University of New York Tirana

Department of Computer Science

Process Optimization using Process Mining

A study case for Albania

Author:
Ana KALEMI

Supervisor:
Dr. Marenglen BIBA

A thesis submitted for the degree of
BSc Computer Science

2023

Abstract

Process mining is a powerful data analysis technique that uses event logs to visualize, analyze, and improve business processes. The primary objective of this research is to emphasize the significance of implementing this approach to optimize business processes in Albania. The advantages were observed throughout the analysis of the illustrative study case with the event log synthesized from a loan application system of a financial institution. The event log is the foundation of process mining as it contains the necessary data from where inefficiencies, bottlenecks, and deviations can be identified. Moreover, the observations can lead to recommendations and enhancements that can improve the overall process performance. The results contribute to the growing body of literature on process mining and offer practical implications for organizations seeking to adopt process mining in their operations.

Keywords: *process mining, optimization, data visualization, process redesign, business process re-engineering, operational efficiency.*

Contents

List of Figures

List of Tables

1 Introduction

Process optimization is a critical component of business operations, as it aims to identify and eliminate inefficiencies in processes, leading to increased productivity, reduced costs, and improved quality. Process mining, which involves analyzing event logs to extract information about processes, has emerged as a powerful technique for process optimization. By leveraging process mining, organizations can gain insights into their processes.

The thesis aims to explore the potential of using advanced process mining techniques as a tool for gaining novel insights from event data and for identifying performance or compliance issues. Process mining is an emerging field with significant potential for adoption and as such, a case is made for its application in the context of the Albanian market by considering the possible benefits as well as the challenges associated with it.

After a thorough introduction of the necessary concepts in the Background section (2), its potential is further explored in the Methodology section (3). An innovative way in enabling organizations in Albania to provide faster and more efficient services is suggested, knowing that there is limited research as of now towards this topic. The study aims to investigate the application of process mining in the context of an online loan application system, using a case study of a financial institution. Through this analysis, the goal is to provide practical insights into how process mining can be used to optimize the loan application process for improved performance. The results are then observed and discussed further on in the respective sections (4, 5), along with the possible areas for future research (6).

2 Background

Process mining is a powerful technique used to analyze and improve business processes. According to the paper "Process Mining: A 360 Degree Overview" by Wil M. P. van der Aalst [1], process mining aims to extract knowledge from event logs. Event logs are collections of data entries that record activities performed by users or systems. By analyzing these logs, process mining can reveal hidden patterns and inefficiencies in business processes, providing valuable insights for process improvement. For a formal definition of the Event Log refer to the Definition 1.

Definition 1 (Event Log)**.** *An event log is a tuple $L = (E, \#, \prec)$ consisting of a set of events $E \subseteq \mathcal{U}_{ev}$, a mapping $\# \in E \to \mathcal{U}_{map}$, and a strict partial ordering $\prec \subseteq E \times E$ on events. For any $e \in E$ and $att \in dom(\#(e))$: $\#_{att}(e) = \#(e)(att)$ is the value of attribute att for event e. For example, $\#_{act}(e)$, $\#_{case}(e)$, and $\#_{time}(e)$ are the activity, case, and timestamp of an event e. The ordering of events respects time, i.e., if $e_1, e_2 \in E$, $\#_{time}(e_1) \neq \perp$, $\#_{time}(e_2) \neq \perp$, and $\#_{time}(e_1) < \#_{time}(e_2)$, then $(e_2) \not\prec (e_1)$ [1].*

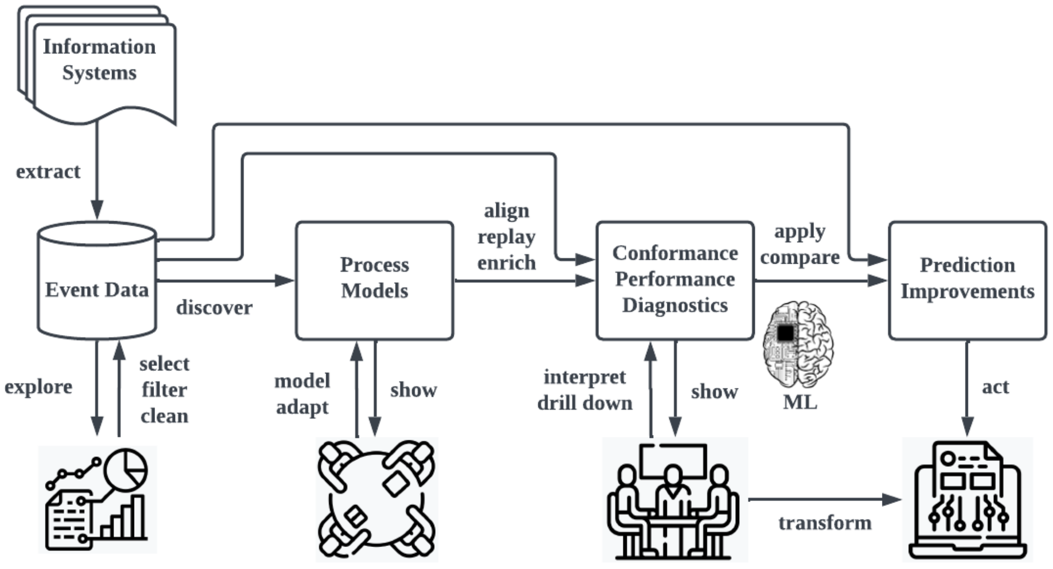


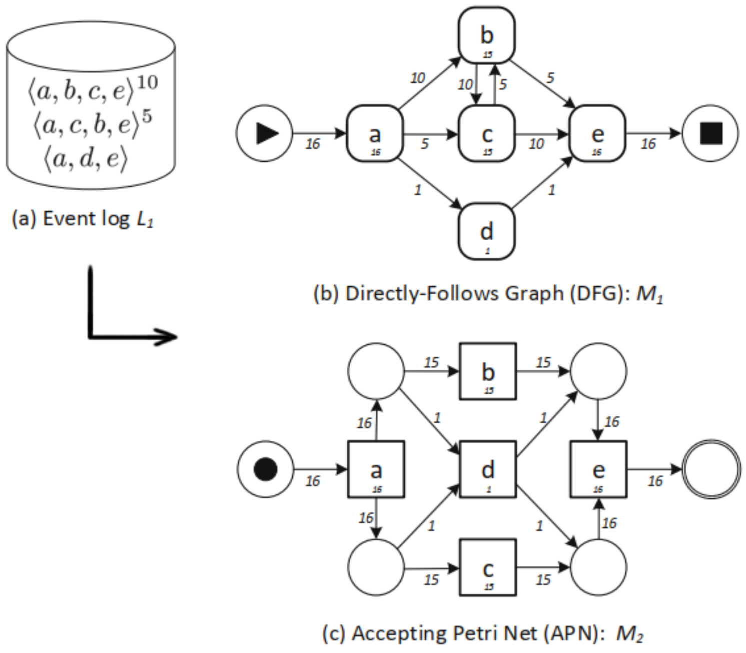
Figure 1: Process Mining Overview

Figure 1 provides a high-level overview of the process mining methodology, which involves three primary techniques: process discovery, conformance checking, and process enhancement. To conduct process mining, event data must first be extracted from the systems that support and store the necessary information enabling processes to be analyzed. This data is often stored in multiple database tables and requires conversion into a format suitable for process mining, with events commonly including a case identifier, activity name, timestamp, and optional attributes like resource, location, and cost.

Once the event data has been extracted and prepared, it can be explored, filtered, and cleaned, with data visualization tools like dotted charts and sequence diagrams used to gain insights. Process discovery techniques are then employed to automatically generate process models. Commercial tools typically rely on the Directly-Follows Graph (DFG). Other high-level models are available, which can better express concurrency than DFGs, such as Business Process Model and Notation (BPMN), Petri nets, and Unified Modeling Language (UML) activity diagrams.

The subject of process discovery will first be explored in abstract terms, independent of the selected process modeling notation. Therefore, the behavior of a model will be described as a set of traces. The formal definition is provided below.

