UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Software Engineering Curriculum

Mir Jalal Hashimli

Automated Variant Analysis for Business Process Improvement

Master's Thesis (30 ECTS)

Supervisor(s): Fredrik Milani, PhD
Katsiaryna Lashkevich, PhD
David Chapela, PhD

Tartu 2023

Automated Variant Analysis for Business Process Improvement

Abstract:
The execution of business processes results in variants. While some of the variants perform better than others, the majority are not optimal. Different techniques can be used to go through the previous executed cases, identify what caused them to reach a successful conclusion given a specific metric, and recommend an action to the process worker. The thesis presents a case study in which a conceptual framework for business process variant analysis is implemented as a software solution. The result provides the most efficient process variant by comparing different impacts on the process cycle time efficiency. It is concluded that considering the best variant is the key to improving business processes.

Keywords: Automated Variant Analysis, Business Process, Control flow, Process Clustering

CERCS: P170 - Computer science, numerical analysis, systems, control

Automatiseeritud variantide analüüs äriprotsesside täiustamiseks

Lühikokkuvõte: Äriprotsesside elluviimise tulemuseks on erinevad variandid. Kuigi mõned variandid toimivad paremini kui teised, pole enamik neist optimaalsed. Eelnevalt teostatud juhtumite läbimiseks saab kasutada erinevaid tehnikaid. On võimalik tuvastada, mis pani nad konkreetse mõõdiku alusel edukale järeldusele jõudma ning vastavalt sellele on võimalik protsessitöötajale toimingut soovitada. Käesolev uurimistöö esitab juhtumiuuringu, milles on tarkvaralahendusena realiseeritud äriprotsesside variantide analüüsi kontseptuaalne raamistik. Selle tulemuseks on kõige tõhusama protsessivariandi leidmine läbi erinevate mõjude võrdlemise protsessitsükli aja efektiivsusele. Jõutakse järeldusele, et parima variandi kaalumine on äriprotsesside täiustamise võti.

Võtmesõnad: Automatiseeritud variantide analüüs, äriprotsess, juhtimisvoog, protsesside rühmitamine

CERCS: P170 - Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Contents

List of Figures

List of Tables

1 Introduction

Business processes are vital assets for companies, significantly affecting how their products and services are perceived in the market [1]. One example of a business process can be the "Order fulfillment process" in an e-commerce company, which involves the steps required to receive and process customer orders until the products are delivered to the customers. A business process is defined as a set of tasks performed by resources in coordination [10]. People, machines, or software perform the activities and are connected to each other by a control flow. It defines the order of the activities are performed. The control flow can be sequential, parallel, or a combination. A subset of executions of a business process that can be distinguished from others based on a given predicate (e.g., the executions of a process in a given country) is called a process variant [5].

Real-life business processes are dynamic, flexible, and adaptable over time, so in different periods of time could exist different execution versions of a given process [3]. For example, a business process could be executed differently depending on the day, year's season, customer type, product type, etc. While the execution of other business processes could influence the execution of a business process, it can also change over time to adapt to the changing business environment. Process mining is widely used to discover, analyze, and improve business processes in various industries.

Process mining techniques are concerned with deriving process-based insights from historical records of business operations recorded in information system event logs [7]. Process mining gives a few advantages to the users, especially in knowing how the process really works in an organisation [8].

Process variant analysis is a family of techniques to analyze event logs produced during the execution of a process, in order to identify and explain the differences between two or more process variants [5]. It is impossible to analyze all the business process variants manually. The primary motivation for this thesis is to improve the performance of business processes. The thesis aims to automatically analyze the variants of a business process and identify the best-performing variant.

Therefore, this thesis focuses on developing a framework, algorithms, and tools for identifying, analyzing, and suggesting process changes to improve the business process.

A control flow-based approach is proposed for implementing a clustering algorithm to automatically analyze the variants of a business process and identify the best-performing variant. The analysis is based on the execution logs of the business process.

Therefore, we have identified a gap in automating variant analysis for business processes. To address this gap, we ask (RQ1) *how can improvement opportunities related to process changes be identified and analyzed from event logs?*

The approach is split into discovery, analysis, and identification:

1. Discovering process model considering variants via trace clustering

2. Analyzing variants from the perspective of time

3. Identifying improvement opportunities

Experiments on synthetic and real-world event logs show that the approach identifies improvement opportunities of variants from the perspective of time.

The rest of the paper is structured as follows. Chapter 2 gives background information about business processes and process mining. Chapter 3 reviews related work that has been done until now. The methodology used in the thesis has been described in Chapter 4. Chapter 5 explains the approach for variant analysis for improvement opportunities. Chapter 6 covers the implementation of the approach, and Chapter 7 discusses the conclusion and future work.

2 Background

This section explains the concepts used in this thesis, like business process management, variant analysis, process mining, and algorithms.

2.1 Business Process Management (BPM)

Business process management encompasses a range of principles, methodologies, and approaches that promote the facilitation, supervision, structuring, execution, and evaluation of business processes [2]. BPM is the science of supervising how tasks are executed in an organization to ensure consistent results and opportunities for improvement [1]. It is a significant and popular subject due to its practical relevance, presenting numerous challenges for software developers and scientists [2].

2.1.1 Business Process Management Lifecycle

The BPM Lifecycle refers to the sequential stages or phases through which a business process undergoes during its management and improvement journey [2]. Figure 1 shows an overview of BPM Lifecycle. The BPM lifecycle starts with the *process identification* meaning the identification of the process, its architecture, and performance measures. Then, it continues with *process discovery* that documents the current state of the process. The next stage is *process analysis* where the problems affect the process are detected, following with *process redesign*. This stage is identification of changes to redesign the process and tackle the issue found in analysis stage. These changes are implemented in the *process implementation* phase based on desired process models. Lastly, after the implementation *process monitoring* is the final step to monitor the performance of the process and start again the cycle to improve it continuously.

Before analyzing any process, *process performance metrics* should be clearly explained to determine whether a process is in "good shape" or in "bad shape" [1].

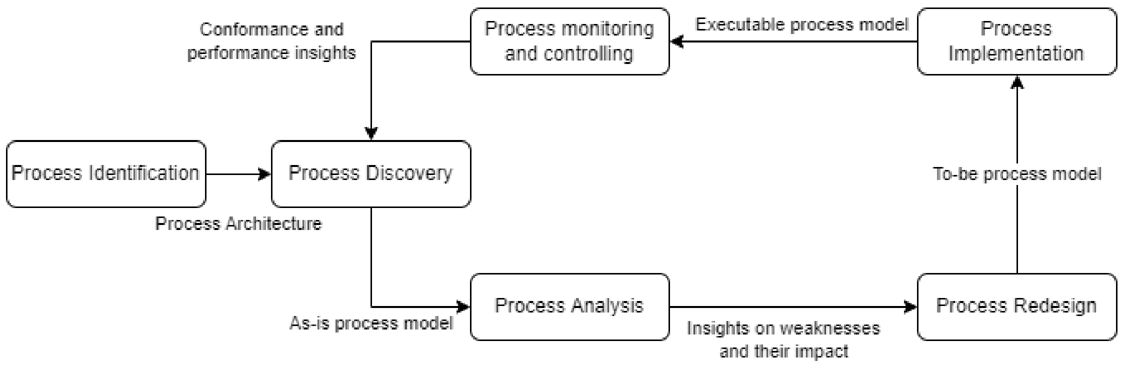


Figure 1. BPM Lifecycle [1]

2.1.2 Process performance metrics

BPM activities can be measured based on cost, quality, flexibility, and time [1]. The first three measures present how much the process is cheaper, better quality, and easy to adapt to changes [1]. The fourth dimension of process performance metrics measures how fast the process can achieve the desired outcome. Time-related measures are one of the broad and recurrent classes of measures [1]. For example, when we look at the "Order Fulfillment Process," a potential process performance metric can be the time between order placement and the package delivery to the customer's specified address. It is generally called cycle time. In other words, cycle time refers to the total time it takes to complete one full cycle of one process, from when it is fired to the end. Another two time-related metrics can be process time and waiting time:

- The processing time is the time taken to complete the actual tasks and activities involved in processing an order [1].
- Waiting time refers to the duration during which a task or activity is delayed or suspended, contributing to inefficiencies and potential disruptions in the process flow. Waiting times can arise due to factors such as resource unavailability, dependency on external inputs, and inadequate task prioritization, all of which impede the seamless progression of the process [1].

2.2 Variant Analysis

A subset of executions of a business process that can be distinguished from others based on a given predicate (e.g., the executions of a process in a given country) is called a process variant [5]. Process variants are very complex objects as they may vary significantly due to the complexity of data required to describe them even though they satisfy the same set of constraints [6]. Variant analysis involves systematically examining and comparing different variations or instances of a process to identify patterns, deviations, and opportunities for optimization [5]. It helps organizations understand the diverse ways a process can be executed, uncover inefficiencies, and make informed decisions about process improvements based on empirical data and insights gathered from these variations. There are several comparison parameters to do variant analysis. One of the most common methods is clustering variants based on control flow and performance analysis. Control flow analysis involves examining the order and logic of process steps to identify common patterns and deviations from the expected flow [18]. Another way is to compare variants related to time. The impact of the differences between the variants and identifying improvement opportunities are assessed by the increase in total cycle time, processing time, and waiting times. In other words, it is assessed by how much a particular difference between the variants increased the total process duration.

