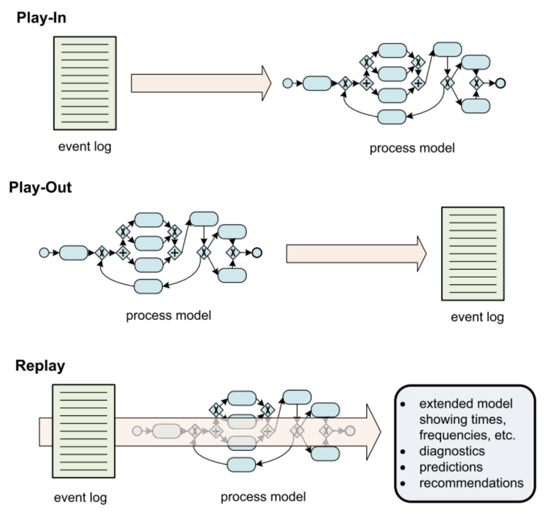


Figure 5: Play-In, Play-Out, Replay (W. Van der Aalst, 2016)

5.6 Root Cause Analysis

Process mining allows the detection of bottlenecks in a business process. Providing valuable information to organizations on where to improve their operations. However, solely indicating a bottleneck does provide organizations small information on how to deal with the bottleneck. Information on how the bottleneck originates can provide insights into how to oppose the bottleneck. With a root cause analysis, a more refined analysis can be performed to correlate different process characteristics.

De Leoni et al (De Leoni et al., 2016) propose a framework on how to correlate process characteristics to perform a more refined analysis. Analysis use cases can be used to provide valuable information to organizations. An analysis use case is a triplet consisting of a dependent characteristic, a set of independent characteristics and an event-selection filter.

The result of performing an analysis use case is a decision or regression tree where the independent characteristic relates to the dependent characteristic. So that the independent characteristic explains the dependent characteristics. The event-selection filter selects the events for which an analysis is made. For example, events performed by a specific resource. In case there is little information about the traces stored in the event log, the log and traces can be manipulated to add process characteristics.

Following the paths to the leaf nodes in the decision or regression tree explains how the dependent variable is impacted. The traces in the leaf nodes can also be used as a cluster for other process mining activities, such as process discovery (De Leoni et al., 2016).

5.7 Conclusion on Process Mining

Process mining is a combination of BPM and data science. For the project, process discovery with enhancement are relevant. Conformance is not relevant as the workflows cannot differ from the structure of the SoliTrust Petri net. The time perspective and case perspective are relevant for the project. As conclusions can be drawn about the throughput time of workflows and what workflows are troublesome. The project is conducted in an offline setting with post-mortem data. Play-In and replay are relevant techniques to find potential bottlenecks in the process. Because the discovered process model can be extended. Root cause analysis is also relevant for the project as this can unveil potential causes why a bottleneck occurs.

6. Event Log Pre-processing

Event logs store information about the execution of the business process. Process mining algorithms can use an event log as input and extract the knowledge to gain meaningful insights into the process. Therefore, the log used as input has a great impact on the result of the process mining algorithm. Or in other words; "garbage in – garbage out", referring to low-quality data implying low final-quality knowledge. Therefore attention should be given to the event log as input to succeed in process mining. This section discusses how an event log can be preprocessed to serve as input for valuable results. These pre-processing steps oppose the process characteristics and quality issues within the event log discussed in section 4.2.

6.1 Pre-processing theory

Real-life processes can be complex, hence the data in the event log can be voluminous and of high variability. Using this raw event log as input for process mining algorithms infers spaghetti-like process models which are difficult to analyze. With the help of pre-processing; algorithms can provide simpler process models which are easier to analyze. Which is visually shown in Figure 6.

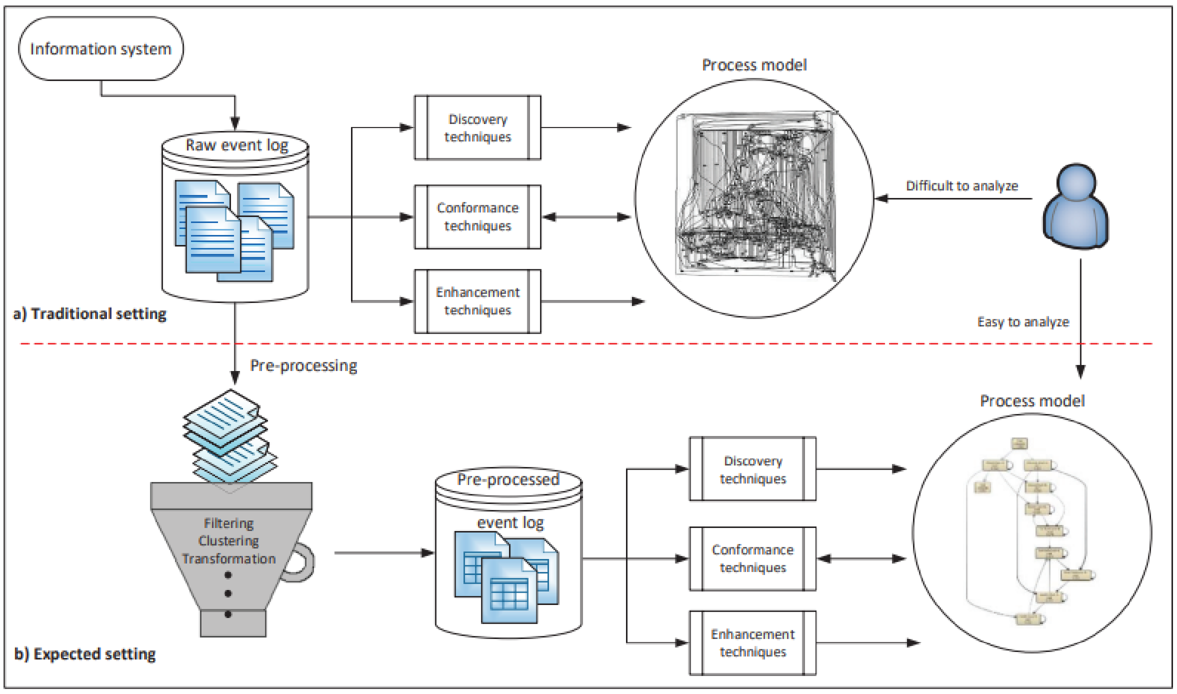


Figure 6: Event Log Pre-Processing (Marin-Castro & Tello-Leal, 2021)

Marin-Castro (Marin-Castro & Tello-Leal, 2021) proposes a taxonomy for preprocessing techniques, where event log preprocessing can be organized into two main groups; transformation techniques and detection and visualization techniques.

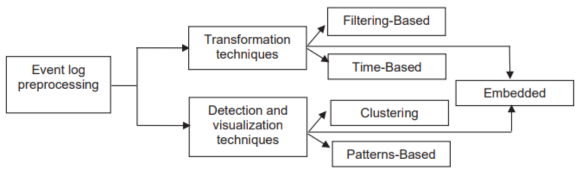


Figure 7: Pre-Processing Taxonomy

6.1.1 Transformation techniques

Transformation techniques search for changes in the original structure of the raw event log to improve the quality. For transformation techniques, two approaches are determined; filtering-based and time-based.

Filtering based

Filtering based aims to exclude events or traces with a lower frequency, focusing on the likelihood of occurrences for an event or a trace. Removing these events or traces from the log excludes them to appear in the constructed process model. On the other hand time-based focuses to maintain and correct the order of events within the event log.

Filtering techniques search for a typical behaviour in the log to eliminate. It fundamentally addresses the concerns for noise and anomalous events in the event log which may affect the performance of future process mining tasks. By filtering out infrequent behaviour in the log based on frequency.

Time-based

Time-based transformation techniques are another way to preprocess the event log. Timestamps can point out performance issues in the process. Time-based transformation techniques can repair and change the timestamps present in an event log. Furthermore an incorrect ordering of events can harm the result of process mining techniques. Especially discovery algorithms benefit from an event log where the events are properly sorted. Strategies concerning the information in timestamps and the ordering of events are therefore of interest to improve process mining outcomes.