Definition 2 (Process Model)**.** *$\mathcal{U}_M$ is the universe of process models. A process model $M \in \mathcal{U}_M$ defines a set of traces $lang(M) \subseteq \mathcal{U}_{act}{}^*$ [2].*



$\langle a,b,c,e\rangle^{10}$
$\langle a,c,b,e\rangle^{5}$
$\langle a,d,e\rangle$
(a) Event log L_1
(b) Directly-Follows Graph (DFG): M_1
(c) Accepting Petri Net (APN): M_2

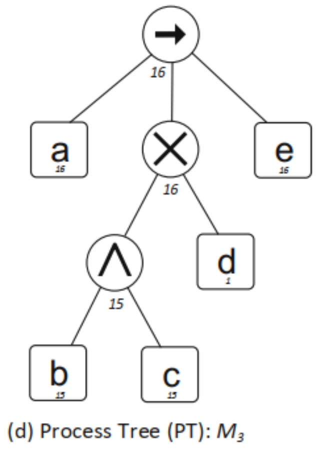


Figure 2: Three process models learned from event log $L_1 = [\langle a,b,c,e\rangle^{10}, \langle a,c,b,e\rangle^{5}, \langle a,d,e\rangle]$.

Consider the process models M_1 (DFG), M_2 (Petri net), and M_3 (process tree) in Figure 2 $lang(M_1) = \langle a,b,e\rangle, \langle a,c,e\rangle, \langle a,d,e\rangle, \ldots, \langle a,b,c,b,c,b,c,e\rangle, \ldots$ contains infinitely traces due to the cycle involving b and c. A process discovery algorithm is designed to produce a process model that describes the observed behavior.

Definition 3 (Process Discovery Algorithm)**.** *A process discovery algorithm is a function $disc \in \mathcal{B}(\mathcal{U}_{act}*) \rightarrow \mathcal{U}_M$, i.e., based on a multiset of traces, a model is produced [2].*

Various techniques, such as the Alpha algorithm, region-based approaches, inductive mining techniques, and the split miner, can be used to discover process models. The aim of these models is to describe all the observed behavior or just the dominant behavior. It is important to note that the event log only contains example behavior which may be incomplete and contain infrequent behavior.

Hence why, Wil M. P. van der Aalst [1] emphasises that the combination of a process model and event data is used to perform conformance checking and performance analysis, where events in the log are compared to activities in the process model to identify commonalities and discrepancies. Performance diagnostics, such as service times and waiting times, can be overlaid onto the process model since events have timestamps. Once the control-flow is discovered, the process model can be transformed into a stochastic model that includes probabilities and delay distributions. After applying conformance checking and performance analysis techniques, performance and compliance issues can be identified, and root-cause analysis can be performed.

The right-hand side of the Figure 1 illustrates how process mining can be used to transform and improve the process as well as automatically address observed and predicted problems. By combining event data and process models, it is possible to generate Machine Learning (ML) problems, which can be used to predict outcomes. This combination can then trigger corrective actions or even complete workflows to address identified problems. This way, event data can be transformed into actions that actively address performance and compliance issues.

As process mining is a rapidly evolving field, there are several directions for the future. Wil M. P. van der Aalst [1] discusses that it can be integrated with AI techniques, such as machine learning and natural language processing, to improve the accuracy and efficiency of process analysis. Process Mining can also be used for real-time process monitoring and alerting, which can help organizations to detect and address issues in a timely manner.

2.1 Tools

There are numerous open-source and commercial process mining tools available such as ProM, bupaR, PM4Py, RapidProM, Celonis, Disco/Fluxicon, Lana/Appian, Minit, Apromore, myInvenio/IBM, PAFnow, Signavio/SAP, Timeline/Abby, and ProcessGold/UiPath. These tools have been used to analyze processes with billions of events and have established themselves as reliable discovery methods. Throughout this paper ProM 6.12 is used as it is a stable release version specifically designed for researchers. Additionally the Celonis Execution Management System (EMS) platform is used to apply real-time business intelligence to the system's data.

ProM is an extensible framework that supports a wide variety of process mining techniques in the form of plug-ins. Implemented in Java, it is platform independent, and can be downloaded from ProM's official website. ProM offers over 1500 plugins, as a result, it is heavily utilized in this study due to its comprehensive implementation of various miners, customizable through many parameters, as well as its array of advanced conformance checking techniques. To facilitate ease of replicability of the study and its results, all plugins used in the subsequent sections are explicitly identified along with their corresponding parameter values. If there are unmentioned parameters, the default value of the parameter given by ProM is used.

Celonis, utilized in the paper to uncover insights, identify inefficiencies, and optimize processes, is a leading process mining tool that empowers organizations to gain deep insights into their operational processes and drive data-based improvements. The Academic Alliance provides free software access in the Celonis' official website, enabling researchers with advanced technology for comprehensive analysis of complex business processes, to conduct cutting-edge research.

2.2 Petri Nets

Petri nets are extensively studied as process modeling language that enable the modeling of concurrency. Having a solid understanding of Petri nets is crucial for comprehending the paper, as they are frequently employed throughout the study and are referenced in other definitions as well. According to Jensen [3], Petri nets consist of a set of places, transitions, and arcs, which are used to describe the flow of tokens between the places and transitions. While the network structure remains static, tokens can flow through the network based on the firing rule (Figure 4).

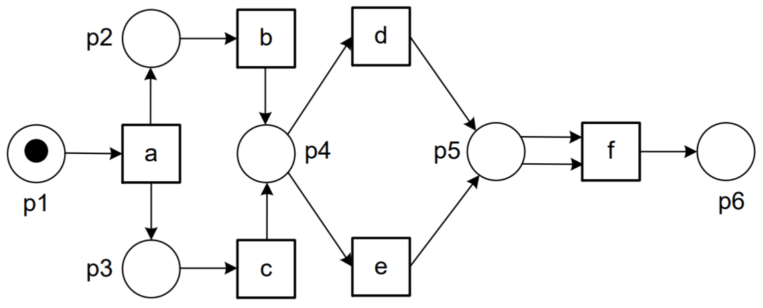


Figure 3: Example of a Petri net

The state of a Petri net is determined by the distribution of tokens over places, which is referred to as its marking. Initially, there is only one token in the marking, and the "start" is the only marked place. The dynamic behavior of a marked Petri net is defined by the firing rule. If all of the input places of a transition have tokens, then the transition is considered enabled. An enabled transition can fire, resulting in the consumption of one token from each input place and the production of one token for each output place.

Definition 4 (Petri net)**.** *A Petri net is a triplet $N = (P, T, F)$ where P is a finite set of places, T is a finite set of transitions such that $P \cap T = \emptyset$, and $F \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs, called the flow relation. A marked Petri net is a pair (N, M) where $N = (P, T, F)$ is a Petri net and where $M \in \mathbb{B}(P)$ is a multi-set over P denoting the marking of the net. The set of all marked Petri nets is denoted $\mathcal{N}$ [4].*

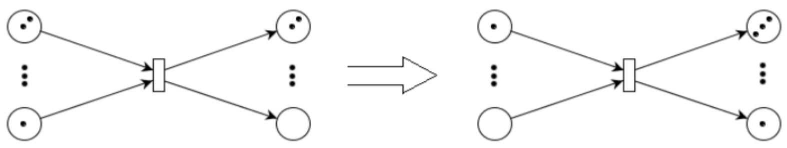
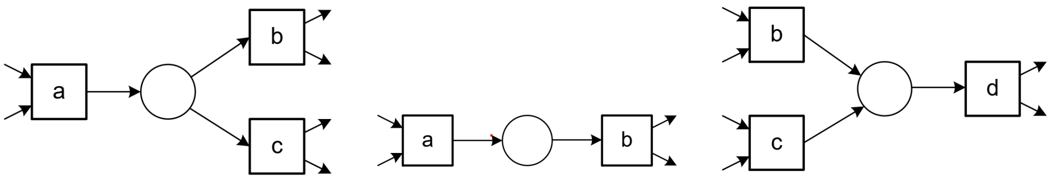


Figure 4: Firing of a Transition

Definition 5 (Firing rule)**.** *Let (N, M) be a marked Petri net with $N = (P, T, F)$ and $M \in \mathbb{B}(P)$. Transition $t \in T$ is enabled, denoted $(N, M)[t\rangle$, if and only if $\bullet t \leq M$. The firing rule $\_[\_\rangle\_ \subseteq \mathcal{N} \times T \times \mathcal{N}$ is the smallest relation satisfying for any $(N, M) \in \mathcal{N}$ and any $t \in T$, $(N, M)[t\rangle \Rightarrow (N, M)[t\rangle(N, (M \setminus \bullet t) \uplus t\bullet)$ [4].*


(a) XOR-split pattern: a → b, a → c, and b#c

(b) Sequence pattern: a → b

(c) XOR-join pattern: b → d, c → d, and b#c

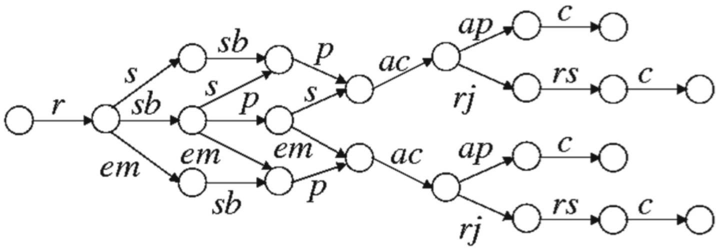
Figure 5: Petri Net patterns.

