DISSERTATIONES
INFORMATICAE
UNIVERSITATIS
TARTUENSIS
62

KATSIARYNA LASHKEVICH

Data-Driven Analysis and Optimization of Waiting Times in Business Processes

TARTU ÜLIKOOL
UNIVERSITAS TARTUENSIS
1632

DISSERTATIONES INFORMATICAE UNIVERSITATIS TARTUENSIS
62

DISSERTATIONES INFORMATICAE UNIVERSITATIS TARTUENSIS

62

KATSIARYNA LASHKEVICH

Data-Driven Analysis and Optimization of Waiting Times in Business Processes

UNIVERSITY OF TARTU
Press
1632

Institute of Computer Science, Faculty of Science and Technology, University of Tartu, Estonia.

Dissertation has been accepted for the commencement of the degree of Doctor of Philosophy (PhD) in Computer Science on December 3, 2024 by the Council of the Institute of Computer Science, University of Tartu.

Supervisors

Assoc. Prof.	Fredrik Payman Milani
	University of Tartu, Estonia
Prof.	Marlon Gerardo Dumas Menjivar
	University of Tartu, Estonia

Opponents

Prof.	Luise Pufahl
	Technical University of Munich, Germany
Prof.	Remco Dijkman
	Eindhoven University of Technology, the Netherlands

The public defense will take place on January 10, 2025 at 10:15 in Narva Rd. 18-2048.

The publication of this dissertation was financed by the Institute of Computer Science, University of Tartu.

ISSN 2613-5906 (print) ISSN 2806-2345 (pdf)
ISBN 978-9916-27-772-0 (print) ISBN 978-9916-27-773-7 (pdf)

Copyright © 2024 by Katsiaryna Lashkevich

University of Tartu Press
http://www.tyk.ee/

To my husband,
with all my love and gratitude

ABSTRACT

In the context of a business process, a waiting time is an idle period of time typically occurring during a transition between two consecutive activities. Waiting time is considered to be a form of waste in a business process, which should be minimized or eliminated, where possible. Accordingly, there are several methods to identify and quantify waiting times for the purpose of business process improvement. However, these existing methods fall short of attributing the identified waiting times to specific causes, thus limiting the ability of the users of these methods to translate the findings into possible business process redesigns. This thesis addresses this gap by exploring how to identify the causes of waiting times from event logs of business process executions, and how to use this information to recommend redesigns that effectively reduce these waiting times.

The research reported in this thesis is structured around two key questions: (1) How to identify causes of waiting times from event logs? and (2) How to identify redesign options to reduce the identified waiting times? The thesis reports on three research contributions aimed at addressing these questions.

First, the thesis proposes a process mining-based approach to discover and analyze waiting times associated with batch processing. This approach conceptualizes three distinct types of waiting times resulting from different batching strategies and provides a method to identify these types of waiting times from event logs and highlight potential redesign options.

Second, the thesis extends the previous method to diagnose multiple causes of waiting times across all activity transitions within a process. Specifically, the thesis proposes an approach to decompose the waiting time observed in each activity transition of a process into five direct causes: batching, resource contention, prioritization, resource unavailability, and extraneous factors. Compared to previous approaches, this decomposition enables a more detailed analysis of waiting time inefficiencies and provides a structured approach to identify and prioritize opportunities to address these inefficiencies.

Third, to assist business analysts during process redesign, the thesis proposes a prompting method for fine-tuning a Large Language Model (LLM), enabling it to analyze the causes of waiting times in a business process and to recommend redesign options. This method leverages the previously proposed decomposition of waiting time into five direct causes. It also integrates a structured collection of redesign patterns, to enable the LLM to provide tailored and actionable recommendations to reduce waiting times.

The contributions of the thesis have been implemented in a research prototype, namely Kronos. This prototype has been used to evaluate the proposed methods via computational experiments and user studies. The findings show that the proposed methods help business analysts in diagnosing waiting times and identifying corresponding redesign options.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Acronyms

AI Artificial Intelligence. 15, 36

BPI Business Process Improvement. 9, 52–54
BPM Business Process Management. 9, 23, 24, 29, 31, 33, 35, 36
BPR Business Process Reengineering. 18, 29, 30, 49, 60
BPS Business Process Simulation. 19, 57

CTE Cycle Time Efficiency. 21, 28, 29, 48, 62, 64, 72, 74, 76, 77, 79, 81, 91–93, 101, 105, 108–110, 133, 134, 138, 139

DSR Design Science Research. 20, 62, 80, 111, 132

FIFO First-In-First-Out. 18, 45, 46, 85, 87, 137
FMEA Failure Modes and Effects Analysis. 136

KPIs Key Performance Indicators. 108, 140

LIFO Last-In-First-Out. 18, 45, 46, 137
LLM Large Language Model. 6, 11, 15, 20, 21, 111–118, 120, 122–124, 127–130, 132, 135, 136, 139, 140
LSS Lean Six Sigma. 29

PM Process Mining. 33–36, 40–42, 44–48, 55, 57, 61
PT Processing Time. 91

RAG Retrieval-Augmented Generation. 117

SLR Systematic Literature Review. 17, 24, 50, 53, 118
SUS System Usability Scale. 103, 104, 139

TAM Technology Acceptance Model. 103, 139
TOC Theory of Constraints. 9, 49
TPM Total Productive Maintenance. 48

WT Waiting Time. 91

LIST OF ORIGINAL PUBLICATIONS

Publications in the scope of the thesis

[I] K. Lashkevich, F. Milani, D. Chapela-Campa, and M. Dumas. “Data-driven analysis of batch processing inefficiencies in business processes”. In: *International Conference on Research Challenges in Information Science*. Springer, 2022, pp. 231–247. DOI: 10.1007/978-3-031-05760-1_14.

Author contribution: Lead author. Substantially contributed to the study design, evaluation, and writing.

[II] K. Lashkevich, F. Milani, D. Chapela-Campa, I. Suvorau, and M. Dumas. “Why am I waiting? Data-driven analysis of waiting times in business processes”. In: *International Conference on Advanced Information Systems Engineering*. Springer, 2023, pp. 174–190. DOI: 10.1007/978-3-031-34560-9_11.

Author contribution: Lead author. Substantially contributed to the study design, evaluation, and writing.

[III] K. Lashkevich, F. Milani, D. Chapela-Campa, I. Suvorau, and M. Dumas. “Kronos: Discovery and analysis of waiting time causes”. In: *5th International Conference on Process Mining (Doctoral Consortium/Demo)*, 2023.

Author contribution: Lead author. Substantially contributed to the study design, evaluation, and writing.

[IV] K. Lashkevich, F. Milani, D. Chapela-Campa, I. Suvorau, and M. Dumas. “Unveiling the causes of waiting time in business processes from event logs”. In: *Information Systems* (2024). DOI: 10.1016/j.is.2024.102434.

Author contribution: Lead author. Substantially contributed to the study design, evaluation, and writing.

[V] K. Lashkevich, F. Milani, M. Avramenko, and M. Dumas. “LLM-assisted optimization of waiting time in business processes: a prompting method”. In: *International Conference on Business Process Management*. Springer Nature Switzerland, 2024, pp. 474–492. DOI: 10.1007/978-3-031-70396-6_27.

Author contribution: Lead author. Substantially contributed to the study design, evaluation, and writing.

[VI] K. Lashkevich, F. Milani, M. Avramenko, and M. Dumas. “Redesigning business processes to reduce waiting times using large language models”. In: *International Conference on Business Process Management (Demos/Resources Forum)*, 2024.

Author contribution: Lead author. Substantially contributed to the study design, evaluation, and writing.

Publications out of the scope of the thesis

[VII] F. Milani, K. Lashkevich, F. M. Maggi, and C. Di Francescomarino. “Process mining: a guide for practitioners”. In: *International Conference on Research Challenges in Information Science*. Springer, 2022, pp. 265–282. DOI: 10.1007/978-3-031-05760-1_16.

[VIII] K. Lashkevich, L. M. Mediavilla Ponce, M. Camargo, F. Milani, and M. Dumas. “Discovery of improvement opportunities in knock-out checks of business processes”. In: *Research Challenges in Information Science: Information Science and the Connected World*. Springer, 2023, pp. 381–397. DOI: 10.1007/978-3-031-33080-3_23.

[IX] K. Lashkevich, F. Milani, and N. Danylyshyn. “Analysis templates for identifying improvement opportunities with process mining”. In: *European Conference on Information Systems*, 2023.

1. INTRODUCTION

Organizations that aim to maintain efficiency and continuously improve their operations need to actively manage their business processes. Business processes comprise sequences of events, activities, and decision points involving actors and objects, collectively leading to the achievement of a business goal [1]. These processes define *how* work is performed within the organization and thus largely determine its performance. Consequently, the way business processes are established and managed impacts the organization's efficiency, adaptability to change, and potential for further development and improvement.

An essential part of business process management is regularly conducting process analysis to identify and address process weaknesses or inefficiencies [2]. One of the key performance measurements is temporal performance, which assesses the processes from the perspective of time, e.g., how quickly cases are processed or whether delays occur. Temporal performance is typically measured by cycle time, which spans from the start to the end of a process execution for a case. Cycle time includes processing time (the time when activities are executed to process a case) and waiting time (the time in between the activity executions) [1].

Waiting time generally does not add value to a business process and, in such cases, is considered waste [3]. Typically, waiting times arise when a case transitions between activities, i.e., when the execution of a case moves from one activity to another. Although it is impractical to eliminate waiting times in business processes, there are various approaches to reduce them [4]. Most approaches indicate that to address waiting time effectively, it is crucial to understand its causes and then address the causes of waiting times [5]. There are different reasons why waiting times occur during activity transitions. For instance, when two consecutive activities in a case are executed by different resources (a.k.a. a handover [6]), the processing of a case is put on hold until the next resource becomes available. In this scenario, the cause of the waiting time is resource contention, i.e., a resource is not available to execute an activity instance because they are busy executing other activity instances [1]. Waiting times may also be caused by data exchanges [7], coordination issues, or synchronization points [8]. When analyzing waiting times, analysts typically identify their causes manually using root-cause analysis techniques such as the cause-effect diagrams (e.g., 6M's diagram) [1].

Nowadays, business process management is supported by process mining techniques. Process mining techniques enable the analysis of data generated by business process executions (a.k.a. event logs) to analyze and improve business processes [9]. These techniques support a wide range of use cases, including the discovery of waiting times and the analysis of temporal performance [10]. Furthermore, the application of process mining techniques can be enhanced with evolving Artificial Intelligence (AI) techniques, such as Large Language Models (LLMs) [11], to make process analysis and improvement more accessible to a broader audience of process analysts, including those not familiar with process mining. All these establish a ground for the hypothesis that the analysis of waiting time causes can be further researched and addressed with these techniques.

1.1. Problem statement

Process mining techniques allow for uncovering (unexpected) patterns occurring in business processes, e.g., social networks among the resources involved in the process [12] or resource availability calendars describing their working schedules [13]. These techniques also enable discovering performance issues [14] and associated improvement opportunities [9]. In particular, they support the discovery of wastes, such as waiting times [9]. However, while existing process mining techniques enable analysts to identify, quantify, and visualize waiting time inefficiencies [15] (e.g., activity transitions with high waiting times, i.e., bottlenecks), they provide limited support for identifying and quantifying the causes of waiting times. Therefore, this thesis sets the following research question (RQ):

RQ1 How to identify causes of waiting times from event logs?

When waiting time inefficiencies and their causes are identified, analysts proceed to process redesign, determining how the process can be changed to eliminate or reduce these inefficiencies. This activity is predominantly manual, relying on the analysts' knowledge and experience in process improvement. Several research studies now provide collections of proven redesign patterns and semi-automated approaches to identifying relevant redesign options. However, these solutions cover multiple performance perspectives (e.g., time, cost, customer-centricity) and do not specifically focus on providing redesign options based on the causes of waiting times. Therefore, this thesis sets the following research question (RQ):

RQ2 How to identify redesign options to address improvement opportunities to reduce waiting times?