6.1.2 Detection and visualization techniques

Detection-visualization techniques is a preprocessing group aiming to recognize, group and isolate events or traces which may cause quality issues in the log. The event log is divided into subsets by clustering techniques, these subsets are analyzed for noise and anomalous elements. It is possible to extract imperfection patterns from the formation of similar clusters (Fani Sani, 2020; Marin-Castro & Tello-Leal, 2021).

6.2 Data Preparation Method

As the quality of the event log is important, a method is used to preprocess the event log to ensure the quality of the process mining algorithms. The data preparation method should deal with the characteristics of the event log mentioned in section 4.2.

The event log is extracted from one process. Which has as benefit that the event log does not have to be compiled from multiple sources. As the system automatically stores the execution of processes within the Petri net. Thereby guaranteeing the same fine granularity in the event log.

Because the Petri net fires every transition, even when the transition does not execute an activity for the business process, the event log contains noise. As it stores transitions that are not relevant to the analysis of the business process. Trace attributes are not found in the event log and are constructed from other available databases in the SQL server of SoliTrust.

The Petri net is designed to handle workflows for financial analysis. As the customers come from different sectors implying variation in the output and input. The Petri net has a flexible setup so that only a partition of the transitions is actually run. Causing the event log to have a heterogeneous mix of traces with diverse and unstructured behaviours.

For these reasons, the data preparation method should pre-process an event dealing with fine-granular, voluminous and heterogeneous characteristics. As a method, the process in Figure 8 is proposed.

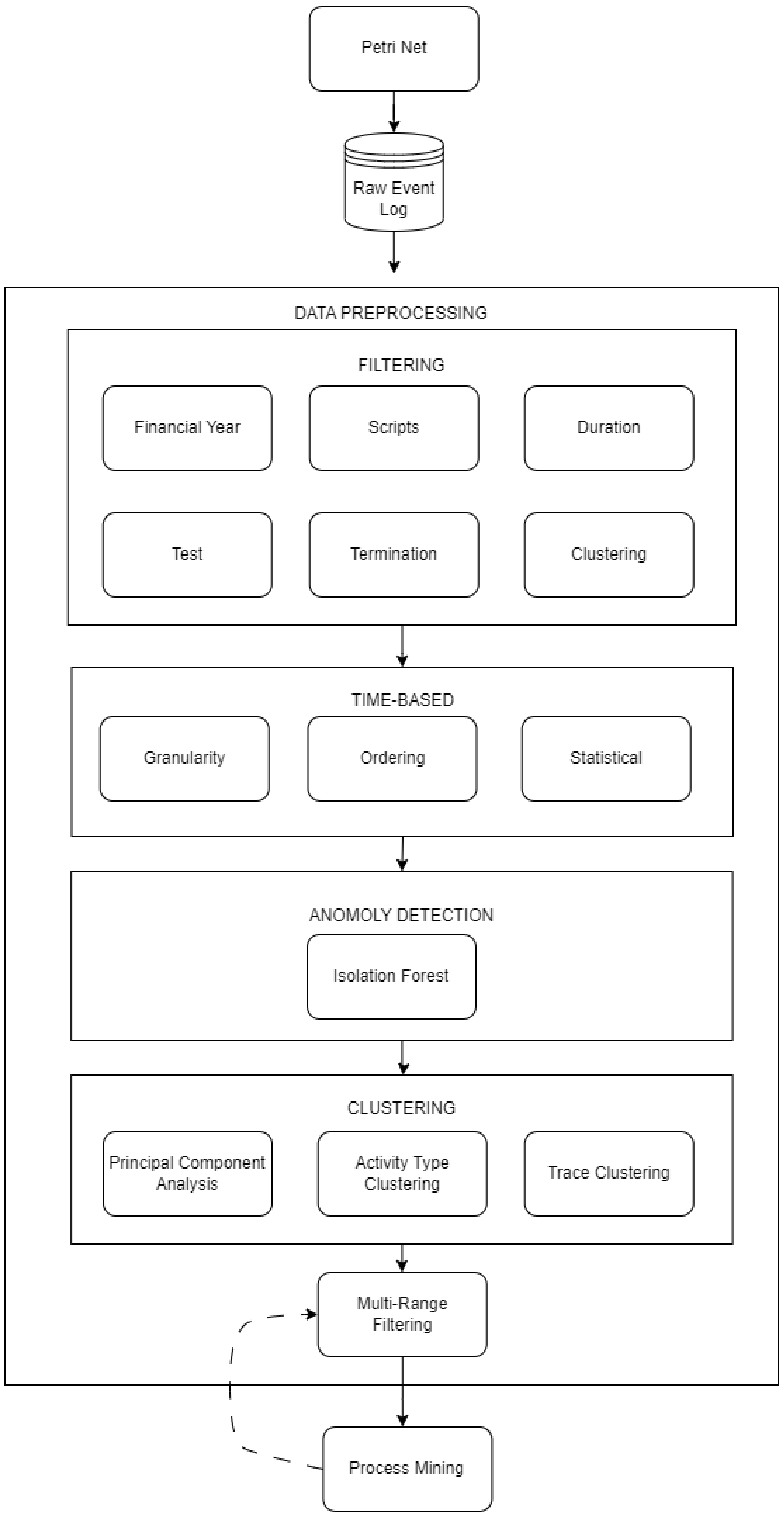


Figure 8: Event Log Pre-Processing Method

6.2.1 Filtering

Figure 8 starts at the Petri net filling the raw event log with data. This raw event log contains over 4 million events. To reduce the lavish data, the first step in the data preprocessing step is filtering.

The first filtering step of the data pre-processing uses attributes extracted from the trace or event attributes to exclude events or traces to preserve wanted behaviour. Attributes and values used to include and exclude events or traces are:

- *Financial Year: 2022*
  For the graduation project only workflows for the financial book year 2022 are considered. This is the year before the graduation project and contains the behaviour of the Petri net in its most recent shape. Furthermore, taking a financial year should still provide plenty of data points for all kinds of workflows in the Petri net. In addition, taking the financial book year 2022 is set with the approval of SoliTrust.
- *Scripts: Empty*
  There are 2816 distinct transitions stored in the event log. However, not all fired or stored transitions do execute a process or activity when fired. Some transitions are in the Petri net for an easier layout. To only consider transitions actually executing activities, events with no scripts are excluded from the event log.
- *Duration: 0*
  In case an event has a duration of zero seconds, the event is excluded from the event log. Because the event has not been executed but does contain a script. The timestamps in the event log are stored with precision to the second. In case, a transition is executed but takes less than a second, it is also excluded. However, this is not seen as an issue. As improving these transitions is not the goal of the graduation project because of lesser yield. An exception is made for events which cause an error and have a duration of zero. These are of interest for the graduation project to reduce errors.
- *Test: True*
  Sometimes test workflows are run through the Petri net. These test runs are excluded from the event log as they do not represent the behaviour of interest.
- *Termination: False*
  A workflow through the Petri net can be terminated, either by the user or the network itself. When a workflow is terminated, the events remaining or being executed are stored in the event log. When the workflow is run over, the same case id is used so the events executed twice are stored double in the event log. Keeping the terminated events provides wrong inputs for process mining algorithms. However, the trace is kept because it is of interest as something causes the workflow to be terminated which could be improved.
- *Attributes*
  Some attributes are not relevant as input for process mining algorithms. These are removed from the event log. For example, which processor core of the computer the workflow is run on, is excluded from the event log.

Figure 9 shows the decline in the number of events in the log when the event attribute filters are applied. Filtering on the financial year, script, duration, test and termination reduces the number of events from around 4.5 million to around 386 thousand.

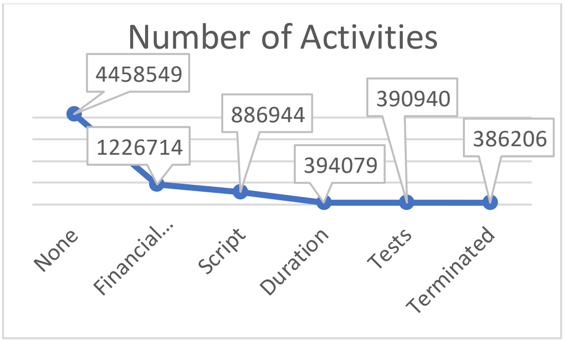


Figure 9: Event Attribute Filtering

6.2.2 Time-Based

For the analysis of the Petri net, the time viewpoint is used to determine throughput time and determining potential bottlenecks. Errors in the timestamps can provide misleading results in the performance of the Petri net. In addition, the correct ordering of events is also addressed in the second step of preprocessing.