2.3 Process Mining

Process Mining is a data-driven technique that leverages event logs to unveil the real-world execution of these processes, offering insights into their actual performance [4]. Process Mining is a field of research situated between Business Process Management and Data Mining that aims to automate BPM tasks by utilizing event data collected from information systems, with the overarching goal of enhancing process efficiency, identifying bottlenecks, improving compliance, and providing valuable insights for informed decision-making [4] [7].

Process mining techniques aim to derive knowledge of the execution of processes using automated analysis of behavior recorded in event logs [7]. Process mining starts by gathering information about the processes as they take place [9]. Process mining is used to identify, analyze, and monitor business processes. Process mining uses process models in various ways. Process models are generated through the analysis of event data, functioning as templates for comparison, or employed to forecast potential bottlenecks [4]. A well-known challenge for any process mining technique is to strike the right balance between the behavioral quality of a discovered model to the event log and the model's complexity as perceived by stakeholders [7].

Process models are representations that depict the sequence of activities, tasks, decisions, and interactions that constitute a business process [4]. These models serve as abstractions of real-world processes, providing a structured and visual way to understand, communicate, and analyze how work is carried out within an organization [1].

Table 3 shows and example of event log for "Order Fulfillment Process". The table provided captures the chronological sequence of activities in an "Order Fulfillment Process" within an e-commerce company. Each row in the table represents a specific activity performed for a particular case, which corresponds to a customer order. The "Case ID" uniquely identifies each order. The "Activity Name" column specifies the action taken, such as "Receive Order," "Process Payment," "Pick Items," "Pack Items," and "Dispatch Shipment." The "Start Timestamp" indicates when the activity began, while the "Stop Timestamp" marks its completion. These timestamps offer insights into the duration of each activity. Finally, the "Resource" column designates the department or team responsible for executing the activity, such as "Customer Service," "Finance Department," "Warehouse Staff," "Packaging Team," and "Logistics Team."

Applied to the "Order Fulfillment Process" event log, Process Mining would automatically generate a visual model illustrating the sequence of activities from receiving an order to dispatching a shipment. It would then compare this model to the actual event data to identify deviations and inefficiencies, such as prolonged durations during activities like processing payments or packing items. By analyzing timestamps and resources, Process Mining would uncover patterns, bottlenecks, and potential optimization opportunities. Additionally, it could highlight frequent process variants, allowing you to understand how different sequences impact overall efficiency.

Case ID	**Activity Name**	**Start Timestamp**	**Stop Timestamp**	**Resource**
1	Receive Order	2023-08-10 09:00 AM	2023-08-10 09:15 AM	Customer Service
2	Receive Order	2023-08-10 09:20 AM	2023-08-10 09:30 AM	Customer Service
2	Process Payment	2023-08-10 09:35 AM	2023-08-10 09:50 AM	Warehouse Staff
3	Pick Items	2023-08-10 09:55 AM	2023-08-10 10:10 AM	Finance Department
1	Process Payment	2023-08-10 10:15 AM	2023-08-10 10:30 AM	Finance Department
4	Pick Items	2023-08-10 10:30 AM	2023-08-10 10:45 AM	Warehouse Staff
3	Pack Items	2023-08-10 10:50 AM	2023-08-10 11:00 AM	Packaging Team
2	Pack Items	2023-08-10 11:05 AM	2023-08-10 11:20 AM	Packaging Team
4	Dispatch Shipment	2023-08-10 11:25 AM	2023-08-10 11:40 AM	Logistics Team
2	Dispatch Shipment	2023-08-10 11:45 AM	2023-08-10 12:00 PM	Logistics Team

Table 1. Example Event Log for Order Fulfillment Process

2.4 Clustering Algorithms

Clustering techniques find applications across various domains and problem contexts [14]. Clusters represent the aggregation of large volumes of data into groups (modules) where entities within a group share stronger internal relationships compared to those across different groups [14]. In the context of process analysis, clustering has proven valuable. In [12], the authors present a novel method for clustering process executions into variants, incorporating additional attributes to enhance the clustering process. Moreover, this approach uses the starting time of each process instance as an additional feature to those considered in traditional clustering approaches by combining control flow and time features [13]. The process mining approach is very much useful in efficiently detecting the intrusions, as previous methods which were used with data mining approaches were not able to detect the intrusions in efficient manner [15]. Another essential facet of process mining is process identification, which involves deriving intelligible and representative process models from event logs. This enables stakeholders to uncover genuine insights into their business processes' actual behavior [7].

To delve into the foundations of Process Mining techniques, it is important to highlight that these techniques usually originate from event logs. An event log can be defined as a chronological record of events or activities captured from a system or process. Typically, an event log records each activity's occurrence, the corresponding case or process instance, the involved resources, as well as the start and end times. Table 1 provides an illustrative example of a set of events in an event log, each with attributes including case ID, activity, resource, start time, and end time.

It's worth noting that event logs can store additional information beyond these minimum attributes. However, for the temporal analysis conducted in this thesis, the case ID, activity, resource, start time, end time, resource, and one cohort column serve as the fundamental requisites.

3 Related Work

This section reviews relevant research on automated variant analysis and its application to business process improvement.

While identifying differences between process variants leads to valuable insights, it is important to ensure that these insights are actionable, meaning that they lead to process changes that effectively improve the performance of a business process [5].

Currently, known techniques are primarily theoretical, so applying them in real-life business processes is complex. This thesis contributes to implementing a software solution that creates a hands-on experience for the end users. This also improves the performance of the results by presenting the desired outcome.

Automated variant analysis has gained significant attention in business process management to identify and analyze variations in process execution.

In [16], the authors propose a technique for discovering process variants using process mining techniques. This study demonstrated the effectiveness of using process mining tools to identify and analyze process variants to improve process performance.

In [17], it's explored using automated variant analysis to identify process improvement opportunities. It's been proposed a framework for identifying and analyzing process variants based on control flow and resource usage and demonstrated the effectiveness of this approach in identifying process improvement opportunities.

In [18], it has been focused on using variant analysis to improve process efficiency and effectiveness. It presents a technique for identifying and analyzing process variants based on case attributes, such as customer segment or product type. It's demonstrated that this approach can be used to identify process improvement opportunities and optimize resource allocation.

Despite the many potential benefits of using automated variant analysis for business process improvement, there are also several challenges and limitations to consider. One issue is the quality and completeness of the data being used for the analysis, which can affect the accuracy and reliability of the results [18]. Further research is needed to explore the business process improvement opportunities based on the effectiveness of different approaches to variant analysis.

4 Methodology

This section represents a description of the Design Science Research (DSR) Methodology, the justification for choosing it, and an overview of how this thesis implements the steps.

Our research design follows the iterative and cyclical approach of Design Science Research. This involves a continuous design, build, and application evaluation loop based on feedback and insights. The design objective is to create a user-friendly web application that assists users in uploading an event log, doing variant analysis, and comparing these variants.

There are several phases of DSR:

- **Problem Identification:** In this stage, we outline the research problem and the potential value of finding a solution. This thesis initiates the problem identification process within Section 1, where we establish the significance of waiting time analysis for business processes. Additionally, we recognize a gap in the existing literature and formulate the central research question.
- **Definition of Design Objectives:** After recognizing the gap outlined in Section 1, we establish specific criteria that our solution should meet:
  - Our approach should process event logs instead of relying solely on process models, indicating a data-driven orientation.
  - Our approach should systematically identify improvement opportunities by implementing variant analysis from waiting time perspective.
- **Design and Development:** Section 5 details the comprehensive concepts and procedures underlying our proposed approach, including its practical implementation.
- **Demonstration:** Within Section 6, we demonstrate the operational efficacy of our approach by applying it to address inefficiencies in a deliberately constructed business process. This involves generating a synthetic event log and utilizing our approach to identify improvement opportunities.
- **Evaluation:** The evaluation phase is extensively addressed in Section 6, where we assess the performance of our approach using a synthetic event log. Furthermore,

we benchmark our approach against an established baseline using pertinent metrics by applying it to a real-world event log.

- **Communication:** The task of effectively conveying the research problem, the derived solution, and the practical value and effectiveness of this solution to researchers and pertinent audiences is addressed in this public access manuscript. Additionally, the implementation's source code is publicly available on a GitHub repository[1][2], complete with installation instructions and guidance on usage.

Our research methodology combines the strengths of DSR with Agile principles to foster flexibility, collaboration, and continuous improvement. While DSR provides a structured framework for creating and evaluating the web application, Agile methodologies enhance the development process by emphasizing iterative cycles, adaptability, and engagement.

Agile implementation within DSR:

- **User Stories and Backlog:** Translate the design objectives into user stories that capture specific web application functionalities. These user stories form the product backlog, which serves as a dynamic list of features to be implemented.

- **Sprints:** Adopt Agile sprints, lasting 1-2 weeks, during which a subset of user stories is selected from the backlog. These user stories are then developed, tested, and integrated into the application.

- **Adaptive Planning:** At the end of each sprint, conduct a sprint review and retrospective. Reflect on what went well, what could be improved, and adjust the project plan accordingly.

To validate the effectiveness of this methodology, it will be important to thoroughly test the tool and software solution using a variety of event logs and process models. The results of this testing should be documented and analyzed to determine the accuracy and reliability of the methodology. Additionally, it will be important to consider any limitations or potential challenges that may arise during the implementation of this methodology.

5 Automated variant discovery, analysis, and improvement opportunities

This section represents an overview of the approach proposed in this thesis, then discusses every step in detail.