In summary, this thesis focuses on two key research questions. The first question investigates how to identify the causes of waiting times from event logs, aiming to develop data-driven, process mining-based approaches that provide detailed causal analysis and support analysts in identifying improvement opportunities related to waiting times. The second question explores how to identify redesign options that can effectively reduce waiting times by targeting their specific causes. By addressing these questions, the thesis offers a comprehensive solution (in the form of a set of data-driven approaches) to solving the waiting time problem: first, by identifying and analyzing the causes of waiting times, and second, by providing actionable redesign suggestions for process improvement. This proposed solution could enable organizations to diagnose the causes of waiting times and receive targeted redesign recommendations, thereby facilitating the optimization of waiting times in business processes[1].

[1] We use the term "optimization" in the sense of heuristically searching for process designs with lower waiting times, and not in the sense of finding an optimum design.

1.2. Prior work and research gaps

Several studies have examined and partially addressed the problem raised in RQ1. Process mining techniques have been extensively utilized to identify and quantify waiting times from the event logs. Ali et al. [16] conducted a Systematic Literature Review (SLR) study where they reviewed the process mining techniques available for quantifying waiting times from event logs and derived five different notions of how waiting time can be quantified. Following the waiting time quantification, various case studies demonstrate the application of process mining techniques to analyze waiting times across different domains such as manufacturing production [17], healthcare [18, 19], and e-commerce [20]. These techniques are effectively employed to detect bottlenecks (e.g., in the form of high waiting times [18] or delays (e.g., waiting time exceeding expected values) [20]). While these studies successfully quantify waiting times and identify waiting time-related inefficiencies, they do not attribute waiting times to their specific causes, which is a primary focus of this thesis.

To address the problem of identifying the causes of waiting times from event logs, previous studies have used correlation-based approaches to identify characteristics associated with higher waiting times. These include the involvement of specific activities and resources [21], executing handovers [22], or process deviations [23]. While these studies focus on understanding which characteristics correlate with higher waiting times, this thesis aims to attribute waiting times to their direct causes.

Based on the SLR, Ali et al. [16] identified several sources of waiting times that can be extracted from event logs: batching (i.e., when resources accumulate activity instances to process them together as a group), resource contention (i.e., when an activity instance cannot start because the required resources are occupied with another activity instance), prioritization (i.e., when an activity instance does not start because the assigned resource starts a higher priority activity instance), and resource unavailability (i.e., when an activity instance cannot start because the required resources are off duty). Subsequently, process mining techniques have been proposed to identify and analyze these sources from event log data.

More specifically, process mining techniques have been developed to discover batch processing behavior from event logs. For instance, Martin et al. provide techniques to identify batch processing activities and their characteristics (e.g., batch size, duration of batch processing activities, and waiting time of a case in a batch) [24], as well as batch activation rules [25]. Additionally, process mining techniques allow analyzing the impact of batching on overall process performance by simulating and comparing how performance (in terms of time and cost) can be improved with different batching strategies [26]. While these studies use event logs to analyze batch processing performance, they do not specifically focus on the waiting times associated with batch processing.

Waiting times associated with resource contention often lead to case buildups and are frequently analyzed in the context of queues, especially in domains where queues are common, such as healthcare. Therefore, several studies combine process mining techniques with queuing theory to analyze waiting times caused by queuing effects [27, 28]. However, in these studies, all waiting times were attributed to queuing (resource contention), while other potential causes (such as prioritization or resource unavailability) could also contribute to waiting times.

Multiple studies indicate that prioritizing certain cases reduces their waiting times at the cost of increasing the waiting times of non-prioritized cases [29, 30, 31]. Process mining techniques have been effectively used to identify when prioritization occurs and which prioritization strategies are applied. For instance, Suriadi et al. [32] propose a process mining technique to discover how resources prioritize work, identifying First-In-First-Out (FIFO), Last-In-First-Out (LIFO), and rule-based prioritization patterns. Existing research focuses on discovering prioritization and its strategies but leaves the discovery of associated waiting times out of scope.

Process mining techniques are also applied to identify waiting times stemming from resource unavailability, especially relevant for processes with complex schedules and varying resource availability patterns, such as healthcare [33]. For instance, Toosinezhad et al. [34] introduce a process mining approach to detect process patterns preceding bottlenecks, particularly resource unavailability. Although these studies recognize resource unavailability as a potential source of waiting time, they do not specifically focus on attributing and quantifying these waiting times. Additionally, studies have focused on mining resource availability patterns from event log data. For example, López-Pintado and Dumas [13] and Martin et al. [35] suggest techniques for discovering resource availability calendars to describe working schedules. This thesis extends this research by identifying and quantifying waiting times caused by resource unavailability.

The techniques proposed in the reviewed studies enable data-driven discovery and analysis of specific issues related to waiting times. However, these techniques focus on identifying the presence of particular phenomena (e.g., batching) and their descriptions (e.g., batching strategies) or identifying a singular source of waiting time (e.g., resource contention). They have not yet been combined to discover multiple causes of observed waiting times in activity transitions or to directly attribute waiting times to specific causes. Therefore, existing research leaves a gap in discovering and quantifying the potential causes for waiting times in each activity transition and their contribution to temporal process performance. Thus, we formulated the following research gap (RG):

RG1 Existing studies have not introduced an approach to attribute waiting times observed in business process event logs to their causes.

To reduce waiting times and improve the temporal performance of business processes, several methodologies have been developed, such as Six Sigma [36], Lean [37] and Business Process Reengineering (BPR) [38, 1]. These methodologies outline how the waiting times can be reduced, e.g., by reducing the batch sizes [37]. However, while these methodologies provide frameworks for identifying inefficiencies, they do not specifically describe how to redesign processes to reduce waiting times based on their causes. Several research studies, particularly case studies, offer recommendations on how waiting times can be reduced in specific domains, such as healthcare [39, 27], e.g., through resource capacity planning and appointment scheduling to optimize patient flow and minimize waiting times [40]. However, they focus on specific domains and offer a limited set of redesigns applicable in particular settings.

Business Process Simulation (BPS) is a widely used technique for the quantitative analysis of business processes [1]. BPS attempts to replicate real-world or hypothetical process behaviors within a computational environment, enabling the analysis of process model variations and predicting process performance under varying conditions [41]. More specifically, BPS allows modifying the business process models (e.g., their control flow, data/rules, resource organization perspectives) for simulation experiments answering "what-if" questions [41]. In particular, BPS is reported to be a valuable tool for testing different redesigns and evaluating their effectiveness in reducing waiting times. Simulation allows for experimenting with various process parameters, such as altering control flows, changing the number of resources and their schedules, or adjusting work procedures [42, 43, 44]. Although BPS is a valuable tool for testing different redesign scenarios in a controlled environment and validating expected gains, analysts first need to identify which specific redesign options to test, which is addressed in this thesis.

Several studies have proposed collections of process redesign options. Reijers and Mansar [45] introduced a framework with 29 redesign heuristics, detailing their application and impacts on process performance. Fehrer [46] built on this work, extending the collection to 50 redesign patterns for process improvements observed in the field. In our previous research [47], we proposed 64 redesign options derived from case studies on process improvement, connecting them to specific improvement opportunities (e.g., bottlenecks) that these redesign options can address. These studies offer comprehensive overviews and broad collections of redesign patterns for process improvement but do not focus on patterns applicable for reducing waiting times by targeting their causes.

Process redesign is predominantly a manual activity executed by process analysts [48], though recent advancements in process mining have led to semi-automated redesign approaches. Tools like the PrICE toolkit [49] use process mining to generate as-is process models and assist analysts in exploring redesign options. The assisted business process redesign (aBPR) tool [50] employs algorithms to recommend redesign patterns based on the analysis of as-is models. Mustansir et al. [48] proposed an approach for the automatic extraction of redesign suggestions by classifying end-user feedback available in natural language. The proposed approaches and tools partially automate activities conducted by process analysts. However, they still largely rely on inputs from process analysts (e.g., process models and decision interventions), use a limited number of predefined redesign patterns, and do not specifically target the causes of waiting times when offering redesign options. Additionally, a set of research studies offered algorithms aimed at optimizing specific process patterns (such as batching, resource allocation, and scheduling) within the context of ongoing cases to reduce waiting times [51, 52, 53]. While these research studies enable the optimization of case execution to improve their performance on a day-to-day basis, they do not address how processes can be redesigned to achieve long-term temporal performance improvements. This leaves the following research gap:

RG2 Existing studies have not provided automated methods to recommend redesign options that reduce waiting times by targeting their causes.

In this thesis, we propose a set of approaches to address these research gaps by enabling data-driven identification and analysis of the causes of waiting times. Additionally, we propose a method enabling automated redesign recommendations to reduce waiting times by specifically targeting their identified causes.

1.3. Research methodology

To answer the set research questions, we follow the Design Science Research (DSR) methodology [54]. DSR is suitable since it implies the continuous development and evaluation of artifacts to improve their performance in the field of information systems research. Using this methodology, we identified the research problems in waiting time cause analysis and redesign, defined two research questions (RQ1 and RQ2), and identified the gaps in the state-of-the-art related to these questions. This allowed us to define objectives and requirements for developing and evaluating the artifacts.

Further, we designed and developed three artifacts to address our research questions. The first artifact, related to RQ1, is an approach for identifying and analyzing waiting times associated with batch processing. The second artifact, also related to RQ1, is an approach for decomposing waiting times in each activity transition into multiple direct causes. The third artifact addresses RQ2 by proposing a prompting method to tune an LLM to analyze waiting time causes and recommend redesign options to reduce waiting times. After developing these artifacts, we evaluated them using synthetic data and real-life event logs. Based on the evaluation results, we identified opportunities for improving the artifacts and, when possible, enhanced them accordingly. The detailed description of how each artifact was developed using the DSR methodology is provided in the respective chapters presenting the artifacts (first artifact in Chapter 4, second artifact in Chapter 5 and third artifact in Chapter 6).

1.4. Contributions

This thesis provides three contributions to the field of business process analysis and improvement using process mining:

Contribution 1: We propose a process mining approach for the discovery and analysis of waiting time inefficiencies caused by batch processing (Chapter 4). To address the RG1, we explore batching as a potential cause of waiting times in business processes. For that, we conceptualize three types of waiting times that can be observed when different batch processing strategies are applied. Then, we propose a process mining approach that allows for discovering these types of waiting times associated with batch processing from the event logs and identifying improvement opportunities based on the impact of each waiting time type on the Cycle Time Efficiency (CTE). The proposed approach has been implemented as a process mining technique and evaluated via experiments on synthetic event log data. The applicability of the proposed approach has been demonstrated on a real-life event log.

Contribution 2: We propose a process mining approach to decompose observed waiting times in each activity transition into multiple direct causes (Chapter 5). Given an event log, the approach allows for discovering 5 direct causes of waiting times in an activity transition: batching, resource contention, prioritization, resource unavailability, and extraneous factors. The discovered waiting time causes are quantified and analyzed to identify improvement opportunities by assessing the impact of each waiting time cause on the process CTE. The accuracy of the proposed approach in quantifying the waiting time causes has been evaluated via experiments on synthetic event log data. The applicability of the approach has been demonstrated on a real-life event log. In addition, the proposed approach has been implemented as a software tool called Kronos. The usefulness and usability of Kronos in identifying waiting time-related improvement opportunities have been evaluated via the user study with process analysts.

Contribution 3: We propose a prompting method for an LLM that tunes it to analyze the causes of waiting times and recommend redesign options to reduce these waiting times by targeting their causes (Chapter 6). The proposed method uses the *Contribution 2* for input data sources and includes a prompt tailored to analyze these sources and recommended relevant redesign options to address the waiting time causes. The proposed prompting method has been implemented as a conversational interface and integrated with Kronos. The accuracy of the prompt-enhanced LLM in identifying waiting time durations and their causes is verified via computational experiments on synthetic and real-life event logs. The effectiveness of the proposed prompting method in assisting analysts with analyzing waiting time causes and identifying redesign options is validated via the user study. More specifically, the user study compares two approaches: (1) a baseline ("zero-shot") approach involving a minimalistic prompt; and (2) an enhanced approach (where the prompt includes descriptions of redesign patterns that may lead to redesigned processes with reduced waiting times). The analysts have compared the recommended redesigns in terms of desirable properties of recommendation systems, such as relevance, usefulness, and diversity of the recommended redesigns.