- *Granularity*
  In case timestamps have a different level of granularity, the events can be wrongly ordered. Which may affect the process models discovered. The event log is checked whether timestamps hold the same granularity. In case the granularity is different, all timestamps are transformed

to the lowest granularity available in the log.

- *Ordering*
  Checking whether the events are in the right order is important for the quality of especially process mining discovery algorithms. The events should be ordered in occurrence in each trace. In case the ordering is mistaken, the ordering is changed to the correct order.

6.2.3 Outlier Detection

To remove outliers, anomalous traces are detected using isolation forest. Isolation forest uses the decision tree algorithm to isolate outliers. By randomly selecting a feature from the set of features and applying a split on the feature. The random partitioning of features produces shorter paths in trees for anomalous data. Thereby distinguishing anomalies from other data points in the available data.

It does not construct a profile of what is normal behaviour and does not make use of point-based distances. It uses the principle that anomalous observations are few and different. Which should make them easier to identify.

6.2.4 Principal Component Analysis

It could be that a pre-processing technique suffers from dimensionality. Therefore, a principal component analysis is used to reduce the number of features. In the pre-processing method, the principal component analysis is used to select the features until the set variance is explained. These features can be used by, for example, the clustering algorithm to cluster the traces (Zandkarimi et al., 2020).

6.2.5 Clustering

The event log of SoliTrust consists of heterogeneous traces. Using all traces as input for process mining may result in complex outcomes. Clustering the traces into similar clusters to analyze may prevent the variability of the traces to be represented in the outcome. Thereby resulting in simpler outcomes.

To cluster traces in the event log, the K-Means algorithm is chosen. The K-Means algorithm is chosen because it is fast, the number of clusters can be chosen and resulted in sufficiently occupied clusters. The algorithm aims to minimize the distance between data points in the dataset. To not overload the algorithm with features, features resulting from the principal component analysis are used.

For K, tests have been performed to see what value of K suits best. By determining K, a trade-off is made between the distribution of traces over the clusters and the total number of clusters. When clusters are assigned a very low number of traces, it is less beneficial to analyze these traces. Otherwise, when there is one cluster with a large number of traces, there is still too much variability causing complex models. Furthermore, a reasonable amount of clusters to analyze is desirable. Else, too many clusters need to be analyzed. After performing a test with several K's, K is set to 4. As 4 clusters are seen as a reasonable amount of clusters to analyze and all clusters had a reasonable amount of traces.

Other clustering algorithms DBSCAN, OPTICS and Agglomerative clustering showed less satisfactory results. The algorithms resulted in low-occupied clusters with one main cluster. Other trace clustering algorithms such as hybrid trace clustering are seen as too expensive processing for the event log because it requires a discovery step during the process (Zandkarimi et al., 2020).

6.2.6 Multi-Range Filtering

Multi-range filtering is used as the last pre-processing step before using the event log as input for process mining algorithms.

The first filtering step in the pre-processing uses attributes to exclude events and traces. Multi-range filtering uses frequency to exclude or incorporate traces and events. Many filtering techniques or tools use frequency as an ultimate measure. Where only the most occurring traces or events are kept as input for process mining. However, users therefore have little influence on what information stays in the model and what can be left out. Because everything below or above a certain threshold is used.

However, depending on the goal of process mining, the information contained in infrequent activities can be of interest. Multi-range filtering allows for simpler models to be constructed while incorporating activities and traces to the user's needs by setting multiple ranges as input. With multiple ranges, both the big-picture and potentially valuable infrequent behaviour can be used to discover and analyze a process model. Allowing, otherwise disregarded behaviour, also to be analyzed to gain insights in the process (Vidgof et al., 2020).

After applying the multi-range filtering to the sub-filtered event logs, process mining algorithms should be able to construct understandable process models. After analyzing the constructed process models, new ranges can be set to multi-range filtering to start an iterative analysis. Based on the results of previous ranges, new insights can be discovered.

6.2.7 Generalization

In the case of fine-granular events in the event log, generalization can be used to abstract events from the event log. Which combines possibly multiple events into one allowing for a simpler comprehensible output. It is chosen not to apply generalization as a processing step because possible relevant information can be discarded. In the search for improvement points in the Petri net, possible bottlenecks must be kept in the event log. Thereby, most grouping techniques apply semantic ontologies, are partly automated and lack domain significance (Bose et al., 2013).

6.2.8 Trace Attributes

The event log of SoliTrust does not contain other trace attributes than the case id. To perform analysis with a case perspective, trace attributes are composed. Attributes composed for traces are the views filled during the case, for what customer the case is performed, the ERP system and what type of file the input consists of. These trace attributes are compiled out of data existing in other data tables in the SQL server or are constructed from event attributes.

6.2.9 Sublogs

Not always, the entire business process is needed as input for process mining. It could also be interesting to zoom in on specific parts of the process. Therefore, the attributes are enhanced with an attribute event type. The event type is constructed from the script run so that only likewise activities are grouped in eight sub-logs. Taking a sub-log could still provide complex process mining outcomes. Therefore, the pre-processing method is also applied to sub-logs before being used as input for process mining. Nevertheless, taking a sub-log allows for a zoomed-in analysis instead of the whole event log. The table in Appendix 13.5 shows what sub-logs are available along with statistics on the number of activities and events.

6.3 Applying Pre-Process Method

The goal of the event log pre-processing method is to prepare the event log as input for process mining algorithms. Before applying the event log pre-processing method, process discovery resulted in large spaghetti-like models. Applying the pre-processing method allows for simpler models to be discovered. Showing the functioning of the pre-processing method. In addition, the pre-processing method is flexible in the setup. The user can select different sub-logs, clusters, filtering intervals, contamination and explained variance in the process. Allowing the user to select the required input for process mining.

Figure 10 shows a process model discovered without good application of the event log pre-processing method. In Figure 11, the event log is pre-processed to provide a simpler outcome. Which shows that the event log pre-processing method is effective for simpler models to be discovered.

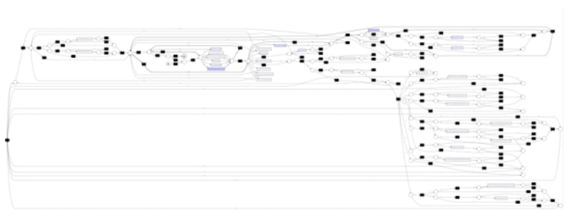


Figure 10: Complex Process Model Discovered

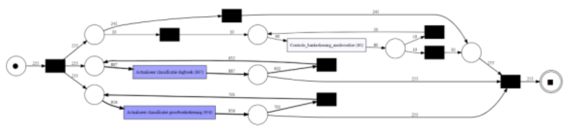


Figure 11: Simpler Process Model Discovered

7. Dashboard

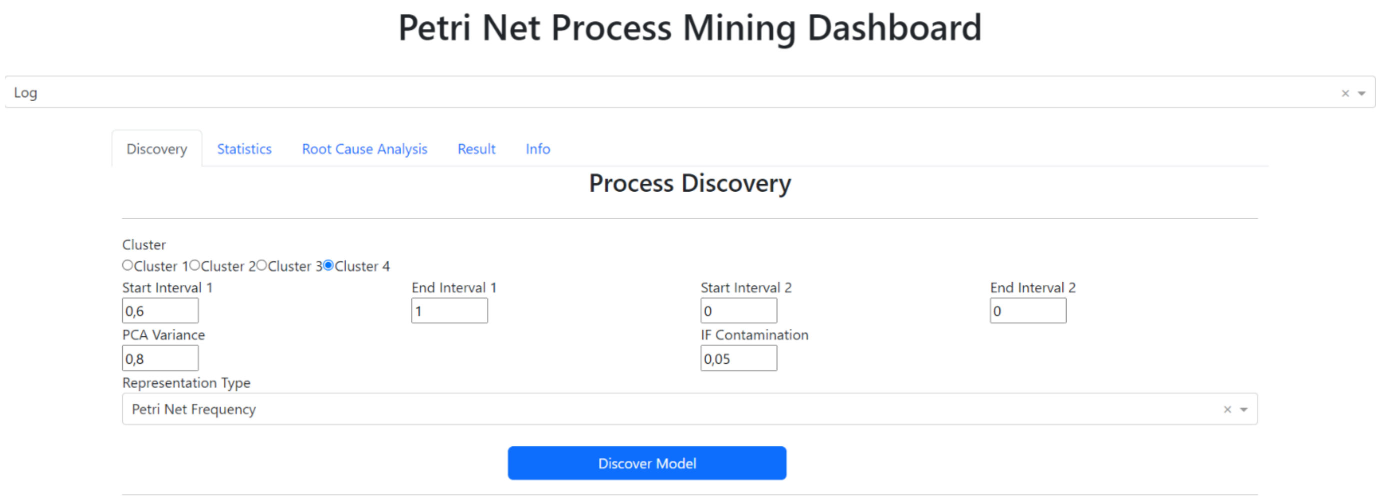
SoliTrust continuously wants to improve the Petri net. Therefore, a dashboard is created which incorporates process mining to provide information on the business process of SoliTrust. The advantage of the dashboard is that multiple process mining functionalities are combined in one tool. This section discusses the dashboard as well as the functionalities within the dashboard.