[1]https://github.com/mir-jalal/ava-core
[2]https://github.com/mir-jalal/ava-face

The overview of the approach consists of 4 steps. They are variant discovery, variant identification, variant analysis (comparison), and identification of improvement opportunities from variant analysis (comparison).

The Discovery phase is used to identify the most frequent paths through the business process by analyzing execution logs after discovering process variants. Clustering algorithms based on control flow will be implemented for variant identification. A comparison technique will be applied over variants to identify the best-performing variant.

Existing tools are used to collect event logs for variant discovery. The software solution takes event logs from the existing storage and clusters process models and stores. Then it compares identified process variants and determines which are meaningful and should be further analyzed. This may involve clustering variants based on common characteristics, such as control flow, performance, or other case attributes. Control flow will be the main characteristic of cluster traces for the implementation of the solution.

After that, metrics based on the time perspective will be derived from the analysis of the event logs. It will be listed all the identified differences with their values related to cycle time, process time, waiting time, and cycle-time efficiency. Table 2 describes the formulas for the given metrics and their components. The impact is assessed by these metrics and rank them by their impact on the process cycle-time-efficiency in descending order.

1. ***Variant Discovery***. We identify the clusters from the perspective of control flow. The input is an event log. This corresponds to step 1 in Figure 2.
   - **Input:** Event Log
   - **Output:** Identified clusters from the perspective of control flow
2. ***Variant Analysis***. We compute metrics given in Table 2. This corresponds to step 2 in Figure 2.
   - **Input:** Identified clusters from the perspective of control flow.
   - **Output:** The calculated metrics result from the analysis.
3. ***Suggesting Improvement Opportunities***. We compute improvement opportunities based on CTE impact. This corresponds to the step 3 in Figure 2.
   - **Input:** Identified clusters from the perspective of control flow.
   - **Output:** The calculated metrics as a result of the analysis.

Metric	Formula for the log with start and end timestamps
Cycle time (CT)	
CT of a trace	The time between the first start timestamp and the last end timestamp of the trace.
Total CT of a variant	The sum of CTs of the traces included in the variant.
Average CT of a variant	The sum of CTs of the traces included in the variant/number of traces in the variant.
Processing time (PT)	
PT of an activity execution	The duration between the start and end timestamps of the activity execution.
Total PT of an activity	The sum of PTs of all executions of the activity.
Average PT of an activity	The sum of PTs of all executions of the activity/number of this activity executions.
Total PT of a trace	The sum of PTs of all activity executions included in this trace.
Total PT of a variant	The sum of PTs of all traces included in the variant.
Average PT of a variant	The sum of PTs of all traces included in the variant/number of traces in the variant.
Cycle time efficiency (CTE)	
CTE of the process	Total PT / total CT of all traces of the process.
CTE of the variant	Total PT / total CT of all trances of the variant.
Frequencies	
The total frequency of a variant	The number of traces (process executions) included in the variant (e.g., variant 1 has 100 traces, variant 2 has 150 traces).
Case frequency of a variant	The number of traces included in the variant / total number of traces in the process (e.g., variant 1 has 100 traces, and there is 500 traces in the process: 100/500=0,2=20%).
The total frequency of the activity in the variant	The number of activity executions in the variant (e.g., activity "check order" was executed 100 times in variant 1)
The average frequency of the activity in the variant	The number of activity executions in the variant / the number of variant executions (e.g., activity "check order" is executed 1.5 times on average in variant 1)

Table 2. Metrics

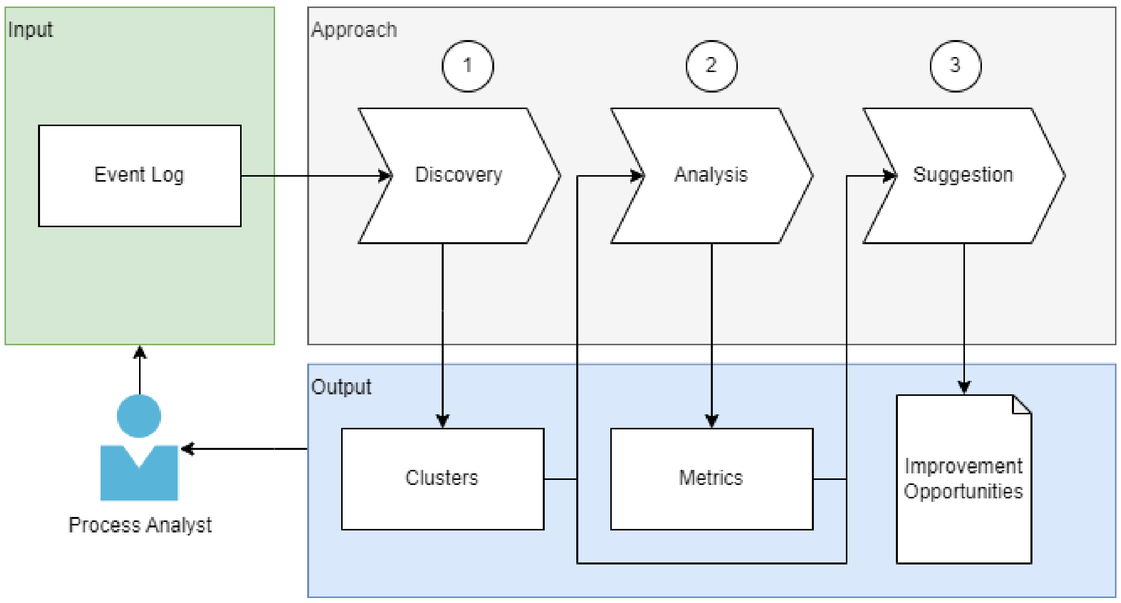


Figure 2. Overview of the proposed approach

5.1 Variant Discovery

It is the starting point of the application. The implementation phase serves as the foundation of the proposed application, and it begins by ingesting an event log. It's strictly required to have the following fields in the event log: case id, starting timestamp, end timestamp, resource, activity, and at least one cohort. The cohort field is required to compare the variants. These fields are vital for enabling comprehensive analysis and comparison of process variants.

A clustering technique was employed to facilitate meaningful analysis and uncover underlying patterns within the event data. The primary objective of clustering is to group similar instances together while distinguishing them from dissimilar ones. In the context of this research, trace clustering was adopted, specifically focusing on the control-flow perspective.

The control flow of a process refers to the sequence in which activities occur over time. In this context, the activities are extracted from the event log and ordered based on their timestamps. This time-ordered sequence of activities forms the basis of the control flow. However, the technique is modular and can be extended with any other clustering algorithms in the future. Various metrics or similarity measures, such as clustering-based on-time performance or more sophisticated control flow, like considering parallel relations, can be employed.

Once trace similarity is established, the clustering algorithm groups traces with high similarity into clusters. The result is a set of clusters containing traces that share similar control flows. This grouping enables the identification of distinct process variants.

Algorithm 2 shows overview of the algorithm for trace clustering:

1. The algorithm takes as input an event log (represented as a pandas DataFrame) and log identifiers (such as case IDs, activity names, start times, and end times).
2. The algorithm starts by obtaining the mapping from activities to characters using the `GetActivityMapping` function. This mapping is essential for creating a unique character sequence to represent each activity sequence.
3. The algorithm initializes an empty dictionary called `clusters` to hold the mapping from character sequences to case IDs that share the same activity sequence.
4. For each case in the event log, the algorithm iterates through the corresponding events in chronological order. It constructs an activity sequence by concatenating the characters corresponding to the activities using the activity-to-character mapping.
5. If the generated activity sequence is already a key in the `clusters` dictionary, the algorithm appends the current case ID to the list of case IDs associated with that sequence. Otherwise, a new entry is created in the `clusters` dictionary with the current case ID.
6. Once all cases have been processed, the algorithm iterates through each unique activity sequence in the `clusters` dictionary. For each cluster, it sets the `cluster_id` column of the corresponding cases in the event log to the current cluster ID.
7. Finally, the algorithm returns the modified event log, where each case is assigned a cluster ID based on its activity sequence.

As a result, we will obtain the list of process variants with different orders of activities. It's achieved by adding a new column called cluster-ID. This column serves as a key reference for subsequent analysis and visualization.

```
Algorithm 1: Variant Discovery from the perspective of control flow
  Data: Event log, filter cohort, and cohort values for filtering
  Result: A list of the clusters with different activities
1 $log \leftarrow identifyCohorts(log, filterCohort, filterValue)$;
2 $log \leftarrow discoverEnablementTimes(log)$;
3 $log \leftarrow clusterTraces(log)$;
```

Algorithm 2: Cluster Traces

```
Data: Event log: event_log, log identifiers: log_ids
Result: Event log with added cluster IDs
Input: Event log data and log identifiers;
Output: Event log with added cluster IDs;
Function GetActivityMapping(event_log, log_ids);
Function SortValues(column);
Function IsIn(values);
Function Loc(condition, column);
Function Return(result);
foreach case_id, events in event_log.GroupBy(log_ids.case_id) do
    sorted_events ← events.SortValues (log_ids.end_time, log_ids.start_time);
    activity_sequence ← "";
    foreach activity in sorted_events[log_ids.activity] do
        activity_sequence ← activity_sequence + mapping[activity];
    if activity_sequence in clusters then
        clusters[activity_sequence] += [case_id];
    else
        clusters[activity_sequence] ← [case_id];
cluster_id ← 0;
foreach cluster in clusters do
    event_log.Loc(event_log[log_ids.case_id].IsIn(clusters[cluster]),
     'cluster_id') ← cluster_id;
    cluster_id ← cluster_id + 1;
Return(event_log);
```

5.2 Variant Analysis

The next step involves performing a comprehensive analysis of these variants after applying the trace clustering methodology to the event log and identifying distinct process variants. This step takes the result of variant discovery from step 1 and computes the metrics given in Table 2.