1.5. Outline

The rest of this thesis is structured as follows. In Chapter 2, we introduce background notions and concepts of waiting time analysis and redesign. In Chapter 3, we review the state-of-the-art studies related to the data-driven analysis of waiting times, waiting time causes, and process redesign. Chapter 4 explores batching as a source of waiting time – it introduces different types of waiting times due to batching and proposes an approach for identifying these waiting times from the event log. Chapter 5 introduces the conceptualization of waiting time causes in activity transitions and presents an approach for their discovery, quantification, and analysis based on the event log data. Chapter 6 proposes a prompting method to tune the LLM to assist analysts in analyzing the waiting time causes and recommending redesign options to reduce waiting times. Chapter 7 concludes this thesis and discusses future research directions. Table 1 summarizes the addressed research questions, research gaps associated with each research question, contributions addressing those research gaps, chapters where these contributions are presented, and publications related to each contribution.

Table 1. Research questions, research gaps, thesis contributions, and publications (for detailed publication information, see the List of Original Publications on page 13).

Research question (RQ)	Research gap (RG)	Contribution
RQ1 - How to identify causes of waiting times from event logs?	**RG1** - Existing studies have not introduced an approach to attribute waiting times observed in business process event logs to their causes.	**#1** - Discovery and analysis of waiting times due to batching (Chapter 4) **Publication**: - (2022) RCIS Conference paper [55]
		#2 - Discovery and analysis of other waiting time causes (Chapter 5) **Publications**: - (2023) CAiSE Conference paper [56] - (2023) ICPM Demo paper [57] - (2024) IS Journal paper [58]
RQ2 - How to identify redesign options to address the improvement opportunities to reduce waiting times?	**RG2** - Existing studies have not provided automated methods to recommend redesign options that reduce waiting times by targeting their causes.	**#3** - LLM-assisted analysis and redesign of waiting times (Chapter 6) **Publications**: - (2024) BPM Conference paper [59] - (2024) BPM Demo paper [60]

2. BACKGROUND

This chapter provides an overview of the key concepts used in the thesis. First, in this chapter, the field of Business Process Management (BPM) is introduced, and the BPM lifecycle is outlined, with a focus on process analysis — particularly performance measurement — and process redesign. Second, process mining and its techniques are presented.

2.1. Business process management

2.1.1. Business process management lifecycle

Business processes form the foundation of organizations' operation as they define *how* work is performed [61]. A business process is a sequence of activities that transform inputs into outputs, adding value to an organization and its customers [1]. These activities are discrete units of work that represent the steps executed in the process. Each execution of a business process creates a *process instance*, which corresponds to a specific *case*. For example, in an "order-to-cash" process, each order execution is a separate *process instance*, and the order itself is called a *case*. Each execution of an activity within the process is called an *activity instance* and is linked to the particular case. Activity instances are performed by *resources*, which can be human participants, machines, or software services [16].

Business processes are critical assets for organizations, largely defining their overall performance. Therefore, it is essential to manage these processes effectively. The specialized body of knowledge that focuses on managing and optimizing business processes is known as Business Process Management (BPM) [1]. BPM is commonly viewed as a lifecycle [62] consisting of process identification, discovery, analysis, redesign, implementation, monitoring, and controlling business processes (Figure 1) [1]. First, the BPM team (e.g., process analysts) identifies the process for analysis and discovers the process and its (performance) characteristics. A typical outcome of the process discovery phase is "as-is" process models [1]. Then, a modeled process can be analyzed to identify issues, "weak parts of the process" (i.e., the parts of the process that negatively affect its performance), and improvement opportunities [1]. In this thesis, we use the term *improvement opportunities* to collectively refer to process issues, inefficiencies, and weaknesses that have the potential to be addressed for enhanced process performance.

Once improvement opportunities have been identified, analysts further examine and classify them to determine which ones should be prioritized and addressed. At the phase of process redesign, analysts can propose a redesigned version of the process (a.k.a. "to-be" process model), which would address the improvement opportunities identified and prioritized in the "as-is" process. Further, the redesigned version of the process is implemented, followed by monitoring and controlling

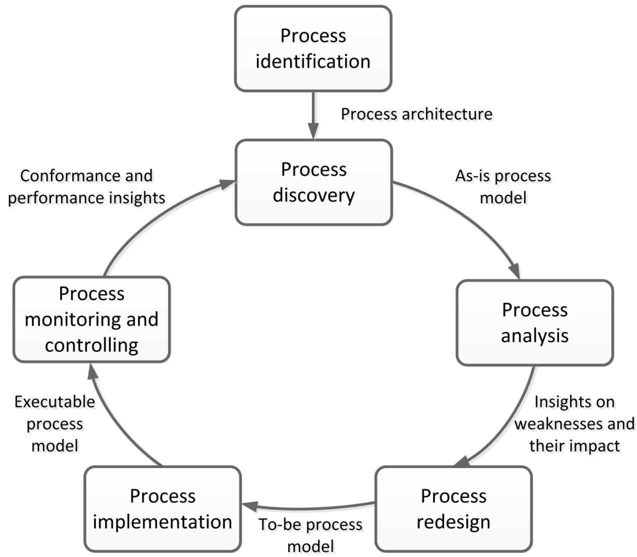


Figure 1. The BPM lifecycle [1].

to assess the process's performance against the set objectives and identify necessary adjustments [1]. This thesis concentrates on the process analysis phase, particularly on analyzing waiting times in business processes and identifying associated inefficiencies. Additionally, our focus covers the process redesign phase, where we explore how to address these inefficiencies to reduce waiting times by redesigning the process.

2.1.2. Process analysis and performance measurement

The process analysis phase of BPM lifecycle focuses on identifying, analyzing, and assessing the issues and improvement opportunities in the business processes [1]. Whenever possible, the identified improvement opportunities are quantified using performance measures [1]. Therefore, as a starting point for process analysis, analysts measure the performance of the current ("as-is") processes [63].

Business process performance can be measured from various perspectives. For instance, Figure 2 provides an overview of 11 performance perspectives, identified based on a SLR of studies discussing business process performance measurement [64]. *Financial performance for shareholders and top management* focuses on strategic financial metrics and includes indicators such as revenue growth [65] and return on investment [66]. The *supplier performance* perspective measures the effectiveness of external collaborations and dependencies, capturing metrics such as external delays [67] and external mistakes [68]. The *customer performance* perspective assesses outcomes related to external quality and meeting end-user needs, such as customer satisfaction [69] and return rates [68]. The *society/environmental performance* perspective evaluates the impact on other stakeholders and the environment during process execution, including metrics for social responsibility and sustainability [70].

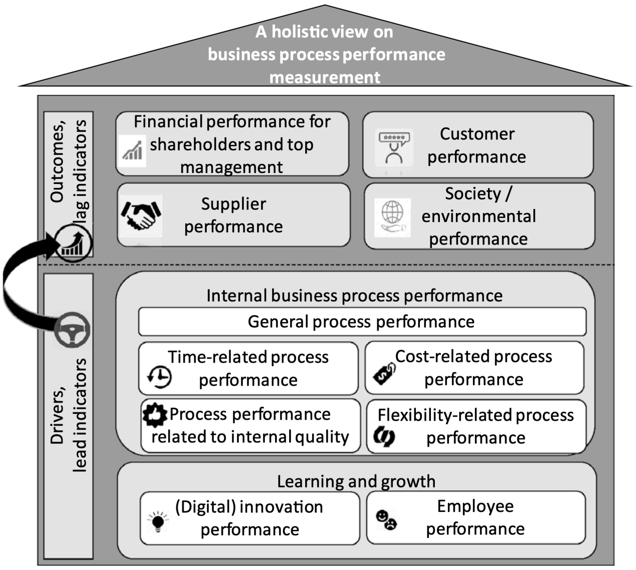


Figure 2. Business process performance perspectives [64].

A group of performance perspectives focuses on assessing the internal business process performance. This includes perspectives related to *time*, *cost*, *flexibility*, and *internal quality*[1]. Additionally, the perspective of *general process performance* enables the assessment of process aspects not specifically related to time, costs, quality, or flexibility, such as process complexity [71], perceived sales, and management performance [66]. Another set of perspectives focuses on performance related to learning and growth. The *(digital) innovation performance* perspective evaluates process innovation and innovation projects, including indicators such as the degree of digitalization [71] and the novelty of output [66]. The *employee performance* perspective assesses staff contributions to process work and personal development, using indicators such as employee workload [72], retention, and motivation [65].

The identified 11 performance perspectives are categorized into those providing lag indicators (measuring the performance outcomes) and lead indicators (measuring the performance drivers) [64] (see Figure 2). These categories are interconnected: improvements in lead indicators impact lag indicators, i.e., strengthening the drivers of performance will ultimately enhance the outcomes. For example, an increase in employee competencies (i.e., the "employee performance" perspective) is expected to positively influence the quality of products and services (i.e., "process performance related to internal quality") [64]. Similarly, investing

[1] These four perspectives are discussed in more detail below.

in innovation, such as integrating digital solutions (i.e., "(digital) innovation performance"), can help reduce customer waiting times (i.e., "time-related process performance) [73]. These improvements in internal business process performance will, in turn, enhance customer perceptions (i.e., "customer performance"). The outcomes from these earlier perspectives will then drive financial performance, ultimately supporting the achievement of the organization's strategy, mission, and vision [64]. According to the framework presented in Figure 2, his thesis focuses on improving internal business process performance, specifically time-related process performance, which can significantly enhance overall company outcomes.

One widely recognized framework for evaluating internal business process performance is the Devil's Quadrangle, which considers four performance dimensions: time, cost, quality, and flexibility [1] (Figure 3). The time dimension refers to the temporal performance of the process. The cost dimension considers the operational expenses associated with the process. The quality dimension refers to how the process and its outcomes are perceived by its users (i.e., both internal and external customers). The flexibility dimension indicates the extent to which the process provides alternatives (in terms of resources and solutions) in delivering the product [74]. The four dimensions are interdependent and often involve trade-offs, as improving one dimension may negatively impact another. For example, reducing process time might lead to increased operational costs, while improving process quality could result in a loss of flexibility [74]. Therefore, a balanced approach is necessary to optimize overall process performance.

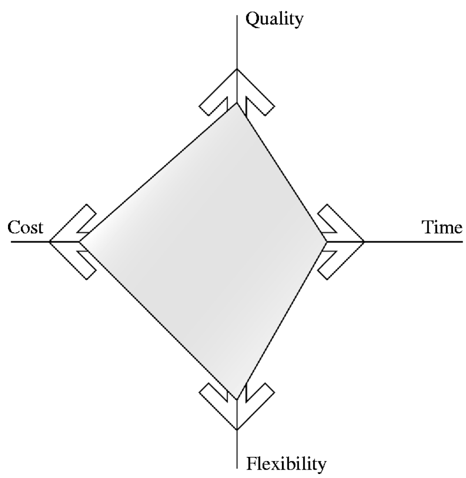


Figure 3. The Devil's Quadrangle [74].

To assess each dimension of the Devil's Quadrangle and monitor the impact of process redesigns, organizations often use Process Performance Indicators (PPIs) [75, 76] – quantifiable metrics that evaluate the efficiency and effectiveness of business processes [77]. Time-related metrics focus on the efficiency or speed of the whole process or its parts (e.g., selected activities or fragments [76]).

Cost-related metrics measure how many resources are required to produce the output [78], including direct (e.g., purchasing costs of raw materials) and indirect costs (e.g., indirect labor or machine depreciation) [1]. Metrics related to the quality perspective measure how well the output meets customer objectives, e.g., defect and rework rates [78]. Flexibility-related metrics assess the degree of process variation and adaptability, such as the number of distinct options to execute a task, the ability to handle variations in demand [45, 1], and the number of variants in the process [76]. This thesis focuses on the temporal performance of business processes. Therefore, below we discuss time-related metrics in more detail.