7.1 Goal Dashboard

The goal of the dashboard is to detect bottlenecks in the Petri net. Bottlenecks are seen as activities which cause rework or lengthen the duration of a workflow.

Rework is seen as starting an activity in the Petri net again. Thereby also triggering activities behind the started activity. The activity must be started by an human intervention. In an discovered process model, rework can be detected with a loop construct. Which means a transition to a previous activity. Because the process stops there and starts again in front of that activity.

The dashboard incorporates process discovery, statistics and root cause analysis to find bottlenecks in the Petri net. Process discovery is used to discover a process model. After which replay is


Petri Net Process Mining Dashboard

Log

Discovery | Statistics | Root Cause Analysis | Result | Info

Process Discovery

Cluster
○Cluster 1 ○Cluster 2 ○Cluster 3 [✓ ●Cluster 4]

Start Interval 1: 0,6
End Interval 1: 1
Start Interval 2: 0
End Interval 2: 0
PCA Variance: 0,8
IF Contamination: 0,05
Representation Type: Petri Net Frequency

Figure 12: Landing page dashboard

used to highlight activities with a higher frequency or duration. This is seen as a method which unveils bottlenecks in literature and therefore has been included in the dashboard.

Root cause analysis provides insights into why a deviation occurs. Giving a direction to SoliTrust on how to solve the bottleneck, a root cause analysis perspective is included into the dashboard.

7.2 Dashboard outline

The dashboard is created in Dash where callbacks are performed on process mining tasks. Both the process mining tasks as well as the dashboard are written in Python. The dashboard uses Dash as a package while the tasks for process mining utilize the Fraunhofer package for process mining in Python PM4PY. The dashboard consists of three parts; discovery, statistics and root cause analysis. Conformance checking is not part of the dashboard as it is not seen as relevant to the project. As the process can not deviate from the original Petri net (*Dash*, n.d.; *PM4PY*, n.d.).

As input the dashboard uses the event log of SoliTrust and the trace attributes. The event log is already enhanced in the SQL server of SoliTrust. Both the event log and trace attributes can be provided as a CSV file. Given event attributes the event log is divided into eight clusters which can also be selected for analysis.

The main advantage of the dashboard is that multiple event log pre-processing steps and process mining algorithms are combined in one tool. Therefore, no knowledge of other process mining tools is required to execute every step in the process. Furthermore, the results are presented in one go and no intermediate steps are required.

Figure 12 shows what the starting page of the dashboard looks like. The figure shows the title on top. Below the title, the entire log or a sub-log can be selected to be analyzed. In the tabs for discovery and root cause analysis, settings can be made for the algorithm. The tab statistics show data tables regarding the whole event log. The result tab shows the figure resulting from either the discovery or root cause algorithm. Both the statistics and root cause analysis tab are provided in Appendix 13.3.

7.3 Process Discovery

The first part of the dashboard is process discovery. Discovering a process model can provide insights into what the process looks like. In addition, the discovered process model could be used as input for other process mining algorithms.

Taking the entire event log as input for process discovery results in complex, hard-to-analyze process models. Therefore the event log is pre-processed as discussed in Chapter 6. As a discovery algorithm, the inductive miner (IMd) is used. This miner is chosen as the inductive miner can produce sound and valid process models. Furthermore, the inductive miner is scalable and allows for a large number of events to be handled. Thereby, the output of the inductive miner can be transformed into a Petri net which is the desired representation type. The log is replayed on the constructed Petri net so that frequencies or duration of activities can be added to the resulting figure. The duration and frequency of activities can be used to detect bottlenecks in the constructed process model. Also, constructs in the process model can indicate a potential bottleneck. When a loop construct occurs, it indicates that the activity starts again and causes rework (Ceravolo et al., 2018; Christian W. Günther & Wil M.P. van der Aalst, 2007; Denisov et al., 2018; Leemans et al., 2018; Sander J.J.

Leemans et al., 2013; Weijters & Ribeiro, 2011; Wen et al., 2009).

An output example of process discovery is provided in Figure 11.

7.4 Statistics

The second part of the dashboard contains statistics about the traces in the event log. There are two data tables on this tab, one shows the performance of cases with the number of activities, duration and a rework ratio. The second table shows what activities cause rework to start.

Originating the cause of rework is important for SoliTrust. As rework requires manual intervention. These interventions are costly for SoliTrust, so eliminating these interventions is beneficial. To construct the statistics for which activities enable rework, the context of the activity is used.

The event log contains all activities executed during a workflow in the Petri net. However, a workflow can execute the same activity in the Petri net automatically. As an example, when 100 CSV files need to be opened for a workflow. The activity open CSV file occurs 100 times in the event log. Applying replay and the normal rework statistic does not provide the correct causes of rework in the Petri net. As these algorithms cannot distinguish between activities occurring multiple times but are started by the Petri net and activities which are occurring multiple times but started by an intervention.

To only take into account the interventions, which are of interest for SoliTrust, the surrounding context of the activity is taken into account. First, the event log is filtered for activities which are reworked, which is done by event attribute filtering.

Thereafter, for each activity in a trace, it is checked whether the input of that activity is an output of another activity in the trace. If the event has an input which is an output of another activity, it can be assumed that the activity is triggered by the output of that activity. Thereby only taking starting points of rework into account.

In other words, it is checked whether an activity that has occurred multiple times in a case could be fired multiple times. If not, the activity must be the start of rework.

At last, a filter is applied to remaining activities or only takes into account activities which are of higher level in the Petri net. In addition, the number of times an activity starts rework is counted and the activities are sorted.

7.5 Root Cause Analysis

The third part of the dashboard is root cause analysis which results in a decision tree. The dependent and independent characteristics can be selected to be used in the analysis. Furthermore, the features for the decision tree can be selected. Incorporated dependent variables are rework ratio and case duration. Case characteristics can be used as features in the decision tree. To construct the decision tree, two classes are calculated; lower and higher. The class lower consist of all traces from the lowest value until the average. While higher contains all traces between the average and the maximum of the dependent variable.

The decision tree makes choices inferred from the independent characteristics of the trace. Thereby splitting the traces to group the classified traces. The Gini index is used to measure the impurity of a node. A higher Gini index indicates that there are traces present in the node which have another classification. While a low Gini index indicates that traces present in the node are mostly of the same classification. The lower the Gini index, the purer the traces in the node are.

To prevent the tree from over- or underfitting, the maximum depth, minimum samples in a leaf node and the number of leaf nodes can be adjusted. Thereby, the classes are balanced so that each class has even priority in a node.

With this setup, the user can perform use-case analysis with different independent as well as dependent characteristics. Allowing for different types of questions about the performance and how the performance originated to be answered.

Preferably, nodes want to be found with a high sample of traces and a low Gini index. Indicating that there is a lower chance of misclassifying a trace in the node and improving more samples results in more workflows being improved.

An example decision tree as output for a root cause analysis is shown in Figure 13. Here, the ERP system LN shows a higher duration for 13 samples.