These metrics collectively provide insights into various aspects of the process execution, such as time efficiency, resource allocation, and the occurrence of different process variants.

1. **Cycle Time (CT) Metrics**:
    - **Cycle time (CT) of a trace**: This metric represents the time a single process instance (trace) takes to complete, from its initial start timestamp to its final end timestamp.
    - **Total CT of a variant**: The sum of the cycle times for all traces included in a specific process variant. It reflects the overall time taken for the entire execution of that variant.
    - **Average CT of a variant**: Calculated by dividing the sum of cycle times of all traces in a variant by the number of traces in that variant. It provides an average duration of execution for that variant.

2. **Processing Time (PT) Metrics**:
    - **PT of an activity execution**: This metric represents the duration between the start and end timestamps of a single execution of an activity.
    - **Total PT of an activity**: The sum of processing times for all executions of a particular activity. It gives an idea of the cumulative time spent on that activity across all traces.
    - **Average PT of an activity**: Calculated by dividing the sum of processing times for all executions of an activity by the total number of executions of that activity. It provides an average time taken for a single execution of that activity.
    - **Total PT of a trace**: The sum of processing times for all activity executions within a trace. It measures the cumulative processing time for the entire trace.
    - **Total PT of a variant**: The sum of processing times for all traces included in a specific process variant. It reflects the overall processing time for the entire execution of that variant.
    - **Average PT of a variant**: Calculated by dividing the sum of processing times of all traces in a variant by the number of traces in that variant. It provides an average processing time for that variant.

3. **Cycle Time Efficiency (CTE) Metrics**:
    - **CTE of the process**: Calculated as the total processing time divided by the total cycle time of all traces in the process. It provides an efficiency measure of how much processing time is spent relative to the overall cycle time.
    - **CTE of the variant**: Similar to the process-level CTE, this metric calculates the efficiency of a specific variant by dividing the total processing time by the total cycle time of all traces within that variant.
4. **Frequency Metrics**:
    - **Total frequency of a variant**: The number of traces (process executions) included in a specific variant. It indicates how frequently that variant occurs in the dataset.
    - **Case frequency of a variant**: Calculated by dividing the number of traces in a variant by the total number of traces in the entire process. It represents the proportion of cases that belong to that variant.
    - **Total frequency of the activity in the variant**: The total number of times a specific activity is executed within a variant. It shows how often that activity is performed in the context of that variant.
    - **Average frequency of the activity in the variant**: Calculated by dividing the number of activity executions in the variant by the number of variant executions. It provides an average occurrence rate of that activity within the variant.

5.3 Improvement Opportunities

Building upon the insights obtained from the variant analysis metrics in the previous section, the "Improvement Opportunities" phase delves into identifying specific areas within the process where optimization efforts can lead to enhanced performance and efficiency. This step involves assessing the impact of waiting time on the Cycle Time Efficiency (CTE) metrics to uncover potential improvements that could result in more streamlined process execution.

The waiting time for activities within a process plays a pivotal role in the overall cycle time and subsequently influences the efficiency of the process. Waiting time represents when an activity is not actively being executed due to dependencies, resource availability, or other factors.

The impact of waiting time is measured as follows:

- For each activity within the process, we identify instances where waiting time occurs. This can be determined by analyzing the time gaps between the end timestamp of one activity and the start timestamp of its subsequent activity within the same trace.

- For each activity, we compute the impact of waiting time on the CTE metrics. This involves simulating a reduction in waiting time and calculating the resulting change in CTE for the affected variant.
- We calculate the difference in CTE before and after the reduction in waiting time for each variant. This provides a quantifiable measure of how improvements in waiting time can affect the overall process efficiency.

Based on the computed CTE impact, you can identify improvement opportunities that target reducing activity waiting time. Opportunities are prioritized based on the magnitude of CTE improvement.

6 Implementation

In the implementation phase of our approach, we've integrated two integral components: the backend and the frontend. This division facilitates a clear separation of concerns, enabling efficient data processing and user interaction management.

6.1 Backend

The backend is the computational powerhouse that drives the analysis, computations, and data processing. To construct this backbone, we've employed a range of technologies, including Python[3], Flask[4], Celery[5], Pandas[6], and NumPy[7].

- **Python:** The programming language forms the foundation of our backend. Its versatility and extensive libraries provide a solid framework for data manipulation, calculations, and logical operations.
- **Flask:** A lightweight and flexible web framework, Flask seamlessly handles the server-side operations. It allows us to define routes, endpoints, and APIs, facilitating communication between the frontend and backend. Flask's simplicity aligns well with our goal of delivering efficient data processing.

[3]https://www.python.org/
[4]https://flask.palletsprojects.com/en/2.3.x/
[5]https://docs.celeryq.dev/en/stable/getting-started/introduction.html
[6]https://pandas.pydata.org/
[7]https://numpy.org/

- **Celery:** A prominent feature in our backend architecture, Celery facilitates the execution of time-consuming or computationally intensive tasks asynchronously. It operates in a distributed and parallel manner, allowing us to perform multiple tasks concurrently without blocking the main application thread. This is particularly crucial for heavy-duty data analysis and computations.

- **Pandas:** The Pandas library empowers us to manage and manipulate large datasets with ease. Its DataFrame structure simplifies data organization, aggregation, and filtering, which are crucial steps in our analysis pipeline.

- **Numpy:** With its efficient numerical computations, NumPy contributes to the backend's computational efficiency. We can perform mathematical operations and array manipulations with remarkable speed, enhancing our approach's overall performance.

6.2 Frontend

To interact with the implementation of our approach easily, we provide a basic user interface[8] built-in React[9] open-source library for user interfaces. In building the front end, we leverage additional technologies alongside the React open-source library for enhanced user experience.

The web application is deployed in `compare.cloud.ut.ee`

- **Material-UI:**[10] We employ the Material-UI library to ensure a visually appealing and responsive design. Material-UI provides pre-designed components, typography, and theming, allowing us to construct an aesthetically pleasing user experience while maintaining consistency throughout the application.

- **Highcharts:**[11]For data visualization, Highcharts comes into play. This versatile JavaScript charting library assists us in creating interactive and insightful graphical representations of the analysis results. We will use Highcharts demonstrate computed metrics, so users can gain a deeper understanding of the processed data.

- **Cytoscape:**[12] As a graph theory library, Cytoscape enhances the user's ability to visualize and explore complex relationships within the data. Its functionalities aid in presenting intricate networks and hierarchies in a comprehensible manner. We use Cytoscape to visualize process maps.

[8]https://compare.cloud.ut.ee
[9]https://react.dev/
[10]https://mui.com/
[11]https://www.highcharts.com/
[12]https://js.cytoscape.org/

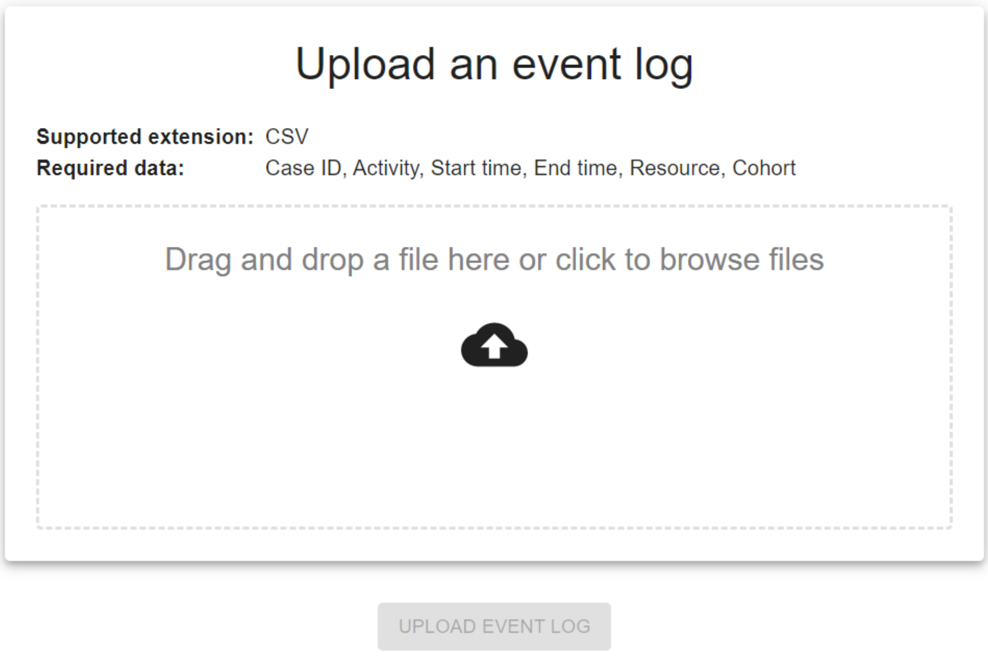
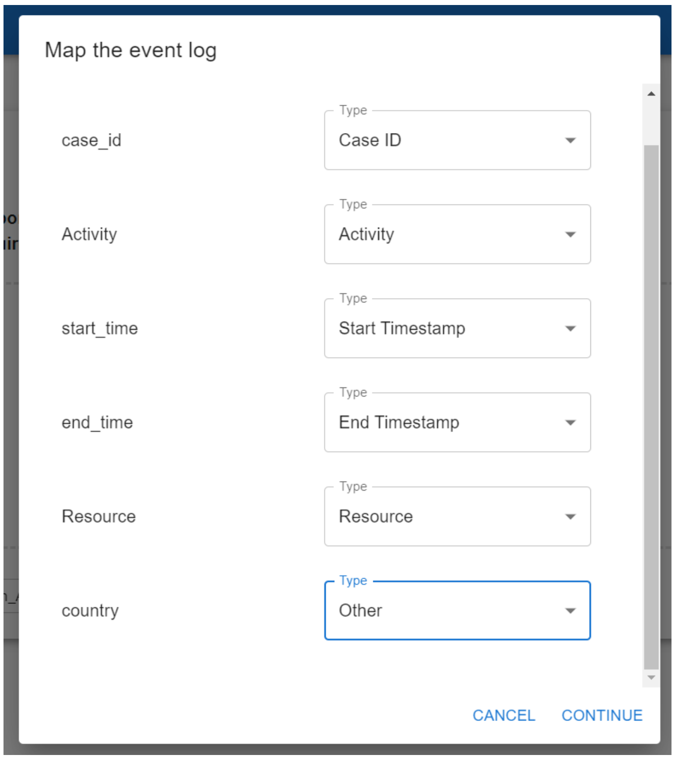