A common measure of temporal process performance is *cycle time* (a.k.a. throughput time), i.e., the duration between the start and end of a case in a process [1]. For example, in a loan application process, the cycle time spans from when a customer submits the loan application to when it is either approved or rejected [16]. Cycle time encompasses *processing time* and *waiting time* [1]. Processing time is the time a resource spends executing activity instances [1]. The remaining time of the cycle time is referred to as waiting time, i.e., the time in-between the execution of activity instancies [16].

Waiting time is considered a source of waste in business processes when it does not add value [3]. For example, waiting time is considered a waste when cashiers need manager approval for an exception to the standard cash receiving process, but the manager is occupied with other responsibilities, resulting in waiting time for both cashiers and customers [79]. Similarly, in a university admissions process, when the admissions office finds an application incomplete, an email is sent to the candidate requesting missing information or documents, placing the application on hold until the candidate responds — this hold time constitutes waiting waste [1].

However, not all waiting times are considered a waste and amenable to elimination [3]. There are several cases where waiting times are an inherent necessity. For example, in healthcare processes, after treatment is conducted, a waiting time might be required for the treatment to take effect before the next activity can be performed and, thus, this waiting time is not considered a waste [80, 81]. In production processes, intermediate products must wait to, e.g., dry after being painted before any further processing can occur [82]. Similarly, financial controls can be a necessary part of the process and add value to the company. Although these controls may introduce waiting times, such waiting times are not considered waste and are not subject to elimination [3]. Some processes require waiting to comply with internal or external regulations. For example, in a travel reimbursement process, organizational rules require that permission from the employee's supervisor is obtained for international trips, which may introduce waiting times before any arrangements can be made [83]. In some processes, extended waiting times help to reduce waiting times for more urgent cases. For example, in emergency room processes, patients with severe conditions experience shorter waiting times, resulting in longer waiting times for patients with less severe conditions. The increased waiting times for non-critical patients are necessary in this context to ensure that the most critical patients are served as quickly as possible and, therefore, are not amenable to complete elimination [29]. Regardless of these considerations, this thesis addresses the identification of waiting times without distinguishing between those that are amenable to elimination and those that are not.

In contrast to waiting time, processing time represents the period during which actual work is performed and is the time when value is added to the process [84, 75]. Therefore, another key temporal metric examined by process analysts is the *theoretical cycle time*, i.e., the amount of time a process instance would take if there were no waiting times. This metric reflects the total time spent on actual work (processing activity instances) and indicates the value-adding portion of the process duration [1]. By comparing the theoretical cycle time to the actual cycle time, analysts can determine the extent to which the process can be streamlined [85, 86].

Theoretical cycle time allows for the calculation of another temporal performance metric known as *cycle time efficiency (CTE)*. CTE measures the proportion of value-added time within the total cycle time of a process [1, 75]. CTE is calculated by dividing the theoretical cycle time (TCT) by the cycle time (CT) and is expressed as a percentage:

$$\mathrm{CTE} = \frac{\mathrm{TCT}}{\mathrm{CT}}$$

For example, given that the total cycle time (CT) of the process is 100 hours and the theoretical cycle time (TCT) is 10 hours, the cycle time efficiency (CTE) is calculated as 10/100 = 0.1 or 10%. CTE close to 100% indicates that the process has comparatively low waiting to the processing time and, thus, little room for improvement. However, low CTE (as in the above example) indicates a comparatively high waiting time to processing time and, thus, highlights an improvement opportunity [1].

Utilizing CTE has proven valuable for analyzing temporal process performance and identifying time-related inefficiencies [87, 88, 89, 90, 91, 92]. For instance, in [90] and [91], CTE was used to assess the temporal efficiency of healthcare processes. In their analyses, low CTE indicated the presence of waiting time inefficiencies. Similarly, in [92], CTE was applied to measure the temporal performance of the scheduling process for university counseling, where the CTE of 6.9% indicated high waiting times. In [87], CTE was applied for measuring time-related performance in factories to detect inefficiencies. Specifically, they assessed the performance of production cells by comparing actual CTE values against benchmarks. This comparison highlighted non-value-added parts of the process and quantified the extra costs they induced, which could then be targeted. Ignizio [88] used CTE to identify workstation instability in production processes. They found that monitoring the decline in workstation CTE was the most effective indicator for detecting their instability, as it provided a clear signal of emerging issues in the production flow.

In addition to evaluating the temporal performance of business processes, CTE has been utilized to assess the (expected) effectiveness of process redesigns[2] [89, 90, 93]. For instance, in [90], the CTE of a redesigned healthcare process was compared to that of the existing process to demonstrate temporal improvement [90]. This comparison highlighted the reduction in waiting times, justifying the effectiveness of the redesign in enhancing temporal process performance. Similarly, in [93], CTE values of the existing and redesigned procurement planning processes were compared as a key metric to evaluate the effectiveness of the process redesign. In this thesis, we also employ the CTE metric to analyze the temporal performance of business processes, identify inefficiencies related to waiting times, and assess potential temporal improvements.

2.1.3. Process redesign

Process redesign is a phase of BPM aimed at enhancing the efficiency, effectiveness, and adaptability of business processes [1]. To support process redesign, various methodologies have been proposed, with some of the most popular being BPR, Lean, and Six Sigma. BPR focuses on fundamental rethinking and radical redesign of business processes to achieve dramatic improvements in critical measures of performance, such as cost, quality, and speed (i.e., time) [38]. Lean prescribes eliminating waste and non-value-added activities [94]. Six Sigma aims to reduce variability and defects through data-driven decision-making and statistical analysis [95]. Subsequently, Lean Six Sigma (LSS) is a methodology that combines Lean and Six Sigma approaches and aims to eliminate nine sources of waste (defects, overproduction, overprocessing, transportation, motion, inventory, underutilized employees, waiting, and behavior waste) and provide goods and services at a rate of 3.14 defects per million opportunities [95].

Process redesign methods can be categorized into two types: transformational and transactional. Transformational (a.k.a. radical) process redesign, such as BPR, involves substantial changes to existing processes with the goal of achieving significant performance improvements. Transformational process redesign is characterized by high risk, significant resource investment, and potential disruption to the organization [61]. In contrast, transactional (a.k.a. incremental [61]) process redesign focuses on gradual partial improvements to existing processes [96]. For instance, process redesign heuristics provide guidelines on how to modify specific process elements, such as resource allocation or control flow, to enhance process performance [45]. Such methods are typically less risky and more manageable, fostering a culture of continuous improvement within organizations [96]. This thesis focuses on identifying transactional process redesigns

[2]It should be noted that for calculating expected CTE (e.g., after the process redesign), some studies (e.g., [89] and [90]) conduct value-added analysis to deduct the duration of non-value-added activities from the expected theoretical cycle time. In this thesis, we assume that all activities are value-added and focus on waiting time-related inefficiencies.

that can be applied to reduce waiting times and, therefore, improve the temporal performance of business processes.

Redesign methods can either take an inward-looking or outward-looking perspective. Inward-looking methods focus on improving business processes from the organization's own viewpoint, using internal data and requirements as the sources of information for process redesign [96]. For instance, the positive deviance method (a.k.a. constructive deviance [97]) involves identifying beneficial variations in process executions that contribute to organizational advantages (e.g., lower resource consumption or faster procedures). These positive deviations are used as a source for identifying actions for improvement [98]. Conversely, outward-looking methods consider the needs and requirements of customers or third parties when redesigning a business process [96]. This thesis focuses on inward-looking methods, primarily driven by the organization's need to reduce waiting times in business processes. In addition, we rely on process execution data to identify improvement opportunities for process redesign.

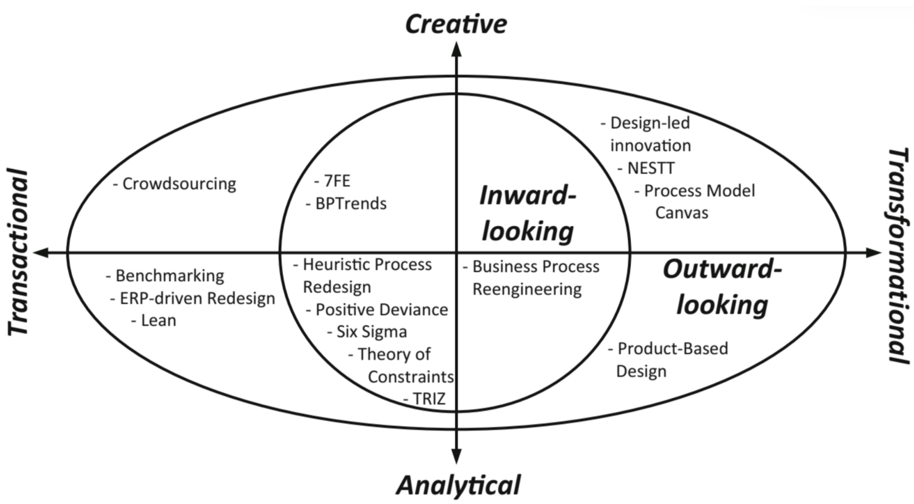


Figure 4. The Redesign Orbit: A spectrum of business process redesign methods [1].

Additionally, redesign methods can be categorized as either analytical or creative. Analytical methods employ quantitative and qualitative techniques to analyze the current state of business processes, e.g., benchmarking, Lean, Theory of Constraints, Six Sigma, and BPR. In contrast, creative methods rely on human creativity to generate new ideas. These methods often involve activities like brainstorming, where participants collaboratively come up with innovative ways to redesign business processes [96]. In this thesis, we focus on analytical methods since our approaches are data-driven.

Figure 4 illustrates process redesign methods across three dimensions: transactional vs. transformational, analytical vs. creative, and inward-looking vs. outward-looking. Most existing process redesign methods are classified as analytical, transactional, and inward-looking [96], e.g., redesign heuristics [45]. These methods aim to achieve incremental process improvements by addressing specific inefficiencies identified through the analysis of process data. Thus, this thesis also falls into this category of process redesign methods, specifically on identifying transactional process redesigns to address waiting time inefficiencies discovered through data-driven analytical methods.

2.2. Process mining

Traditionally, most activities in the BPM lifecycle are performed manually by process analysts, managers, and other business stakeholders [99]. For instance, discovering business processes often requires time-consuming data collection methods such as interviews, focus groups, and field observations. This is followed by the manual analysis and validation of the collected data to create accurate business process models [99]. Nowadays, a range of techniques are offered to automate the BPM lifecycle activities [99]. For instance, business processes can be automatically discovered and modeled based on the process execution data recorded by systems, such as Enterprise Resource Planning (ERP) or Customer Relationship Management (CRM) systems [99]. These techniques belong to the field of process mining [9]. In this section, process mining and its techniques are presented.

2.2.1. Event logs and activity instance logs

Modern IT systems record and store process execution data in *event logs*, i.e., sets of timestamped events capturing the execution of the activities in a process [1]. An activity (a.k.a. activity model) represents a task that can be executed multiple times within a business process for different cases. For example, activity "Register invoice" can be performed for multiple invoices in an invoice handling process. An activity instance refers to a single execution of a specific activity, e.g., activity instance of "Register invoice" executed on 12/10/2024 from 08:30:12 to 11:30:00 for case 510 [100].

An activity instance can transition between various states during its execution. These states and their corresponding transitions are represented in a state transition diagram, as shown in Figure 5. When an activity instance is created, it enters the *init* state, indicating that it is recorded in the system. Then, when an activity instance is enabled, it enters the *ready* state, indicating that it is ready to be executed. During the init and ready states, the execution of an activity instance is *not started*. From the ready state, the activity instance can be started, i.e., enter the *running* state. Once the execution of an activity instance has been completed, an activity instance enters the *terminated* state. If a particular activity instance is not required for a specific case, it can be skipped. This is represented by a transition from the not started to the *skipped* state. An activity instance is considered *closed* when it is in either the terminated or skipped state [100].