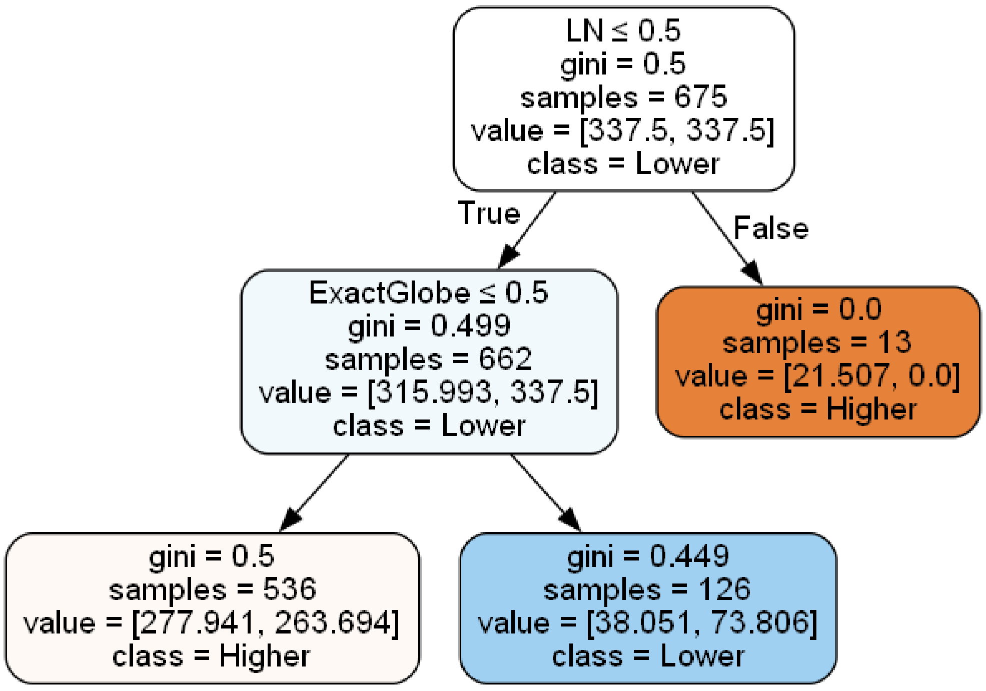


Figure 13: Example Output Root Cause Analysis

8. Validation and Continuous Implementation

This section discusses how the dashboard is validated and implemented. Thereby discussing research question 5.

8.1 Validation

After the first version of the dashboard was constructed, a presentation was given to 6 employees of SoliTrust. In this presentation, a demo of the dashboard was given as well as an explanation of why these process mining algorithms were used. This is one large iteration in the development of the dashboard.

During the discussion, these specific improvement points were given:

- More concrete, work down to a single activity as bottleneck. Giving a high-level overview of problems does not provide enough insights into where to improve the Petri net.
- Other perspectives such as file type, customer and ERP system are also relevant to discover bottlenecks in the process.
- Explanation of settings. For example, what does minimum samples in leaf node mean and how do we need to alter it to make an analysis.
- Instead of reworked activities, look for activities that start the rework. Do not take into account all activities that are reworked. Some activities are executed correctly but are started by the process itself or are executed after rework.

Thereafter, a different approach is used to identify the rework enablers as well as calculate the rework in a trace. Leading to clear and understandable outcomes for detecting single activities that start rework.

In addition, more attributes are extracted for traces so that other additional root cause analyses could be performed. Allowing for more use cases to be analyzed for SoliTrust.

The final dashboard is checked and explained to one employee of SoliTrust concerning the code and reasoning behind the dashboard.

Preliminary results from the dashboard were also checked with SoliTrust's employees as well as the company supervisor. After a number of iterations and evaluations, the dashboard is seen as valid.

8.2 Continuous Implementation

The dashboard supports the enactment and monitoring phase in the BPM lifecycle. Based on data on the execution of the business process, bottlenecks can be detected so that adaptations in the process can be made. In case the process changes drastically, a new design phase can be started where new requirements can be set. In the case of SoliTrust, the dashboard supports the triggering of the adjustment phase. Here, a smaller change in the process is made rather than changing the entire process.

In case a bottleneck is detected, SoliTrust can perform two things to solve the bottleneck. To start, SoliTrust can look into the activity itself and improve this activity. Meaning adjusting the code used by that activity to better perform or adjusting the layout of the Petri net. Thereby relocating the activity in the Petri net. Also, activities can be removed or added to the Petri net to resolve a bottleneck.

As the dashboard can analyze future event logs, SoliTrust can continuously improve the Petri net and close the BPM lifecycle.

9. Results

This section presents the outcomes of applying the incorporated process mining techniques in the constructed dashboard. The results of the research questions are discussed in section 11. The goal was to improve the business process performance. Hence, bottlenecks in the process are sought. Detecting the bottlenecks is the first step to improving the process, along with how these bottlenecks originate. Which is in line with the monitoring of activities and diagnosing problems in the BPM lifecycle. Whereafter, improvements can be made in the process with adjustments or (re-)design of the process.

To detect bottlenecks in the business process of SoliTrust, the flow-, case- and time perspectives are taken into account. SoliTrust is particularly interested in detecting activities which cause rework. Because a human intervention in the process is required, which costs SoliTrust valuable time which could be spent on other activities. In addition, reducing the throughput time of a workflow allows for a faster and increased throughput. Hence, bottlenecks in the business process are seen as activities which enable rework in the process or extend the duration of the process.

This part is split into three sections. First, process discovery is discussed. Thereafter, statistics are discussed using the method to detect rework enablers explained in section Statistics7.4. At last, root cause analysis results are presented. The results when applying the event log in the dashboard provide an answer to research question 6.

9.1 Process Discovery

Obtaining results

Constructed process models are analyzed visually by the user. Where there is searched for loop constructs or strange patterns in frequency or duration. For example in Figure 11, each place has a silent transition towards the place in front of the transition, meaning a loop construct. Which can indicate that the transition is a bottleneck or is executed automatically by the Petri net.

By changing the settings, a new model can be constructed. Depending on previous outcomes, the user can select new settings to zoom in or out into the model. Which is done by trial and error and depends on the goal of the user. For the graduation project, each (sub)log is analyzed with different settings. In total, around 40 discovered process models are analyzed.

Results

Constructed process models showed that conversions as well as call bronbestanden were incorporated and highlighted in the process models. Showing that these activities occur relatively often. Thereby, a loop construct was visible around these activities. Indicating that these activities start rework in the process.

Thereby, the activity Bankafschrift Matching can be seen as a bottleneck. This activity is the start of a loop construct in several process models. Indicating that the activity enables rework.

Furthermore, the activities

JSON: SoliTrust_Werkprogramma_Debiteuren, JSON: SoliTrust_Werkprogramma_Crediteuren and JSON: SoliTrust_Werkprogramma_Fraude

also show a large frequency and start of loop constructs. Showing that rework is caused and therefore these activities might also be relevant for improvement.

9.2 Statistics

Section 7.4 presents a methodology to unveil activities which cause rework in the process. Thereby only taking into consideration the activities where human intervention is necessary. It uses the input and output of other reworked activities to discover whether an activity was enabled and thereby enables rework. The top 5 activities which enable rework in the process are:

1. Call bronbestanden
2. Conversie tabel Verkoopfactuur
3. Conversie tabel Artikelmutatie
4. Conversie tabel Inkoopfactuur
5. Conversie tabel Algemeen validatie bron

9.3 Root Cause Analysis

For the root cause analysis, two dependent characteristics are chosen; rework ratio and duration. As independent characteristics the customer, file type, views and ERP system of the traces are used. This section first discusses the dependent characteristic duration for each of the independent characteristics. Thereafter, the rework ratio is also discussed as a dependent characteristic.

For each dependent and independent variable, the settings are set so that the decision tree underfits the data. Thereafter, the settings are changed so that the tree is allowed to grow. The analysis stops when the tree starts to overfit the data or no new changed in the tree are present.

9.3.1 Duration

Customer

There are traces performed for specific customers which tend to have an increased duration. These customers count up to around 3.5% of the total workflows performed each. Larger customers around 25% of the total workflows are also classified as higher. However, these customers do have a high Gini index indicating impurity. Traces are being classified as higher, as well as the node having a low Gini index. However, these are small customers who count up to 0.5% of the total workflows performed each.

File Type

CSV files often result in an increased duration of the workflow. Especially in combination with .bak, .xml and .xlsx files. Thereby, traces with DBC files also classify as traces with a higher duration.