Upload an event log
Supported extension: CSV
Required data: Case ID, Activity, Start time, End time, Resource, Cohort
Drag and drop a file here or click to browse files
UPLOAD EVENT LOG

Figure 3. Graphical Interface of the approach implementation

The first step in the user interface is the "Upload Page." This is where the user can upload an event log file. Once the user uploads the event log file, they can initiate the application by clicking on the "Upload Event Log" button as shown in Figure 3. This action will trigger the back end to process the uploaded log and start the analysis.

After the event log is uploaded and the application has started, the user will be directed to the "Cohort Selection" page. The user needs to specify a "cohort field" and its corresponding values as shown in Figure 4. A cohort field is a data column or attribute in the event log that categorizes the data into two variants based on some criteria. For example, it could be a field that classifies events by location, department, user type, or any other relevant grouping.

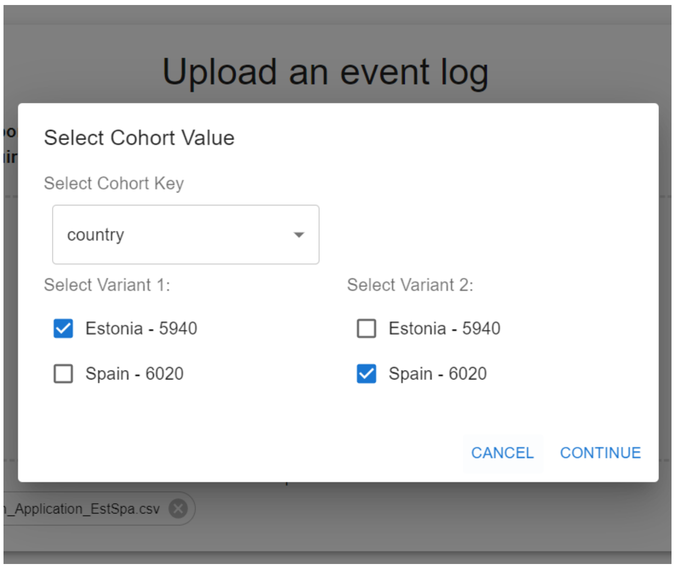
Once the user selects the cohort field and its values, they proceed to the next step where the back-end takes over. The application will start processing the event log data, performing various analyses and calculations based on the selected cohort(s) and any other relevant parameters. This analysis is done on the server-side (back-end) to offload the computational load from the user's device. As soon as the back-end analysis is completed, the user interface will display the "Dashboard" page.



Map the event log

case_id	Type: Case ID
Activity	Type: Activity
start_time	Type: Start Timestamp
end_time	Type: End Timestamp
Resource	Type: Resource
country	Type: Other

Figure 4. Graphical Interface of the Field Selection



Upload an event log

Select Cohort Value

Select Cohort Key: country

Select Variant 1:

- [x] Estonia - 5940
- [ ] Spain - 6020

Select Variant 2:

- [ ] Estonia - 5940
- [x] Spain - 6020

Figure 5. Graphical Interface of the Cohort Selection

Log Name	Process Type	Cases
Synthetic Event Log	Loan Application	2000
Real World Event Log	Device Repair	1142

Table 3. Summary of the event logs used for evaluation

7 Evaluation

This section covers demonstration and evaluation phases of the Design Science Research thesis. First we start introducing event logs used in this thesis, then we look at demonstration of the results in details. We begin with synthetic event log to show the approach meets the Design Objectives. Then we apply the approach to a real-world event log.

7.1 Datasets and Features

7.1.1 Synthetic event log

We built a hypothetical Load Application process[13]. Table 4 shows the description of the event log format. Each row in the event log corresponds to a specific event, which can be considered an occurrence or action associated with a particular case or instance. In the provided event log, each event is characterized by several attributes.

- **case_id:** This attribute identifies the unique case or instance to which an event belongs. It groups together all the events related to a particular process instance. For example, case 0 represents a specific process.
- **activity:** The "Activity" attribute describes the nature of the event, indicating the action or step that occurred within the process. For instance, "Check application form completeness" and "AML Check" are examples of activities in the event log.
- **start_time and end_time:** These attributes represent the timestamps when an event started and ended, respectively. They provide crucial information about the temporal aspect of the process. For instance, the "Check application form completeness" event for case 0 started at 2017-01-02 09:00:00 and ended at 2017-01-02 09:21:18.882.
- **resource:** The "Resource" attribute identifies the entity responsible for carrying out the event. This could be an individual, role, or department involved in the process. For instance, "Clerk-000001" and "Loan Officer-000001" represent specific resources responsible for different stages of the process.

[13]https://gist.github.com/mir-jalal/66d524de9bc6b34852afd87400a10db2

Column name	Type	Note
case_id	number	
activity	string	
start_time	timestamp	
end_time	timestamp	
resource	string	
country	string	cohort

Table 4. Summary of the synthetic event log used for evaluation

- **country:** This attribute denotes the country associated with the event. It provides information about where the process is taking place. This will be the cohort column for the synthetic event log.

Each row in the event log encapsulates the details of a single event, capturing its timing, the activity it represents, the involved resource, and the corresponding case. This granular representation of events allows for detailed process analysis, such as identifying bottlenecks, analyzing resource allocation, and understanding the sequence of activities.

The provided example illustrates various events related to loan application processing. For instance, case 0 goes through multiple stages, including a form completeness check, AML check, risk assessment, loan offer design, loan approval, and more. These events collectively represent the journey of processing a loan application.

7.1.2 Real world event log

The second experiment will be conducted on real-world event log[14]. Table 5 shows the description of the event log format. Each row in the event log corresponds to a specific event, which can be considered an occurrence or action associated with a particular case or instance. In the provided event log, each event is characterized by several attributes.

- **creator:** This column specifies the entity or tool that created the event.
- **variant:** The "variant" column refers to a particular variant or version of the process associated with the event. This information can be used to differentiate between different process paths or scenarios.
- **variant_index:** This column provides an index or identifier associated with the specific variant.
- **defectFixed:** This column indicates whether a defect was fixed during the event. It could have values like "true" or "false," indicating whether a defect was addressed.

[14]https://gist.github.com/mir-jalal/86e286a5d310c0ae10a285d9476b6e1d

Column name	Type	Note
case_id	number	
activity	string	
start_time	timestamp	
end_time	timestamp	
resource	string	
variant	string	cohort
variant_index	number	
defectFixed	bool	
defectType	string	
numberRepairs	number	
phoneType	string	

Table 5. Summary of the real world event log used for evaluation

- **defectType:** This column specifies the type of defect that was addressed or analyzed during the event.
- **numberRepairs:** This column represents the number of repairs or fixes associated with the event. It indicates how many times a repair occurred within this event.
- **phoneType:** This column indicates the type of phone or device associated with the event.

Table 5 shows the columns that we have as a real world event log.

7.2 Experiment 1: Demonstration of approach with synthetic event log

We conducted a cohort analysis on the synthetic event log, categorizing cases based on the country of origin: Estonia and Spain.

7.2.1 Results

The metrics obtained are shown in Table 6:

The cycle time efficiency was calculated for each cohort. The results are as follows:

- Cycle Time Efficiency for Estonia: 5.77
- Cycle Time Efficiency for Spain: 5.25

Metric	**Estonia**	**Spain**
Cases	1000	1000
Activity Instances	5940	6020
Unique Activities	10	10
Transition Instances	4940	5020
Unique Transitions	9	9

Table 6. Cohort Analysis Metrics

These results indicate that cases from Estonia tend to exhibit slightly higher cycle time efficiency than those from Spain.