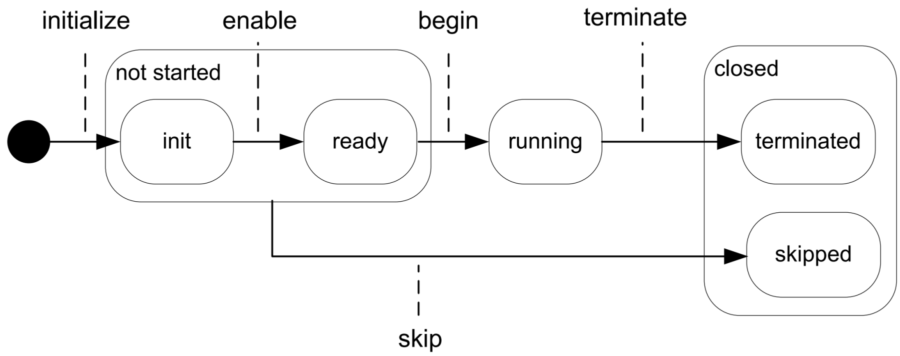


Figure 5. State transition diagram for activity instances [100].

An event log contains information about state changes of each activity instance (e.g., enablement, start, end, or completion of activity instances). In this thesis, we use the concept of *activity instance log* to represent the execution of a set of cases in a process. An activity instance log is an event log in which each entry contains information about the start time and end time of an activity instance [24]. Below, we introduce several notations used in the paper, leading to a definition of an activity instance log.

Table 2. Fragment of an activity instance log composed of 10 activity instances.

Case ID	Activity	Enabled Time	Start Time	End Time	Resource
...					
510	Register invoice	08:30:12	08:30:12	11:30:00	Jack
511	Notify acceptance	09:10:11	09:10:11	10:01:01	Carolyn
511	Post invoice	09:10:11	09:10:11	10:14:15	Sarah
512	Post invoice	10:25:45	10:25:45	10:30:00	Sarah
513	Post invoice	10:34:15	10:34:15	12:00:00	Sarah
514	Post invoice	09:10:11	11:30:00	13:00:00	Jack
515	Register invoice	12:08:10	12:08:10	13:00:00	Sarah
512	Pay invoice	10:30:00	15:55:50	17:00:11	Jack
513	Pay invoice	12:00:00	17:00:11	17:55:40	Jack
511	Pay invoice	10:14:15	17:55:40	18:30:15	Jack
...					

We consider a business process that involves a set of *activities* A. We denote each of these activities with α. An *activity instance* $\varepsilon = (\varphi, \alpha, \tau_e, \tau_s, \tau_c, \rho)$ denotes one execution of activity α, where φ identifies the *process case* to which this execution belongs to, τ_e, τ_s, and τ_c denote, respectively, the instants in time in which this activity instance was *enabled*, *started*, and *completed*, and ρ identifies the *resource* that processed the activity. Accordingly, we use $\varphi(\varepsilon_i)$, $\alpha(\varepsilon_i)$, $\tau_e(\varepsilon_i)$, $\tau_s(\varepsilon_i)$, $\tau_c(\varepsilon_i)$ and $\rho(\varepsilon_i)$ to denote, respectively, the process case, the activity, the enablement time, the start time, the completion time, and the resource associated with the activity instance ε_i. We use (τ_i, τ_j) to denote the time interval between τ_i and τ_j. We write $\omega(\varepsilon_i) = (\tau_e(\varepsilon_i), \tau_s(\varepsilon_i))$ to denote the *waiting time* of ε_i, representing the interval since ε_i became available for processing ($\tau_e(\varepsilon_i)$), until its recorded start ($\tau_s(\varepsilon_i)$). The processing time of ε_i is $pt(\varepsilon_i) = (\tau_s(\varepsilon_i), \tau_c(\varepsilon_i))$. We use $(\tau_i, \tau_j) \in (\tau_k, \tau_l)$ to denote that the interval (τ_i, τ_j) is contained in (τ_k, τ_l), i.e. $\tau_i \geq \tau_k$ and $\tau_j \leq \tau_l$. With $(\tau_i, \tau_j) \perp (\tau_k, \tau_l)$ we denote that both intervals (partially or fully) overlap, i.e. $\exists(\tau_m, \tau_n) \in (\tau_i, \tau_j) \mid (\tau_m, \tau_n) \in (\tau_k, \tau_l)$.

Given the above, an *activity instance log L* is a collection of activity instances recording the data of the execution of a set of cases of a business process. Table 2 shows an example of 10 activity instances from an activity instance log, whereas Figure 6 depicts the corresponding process model.

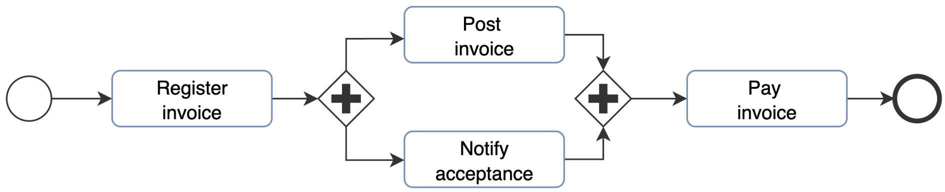


Figure 6. Process model example corresponding to the event log of Table 2.

The approaches proposed in this thesis do not include transforming event logs into activity instance event logs. Therefore, for those interested in transforming event logs into activity instance logs, we recommend utilizing third-party services such as bupaR[3] [101].

2.2.2. Process mining techniques

Process Mining (PM) is a family of techniques that uses the process execution data (event logs) to analyze and improve business processes [102]. PM techniques enable the *data-driven BPM* by supporting one or more phases of the BPM lifecycle [99].

To support process analysts in their data-driven process improvement initiatives, the Process Mining Project Methodology (PM^2) has been proposed [103]. PM^2 provides a structured approach to systematically analyze and optimize business processes using process mining techniques. The methodology consists of six stages (Figure 7). It starts with the (1) planning stage, where initial research questions for the PM project are defined. Then, during the (2) extraction stage, the event data are extracted. After these stages, one or more analysis iterations are conducted, potentially in parallel. Each iteration involves (3) data processing, (4) mining & analysis, and (5) evaluation. An iteration focuses on answering specific research questions through process mining activities and evaluating the discovered process models and findings. Thus, during the (3) data processing stage, the event logs are modified to optimize them for the subsequent mining & analysis stage, e.g., enriching event logs with additional attributes [23] such as timestamps indicating when activity instances are enabled. During the (4) mining & analysis stage, PM techniques are applied to the event logs to answer research

[3]`https://bupaverse.github.io/docs/create_logs.html#Logs:_eventlog_vs_activitylog`

questions and gain insights into process performance, e.g., by applying process analytics [63]. The (5) evaluation stage serves to diagnose the findings obtained through mining & analysis, verify and validate the findings, and identify improvement ideas (e.g., how the process can be redesigned) that can help achieve the project's goals. If findings are satisfactory, they lead to (6) process improvement & support, where the improvement ideas are implemented, followed by supporting operations. In this thesis, we focus on the analysis stages (3-5) of the PM^2 methodology and investigate how waiting times and their causes can be identified, analyzed, and addressed.

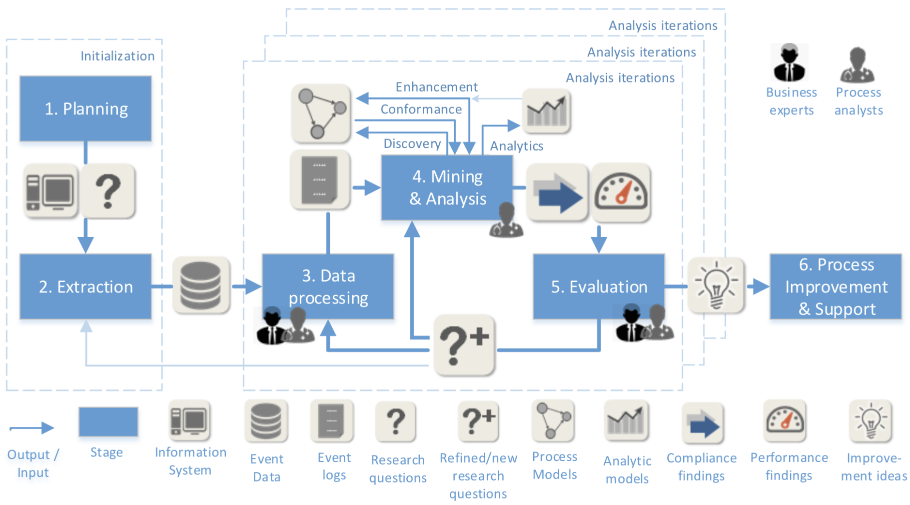


Figure 7. An overview of the PM^2 methodology [103].

Process mining supports a wide range of use cases that can be categorized into five areas: transparency, efficiency, quality, compliance, and agility [10]. Transparency-related use cases aim at discovering and enhancing process models, including social network analysis, model repair and extension, rule mining, and goal mining. Quality-related use cases deal with identifying and comparing process variations (variant analysis) and analyzing reasons for process deviations (deviance mining). Compliance assesses alignment between process models and observed behaviors or predefined rules, and includes conformance and compliance monitoring. Agility predicts future process behaviors (predictive monitoring), tracks changes over time (concept drift), and suggests actions to achieve desired outcomes in ongoing cases (prescriptive monitoring). Efficiency-related use cases involve analyzing process performance to optimize operations [10]. This thesis focuses on applying PM techniques for efficiency-related purposes, specifically for analyzing waiting times and their causes, and identifying redesigns aimed at reducing these waiting times.

Additionally, PM techniques can be categorized as backward-looking (e.g., identifying the causes of bottlenecks) or forward-looking (e.g., predicting the remaining processing time of a running case or providing recommendations to lower the failure rate) [102]. Both backward-looking and forward-looking analyses can trigger actions. This concept is known as *action-oriented process mining* that aims to turn diagnostics into actions [102]. For instance, based on diagnostic results for identifying causes of process inefficiencies (e.g., bottlenecks) in historical data, process redesigns can be implemented. Similarly, diagnosing the current state of a process execution can trigger improvement actions for ongoing cases [102]. This thesis aims to use backward-looking PM techniques to identify and analyze the causes of waiting times. To translate the analysis results into actions, we then focus on identifying process redesigns that can effectively reduce waiting times.

Another categorization of PM techniques examines these methods based on their capabilities and application timeframe. This classification is illustrated in the form of a pyramid in Figure 8. In this pyramid, the capability perspective is illustrated via layers: each layer builds upon the ones below and serves as the foundation for the ones above. Each layer comprises techniques for two types of use cases – tactical and operational, i.e., timeframe perspective. Tactical use cases aim to support managers in process change decisions with timeframes of a few weeks to a few months between decision and implementation. Operational use cases aim to provide information, issue recommendations, or trigger actions for the running cases of business processes to improve their performance on a day-to-day basis [99].

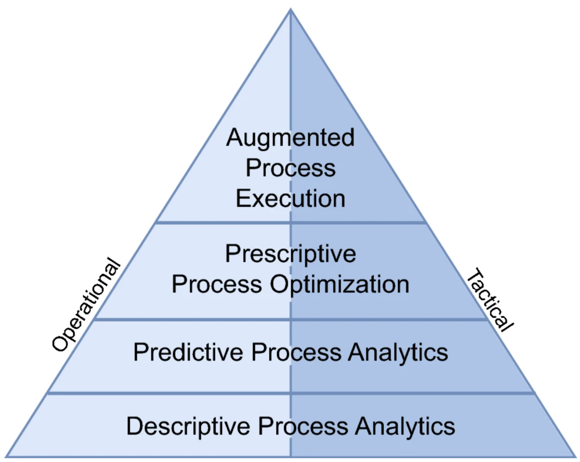


Figure 8. The augmented BPM pyramid: a classification of data-driven BPM approaches [99].