Views

When the view Autorisatie is used during the workflow, the workflow is likely to have a higher duration. Thereby, traces containing the attribute view Salaris are classified as traces which have a higher duration.

ERP system

AFAS and AccountView are ERP systems which tend to cause a higher duration. Because of the high Gini index for the leaf nodes of these ERP systems, there is not a clear conclusion for these ERP systems. The combination of ExactGlobe and ProActive does show a lower Gini index, however only 24 traces are having both these ERP systems.

9.3.2 Rework Ratio

Customer

A few customers tend to increase the rework ratio. One larger customer, with up to around 25% of the total workflows. Though, the Gini index is still relatively high. There is a customer with a lower Gini index where traces are being classified higher for their rework ratio. However, SoliTrust performed around 4% of the total workflows for this customer.

File Type

Workflows utilizing a CSV file classify as workflows with a higher rework ratio. The combination of CSV and XML also results in traces with a higher rework ratio. In case there are CSV and XML files as well as a .bak file, the workflow behaves quite normally. However, when the trace lacks a .bak file, the workflow classifies with a higher rework ratio. The combination of CSV, XML, not .bak and a .xlsx file results in traces being classified with a higher rework ratio.

Views

The view Verkooplevering tends to cause rework in the workflow. Especially the view combination of Verkooplevering and Optie. Besides, traces with the view Salaris and the combination of Bankafschrift and Betaalopdracht result in the trace being classified with a higher rework ratio.

ERP system

AFAS is an ERP system which causes traces to be classified as one with a higher rework ratio. AFAS has a relatively high Gini index indicating impurity and also a significant number of traces having a lower rework ratio. Other, less occurring ERP systems such as ExactOnline and UNIT4 also cause traces to be classified as one with a higher rework ratio. These ERP systems have a lower Gini index. The 11 workflows with UNIT4 ERP system all classify as workflows with a higher rework ratio.

10. Contribution

This section discusses how the project contributes to both science and practice.

10.1 Contribution to Science

The results of this graduation project show that the dashboard can detect bottlenecks in the Petri net of SoliTrust with help of process mining algorithms. With help of process discovery techniques as well as replay and root cause analysis are used as generalizable methods. These methods can be applied to any process as long as the required information is stored in the event log.

The method for detecting activities which start rework however is specifically designed for Petri nets and cannot be applied to every process model. As the method uses attributes for input and output of a transition which are not always available in an event log. Thereby is the method tailored to the process of SoliTrust, to be in line with the hierarchy and events of the SoliTrust process.

Furthermore, the event log pre-processing method shows that these steps can be used to pre-process a voluminous, fine-granular and heterogenous event log. Where before, complex process models were discovered and after applying the method simple understandable models were discovered. In addition, the method is flexible in setup. Therefore, the method can be applied to other event logs with the same characteristics.

10.2 Contribution to Practice

With the dashboard, SoliTrust is able to find bottlenecks in the Petri net. Solving these bottlenecks can improve the performance of the Petri net in terms of decreased rework and duration.

The dashboard is able to find specific activities and trace characteristics of the bottleneck. Informing SoliTrust where the bottleneck is located and a possible cause of the bottleneck.

Which has as benefit that SoliTrust requires fewer human interventions and can decrease the throughput time of a workflow. Therefore, employees can focus on different activities and the total throughput of the system can be increased.

For the duration, the project contributed in recognizing the following characteristics as troublesome;

- 4 file types
- 6 customers
- 4 ERP systems
- 7 views

Regarding the rework, the following number of trace characteristics can be identified as troublesome;

- 3 file types
- 2 customers
- 3 ERP systems
- 6 views

Furthermore, 15 conversions are identified as rework enablers with more than 200 times enabling rework. Alongside call brondbestanden with 341 times enabling rework, it is the activity which causes the most rework.

As the dashboard can analyze future workflows, SoliTrust can continuously improve Petri net performance.

11. Conclusion

The objective of the graduation project is to continuously improve the Petri net to improve workflow handling. The main research question is therefore formulated as follows:

How can SoliTrust continuously improve the Petri net performance utilizing the event log as input for process mining algorithms to detect bottlenecks?

First, a literature study has been performed to provide an answer to research questions 1 & 2. To provide an understanding of what a Petri net and event log are as well as what process mining entails. A Petri net is a static structure which can be used to describe process models. Through the structure, a token flows from a place to a transition to execute activities. In the case of SoliTrust, the transitions of the Petri net handling workflows are logged in an event log. Which consists of traces, events and attributes. This event log can be used as input for process mining to discover, monitor and improve processes.

Since the event log is voluminous, heterogenous and fine-granular, the event log cannot directly be used as input for process mining. Before the event log can be used as input, the event log is pre-processed based on findings during research question 3. For event log pre-processing, a methodology is designed where the event log is filtered based on attributes and anomalies. Furthermore, clusters with similar traces or events are created. Resulting in an input which could be used as input for process mining.

To discover bottlenecks, process discovery with replay, root cause analysis as well as a custom detection algorithm for activities that enable rework are used. Applying these process mining techniques showed specific activities as well as characteristics of traces which cause an increased duration or rework. Which answers research questions 4 & 6.

To combine the event log pre-processing and process mining techniques in one tool, a dashboard is developed. This dashboard allows for future event logs to be analysed. Hence, a continuous implementation. Furthermore, based on expert feedback on the process, the dashboard is evaluated, improved and validated. Which provides an answer to research question 5.

To conclude the main research question, SoliTrust can continuously improve the Petri net performance by applying the process mining techniques in the developed dashboard to detect activities and process characteristics which cause an increased duration or rework ratio. Thereby monitoring the process and closing the BPM lifecycle.

12. Limitations and recommendations

One performance-related aspect of the dashboard is the duration of the workflow. Which is chosen because decreasing the duration of workflows allows for more throughput. However, parts of the activities in the process cannot be optimized in time aspect. As an example, opening a CSV file takes time. When the file is larger for a workflow, it processes longer in the workflow. However, this is not something that can be optimized but which is limited to the process. Therefore, the duration aspect might sometimes not provide meaningful insights.

One recommendation for SoliTrust is to provide more structure to the process in the event log. Activities are now given a type based on the script that is run. However, especially in process discovery, it might be that intermediate activities are filtered out. Giving a better structure or hierarchy might improve the way the event log can be analysed and deal with the fine granularity in the event log.

Furthermore, one recommendation is to store more trace attributes within the event log. These are now constructed from other databases. Assigning attributes to traces in the event log table creates an easier input for the dashboard. Especially trace attributes with regards to whether a trace is successful or not could be interesting for SoliTrust. As the root cause analysis determines the successfulness of a trace with respect to the other traces in the log. However, using this type of classification for traces could misclassify traces. Such as traces with large data, these take longer to process because of computational limitations. But do not need to imply that the trace was executed with faults or bottlenecks. Though, the trace could be classified as one with a higher duration.

Another limitation is that the dashboard only takes into account post-mortem data. Or in other words, data about workflows that already have been performed are used as input for the dashboard. Meaning that the analysis made is based on data from the past. Meaning that the current workflows executed in real-time are not supported by process mining to improve the business process performance. A recommendation to SoliTrust could be to implement process mining in an online setting. So that current workflows are supported by process mining outcomes. Especially remaining processing time and error predictions could be interesting for SoliTrust. As the path through the Petri net is already fixed from the start, the flow perspective is less relevant.

12. References

Batista, E., Solanas, A., Ieee, & Ieee, U. P. U. P. U. P. (2019). Skip Miner: Towards the Simplification of Spaghetti-like Business Process Models. *10th International Conference on Information, Intelligence, Systems and Applications (IISA)*, 40–45. <Go to ISI>://WOS:000589872200007

Bose, R. P. J. C., Mans, R. S., & Van Der Aalst, W. M. P. (2013). Wanna improve process mining results? *Proceedings of the 2013 IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2013 - 2013 IEEE Symposium Series on Computational Intelligence, SSCI 2013*, 127–134. https://doi.org/10.1109/CIDM.2013.6597227

Ceravolo, P., Guetl, C., & Rinderle-Ma, S. (Eds.). (2018). *Data-Driven Process Discovery and Analysis* (Vol. 307). Springer International Publishing. https://doi.org/10.1007/978-3-319-74161-1

Christian W. Günther, & Wil M.P. van der Aalst. (2007). *Fuzzy Mining: Adaptive Process Simplification Based on Multi-perspective Metrics*.