XOR joins and XOR splits are types of transitions used in Petri nets to model decision points in the system. XOR joins illustrated in Figure 5(c), are transitions that combine multiple incoming arcs into a single outgoing arc. Murata [5] explains that an XOR join is depicted by a diamond shape with incoming arrows representing the paths to be merged and an outgoing arrow representing the merged path. When an XOR join is executed, it enables the system to proceed to the next transition only if one and only one of the incoming arcs has a token (i.e., the state of the system allows it to proceed via that specific path). Figure 5(a) presents an XOR split. These type of transitions split a single incoming arc into multiple outgoing arcs. Murata [5] explains that, an XOR split represents a point in the process where the flow can take one of several paths, but only one of them can be executed. An example of a sequence can be seen in Figure 5(b). Sequences in a Petri net are represented by a series of transitions connected by places. The tokens flow through these places and transitions in a specific order, creating a sequence of events in the system. As Aalst, Wil [6] explains, sequence patterns are represented by a series of connected transitions, with each transition representing an activity that must be completed before the next one can start. The Petri net structure ensures that the sequence is executed correctly and that any conflicts or deadlocks are avoided.

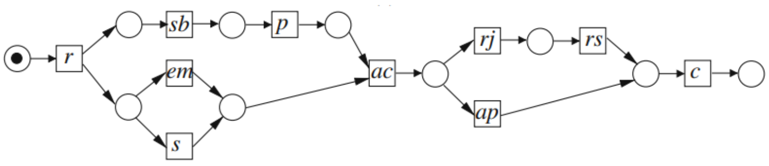
2.3 Region-Based Miner

Region theory approaches look for a process model that is precise and fitting, despite the region-based technique that is applied [7]. The two branches of region-based techniques for process discovery are state-based and language-based approaches. State-based region approaches for process discovery convert the event log to a state-based representation, which will be used to discover the Petri net. The Definition 6 formalizes this representation, while Figure 6(a) displays an example of a transition system and Figure 6(b) the derived Petri net.

Definition 6 (Transition System). *A transition system (TS) is a tuple* S, Σ, A, s_{in}*, where* S *is a set of states,* Σ *is an alphabet of activities,* $A \subseteq S \times \Sigma \times S$ *is a set of (labeled) arcs, and* $s_{in} \in S$ *is the initial state. The notation* $(s, e, s') \in A$ *will be denoted* $s \xrightarrow{e} s'$ *as a shortcut [7].*



(a) transition system



(b) derived Petri net

Figure 6: State-based region discovery.

Given a model that has executable semantics, it can be mapped into a transition system. A transition system is made up of states and transitions. Black circles represent states while arcs represent transitions which are used to connect two states. These arcs are labeled with the name of the activity, and multiple arcs can have the same label. The transition system's behavior can be analyzed to understand how it works. The transition begins at one of the initial states, and any path that starts from this state in the graph represents a potential execution sequence. The execution path is considered successful if it ends at one of the final states. If the execution path reaches a non-final state without any outgoing transitions, it is considered a deadlock. However, the lack of deadlocks does not ensure successful termination. There is a possibility that the transition system may enter a "livelock" state where some transitions are still enabled, but it is impossible to reach one of the final states.

Transition systems are simple but they struggle to efficiently depict concurrency. For example, if there are n parallel activities that must be executed in any order, there are n! possible execution sequences. The transition system requires 2^n states and $n \times 2^{n-1}$ transitions to represent it. This is an example of the well-known "state explosion" problem [8]. Since business processes are inherently concurrent, more sophisticated models, such as Petri nets, are required to accurately convey process mining results.

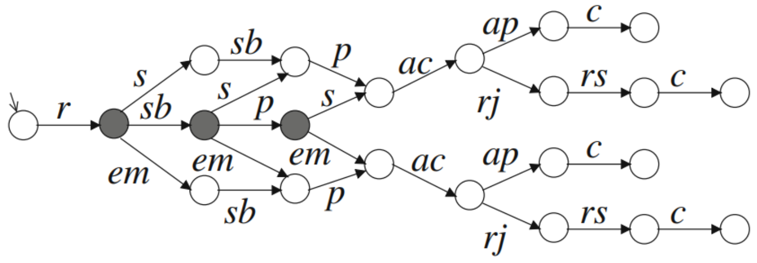
Definition 7 (Multiset representation of traces). *We denote by* $\#(\sigma, \mathsf{e})$ *the number of times that event* e *occurs in* σ*, that is* $\#(\langle \mathsf{e}_1 \dots \mathsf{e}_n \rangle, \mathsf{e}) = |\{\mathsf{e}_i \mid \mathsf{e}_i = \mathsf{e}\}|$*. Given an alphabet* Σ*, the Parikh vector of a sequence* σ *with respect to* Σ *is a vector* $p_\sigma \in \mathbb{N}^{|\Sigma|}$ *such that* $p_\sigma(\mathsf{e}) = \#(\sigma, \mathsf{e})$ *[7].*

There are various methods available to create a transition system from an event log, but the most widely used approach involves examining the set of event multisets present within a subtrace in the log in order to integrate state information.

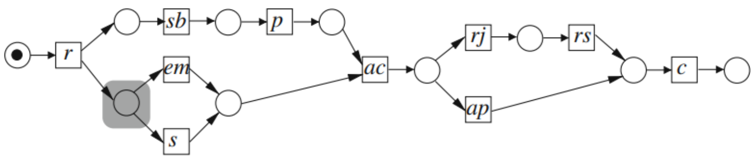
Definition 8 (Multiset State Representation of an Event Log)**.** *Given an event log $L \in \mathcal{B}(\mathcal{U}_{act}{}^*)$, the TS corresponding to the multiset conversion of L, denoted as $TS_{mset}(L)$, is $\langle S, \Sigma, T, s_{p_\epsilon}\rangle$, such that: S contains one state s_{p_ω} for each Parikh vector p_ω of a prefix ω in L, with ϵ denoting the empty prefix, and $T = s_{p_\omega} \xrightarrow{e} s_{p_{\omega e}} \mid \omega e$ is a prefix of L} [7].*

To further demonstrate this section, an example is extracted from "Finding Structure in Unstructured Processes: The Case for Process Mining" [9]. The activities that are included in the event log: *r=register, p=payment, s=ship, c=close, ac=accounting, rs=resolve, ap=approved, sb=send bill, rj=rejected, em=express mail.* The following event log is given: $L_2 = [\langle r, em, sb, p, ac, ap, c\rangle^{10}, \langle r, sb, em, p, ac, ap, c\rangle^{10}, \langle r, sb, p, ac, ap, c\rangle^{10}, \langle r, s, sb, p, ac, ap, c\rangle^{10}, \langle r, sb, s, p, ac, rj, rs, c\rangle^{10}, \langle r, sb, p, em, ac, rj, rs, c\rangle^{10}, \langle r, sb, p, em, ac, ap, c\rangle^{10}]$. An example of TS constructed according to Definition 8, is presented in Figure 6(a).