Table 7 summarizes the waiting time causes identified during the analysis:

Waiting Time Cause	**Estonia**	**Spain**
batching	4h 50m	7h 29m
prioritization	0	0
contention	5h 36m	10h 42m
unavailability	15m	17m
extraneous	4m	5m

Table 7. Waiting Time Causes

Figure 6 demonstrates the result of waiting time analysis on the user interface. It indicates that Spain experiences longer waiting times attributed to resource contention. This suggests that activities often wait due to resources being occupied with other tasks. This observation implies that the Estonia division either faces a lower workload, benefits from more skilled and efficient workers, or has a higher workforce count. This presents an avenue for analysis to enhance the Spain division's performance by addressing factors that lead to resource contention and exploring strategies to improve efficiency.

The process mapping dashboard is a visual aid to complement the obtained metrics and insights. It enables users to interactively explore the process flow, activities, and transitions for both the Estonia and Spain cohorts. The dashboard allows users to zoom in on specific process segments, highlight critical paths, and identify potential bottlenecks. Figure 7 and Figure 8 represent Process Mappings on the user interface for Estonia and Spain, respectively.

To gain deeper insights into the transitions, we demonstrate detailed analysis of individual transitions per cohort within the event log. Figure 9 presents each transition's average duration, relative transition frequency, and Cycle Time Efficiency (CTE) impact. The table is sorted in descending order of CTE impact, highlighting the transitions with the most significant influence on cycle times.

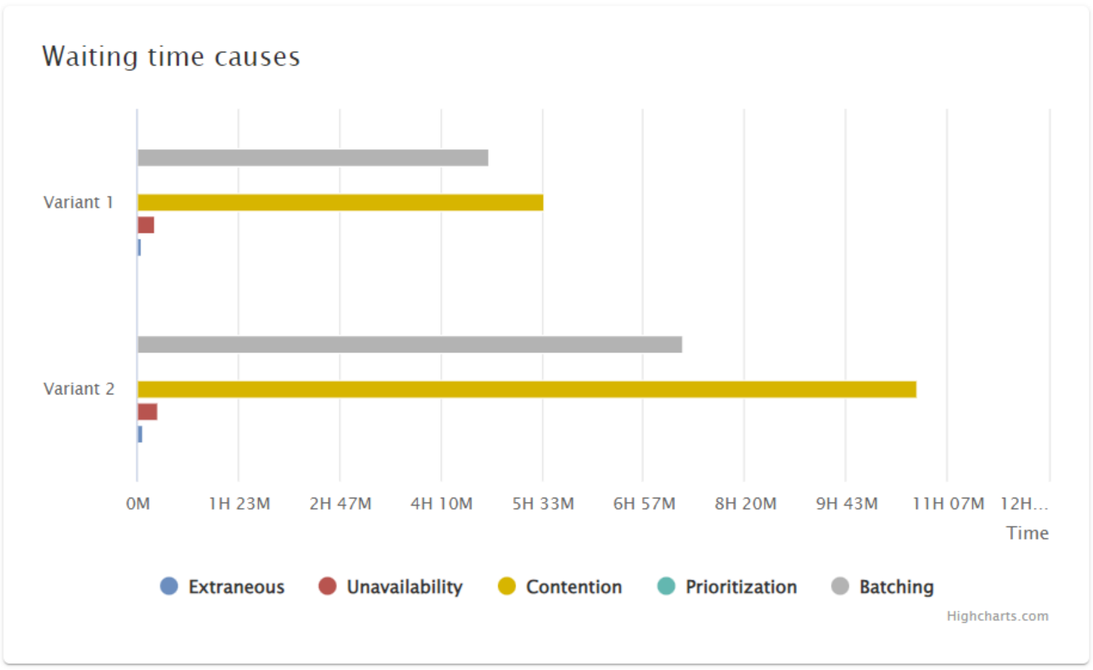


Figure 6. Bar chart of waiting time analysis for synthetic event log

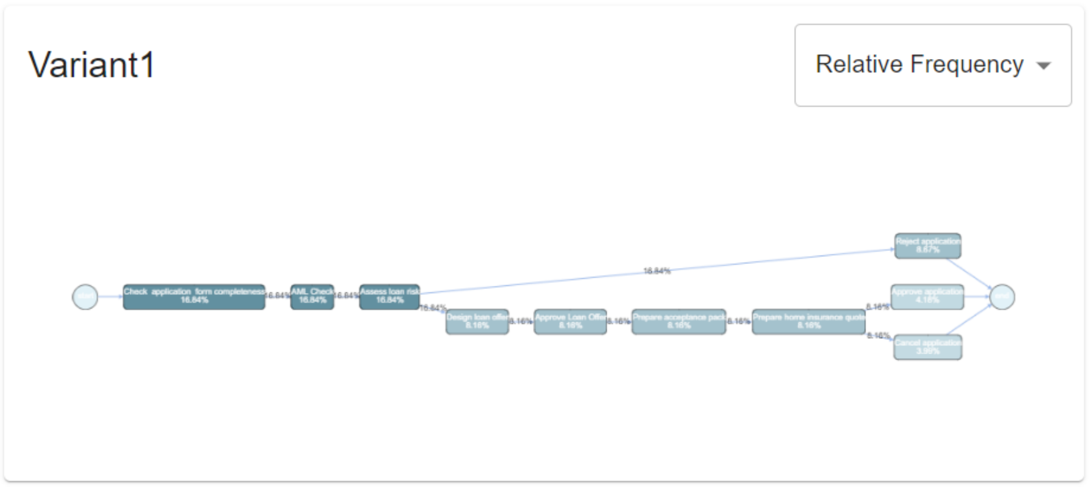


Figure 7. Process Mapping for Estonia of synthetic event log

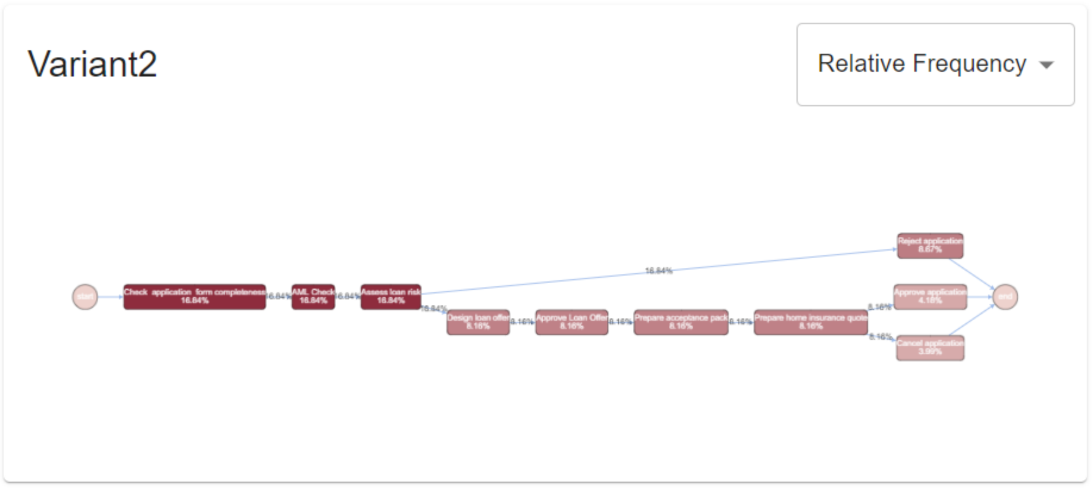
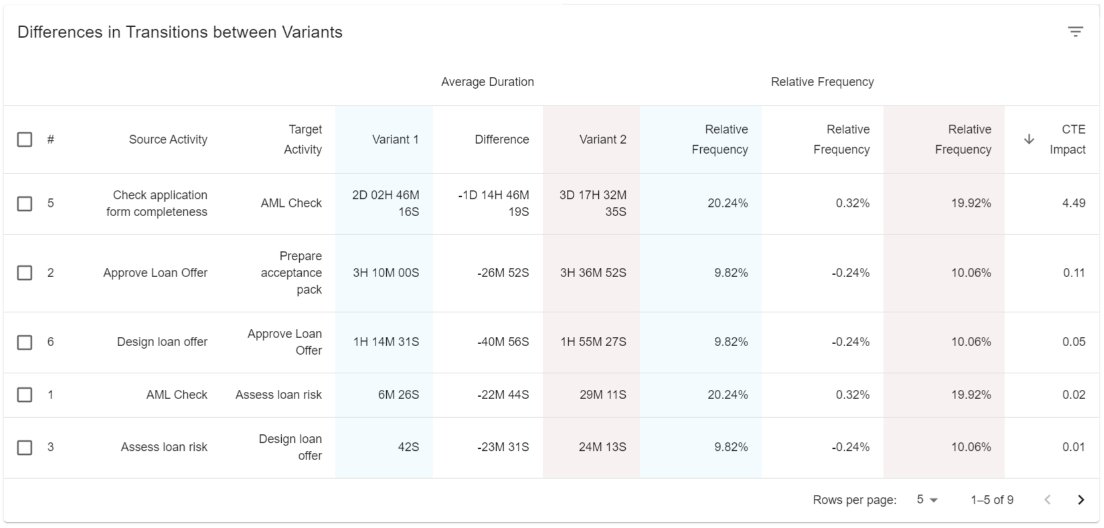


Figure 8. Process Mapping for Spain of synthetic event log


Differences in Transitions between Variants

	#	Source Activity	Target Activity	Average Duration			Relative Frequency			CTE Impact
				Variant 1	Difference	Variant 2	Relative Frequency	Relative Frequency	Relative Frequency ↓	
☐	5	Check application form completeness	AML Check	2D 02H 46M 16S	-1D 14H 46M 19S	3D 17H 32M 35S	20.24%	0.32%	19.92%	4.49
☐	2	Approve Loan Offer	Prepare acceptance pack	3H 10M 00S	-26M 52S	3H 36M 52S	9.82%	-0.24%	10.06%	0.11
☐	6	Design loan offer	Approve Loan Offer	1H 14M 31S	-40M 56S	1H 55M 27S	9.82%	-0.24%	10.06%	0.05
☐	1	AML Check	Assess loan risk	6M 26S	-22M 44S	29M 11S	20.24%	0.32%	19.92%	0.02
☐	3	Assess loan risk	Design loan offer	42S	-23M 31S	24M 13S	9.82%	-0.24%	10.06%	0.01

Rows per page: 5 1–5 of 9

Figure 9. Transition difference table of synthetic event log

7.2.2 Discussion

The analysis of the provided event log has revealed insights into the process behavior and efficiency of both the Estonia and Spain cohorts. The results obtained from the cohort analysis, cycle time efficiency calculation, and waiting time causes provide important information for validating process dynamics and identifying potential areas for improvement.