Dash. (n.d.). 2023.

De Leoni, M., Van Der Aalst, W. M. P., & Dees, M. (2016). A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. *Information Systems*, *56*, 235–257. https://doi.org/10.1016/j.is.2015.07.003

Denisov, V., Belkina, E., Fahland, D., & Van Der Aalst, W. M. P. (2018). *The Performance Spectrum Miner: Visual Analytics for Fine-Grained Performance Analysis of Processes*. https://github.com/processmining-in-logistics/psm

Fani Sani, M. (2020). *Preprocessing Event Data in Process Mining*.

Garcia, C. D. S., Meincheim, A., Faria Junior, E. R., Dallagassa, M. R., Sato, D. M. V, Carvalho, D. R., Santos, E. A. P., & Scalabrin, E. E. (2019). Process mining techniques and applications – A systematic mapping study. *Expert Systems with Applications*, *133*, 260–295. https://doi.org/10.1016/j.eswa.2019.05.003

Gyunam Park and Wil M.P. van der Aalst. (2021). *A General Framework for Action-Oriented Process Mining*.

I. Mukhlash, W. N. R. D. A. and R. S. (2018). Business process improvement of production systems using coloured petri nets. *Bulletin of Electrical Engineering and Informatics*.

Kaouni, A., Theodoropoulou, G., Bousdekis, A., Voulodimos, A., & Miaoulis, G. (2021). Visual analytics in process mining for supporting business process improvement. In *Frontiers in Artificial Intelligence and Applications* (Vol. 338, pp. V–VI). https://doi.org/10.3233/FAIA210089

Kintz, M. (2012). *A Semantic Dashboard Description Language for a Process-oriented Dashboard Design Methodology*.

Kitchenham, B. (2004). *Procedures for Performing Systematic Reviews*.

Leemans, S. J. J., Fahland, D., & Van Der Aalst, W. M. P. (2015). Scalable process discovery with guarantees. In *Lecture Notes in Business Information Processing* (Vol. 214, pp. 85–101). https://doi.org/10.1007/978-3-319-19237-6_6

Leemans, S. J. J., Fahland, D., & van der Aalst, W. M. P. (2018). Scalable process discovery and conformance checking. *Software and Systems Modeling*, *17*(2), 599–631. https://doi.org/10.1007/s10270-016-0545-x

M. Leemans, W. M. P. V. D. A. and M. G. J. V. D. B. (2018). Hierarchical performance analysis for process mining. *ACM International Conference Proceeding Series*.

Manoj Kumar, M. V, Thomas, L., & Annappa, B. (2018). Simplifying spaghetti processes to find the frequent execution paths. In *Smart Innovation, Systems and Technologies* (Vol. 79, pp. 693–701). https://doi.org/10.1007/978-981-10-5828-8_66

Marin-Castro, H. M., & Tello-Leal, E. (2021). Event log preprocessing for process mining: A review. In *Applied Sciences (Switzerland)* (Vol. 11, Issue 22). MDPI. https://doi.org/10.3390/app112210556

Milani, F., Lashkeyich, K., Maggi, F. M., & Di Francescomarino, C. (2022). Process Mining: A Guide for Practitioners. *16th International Conference on Research Challenges in*

Information Sciences (RCIS), *446*, 265–282. https://doi.org/10.1007/978-3-031-05760-1_16

Patrick Konniger, & Dirk Niestadt. (2021, March 30). *Process mining, een introductie*.

Peterson, J. L. (1977). *Petri Nets**.

PM4PY. (n.d.). 2023.

Salimifard, K., & Wright, M. (2001). Petri net-based modelling of workflow systems: An overview. *European Journal of Operational Research*, *134*(3), 664–676. https://doi.org/10.1016/S0377-2217(00)00292-7

Sander J.J. Leemans, Dirk Fahland, & Wil M.P. van der Aalst. (2013). *Discovering Block-Structured Process Models from Event Logs - A Constructive Approach*.

Van der Aalst, W. (2016). Process mining: Data science in action. In *Process Mining: Data Science in Action*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-49851-4

van der Aalst, W. M. P. (2010). Process Discovery: Capturing the Invisible. *Ieee Computational Intelligence Magazine*, *5*(1), 28–41. https://doi.org/10.1109/mci.2009.935307

van der Aalst, W. M. P. (2011). Process Mining. In *Process Mining*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-19345-3

Van Eck, M. L., Lu, X., Leemans, S. J. J., & Van Der Aalst, W. M. P. (2015). *PM 2 : a Process Mining Project Methodology*.

Vidgof, M., Djurica, D., Bala, S., & Mendling, J. (2020). Cherry-picking from spaghetti: Multi-range filtering of event logs. *Lecture Notes in Business Information Processing*, *387 LNBIP*, 135–149. https://doi.org/10.1007/978-3-030-49418-6_9

W. Premchaiswadi and P. Porouhan. (2015). *Process modeling and bottleneck mining in online peer-review systems*.

Weijters, A. J. M. M., & Ribeiro, J. T. S. (2011). Flexible heuristics miner (FHM). *IEEE SSCI 2011: Symposium Series on Computational Intelligence - CIDM 2011: 2011 IEEE Symposium on Computational Intelligence and Data Mining*, 310–317. https://doi.org/10.1109/CIDM.2011.5949453

Wen, L., Van Dongen, B. F., Alves De Medeiros, A. K., & Wen, L. (2009). *Process Mining: Overview and Outlook of Petri Net Discovery Algorithms*. www.processmining.org

Wirth, N. (1971). *Program Development by Stepwise Refinement*.

Y. Caesarita, R. S. and K. R. S. (2018). *Identifying bottlenecks and fraud of business process using alpha ++ and heuristic miner algorithms*.

Zandkarimi, F., Rehse, J.-R., Soudmand, P., & Hoehle, H. (2020). *A Generic Framework for Trace Clustering in Process Mining*. https://doi.org/10.6084/m9.figshare.12607742.v2

13. Appendix

13.1 Literature review

Table 1: Result Systematic Literature Review

RQ1: What is a Petri net and how to enhance the performance				
Search strings:	***Scope***	***Date***	***Date range***	***#Articles***
Scopus				
“Petri net*” AND “performance” AND “bottleneck”	TITLE-ABS-KEY	9-1-2023	Until 20-12-2022	#157
Web of Science				
“Petri net*” AND “performance” AND “bottleneck”	TITLE-ABS-KEY	9-1-2023	Until 20-12-2022	#50
Total:				207
Removing Duplicates:				(-57) 150
Removing based on exclusion criteria:				(-129) 21
Removed after reading abstract:				(-10) 11
Removed after more extensive reading:				(-6) 5
Added by recommendation:				1
Total:				6
RQ2: What is process mining and what models/techniques are available for (continuous) improvement?				
Scopus				
“process mining” AND “discovery” AND “petri net*”	TITLE-ABS-KEY	9-1-2023	Until 9-1-2023	#168
“process mining” AND “performance analysis”	TITLE-ABS-KEY	9-1-2023	Until 9-1-2023	#147
“process mining” AND “discovery” AND “algorithm*”	TITLE-ABS-KEY	9-1-2023	Until 9-1-2023	#523