Definition 9 (Region)**.** *Let S' be a subset of the states of a TS, $S' \subseteq S$. If $s \notin S'$ and $s' \in S'$ then we say that transition $s \xrightarrow{a} s'$ exits S'. Otherwise, transition $s \xrightarrow{a} s'$ does not cross S': it is completely inside ($s \in S'$ and $s' \in S'$) or completely outside ($s \notin S'$ and $s' \notin S'$). A set of states $r \subseteq S$ is a region if for each event $e \in E$, exactly one of the three predicates (enters, exits or does not cross) holds for each of its arcs [7].*



(a) Example of region (three shadowed states). The predicates are r enters, s abd em exits, abd the rest of events do not cross.



(b) Corresponding place shadowed in the Petri net.

Figure 7: Presenting regions

Figure 7 presents a visualization of a region on the TS of the running example. A region corresponds to a place in the Petri net, which is shadowed in the Figure 7(b), and the Petri net flow relation is determined by the role of the arcs. There is an arc from the transition corresponding to an event e to the place when the event enters the region, and from the region to the transition for e when the event exits the region. Events satisfying the do not cross relation are not connected to the corresponding place. The shaded region in Figure 7(b) represents the region depicted in Figure 7(a). In this case, event r is part of the input transitions of the place, while events em and s belong to the output transitions set. The algorithm for Petri net derivation from a transition system involves finding regions and creating the Petri net as demonstrated in the example.

3 Methodology

The methodology employed for the study case is outlined in this section. The first subsection introduces the research questions that the paper aims to address. The second elaborates on the data collection process and provides details about the components of the event log. Finally, the last subsection explains the methodology for process discovery and conformance checking, which is crucial for conducting further data analysis.

3.1 Research Questions

The main objective of this research is to investigate the use of process mining techniques for process optimization in an industrial context in Albania. To reach this objective, it is necessary to address a number of research questions.

- *What is process mining and how can it be applied to optimize processes in Albania?*
  This question seeks to explore the concept of process mining and its potential applications in optimizing processes in Albania. The research will examine process mining techniques and tools to investigate how they can be used to identify inefficiencies and improve process performance.
- *What are the challenges and limitations of process mining in optimizing processes in Albania?*
  The aim is to identify the challenges and limitations of process mining when applied to optimizing processes in Albania. The research will investigate issues such as data quality, data privacy, and the interpretability of process mining results.

3.2 Data Collection

In process mining, the event log dataset is crucial for optimizing processes and identifying inefficiencies. However, as the real data can be sensitive and confidential, using it for research purposes may not be possible. In such cases, synthetic datasets that mimic the characteristics of the real data without disclosing any sensitive information are used.

In this research study aimed at optimizing the loan application process of an Albanian financial institution, a synthetic event log dataset is used. The dataset is designed to replicate the loan application process, including the steps involved, the actors, the time durations and additional information. The synthetic data was generated using simulation tools, and it was calibrated to match the statistical characteristics of the real data.

The dataset used in this study was translated into both Albanian and English, but the later version is used for presenting results in the paper with the aim of facilitating wider accessibility and understanding. Both versions of the event log are available online and can be accessed via Github for further analysis and replication of the study [10].

3.2.1 Event Log

The event log is given in a CSV format. It encompasses all online applications that were filed in 2020-2021 and presents the chronological sequence of events that took place. The loan application process typically involves several stages such as validating the application, the bank's decision to offer a loan or not, the applicant's response, and the validation of the applicant's decision to accept the offer. The event log contains the following columns presented in the Table 1.

Columns	Description	Example
Case ID	Unique identifier of the application	Application_1691306052
Activity	Name of the activity performed	A_Create_Application
Resource	Identifier of an employee or system	User_1
Complete Timestamp	End time of the performed activity	2017-01-01 11:15:21.303
Event Origin	Indicates the type of activity	Loan Application
Number of Terms	Indicates the number of times an applicant should repay	33
Monthly Cost	Offer's monthly repayment to the bank	200
Offered Amount	The amount offered to the applicant	6000
Application Type	Indicates whether this application is for a new credit or raising an existing credit	New Credit
Loan Goal	Indicates the goal that the loan should be spent on	Home improvement
Requested Amount	Applicant's requested loan amount	22000

Table 1: Dataset Columns

Definition 10 (XES Standard)**.** *The XES format utilizes XML to create a structured and standardized format for recording event data. It is designed to enable the exchange of event log information between different tools and application domains, and is primarily used for process mining, which involves analyzing operational processes by examining their event logs [11].*

To get more insights from the event log, it can easily be imported in ProM 2.1. As the event log is in CSV format, it needs to be converted in XES standard, as per the formal Definition 10. Afterwards, information like the following can be obtained from the **Explore Event Log (Trace variants ...)** view.

- The time period covered by the event log is from 01/01/2020 10:16:00h to 01/02/2021 08:04:00h.
- In total there are 4,982 traces, 74,809 events, 13 attributes and 1,424 unique trace variants in the event log.

3.3 Data Analysis

The event log is then analysed through process discovery and conformance checking techniques using ProM. Process discovery involves analyzing the event logs generated by the system to extract information on the sequence of events that took place, while conformance checking compares the actual process execution with the ideal process model and identifies any deviations. In the remaining part of the section are outlined the steps that were performed for the analysis, while the results are presented on section 4.

3.3.1 Process Discovery

Process discovery is a critical component in understanding the behavior of a system or a process. The process discovery phase involves analyzing the event logs generated by the system to extract information on the sequence of events that took place. ProM 2.1, a popular tool for process discovery offers several algorithms to analyze event logs. Region-based miner algorithm was found to be the most effective in identifying the underlying process of interest in this case.

The region-based miner algorithm first divides the log into regions where similar activities occur, and then identifies the transition patterns between these regions. Different algorithms were tried and the region-based miner algorithm provided better fitness and precision compared to other algorithms available in ProM. Additionally, the region-based miner algorithm had several parameters that were adjusted to fine-tune the analysis, including collection type and size limit, key classifier filter, transition label filter and post-mining conversions. The ability to adjust these parameters allows for a more tailored analysis of the log and a better understanding of the system's behavior.

3.3.2 Conformance Checking

Conformance checking is an important aspect of process mining, and ProM 2.1 was used as the tool to perform this task. The *Replay a Log on a Petri Net for Conformance Analysis* plugin is one of the many plugins available in the ProM framework that enables the comparison of event logs and Petri nets. To use this plugin, the Petri net model discovered is used, and the corresponding event log is loaded. The plugin then maps the events to the Petri net model, replaying the log on it and highlighting any deviations found. The results of this analysis can be seen in the Results section (4), where the identified deviations are reported along with their severity. This information can be used to improve the process and optimize its performance.

4 Results

In this section, the results of our analysis using process mining techniques on the event log are presented. The section is divided into three subsections, each detailing a specific aspect of the analysis as presented in section 3, In the first subsection, we discuss the results of the process discovery phase, which involved identifying and visualizing the underlying process model as well as understanding it. In the second, we present the results of the alignment-based analysis, which aimed to compare the actual process execution with the ideal process model and identify areas of improvement. In the final subsection, Celonis is leveraged to delve deeper into the analysis of throughput time and deviations, ultimately resulting in concrete recommendations to enhance the process's efficiency and reduce throughput time.

4.1 Process Discovery

One approach commonly used by most practitioners in Process Discovery is the generation of Directly-Follows Graphs (DFG), as highlighted by the paper "A practitioner's guide to process mining: Limitations of the directly-follows graph" [12]. This approach is also favored by many commercial tools for process mining as the default notation while allowing users to simplify its complexity by filtering out nodes (activities) and edges (paths) based on frequency. ProM allows us to mine a directly-follows graph from an event log through the plugin **Mine with Directly Follows Visual Miner**.