The first layer, descriptive process analytics, comprises techniques focused on describing the current state of the process, e.g., the control flow of the process. A subset of techniques in this layer focuses on performance mining, which involves analyzing processes based on their performance metrics (e.g., cycle time) and visualizing the results using performance dashboards. The second layer, predictive process analytics, builds upon descriptive analytics and goes further by predicting future outcomes, e.g., the remaining time of a running case. The third layer, prescriptive process optimization, focuses on techniques to prescribe actions that, based on the current state and future predictions, can help optimize the process performance. Prescriptive process monitoring represents the operational perspective and includes methods to recommend actions to optimize process performance in real-time or near-real-time, e.g., preventing a loan offer from being rejected by a customer. Automated process optimization covers the tactical perspective and includes methods to recommend how the process can be redesigned for better performance [99]. Recently, AI methods have been increasingly used to address BPM-related tasks. As a result, a range of AI-driven approaches for business process optimization has been introduced, e.g., using machine learning models for predictive monitoring [104]. PM techniques and AI-driven approaches can be used collaboratively, resulting in the development of (AI)-augmented process execution (the top layer of the pyramid in Figure 8). This layer extends beyond informing or recommending decisions, aiming instead to autonomously manage and optimize processes to achieve desired business outcomes within the constraints set by managers [99].

This thesis focuses on leveraging PM techniques related to descriptive process analytics, specifically performance mining, to identify and quantify waiting times in activity transitions. Additionally, we aim to identify the causes of these waiting times, which also falls into the descriptive process analytics. Identifying the root causes of specific outcomes can be seen as a more advanced form of descriptive analytics, often referred to as diagnostic analytics [105]. While diagnostic analytics also analyzes historical data, it goes further by uncovering the underlying reasons behind events or patterns in the data [105]. Furthermore, we intend to utilize PM techniques for prescriptive process optimization, particularly automated process optimization, to provide recommendations for process redesign to reduce waiting times. We present the PM techniques related to descriptive (incl. diagnostic) and prescriptive process analytics for waiting time analysis and process redesign in Chapter 3.

3. STATE OF THE ART

In this chapter, a review of the state-of-the-art in data-driven waiting time analysis and redesign is presented. The state-of-the-art review is structured to align with the research questions defined in this thesis. The first research question (RQ1), "How to identify causes of waiting times from event logs?" focuses on analyzing waiting times in business processes and identifying their causes. Therefore, the state-of-the-art review aims to answer the following question:

Q1 What process mining approaches and techniques have enabled the analysis of waiting times and their causes?

To answer Q1, Section 3.1 reviews how waiting times have been defined and quantified using event log data (Sec. 3.1.1), presents existing process mining techniques for analyzing waiting times (Sec. 3.1.2), and examines the process mining methods for discovering and analyzing the causes of waiting times (Sec. 3.1.3).

The second research question (RQ2), "How to identify redesign options to address improvement opportunities to reduce waiting times?" focuses on redesigning business processes to address waiting times. Consequently, the state-of-the-art review aims to answer the following question:

Q2 What approaches, frameworks, and techniques have been used to identify redesigns to reduce waiting times?

To answer Q2, Section 3.2 reviews the existing methodologies and approaches analysts have used to redesign processes to reduce waiting times (Sec. 3.2.1), presents the collections of redesign options (Sec. 3.2.2), and examines existing solutions enabling semi-automated process redesign (Sec. 3.2.3).

3.1. Data-driven analysis of waiting times and their causes

The first research question (RQ1) of this thesis focuses on analyzing waiting times in business processes and identifying their causes. Therefore, this section presents state-of-the-art process mining techniques for quantifying and analyzing waiting times, and then examines how these techniques are used to identify and analyze the causes of waiting times.

3.1.1. On the notions of waiting time

The first question (Q1) of the state-of-the-art review focuses on the analysis of waiting times and their causes using process mining. However, before conducting the analysis, it is essential to first calculate the durations of these waiting times from event log data, i.e., to quantify waiting times. Existing research provides multiple definitions and methods for quantifying waiting times. Therefore, this subsection explores the different notions of waiting time that have been used in existing research to define and quantify waiting time durations. This discussion is relevant to this thesis, as quantifying waiting time durations serves as the foundation for further analysis and identification of their causes.

Existing research offers different definitions of waiting time when quantifying its duration. From a granularity perspective, waiting time can be measured at the level of an entire case or at the level of an individual activity instance [16]. At the case granularity level, waiting time is defined as periods when no work is being performed on the case as a whole. In contrast, at the activity instance granularity level, waiting time refers to periods when a specific activity instance is waiting to be processed, even if other activity instances within the same case are being actively worked on. Thus, the activity instance granularity level offers a more detailed view of where the waiting times occur compared to the case granularity level. In this thesis, we use the activity instance granularity perspective for quantifying waiting times. This approach allows us to measure the specific periods when individual activity instances are waiting. Following this quantification, we then conduct further analysis to identify the causes of these waiting times.

Ali et al. [16] describe five distinct approaches to measuring waiting time: simple waiting time, control-flow waiting time, idle time, shelf time, and customer-perceived idle time. These five notions of waiting times are depicted in Figure 9. The figure illustrates the execution of three cases in a process with three activities (A, B, and C), where activities B and C are parallel, i.e., they can be executed in any order, or their execution may overlap. The most commonly used approach is to measure *simple waiting time*, i.e., an interval between the start time of an

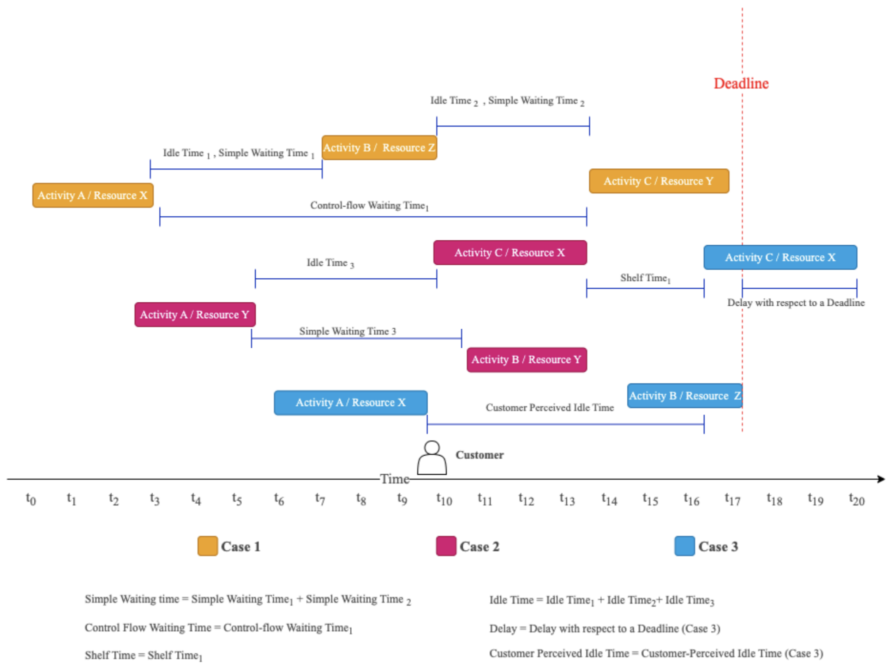


Figure 9. Different notions of waiting times [16].

activity instance and the end time of the most recently completed activity instance of the same case [106, 28, 107, 108, 29, 44, 43, 109, 110]. *Control-flow waiting time* is the interval between the start time of an activity instance and the time it was enabled from a control-flow perspective [111, 112, 113, 114]. For example, in Case 1 depicted in Figure 9, both Activity B and Activity C are enabled (i.e., ready to be executed) when Activity A is completed. Therefore, there are two intervals of control-flow waiting time: (1) between the completion of Activity A and the start of Activity B, and (2) between the completion of Activity A and the start of Activity C. *Idle time* refers to intervals when no work on the case is being done at all [115, 116, 117, 118] and often coincides with simple waiting times. However, in contrast to simple waiting time, idle time considers parallelism, external delays, and other non-activity-related waiting times (as in Case 2 in Figure 9) [16]. The idle time differs from control-flow waiting time in that idle time focuses on the case granularity perspective, identifying periods when no work is being done on the case as a whole. In contrast, control-flow waiting time focuses on the activity instance granularity perspective, identifying periods when individual activity instances are waiting to be executed. The control-flow waiting time of a case is calculated by summing the control-flow waiting times of its activity instances [16]. *Shelf time* corresponds to the waiting time due to external factors, i.e., intervals when the activity instance is enabled, the assigned resource is available, but the activity instance is not started due to external factors [119] (e.g., waiting for a response from a customer [21]). Finally, *customer-perceived idle time* refers to waiting periods observed by the customer [42, 27, 120, 44, 43, 19]. It is quantified as the interval between the end time of one activity instance involving customer interaction and the beginning of another such activity instance within the same case.

To quantify waiting times, this thesis examines waiting time intervals using the definition of control-flow waiting time. Control-flow waiting time is particularly useful as it identifies periods when each individual activity instance waits to be executed once enabled, representing the lowest granularity level among the five notions of waiting times. Therefore, control-flow waiting time can highlight more potential improvement opportunities for optimizing the temporal performance of business processes.

As a result, to enable the waiting time analysis, the waiting time durations need to be first quantified. Waiting times can be measured at different levels of granularity, either at the case level or the activity instance level, with the latter providing a more detailed view. Five distinct approaches to measuring waiting times are available: simple waiting time, control-flow waiting time, idle time, shelf time, and customer-perceived idle time. The thesis specifically focuses on control-flow waiting time due to its ability to quantify periods when individual activity instances are waiting (the lowest granularity level). While we base our research on how the control flow waiting time is quantified, we extend this by identifying the causes of these waiting times.

3.1.2. Data-driven analysis of waiting times

Once the waiting times are quantified, they can be analyzed. Therefore, this section reviews the current state-of-the-art research on process mining approaches for analyzing waiting times. This discussion is particularly relevant to this thesis, as it also focuses on analyzing waiting times using process mining.

The research on process mining for analyzing waiting times can be broadly categorized into descriptive approaches, predictive & what-if approaches, and data refinement approaches [16]. Descriptive approaches focus on extracting statistics and visualizing business processes to analyze waiting times in "as-is" processes [16]. Predictive approaches estimate the remaining time of ongoing cases, including predicting delays [109]. What-if approaches examine the impact of changes in business processes on their temporal performance [16], such as how altering staff schedules can affect resource utilization and waiting times [44]. Data refinement approaches analyze waiting times to improve the quality of event logs [16], e.g., by enhancing the accuracy and precision of timestamps in event logs [121]. In the context of this categorization, this thesis considers descriptive approaches (as we analyze the "as-is" state of waiting times in business processes and their causes) and what-if approaches (as we analyze how the process redesigns could impact the temporal performance).

Following the quantification of waiting times, descriptive process mining approaches facilitate the analysis of waiting times. Thus, multiple case studies demonstrate the application of process mining techniques for analyzing waiting times in different domains. For instance, Uysal et al. [17] present a case study where PM techniques were used to analyze an event log of a production process. In particular, they quantified and visualized waiting times between workstations and identified bottlenecks in the form of the highest waiting times between these stations (see Figure 10). This analysis enabled them to pinpoint critical areas for process improvement.

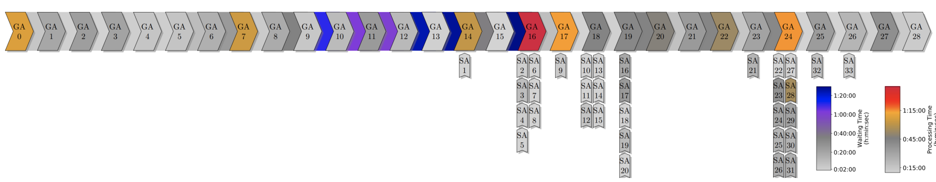


Figure 10. Visualization of median waiting time (blue color scale) and service time (red color scale) on the general assembly (GA) line and subassembly stations (SA). The major bottleneck in the process is formed by the general assembly station GA16 [17].

Erdogan et al. [18] applied PM techniques to analyze waiting times in a hospital emergency process. They identified the highest waiting times between activities and those exceeding expected durations (e.g., waiting times for urgent cases exceeding an average of 30 minutes) as improvement opportunities. Similarly, Singh et al. [19] analyzed waiting times between activities in a healthcare surgical process and identified the highest mean waiting time as a bottleneck, marking it as a key improvement opportunity. Stefanini et al. [122] exploited PM techniques to analyze temporal performance in emergency departments. Among others, they compared waiting times between winter and summer periods and focused on the highest differences to identify areas of improvement, such as when in summer patients waited on average 93 min before the first visit compared to 80 min in winter (see Figure 11).