The cycle time efficiency results demonstrate that both Estonia and Spain cohorts exhibit commendable cycle time efficiencies, with Estonia's cases slightly outperforming those from Spain. This difference could arise due to process optimizations, resource availability, or varying process complexities in the respective cohorts. It's worth exploring the contributing factors behind the higher efficiency of cases from Estonia, as these insights could inform process enhancement strategies that can be applied more broadly.

The waiting time causes analysis, as shown in Table 7, unveils intriguing patterns in both the Estonia and Spain cohorts. Notably, the most substantial causes of waiting time are "contention" and "batching," which are prominent in both cohorts. These findings suggest potential areas for optimization, such as refining resource allocation and streamlining processes to mitigate contention-related delays. Additionally, the shorter waiting times due to "unavailability" and "extraneous" factors in Estonia compared to Spain could be indicative of better resource management practices in the former.

The findings suggest that the process in the Estonia cohort demonstrates better performance and optimization compared to Spain. This realization opens avenues for potential process improvements by emulating certain practices observed in Estonia.

In conclusion, this experiment sheds light on the performance and efficiency of two cohorts within the process. The differences in cycle time efficiency and waiting time causes between Estonia and Spain cohorts provide a foundation for focused process optimization efforts. By leveraging these insights, organizations can work towards enhancing their process performance and delivering more streamlined services. The computed improvement opportunities were found to be accurate, affirming the correctness of the approach.

7.3 Experiment 2: Demonstration of approach with real world event log

For this experiment we will use "Device Repair" event log. "Variant" column has been selected as cohort value. ["Variant 11", "Variant 10", "Variant 12", "Variant 13", "Variant 3"] are selected as the first cohort value. ["Variant 1", "Variant 41", "Variant 40", "Variant 4", "Variant 39"] are selected as the second cohort value.

7.3.1 Results

Some of the metrics obtained are shown in Table 8

Metric	**Variant 1**	**Variant 2**
Cases	179	337
Activity Instances	1075	2064
Unique Activities	8	7
Transition Instances	896	1721
Unique Transitions	10	9

Table 8. Cohort Analysis Metrics

The cycle time efficiency was calculated for each cohort. The results are as follows:

- Cycle Time Efficiency for Variant 1: 1.67
- Cycle Time Efficiency for Variant 2: 1.56

Table 9 summarizes the waiting time causes identified during the analysis:

Waiting Time Cause	**Variant 1**	**Variant 2**
batching	0	0
prioritization	0	0
contention	3s	10s
unavailability	2m 17s	2m 6s
extraneous	4m 58s	7m 11s

Table 9. Waiting Time Causes

Figure 10 demonstrates the result of waiting time analysis on the user interface.

Figure 11 and Figure 12 show the representation of Process Mappings on the user interface for Estonia and Spain, respectively.

Figure 13 presents each transition's average duration, relative transition frequency, and Cycle Time Efficiency (CTE) impact. The table is sorted in descending order of CTE impact, highlighting the transitions with the most significant influence on cycle times.

7.3.2 Discussion

In real-world data analysis, we examined the performance metrics of two different process variants, Variant 1 and Variant 2. As demonstrated in Table 8, we quantified various key metrics to gain insights into their process characteristics.