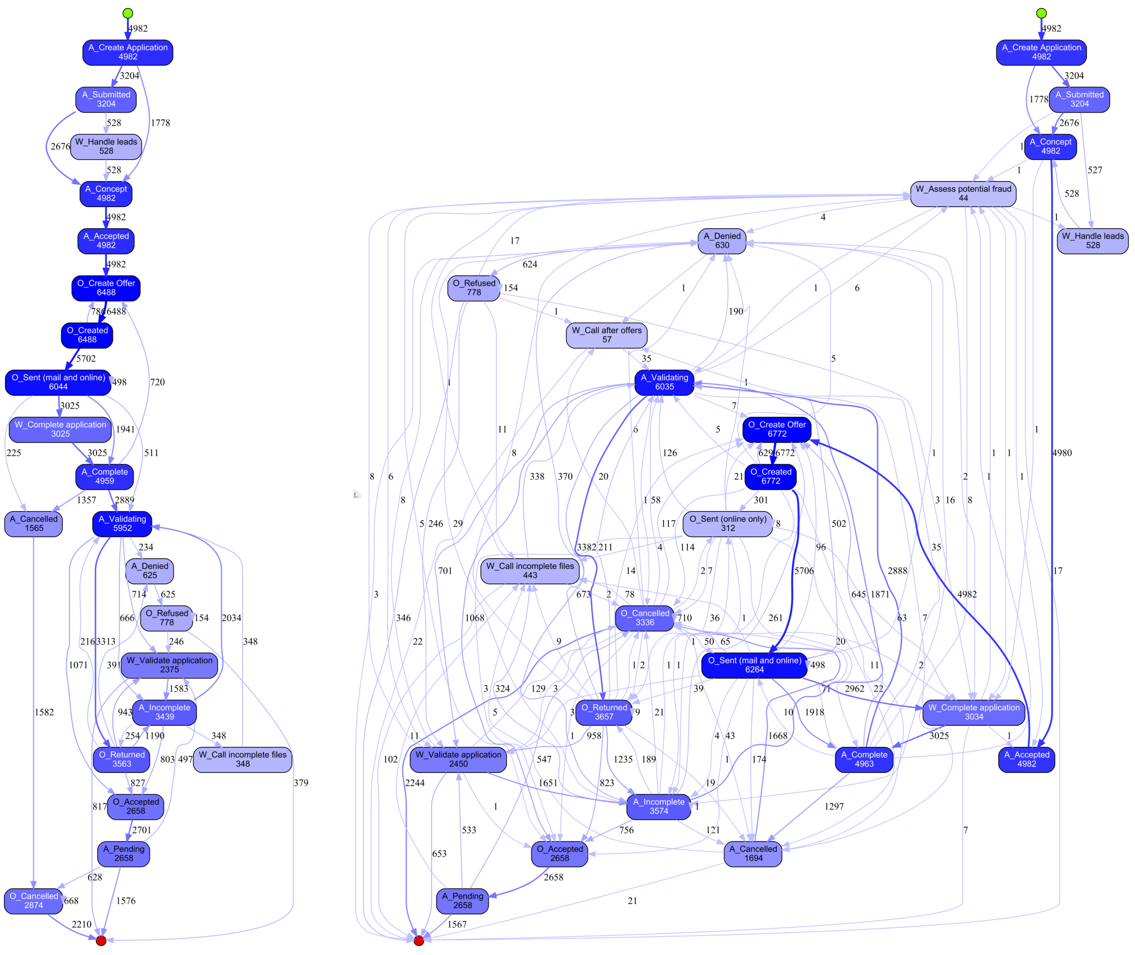


Figure 8: The DFGs of the event log, 80% most frequent paths (left) and unfiltered (right).

Figure 8 shows the what the DFG generated by the event log looks like while keeping only 80% most frequent paths (on the left), as well as keeping every path (on the right). Additionally, edges and nodes of the DFG are labelled with their frequency.

However, using notations like Directly-Follows Graphs (DFGs) instead of more sophisticated ones, such as Petri-Nets, can be problematic because it is challenging to capture concurrency. This limitation often leads to loops being introduced in the model, which can result in unintended behavior. Furthermore, as the frequency of paths and activities approaches 100%, the graph can become densely populated with edges, leading to what is commonly known as Spaghetti Processes [13]. For a more sophisticated process model, Region-based Miner was used for the process discovery using the state-based approach. A transition system was first mined from the given log by using the plugin **Mine Transition System** with the parameters specified as follows:

- Backward Key: Event Name (no Forward Key)
- Collection Type: Multiset
- Collection Size Limit: 15
- Transition Size Limit: 50
- Key Classifier Filter, Top 'Event Name' percentage: 96
- Transition Label Filter, Top percentage: 96
- Post-mining conversions: Remove self loops, Merge states with identical inflow, Merge states with identical outflow

Afterwards, the transition system can be converted into a Petri net by using the plugin **Convert to Petri Net using Regions**. The resulting Petri Net is presented in Figure 9.

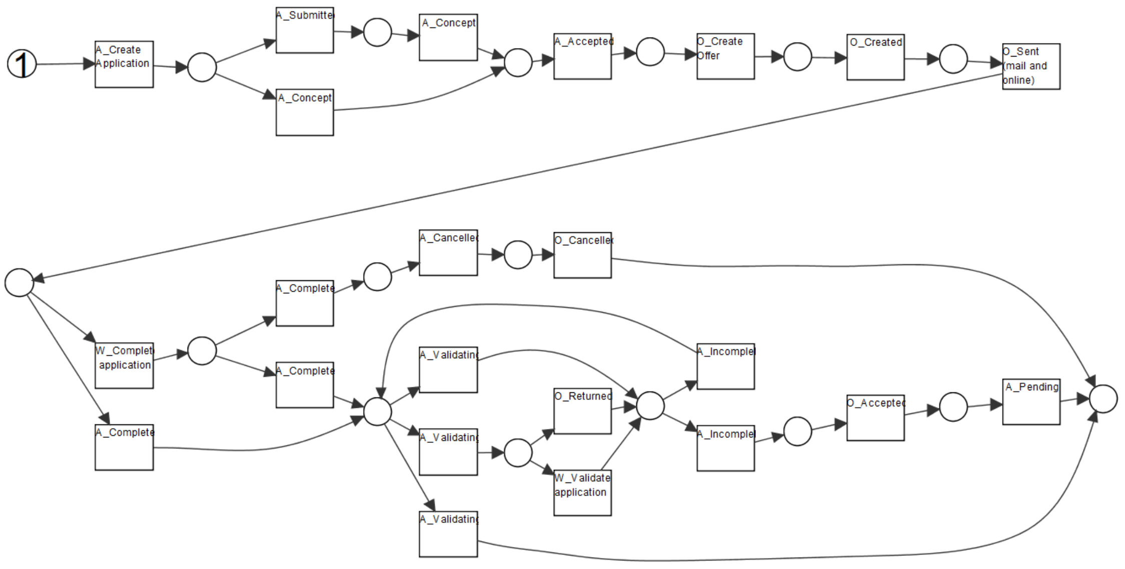


Figure 9: The model discovered by Region-Based Mining

To interpret a Petri net, we need to understand what each element represents and how they are connected. The Petri net consists of places (circles) and transitions (rectangles), which are connected by arcs. Each transition has a unique label that indicates the event name. For instance, the first transition *A_Create_Application* represents the initial step, where the applicant is creating a loan application in the system. Then, an XOR-split follows, meaning that the token can either enable the transition *A_Concept* or the sequential transitions *A_Submitted* and

A_Concept. An XOR-join causes the token to flow through the following sequential transitions: *A_Accepted*, *O_Create Offer*, *O_Created* and *O_Sent* (*mail and online*). The XOR-split that follows, enables one of the paths; the one that enables the transition *A_Complete* or the other that enables the transitions *W_Completed application* and then XOR-splits to enable the transition *A_Complete* and end up in the same place as the former path of the second XOR-split, or to enable the following sequence of transitions *A_Complete*, *A_Cancelled* and *O_Cancelled*, which leads to the ending place where there are no more moves left for the token.

The other path left to explore is the following, in which the token is in the place where the XOR-split enables one of the following transitions; *A_Validating* which leads to the final place, *A_Validating*, (*A_Validating* and *O_Returned*), or (*A_Validating* and *W_Validate_application*). Each one of the last three arcs mentioned are XOR-joined to a place that could possibly introduce a loop by XOR-splitting to enable the *A_Incomplete* and lead the token to the transition described initially in this paragraph, or it can enable the other three sequential transitions which eventually take the token to the final place, *A_Incomplete*, *O_Accepted* and *A_Pending*.