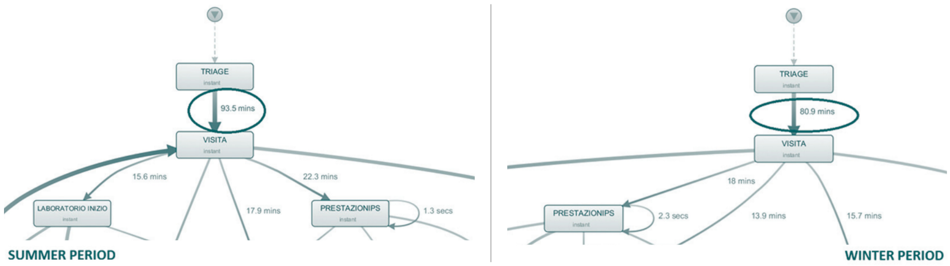


Figure 11. Care pathways of patients in the emergency department in summer and winter periods (key differences highlighted with circles) (excerpt) [122].

Tridalestari et al. [20] combined PM techniques with data mining to analyze waiting times in the e-commerce process. They used process mining to identify before which stages high waiting times are observed, indicating delays. Subsequently, they applied clustering techniques from data mining to determine which specific locations (provinces from where the goods were shipped) were most affected by these delays. Pan and Zhang [22] employed process mining techniques in a case study to analyze a building construction project. In particular, they discovered and examined participant-specific process models to identify inefficiencies, such as delays in transitions between process participants. To identify these inefficiencies, they focused on longer average waiting times in transitions between two participants, which are highlighted with bolder red arrows in Figure 12.

As a result, this review addressed the first part of Q1 ("What process mining approaches and techniques have enabled the analysis of waiting times and their causes?") by focusing on the PM techniques for waiting time analysis. The reviewed studies describe the application of PM techniques to quantify and analyze waiting times. Most of them focus on the highest waiting times as the main inefficiencies and primary targets for improvement. While these studies describe how the waiting times have been analyzed, they do not consider the question of how to attribute waiting times to their causes. Therefore, this thesis extends the existing research by examining how the waiting time causes can be identified from event logs.

3.1.3. Causes of waiting times

When analyzing waiting times, it is crucial for analysts to understand the root causes of waiting in order to identify relevant and effective redesigns [1]. This

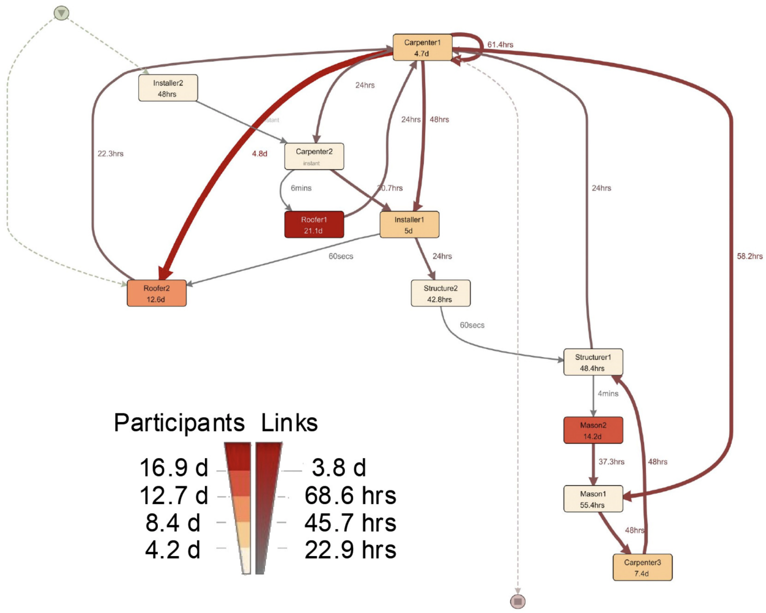


Figure 12. Participant-specific process model of a building construction process from the mean duration perspective [22].

section reviews the causes of waiting times considered in existing studies, such as batching, resource contention, prioritization, and resource unavailability. Additionally, it presents how correlation-based analysis has been applied to identify the patterns occurring in the process that are associated with waiting times. This discussion is particularly relevant to this thesis, as its primary aim is to provide an PM-based approach for discovering and analyzing the causes of waiting times using event logs.

Waiting times can be associated with external and internal sources w.r.t. *the executed process* [16]. Internal sources of waiting time relate to events within the scope of the executed process, e.g., waiting while resources are engaged in activities of the same process. External sources, on the contrary, relate to events outside of the process scope, such as waiting when resources are occupied with activities of a different process within the organization, waiting for a delivery from a supplier, or a response from a customer [21]. Similarly, waiting times can be associated with external and internal sources w.r.t. *the organization* where the process is executed [16]. Thus, internal sources of waiting time relate to the actors inside the organization's scope, e.g., waiting while resources are engaged in activities of the same or different processes in the organization. Consequently, external sources relate to the actors that operate outside the organization where the process is executed, e.g., waiting for input from an external business partner, supplier, or customer [1].

This thesis classifies the sources of waiting times as internal and external w.r.t. *the data captured in event logs* [16]. From this perspective, internal sources of waiting time are those identifiable from the data recorded in event logs and include factors such as batching, resource contention, prioritization, and resource unavailability [16]. In contrast, external sources of waiting time cannot be identified exclusively from event log data due to the absence of necessary information captured in event logs. Identifying such external sources of waiting time requires analysts to access additional data, such as contextual information. Thus, external sources of waiting time include scenarios such as waiting for an external party's input, e.g., customer's response [21], delays caused by exceptions or incidents not recorded in the event log [123], waiting required by procedures (e.g., patients waiting for the treatment to take effect [80, 81]), and waiting when resources are engaged in activities of the same or different processes not documented in the event log [16]. This thesis focuses on the data-driven identification of waiting time causes from event logs and, thus, aims to distinguish and quantify the *internal* sources of waiting times w.r.t. *the event log data*. Further, we discuss the internal sources of waiting times in more detail and review the techniques for their identification and analysis.

Batching occurs when a resource accumulates cases to process them together as a group [26]. Batch processing is often used to solve scheduling problems [124] and reduce processing times and costs [45, 124] by executing multiple cases together. However, introducing batch processing requires accumulating cases, which increases waiting times [45, 125, 126, 116]. In this way, waiting time due to batching occurs when an activity instance is ready to be executed but waits because the resource collects other instances to form a batch. As such, batch processing implies a trade-off between reduced processing times and increased waiting times [127, 116].

Process mining techniques enable the discovery of batch processing behavior from event logs. For instance, Nakatumba [128] mines the resource behavior from event logs (incl. when resources employ batches) and addresses the problem of accurately reproducing batch processing behavior in simulation models. Wen et al. [129] propose a process mining technique to discover batch processing from events logs to discover batch processing workflow models. Pufahl et al. [26] utilize event logs to enhance process models (both visualization and simulation models) by incorporating batch processing into them. Pika et al. [130] propose an approach for discovering batch processing from event logs and discusses, in particular, how to identify batch processing from multiple perspectives of a business process, such as activity, resource, and data perspectives.

Martin et al. [127] proposed a process mining-based approach to discover three types of batch processing based on how activity instances are processed within a batch: in parallel, simultaneously, or concurrently with a partial overlap in processing times. Additionally, they use various metrics, such as frequency of batch processing, batch size, duration of activity instances in a batch, and waiting time of activity instances in a batch, to describe batching behavior. In their subsequent research, Martin et al. [24] extended their approach to discover batch processing behavior at the subprocess level (i.e., when a resource performs a series of activities for a particular case). Furthermore, Martin et al. [25] proposed a technique for identifying rules that trigger batch processing activities. This thesis aims to, among others, identify and analyze waiting times associated with batch processing. Thereby, we build on existing research to identify batch processing activities and their behavior.

Process mining techniques have also been applied to assess batch processing performance and explore their impact on process performance. Thus, in addition to batch discovery, in [131], batch processing behavior is visualized and quantitatively analyzed to identify specific patterns (e.g., how long it typically takes to form a batch in a given batch processing activity) and detect outliers. Similarly, Klijn and Fahland [116] propose several measures for quantifying batch processing performance using PM techniques. These measures include intra-batch case interarrival time (i.e., the time between arrivals of two successive cases within a batch), case interarrival time (i.e., the time between arrivals of two cases considering both batched and non-batched cases), batch interval (i.e, the time between two successive batches are started to be processed), batch size, and batch frequency. Pufal et al. [26] examine how overall process performance can be improved in terms of time (average cycle time) and cost (average cost of process execution) by simulating and comparing different batch processing strategies. While these studies use event logs to analyze the performance of batch processing, they do not specifically focus on the waiting times associated with batch processing. Therefore, we extend existing work by identifying different types of waiting times related to batch processing and their impact on process performance to identify potential improvement opportunities.

Resource contention occurs when an activity instance is ready to be executed but cannot start because the resource(s) required to perform it are occupied with another activity instance [44, 132, 33]. Resource contention often leads to queues, where multiple activity instances accumulate while waiting for occupied resources [1]. Therefore, waiting times related to resource contention are commonly examined in the context of queues. Notably, many case studies examining queues focus on the healthcare domain since queues and long waiting times are common for hospitals [44]. Thus, Antunes et al. [44] apply process mining to analyze queues of patients in an emergency department process. They focused on analyzing waiting times at one particular point of the process, specifically between the patient's arrival and the start of the first activity. They examined the waiting time distribution throughout the day to identify peak times with the longest queues and highest waiting times. Similarly, Petitdemange et al. [132] use PM techniques to analyze waiting times in the phone call queues in the emergency call center process. They analyzed, among others, how long patients typically wait in queues before the phone call is picked up.

Some process mining techniques (often combined with queuing theory techniques) support the identification of queues and waiting times caused by queuing effects, i.e., when activity instances wait in a queue until being processed [133, 27, 44]. For instance, Yampaka & Chongstitvatana [27] describe the use of PM techniques combined with a queuing system to analyze temporal performance in a healthcare process. They employed PM techniques for process discovery and applied queueing theory to assess temporal performance. They noted that customer waiting time in queues was one of the key metrics for their analysis. Nguyen et al. [28] propose a PM-based approach to analyze dynamic flow performance from event logs, specifically examining how queues build up and the amount of waiting time they induce at each stage of the process. In the reviewed studies, since queues were observed, all waiting times were attributed to queuing. Although queuing can indicate resource contention, it is a symptom rather than an underlying cause of waiting times (i.e., we need to understand *what* causes the queue to build up). Additionally, other potential causes, such as prioritization or resource unavailability, which could also contribute to queues and associated waiting times, were not considered in these studies. Therefore, this thesis aims to explore how waiting times can be attributed to multiple causes.

Queuing models are also commonly used for simulation and prediction, in particular, of waiting times [28]. For instance, Whitt [134] proposes a technique to predict each customer's waiting time in the queue before the service is started. Senderovich et al. [135] propose techniques for predicting delays for a running instance of a service process due to the queuing effects. Kyritsis and Deriaz [136] propose using machine learning techniques as an alternative to the queueing models to predict waiting times in queueing scenarios. However, this thesis focuses on the performance analysis and diagnostics of waiting times in business processes to identify their causes, rather than on predicting waiting times.

Prioritization causes waiting times when an activity instance is ready to be executed, but the assigned resource starts a higher priority activity instance that was enabled later (i.e., after the lower priority one) [137, 31, 29]. In this situation, it is assumed that the resources shall follow the FIFO order of processing of the activity instances (i.e., instances enabled earlier are processed first) while some activity instances are prioritized and violate the FIFO order[1]. For example, Premchaiswadi and Porouhan [137] highlight that in the process of manuscript peer review, reviewers may be busy with other commitments and are not able to participate in manuscript revision tasks. This signifies prioritization and subsequently induces waiting times into the review process [31].