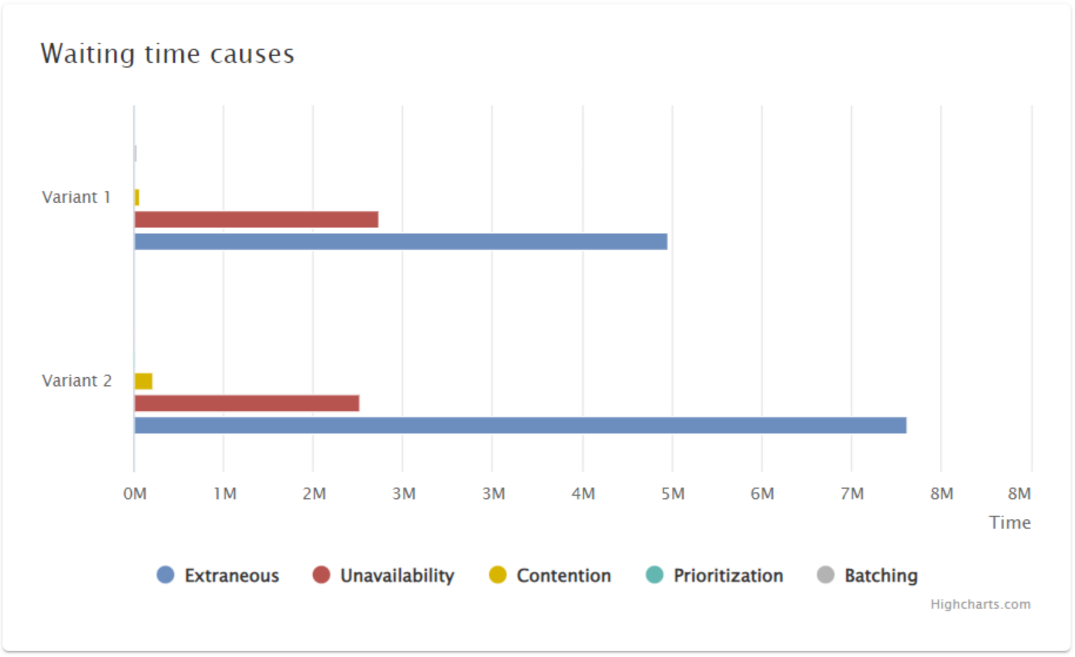


Figure 10. Bar chart of waiting time analysis for real-world event log

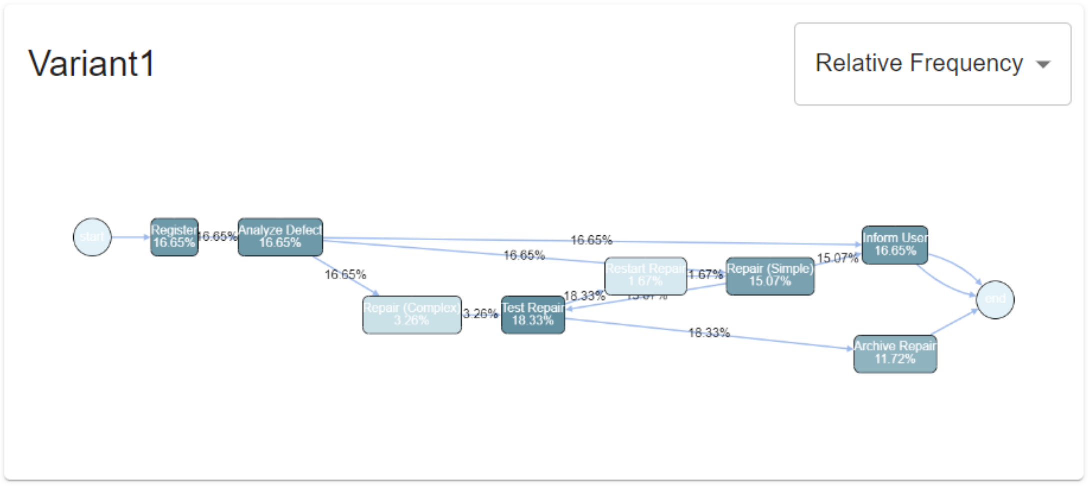


Figure 11. Process Mapping for Variant 1 of real-world event log

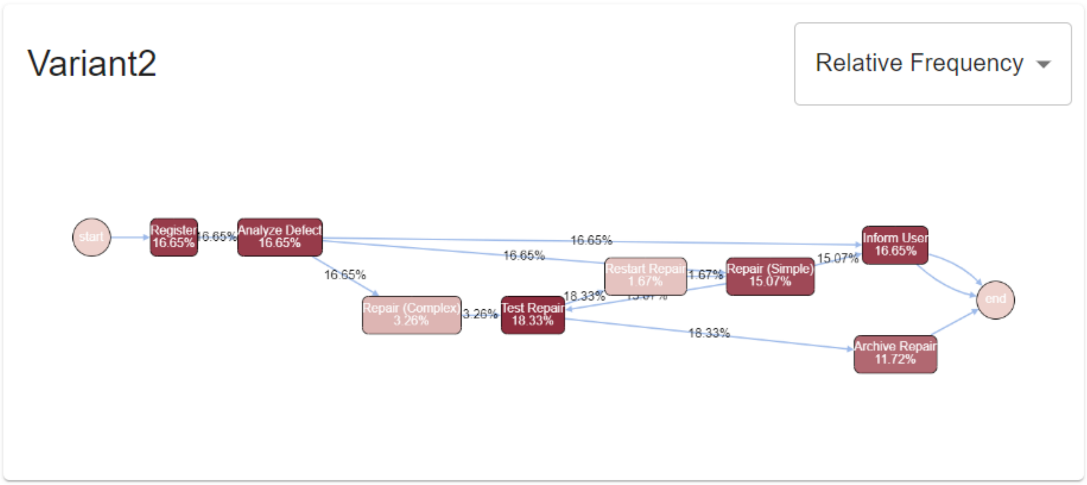
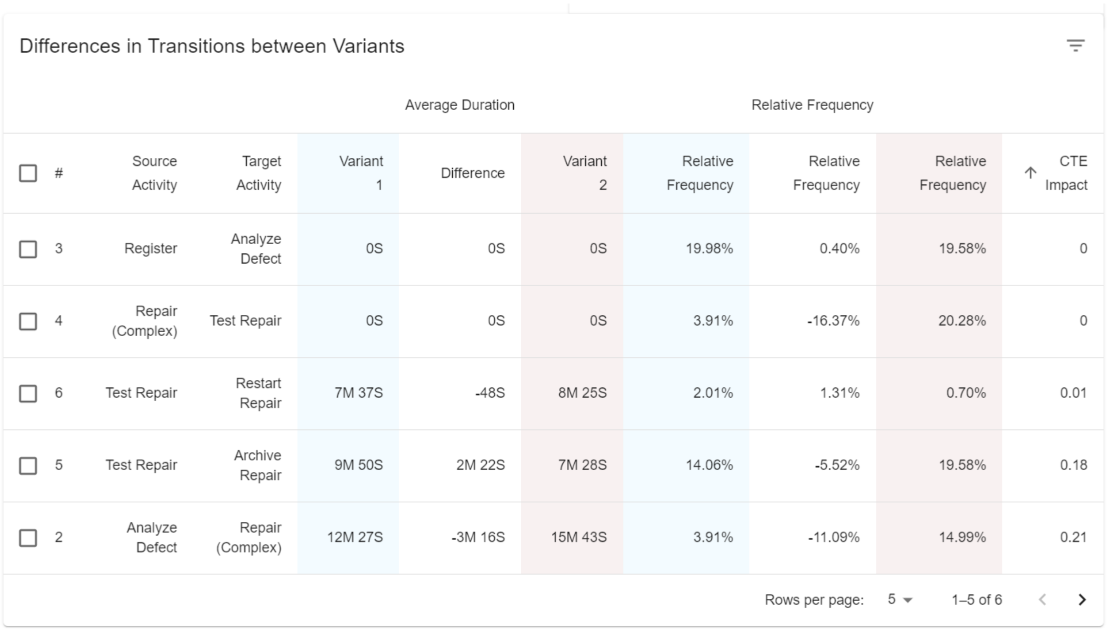


Figure 12. Process Mapping for Variant 2 of real-world event log



Differences in Transitions between Variants

				Average Duration			Relative Frequency			
☐	#	Source Activity	Target Activity	Variant 1	Difference	Variant 2	Relative Frequency	Relative Frequency	Relative Frequency	↑ CTE Impact
☐	3	Register	Analyze Defect	0S	0S	0S	19.98%	0.40%	19.58%	0
☐	4	Repair (Complex)	Test Repair	0S	0S	0S	3.91%	-16.37%	20.28%	0
☐	6	Test Repair	Restart Repair	7M 37S	-48S	8M 25S	2.01%	1.31%	0.70%	0.01
☐	5	Test Repair	Archive Repair	9M 50S	2M 22S	7M 28S	14.06%	-5.52%	19.58%	0.18
☐	2	Analyze Defect	Repair (Complex)	12M 27S	-3M 16S	15M 43S	3.91%	-11.09%	14.99%	0.21

Rows per page: 5 1–5 of 6

Figure 13. Transition difference table of real-world event log

The metrics outlined in Table 8 provide a snapshot of the process behaviors in both variants. Notably, the number of cases, activity instances, and transition instances significantly differ between the two variants. This disparity suggests potential variations in the complexity and scale of the processes represented by each variant. Additionally, each variant's unique activities and transitions offer a glimpse into the distinct process flows they encompass.

To further understand the efficiency of these process variants, we calculated the cycle time efficiency for both Variant 1 and Variant 2.

These efficiency metrics provide insights into how promptly cases are processed within each variant. The relatively close values suggest that both variants exhibit a similar level of cycle time efficiency, despite the differences in their process metrics.

In both variants, the absence of "batching" and "prioritization" delays is noticeable, indicating efficient resource allocation and task scheduling. However, "contention" appears to be a significant waiting time cause in Variant 2, with a longer duration than Variant 1. This suggests that resource conflicts or contention may be more prevalent in Variant 2, leading to delays in task execution.

Furthermore, "unavailability" and "extraneous" causes demonstrate varying impact duration between the two variants. These insights are crucial for understanding the reasons behind waiting time and identifying opportunities for process optimization.

In conclusion, the metrics and results presented in the tables offer a comprehensive view of the process characteristics, cycle time efficiency, and waiting time causes for both process variants.

This showed that our approach to variant analysis provides a contextually rich interpretation of the metrics and results presented in the tables, helping users find design changes for improvement opportunities.

8 Conclusion and Future Work

This thesis addressed the automated variant analysis for business process improvement. In other words, it aimed to solve the challenge of variant analysis through the development and validation of an automated framework.

We observed that there are many techniques to create process models, to do clustering, however many of them are not implemented. Therefore, our approach works with event logs and it's based on clustering from the perspective of control flow. The fulfillment of the research goal was verified through experiments in Section 7.

As future work, our approach could be extended by integrating more clustering techniques, and also more visualization could be added. The approach was modular so it could also be extended by integrating with different applications.

As we conclude this thesis, we look ahead with optimism, acknowledging that the contributions made here are a stepping stone toward a broader and more impactful exploration of automated variant analysis.

References

[1] DUMAS, M., LA ROSA, M., MENDLING, J., AND REIJERS, H. A. Fundamentals of Business Process Management, 2nd ed. 2018 ed. Springer Berlin Heidelberg : Imprint: Springer, Berlin, Heidelberg, 2018.

[2] (2007). Business Process Management Architectures. In: Business Process Management. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-73522-9_7

[3] Luengo, D., Sepúlveda, M. (2012). Applying Clustering in Process Mining to Find Different Versions of a Business Process That Changes over Time. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds) Business Process Management Workshops. BPM 2011. Lecture Notes in Business Information Processing, vol 99. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-28108-2_15

[4] van der Aalst, W. (2016). Process Mining: The Missing Link. In: Process Mining. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-49851-4_2

[5] Taymouri, F., La Rosa, M., Dumas, M., Maggi, F.M.: Business process variant analysis: Survey and classification. Knowledge-Based Systems 211, 106557 (2021)

[6] N. M. b. Mahmod and S. b. Ahmad Radzi, "An approach to analyse similarity of business process variants," 2010 IEEE International Conference on Progress in Informatics and Computing, 2010, pp. 640-644, doi: 10.1109/PIC.2010.5687872.

[7] Using Multi-Level Information in Hierarchical Process Mining: Balancing Behavioural Quality and Model Complexity

[8] Analysis on Online Learning Environment using Process Mining Technique for Personal Knowledge Management Mapping

[9] W.M.P. van der Aalst, A.J.M.M. Weijters, Process Mining: A Research Agenda, Computers in Industry, 53(3):231-244, (2004)

[10] Nakatumba, J.: Resource-aware business process management: analysis and support. Ph.D. thesis, Mathematics and Computer Science (2013)

[11] Ariouat, H., Barkaoui, K., Akoka, J.: Improving process models discovery using axor clustering algorithm. In: Information Science and Applications, pp. 623–629. Springer (2015)

[12] Bose, R., van der Aalst, W.M.: Trace clustering based on conserved patterns: Towards achieving better process models. In: International Conference on Business Process Management. pp. 170–181. Springer (2009)

[13] Luengo, D., Sep ´ulveda, M.: Applying clustering in process mining to find different versions of a business process that changes over time. In: International Conference on Business Process Management. pp. 153–158. Springer (2011)

[14] T. A. Wiggerts, "Using clustering algorithms in legacy systems remodularization," Proceedings of the Fourth Working Conference on Reverse Engineering, 1997, pp. 33-43, doi: 10.1109/WCRE.1997.624574.

[15] V. P. Mishra, J. Dsouza and L. Elizabeth, "Analysis and Comparison of Process Mining Algorithms with Application of Process Mining in Intrusion Detection System," 2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), 2018, pp. 613-617, doi: 10.1109/ICRITO.2018.8748748.

[16] Dijkman, R. M., Song, M., van der Aalst, W. M. P., & Verbeek, H. M. W. (2013). A framework for identifying process improvement opportunities based on process variants. Information Systems and e-Business Management, 11(3), 365-391.

[17] Özcan, E., Dumas, M., & ter Hofstede, A. H. M. (2018). A framework for process optimization based on case attributes. Information Systems and e-Business Management, 16(1), 1-29.

[18] Van der Aalst, W. M. P., van der Aalst, W. M. P., & van der Aalst, W. M. P. (2004). Process mining: discovery, conformance, and enhancement of business processes. Springer Science & Business Media.