The plugin **Multi-Perspective Process Explorer** can be used to analyze the precision and the fitness of this model. After the alignments were computed, to find the percentage of models with the fitting higher than 0.9, *Toggle Trace Views* button was clicked and the traces were filtered by applying the condition *fitness > 0.9*. The number of traces satisfying the condition is 2,669. Hence, the percentage of traces that fit the model with an accuracy of over 90% is $\frac{2{,}669}{4{,}982} \times 100\% = 0.5357 \times 100\% = 53.57\%$. The alignment-based fitness and precision score of the model are also observed by using the *Multi-Perspective Process Explorer*. The information can be found on the information panel by changing the *MPE Mode* to *Show Precision Mode*. The findings are presented in the Table 2.

Attribute	**Value**
% of Traces with Fitting > 0.9	53.57%
Activity precision	99.4%
Alignment based fitness	87.8%

Table 2: Multi-Perspective Process Explorer Statistics

The quality of a process mining result can be determined by referring to the four main quality dimensions which are: fitness, precision, simplicity and generalization, as Wil M. P. van der Aalst [4] explains. A model has perfect fitness if all traces in the log can be replayed by the model, while a good fitness allows for the behaviour seen in the event log. The lack of precision in a model leads to "underfitting" and over-generalizing the example behavior in the log, meaning that it allows for more behavior even when there are no indications in the log that suggest this additional behavior. Meanwhile, an "overfitting" model does not generalize and therefore, only allows for the exact behavior recorded in the log, but as mentioned, logs can be incomplete as they capture only a fraction of the real-world scenario. Hence, precision is attained when a balance between "overfitting" and "underfitting" is achieved. As per the results in Table 2, it can be seen that a balance of structural and behavioral appropriateness is attained.

4.2 Conformance Checking

Conformance checking is used to identify commonalities and discrepancies, where events in the log are compared to activities in the process model. Two common techniques that are used to analyze and optimize processes are the move on model and move on log.

The move on model is a process mining technique used to analyze event logs and identify deviations from the expected process flow based on a pre-defined process model. The technique was introduced by Aalst in 2004 [14] and has since been refined and extended by many researchers.

The move on model technique involves comparing the actual process behavior with the process model to identify cases where process instances deviate from the expected flow.

Move on log is a process mining technique that compares the expected behavior of a process model with the actual behavior recorded in an event log to identify deviations. The technique was introduced by Buijs et al. in 2014 [15] and has been further developed by other researchers. It uses a behavioral profile, which captures the order and frequency of activities in a process, to identify traces in an event log that differ from the expected behavior.

The plugin *Replay a Log on Petri Net for Conformance Analysis* is used with the default parameter configurations to portray the alignment results on the event log and the model from Figure 9. The model projected with alignments is presented in Figure 10.

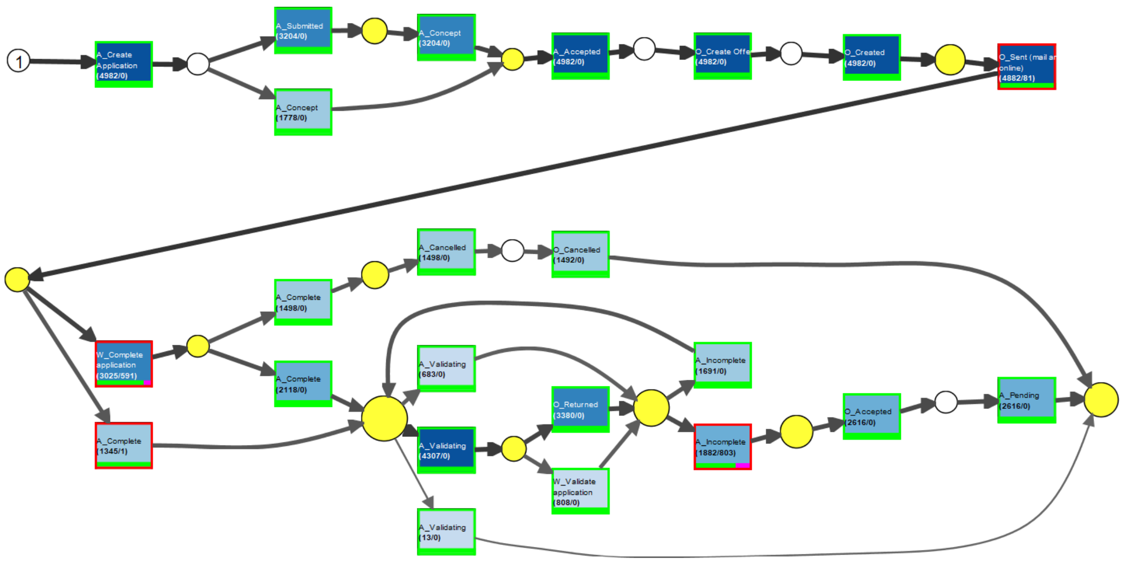


Figure 10: Model Projected with Alignments

The visualization of the *Replay a Log on Petri Net for Conformance Analysis* plugin, enables the observation of moves on model which can be seen in the pink portion of the red-bordered transitions. The three most significant transitions with a number of moves on model are presented in Figure 11. From the highest number of moves on model, there are 803 traces where the activity *A_Incomplete* was not executed, 591 traces where the activity *W_Complete application* was not executed and 81 traces where the activity *O_Sent (mail and online)* was not executed.

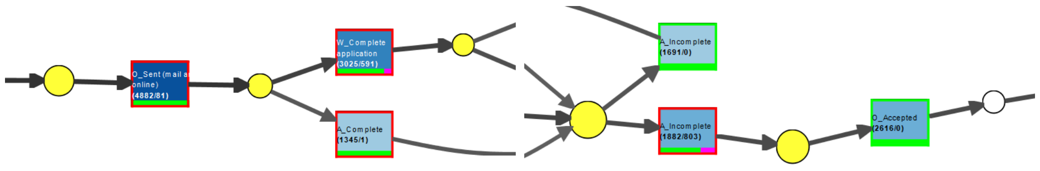


Figure 11: The transitions in the *Model Projected with Alignment*

In these transitions, the moves happen because while model does not allow for the corresponding activity to be skipped, there are traces in the log where the activity is indeed skipped. A way in which this problem can be resolved is by adding an additional transition in this cases that would allow skipping the corresponding activity. Thus, the alignment fitness would increase because the alignments for the other traces would not change.

The move on log is observed on the yellow places in the model projected with alignments. These places correspond to states in the process where the model fails to mimic something that happened in the log. The selected places in Figure 12 filled with red, are the ones that contain the

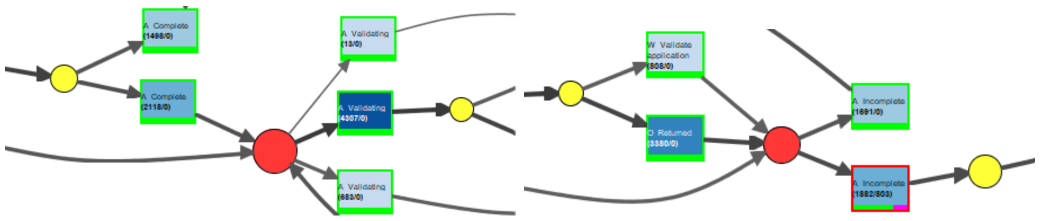


Figure 12: The places in the *Model Projected with Alignment*

most noticeable moves on log.

After selecting a place, a better overview of the moves on log can be obtained by looking at the *Element Statistics* of the *Inspector*. The Table 3 below summarizes the alignment statistics for the first place (left side of the Figure 12), while the Table 4 shows the statistics for the second place (right side of the Figure 12). It can be seen that the most problematic activities which are shown to have significant move on log are *O_ Create Offer*, *O_ Created*, *W_ Validate Application*.