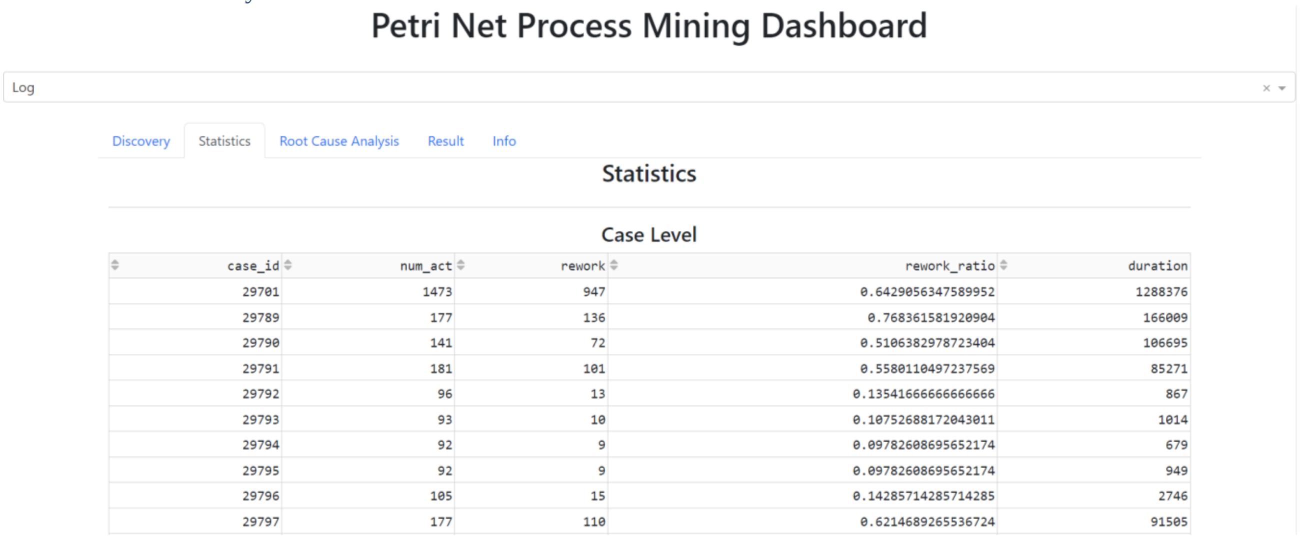
“process mining” AND “contin*”	TITLE-ABS-KEY	9-1-2023	Until 9-1-2023	#294
Web of Science				
“process mining” AND “discovery” AND “petri net*”	TITLE-ABS-KEY	9-1-2023	Until 9-1-2023	#134
“process mining” AND “performance analysis”	TITLE-ABS-KEY	9-1-2023	Until 9-1-2023	#87
“process mining” AND “discovery” AND “algorithm*”	TITLE-ABS-KEY	9-1-2023	Until 9-1-2023	#361
“process mining” AND “contin*”	TITLE-ABS-KEY	9-1-2023	Until 9-1-2023	#169
Total:				1883
Removing Duplicates:				(-1218) 665
Removing based on exclusion criteria:				(-585) 80
Removed after reading abstract:				(-56) 24
Removed after more extensive reading:				(-16) 8
Added by recommendation:				1
Total:				9

13.2 Attributes Event Log

Table 2: Event Attributes Event Log

Attributes in Event log		
Column	***Description***	***Format***
action_id	Id for each event in the event log	Integer
task_id	Id for each workflow in the event log	Integer
script	Script called in the transition	String
command	Command executed by the script	String
in_format	Input place for transition	String
out_format	Output place for transition	String
in_tokens	Tokens used as input	String
trans_id	Id for each transition	String
name	Name for each transition	String
parameters	Variables used in transition	String
started	Starting time of transition	Timestamp
finished	Ending time of transition	Timestamp
success	Return whether transition was successful	0, 1
returncode	Return code when error occurs	-1,0,1,404
std_out	Standard output	String
st_derr	Output what the error is	String
skipped	Value whether transition is skipped	0, 1
out_tokens	Tokens produced by transition	String
params	Variables produced by transition	String
deleted	Value if trace is deleted	0, 1
core	Core of processor used	Integer

13.3 Dashboard layout

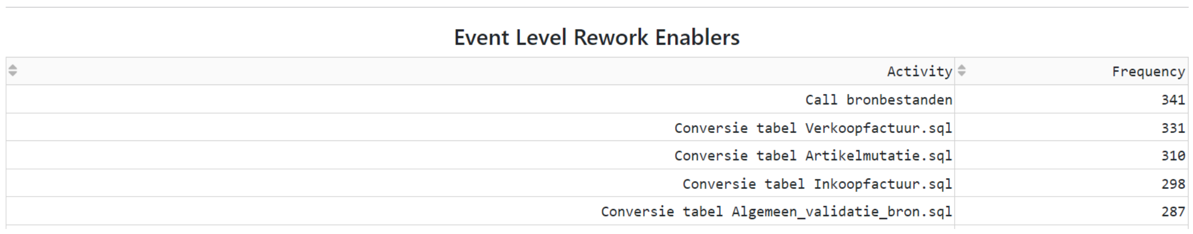


Statistics

Case Level

	case_id	num_act	rework	rework_ratio	duration
	29701	1473	947	0.6429056347589952	1288376
	29789	177	136	0.768361581920904	166009
	29790	141	72	0.5106382978723404	106695
	29791	181	101	0.5580110497237569	85271
	29792	96	13	0.13541666666666666	867
	29793	93	10	0.10752688172043011	1014
	29794	92	9	0.09782608695652174	679
	29795	92	9	0.09782608695652174	949
	29796	105	15	0.14285714285714285	2746
	29797	177	110	0.6214689265536724	91505

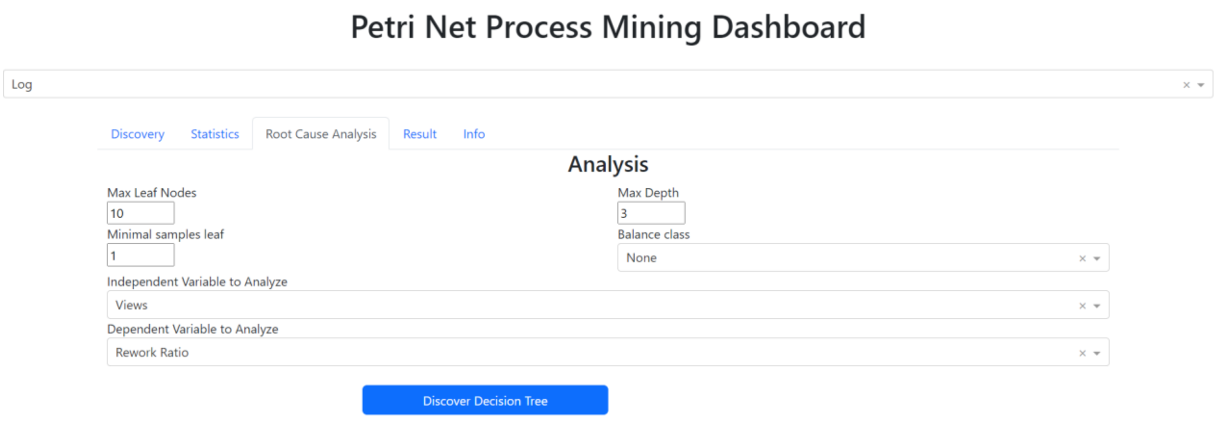
Figure 14: Statistics Tab Trace Level



Event Level Rework Enablers

	Activity	Frequency
	Call bronbestanden	341
	Conversie tabel Verkoopfactuur.sql	331
	Conversie tabel Artikelmutatie.sql	310
	Conversie tabel Inkoopfactuur.sql	298
	Conversie tabel Algemeen_validatie_bron.sql	287

Figure 15: Statistics Tab overview Event Level



Max Leaf Nodes: 10
Max Depth: 3
Minimal samples leaf: 1
Balance class: None
Independent Variable to Analyze: Views
Dependent Variable to Analyze: Rework Ratio

Figure 16: Root Cause Analysis Tab Layout

13.4 Event Log Filtering

Table 3: Filtering Steps Event Log

Filtering Activities			
Filter	***Reason***	***#Activities***	***Percentage***
Book Year	Only take workflows for book year 2022	1,226,714	100%
Script	Only want to take into account activities which perform certain tasks.	886,944	72,30%
Duration	Filter out activities which take zero time, these do not perform any tasks in the workflow. In addition, more improvement is likely with events that take a larger time. However, activities that take no time but have an error are kept in the event log.	394,079	32,12%
Tests	Filter out cases with are run for tests in the network.	390,940	31,87%
Terminated	Cases which are terminated are excluded for the event log.	386,206	31.48%

13.5 Sublogs Type

Table 4: Event Type Sub-Logs

Sublogs					
Additional	***Type***	***Name***	***Distinct Count Activities***	***Distinct Count Script***	***# Events***
orders	00	Crm	20	11	27,660
{l	01	Copy db	94	75	139,357
D:	02	Conversie	61	57	34,625
	03	Samenvoegen Tabellen	6	5	283
	04	Classificatie	22	10	12,950
	05	Validatie	1	1	1,585
	06	Report	36	2	7,915
	07	Export	2345	857	160,526
	General	General	1	1	1,305