Marking	**Move on Log**	**# (Freq)**	**# Traces**
Sink1	A_Cancelled	103	103
	A_Denied	49	49
	A_Incomplete	1	1
	O_Cancelled	297	222
	O_Create Offer	857	669
	O_Created	857	669
	O_Refused	55	47
	O_Returned	173	169
	O_Sent (mail and online)	683	563
	O_Sent (online only)	170	153
	W_Assess potential fraud	3	2
	W_Call after offers	37	37
	W_Call incomplete files	210	204

Table 3: First Sink Statistics

Marking	**Move on Log**	**# (Freq)**	**# Traces**
Sink2	A_Cancelled	9	9
	A_Denied	383	383
	O_Cancelled	59	41
	O_Create Offer	11	11
	O_Created	11	11
	O_Refused	477	383
	O_Returned	10	10
	O_Sent (mail and online)	8	8
	O_Sent (online only)	2	2
	W_Assess potential fraud	30	30
	W_Validate Application	1149	1127

Table 4: Second Sink Statistics

4.3 Process Analysis

The overall count of cases is 4982, and the average duration for processing each case is 22 days. Among all the cases, the most common variant has occurred 320 times, representing 6% of the total cases.

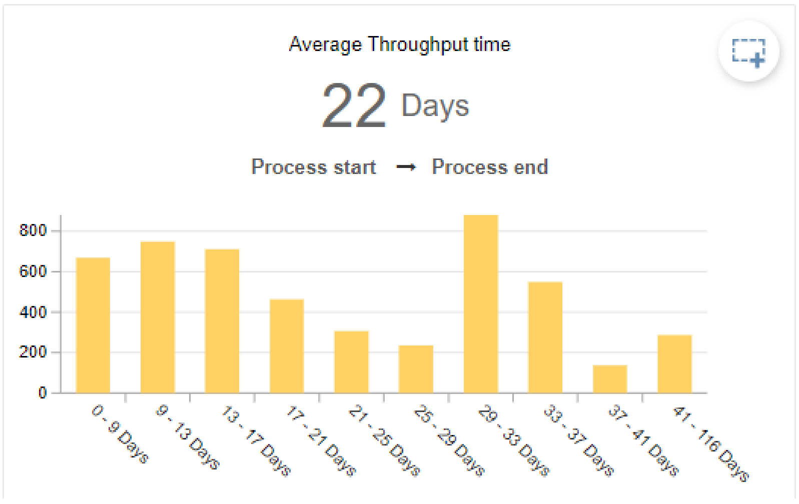


Figure 13: The throughput time of all variants.

The variant is as follows:
⟨Process Start, A_Create Application, A_Submitted, A_Concept, A_Accepted, O_Create Offer, O_Created, A_Complete, O_Sent (mail and online), W_Complete application, A_Cancelled, O_Cancelled, Process End⟩

The number of cases being processed follows a decreasing trend in the throughput time until the 30th day, when there is a sudden increase as shown on Figure 13. Upon investigation, it was discovered that canceled cases deviate from the expected pattern, with around 85% of them taking approximately 31 to 32 days to process.

To provide a more detailed illustration, Figure 14 is presented, depicting the throughput time of two sets of variants respectively: one without the *A_ Cancelled* activity and the other with the *A_ Cancelled* activity. These figures allow for a visual comparison of the respective throughput times.

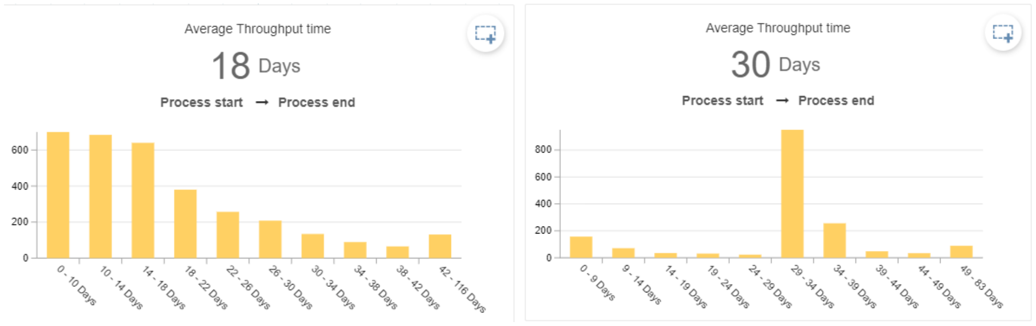


Figure 14: The throughput time of all variants without (left) and with (right) the *A_ Cancelled* activity.

A DFG model was created by filtering with the following activities: *A_ Create Application, O_ Created, W_ Complete Application, O_ Accepted, O_ Refused, O_ Cancelled,* and *A_ Cancelled.* The resulting DFG is illustrated in Figure 15. It shows that the long-lasting edges are the edges when going through activity *O_ Created* or *W_ Complete* application to *A_ Cancelled.* The throughput time in these edges is of 27 days. Cases lasting more than 29 days typically exhibit a key trait of high likelihood of cancellation. Figure 16 illustrates that approximately 80% of such cases ultimately get cancelled.

There are 3288 Cases in total with an average throughput time of 18 days. The most frequent

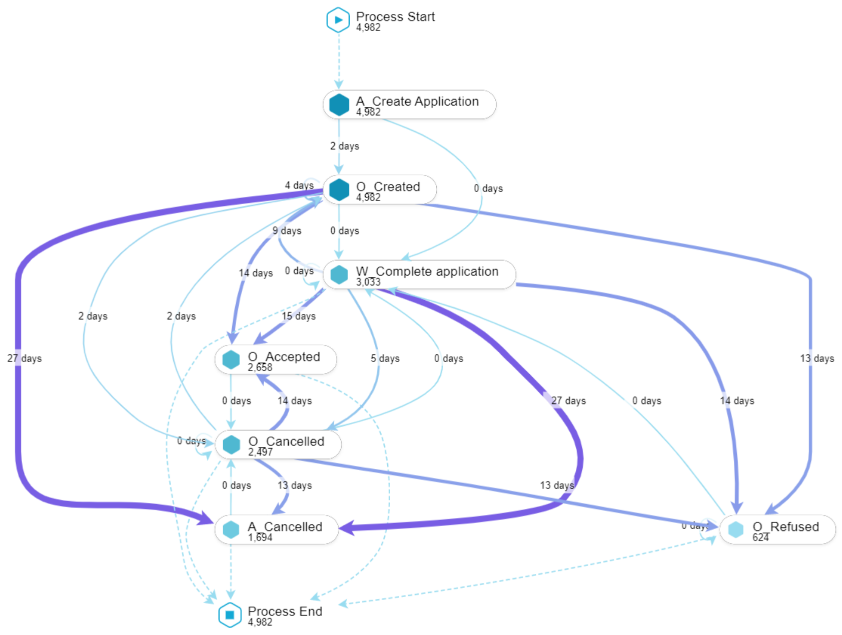


Figure 15: The DFG model annotated with performance.

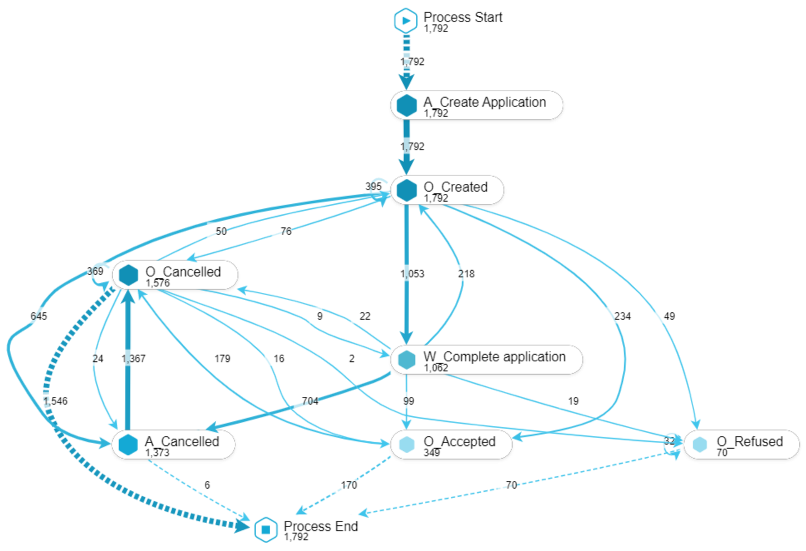


Figure 16: The number of cases that took 30 days or more in each activity.

variant has happened 141 times, thus appearing in 4% of the total cases. The variant is as follows: ⟨Process Start, A_Create Application, A_Submitted, A_Concept, A_Accepted, O_Create Offer, O_Created, A_Complete, O_Sent (mail and online), W_Validating, O_Returned, A_Pending, O_Accepted, Process End⟩

The number of cases decreases as the throughput time increases as can be seen in Figure 17). Two significant workflow bottlenecks were observed, namely *W_ Complete Application* & *W_ Validate Application* with which the underlying issue is that either of them would take 5-9 days regardless of the outcome. There is a clear negative correlation between the number of offers and the throughput time of applications as presented in Figure 17. The more offers there are, the less throughput time the application needs.

The throughput time is influenced by two primary factors:

- The *A_ Cancelled* activity has a throughput time of 28 days in 84% of cases, in which the application was cancelled.
- The time it takes to start the activity *A_ Validating* after the activity *W_ Complete Application*.

To decrease the throughput time, a recommended approach is to send periodic reminders to clients on a weekly basis if they fail to respond, and cancel the application if there is no response after receiving only two reminders. By implementing this strategy, the duration of the *A_ Cancelled* activity would be limited to a maximum of two weeks, while also benefiting from the reminder system that helps customers avoid missing any deadlines related to their loan applications.

In addition, it is worth exploring the process of *A_ Validating*, particularly in successful scenarios where the bank generates an offer. This investigation would contribute to optimizing the overall workflow and further improving process efficiency.

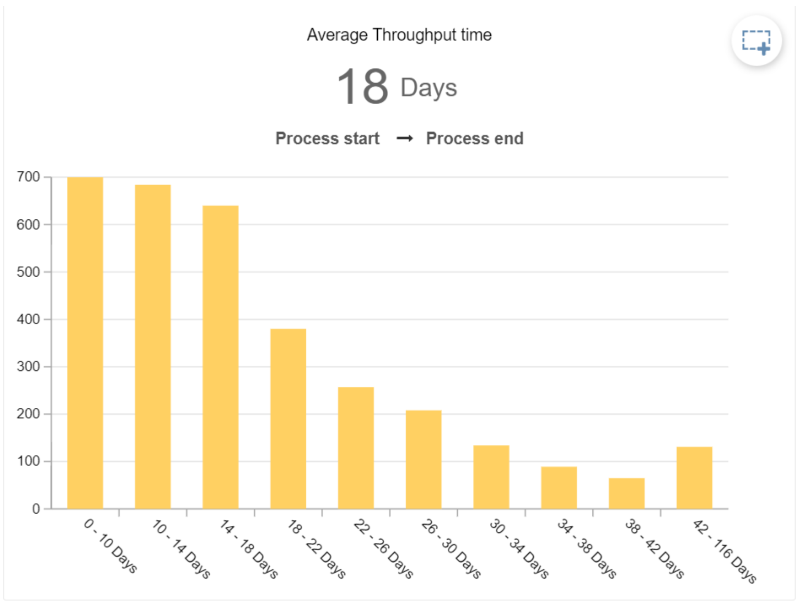


Figure 17: The throughput time distribution for cases without the application being cancelled.

5 Discussion

The study aimed to explore the application of process mining in optimizing processes in Albania. The findings show that process mining is a effective tool that can be used to identify inefficiencies and improve process performance in various industries in Albania. Process mining is a technique used to extract knowledge from event logs to analyze business processes. It involves the use of algorithms and statistical methods to identify patterns and relationships in the data. The results can be used to identify bottlenecks, inefficiencies, and opportunities for improvement in the processes.

In this study, process mining was applied to a case study of a loan application process in a financial institution in Albania. The analysis revealed that the process was inefficient due to extended throughput time caused by the lengthy *A_ Cancelled* activity and the delay in starting *A_ Validating* after *W_ Complete Application.*

The results of the analysis were used to redesign the process, with changes being implemented, such as periodic reminders being sent to clients who had not responded to their loan applications, on a weekly basis. Furthermore, a limit of only two reminders before the application is canceled was set. The *A_ Validating* activity was also optimized to expedite the process of creating an offer in successful scenarios. Through these modifications, the efficiency of the overall workflow was enhanced, and the process performance was improved.

These redesigns led to a more streamlined and efficient workflow, reducing the overall throughput time and improving the process performance. Though the results of this study demonstrate that process mining can be applied to optimize processes in Albania, there are several challenges and limitations that need to be addressed to ensure its effectiveness. These challenges include data quality, data privacy, and the interpretability of the results.

One of the major challenges of process mining is data quality. The accuracy and completeness of the data used in the analysis can significantly impact the quality of the results. In Albania, data quality is a major concern, and the lack of accurate and reliable data can limit the effectiveness of process mining. Data privacy is another significant challenge of process mining in Albania. The privacy of personal and confidential data needs to be protected, and proper measures need to be put in place to ensure that the data is not misused or compromised during the analysis.

The interpretability of the results is another limitation worth mentioning. The analysis results can be complex and difficult to understand, making it challenging to make informed decisions. To overcome this limitation, it is important to involve stakeholders in the analysis and interpretation of the results.

In conclusion, process mining has the potential to optimize processes in Albania, leading to improved efficiency and performance. However, the challenges and limitations discussed in this study need to be addressed to ensure its effectiveness. Addressing these challenges will require a concerted effort from stakeholders, including businesses, and researchers.

6 Future Work

In future work, investigating the integration of Generative AI with Process Mining might prove important. The main idea is to use process mining and generative AI techniques to optimize processes in an automated manner. The aim would be to create a model that can automatically identify bottlenecks, inefficiencies, and other areas of improvement in processes, and generate automated solutions to improve the efficiency and effectiveness of the operations. The research could involve gathering data from real-life operations, and using process mining to analyze and visualize the process flows. Generative AI techniques could be used to generate alternative process flows and evaluate their impact on the operations.

References

[1] Wil M. P. van der Aalst. *Process Mining: A 360 Degree Overview*, pages 3–34. Springer International Publishing, Cham, 2022.

[2] Wil M. P. van der Aalst. *Foundations of Process Discovery*, pages 37–75. Springer International Publishing, Cham, 2022.

[3] Kurt Jensen. *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use, Vol. 2*. Springer-Verlag, Berlin, Heidelberg, 1995.

[4] Wil M. P. van der Aalst. *Process Mining: Data Science in Action.* Springer, Heidelberg, 2 edition, 2016.

[5] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[6] Wil Aalst. The application of petri nets to workflow management. *Journal of Circuits, Systems, and Computers*, 8:21–66, 02 1998.

[7] Adriano Augusto, Josep Carmona, and H.M.W. (Eric) Verbeek. *Advanced Process Discovery Techniques*, volume 448 of *Lecture Notes in Business Information Processing (LNBIP)*, pages 76–107. Springer, Germany, 2022.

[8] Antti Valmari. *The state explosion problem*, pages 429–528. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[9] Wil Aalst and C.W. Giinther. Finding structure in unstructured processes: The case for process mining. pages 3 – 12, 08 2007.

[10] Kalemi Ana. Loan-application-system-dataset.

[11] C.W. Gunther and H.M.W. Verbeek. *XES - standard definition.* BPM reports. BPMcenter. org, 2014.

[12] Wil M.P. van der Aalst. A practitioner's guide to process mining: Limitations of the directly-follows graph. *Procedia Computer Science*, 164:321–328, 2019. CENTERIS 2019 - International Conference on ENTERprise Information Systems / ProjMAN 2019 - International Conference on Project MANagement / HCist 2019 - International Conference on Health and Social Care Information Systems and Technologies, CENTERIS/ProjMAN/HCist 2019.

[13] Wil Aalst. Process mining: Discovering and improving spaghetti and lasagna processes. pages 1–7, 04 2011.

[14] Wil Aalst, A. Weijters, and Laura Măruşter. Workflow mining: Discovering process models from event logs. *Knowledge and Data Engineering, IEEE Transactions on*, 16:1128 – 1142, 10 2004.

[15] J. Buijs, B. Dongen, and Wil Aalst. Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity. *International Journal of Cooperative Information Systems*, 23:1440001, 03 2014.