Multiple studies indicate that prioritization impacts waiting times in business processes [139, 140, 29, 141]. For instance, in healthcare, the most prevalent prioritization method involves classifying patients into urgency groups, which results in shorter or longer waiting times based on the severity of their condition [140]. Thus, in their performance analysis of emergency room processes, Rojas et al. [29]

[1]We note that other orders of processing (a.k.a. queue disciplines) exist, such as LIFO (i.e., instances enabled later are processed first), processor sharing, and priority queue [138]. However, in this thesis, we consider prioritization as any order of processing except for FIFO as in [16].

distinguish cases with different priorities (five types ranging from low to high severity), which impact the waiting times and overall process duration. Similarly, Thullner and Rozsnyai [30] describe how waiting times differ between "premium" and "regular" customers, with orders from "premium" customers prioritized over those from "regular" ones, resulting in shorter waiting times for "premium" customers. Jäger and Roser [31] examine the impact of prioritization on waiting times and experimentally validate (through simulation) that prioritizing certain orders decreases their waiting times at the cost of increasing the waiting times for non-prioritized orders. Although these studies highlight the dependency between prioritization and waiting times, they do not focus on quantifying waiting times specifically due to prioritization or consider other potential causes of waiting.

Several PM techniques allow identifying prioritization and discovering how the prioritization is executed. For instance, Suriadi et al. [32] propose a process mining technique for discovering the prioritization strategies the resource employs from event logs, i.e., the manner in which resources prioritize their work. More specifically, for each activity and resource, they identify which queueing disciple is employed – FIFO, LIFO, or priority-based, i.e., priority is determined by some pre-determined rules. Additionally, rule-based process mining techniques allow for the discovery of specific patterns in process execution based on frequently observed behaviors, such as specific sequences of activities (e.g., activity x occurs at the beginning of every trace) [142] or batch activation rules [25]. Similarly, rule-mining techniques can be applied to discover patterns according to which activity instances are prioritized. In this thesis, we build on these studies to identify activities where prioritization is employed and to discover specific prioritization rules (i.e., rules according to which cases were prioritized). Additionally, we extend the existing research by identifying waiting times associated with prioritization.

Resource unavailability is a source of waiting times when an activity instance is ready to be executed but cannot start because the resource(s) required to perform it are off duty, i.e., not available according to their working schedules [111, 143, 34, 33]. Aissaoui et al. [33] propose an approach to identify bottlenecks and their causes in complex processes with a high number of activities, resources, and flows, such as healthcare processes. Among their findings, they identify bottlenecks where patients are waiting for doctors to arrive at work (e.i., become available). However, in their approach, the identification of bottleneck causes is performed manually. Toosinezhad et al. [34] introduce a process mining-based approach to detect event patterns that frequently precede dynamic bottlenecks, in particular, the unavailability of a worker or machine. Although these studies recognize resource unavailability as a potential source of waiting time, they do not specifically focus on attributing and quantifying waiting times due to resource unavailability. This thesis addresses this gap by identifying and measuring waiting times specifically caused by resource unavailability.

Several studies focused on mining the resource availability patterns from event log data. For instance, Nakatumba [128] mines the resource behavior from event logs (such as the time a resource is available for a specific process, and the periods of time over which a resource works) and addresses the problem of accurately reproducing resource behavior in simulation models. Lòpez-Pintado and Dumas [13] and Martin et al. [35] suggest techniques for discovering resource availability calendars from event logs describing their working schedules. We build on this research to identify resource (un)availability and further extend it by identifying the associated waiting times.

Several existing studies have also utilized correlation-based analysis to discover the process patterns associated with waiting times. For instance, Ferreira & Vasilyev [21] present a technique to identify why some cases in a process take longer time to complete. They identify case characteristics correlating with higher delays, e.g., when a specific activity occurs in a case, or when a specific resource is involved, the case is likely to have higher waiting times. Likewise, De Leoni & van der Aalst [144] combine some of the existing correlation analysis techniques to identify how different process characteristics correlate with the process performance, e.g., if process deviations cause delays. Similarly, Hompes et al. [107] propose an approach based on time series analysis to detect cause-effect relations between process characteristics and performance indicators, e.g., if the waiting time for the receipt of payment depends on the time of day. Pan and Zhang [22] apply correlation analysis to identify associations between extended waiting times (i.e., delays) and such factors as involved resources, resource roles, handovers of work, and the involvement of external participants. Toosinezhad et al. [34] introduce an approach to detect event patterns that frequently precede (and, thus, might lead to) dynamic bottlenecks. While these studies take a correlation-based approach to analyzing waiting times, this thesis aims to identify and quantify the causes of waiting times and consider their impact on process performance.

As a result, this review addressed the second part of Q1 ("What process mining approaches and techniques have enabled the analysis of waiting times and their causes?") by focusing on the PM techniques for the discovery of waiting time causes. The state-of-the-art has revealed a range of PM techniques that enable the data-driven discovery and analysis of specific factors related to waiting times, such as batching, resource contention, prioritization, and resource (un)availability. Batching has been extensively studied using PM techniques to discover batch processing behaviors and evaluate their impact on process performance. Similarly, resource contention has been explored through process mining and queuing theory techniques to analyze queues and the associated waiting times. PM techniques have also been employed to discover prioritization and its strategies. Additionally, PM techniques have been developed to discover resource availability calendars. However, while resource unavailability has been recognized as a source of waiting, less emphasis has been placed on quantifying its effects. Furthermore, correlation-based techniques have been employed to link specific process characteristics or patterns to waiting times, providing insights into potential causes of higher waiting times (i.e., delays).

Predominantly, the proposed PM techniques focus on identifying the presence of a particular phenomenon (e.g., batching) and its description (e.g., batching strategies) or identifying a singular source of waiting time (e.g., due to queueing effects). This thesis builds on existing research by examining the same potential causes of waiting times: batching, resource contention, prioritization, and resource unavailability. It extends this research by considering all these causes collectively and exploring how waiting times can be attributed to multiple causes. Additionally, the analysis of these waiting time causes is further extended by assessing their impact on temporal process performance using the CTE metric.

3.2. Redesign for waiting time optimization

The second research question (RQ2) of this thesis focuses on identifying redesign options to reduce waiting times in business processes. Therefore, this section reviews the state-of-the-art in response to Q2: "What approaches, frameworks, and techniques have been used to identify redesigns to reduce waiting times?" First, it presents the approaches utilized by analysts to redesign business processes with the goal of reducing waiting times. Second, it reviews the collections of redesign options drawn from literature and practical applications. Finally, this section discusses semi-automated process redesign approaches.

3.2.1. Approaches for process redesign to reduce waiting times

This section reviews the approaches employed by analysts to redesign business processes aimed at reducing waiting times. Specifically, it presents methodologies for process redesign and examines how waiting times have been addressed in domain-specific initiatives. This discussion is relevant to this thesis, as it helps to understand how analysts have identified redesign options to reduce waiting times in real-life scenarios.

To reduce waiting times and improve the temporal performance of business processes, several methodologies can be followed. For instance, Six Sigma methodology suggests applying statistical analysis to identify and eliminate inefficiencies such as defects and variability (e.g., in waiting times) [36]. The Lean methodology offers a set of principles to address waiting times, such as employing Total Productive Maintenance (TPM) [37]. TPM is a comprehensive maintenance strategy aimed at maximizing equipment effectiveness by improving its overall performance, reducing waste, and ensuring zero defects and downtime. Specifically, it includes preventive maintenance, which involves performing regular maintenance tasks to prevent equipment failure before it occurs [145]. TPM can contribute to more stable business processes, particularly by reducing waiting times caused by equipment downtime [145]. Another principle proposed within the Six Sigma methodology is batch size reduction, which aims to streamline operations and reduce lead times [37].

The Theory of Constraints (TOC) implies that every system has at least one constraint (a.k.a. bottleneck), which acts as the limiting factor impeding overall system performance [146], such as prolonged waiting times [147]. The TOC prescribes identifying these constraints and first implementing targeted improvements with existing resources (i.e., "exploit constraint," and then using scheduling techniques to synchronize the entire process with the constraint's capacity ("subordinate all resources to global decision"). Once the constraint's performance is maximized, the next step is to elevate the constraint by involving additional resources such as upgrading equipment and hiring more employees [146] (see Figure 13). Business Process Reengineering (BPR) involves fundamentally rethinking and redesigning business processes to achieve significant improvements in critical aspects such as quality, output, and speed [38]. Most distinctively, BPR often includes a complete overhaul of existing processes [1]. However, while these methodologies provide a framework for identifying inefficiencies, they do not specifically describe how to redesign processes to reduce waiting times based on their causes.

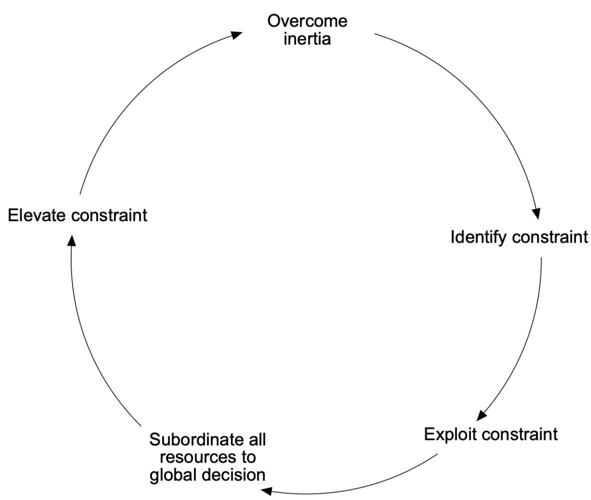


Figure 13. Process of on-going improvement according to TOC [146].

Several research studies, predominantly case studies, feature recommendations on how waiting times can be reduced in specific domains such as healthcare. For instance, in outpatient processes, temporal process performance can be improved through resource capacity planning (to ensure adequate staffing) and appointment scheduling (to optimize patient flow and minimize waiting times) [40]. In the pathology department process, the reduction of patient waiting times by over 50% was achieved by eliminating non-value-added activities, standardizing procedures, balancing workload, and optimizing resource allocation [148]. To improve waiting times in an outpatient process, Lot et al. [39] suggest altering the scheduling pattern, introducing the task tracking system to increase workload visibility, and changing the medical appointment scheduling software. Singh et al. [19] analyzed waiting times in a healthcare surgical process and manually (through interviews with process participants) identified idle periods of the process participants (i.e., when they did not actively participate in task executions). The authors recommended making these resources shared among multiple cases instead of dedicated to a single case. This approach allows resources to begin processing other cases during their idle times (e.g., preparing another patient for surgery), thereby reducing patient waiting times. Yampaka & Chongstitvatana [27] suggest adding more resources to reduce the waiting times of customers in queues. To reduce waiting times in a hospital emergency process, Erdogan et al. [18] suggest performing several activities in parallel (such as conducting patient examinations while simultaneously requesting medical orders), as well as reviewing and refining medical practices.

These studies present various redesign options to modify healthcare processes to reduce waiting times. However, they focus on a specific domain and offer a limited set of redesigns applicable in particular settings. In this thesis, we adopt a domain-agnostic approach, aiming to identify a comprehensive collection of redesign options relevant to addressing the waiting time causes.

3.2.2. Collections of redesign options

Based on the successful process redesigns, existing research has formulated redesign patterns, defined as reusable components that have proven effective in the past and can likely be applied successfully again [149]. Utilizing these patterns can save time and reduce effort since a suitable solution to address process inefficiencies already exists [150]. Consequently, several studies have focused on defining redesign patterns and creating collections of these redesign patterns to assist analysts in their process redesign efforts. Reviewing these collections is relevant because they could potentially contribute to this thesis by offering proven solutions for process redesign aimed at reducing waiting times.

Several collections focus specifically on process *model* patterns that can be applied during process modeling and contribute to process optimization. For instance, van der Aalst et al. [151] described a set of workflow patterns focused specifically on control flow, providing guidance on their implementation. Fellmann et al. [152] conducted an SLR to identify frequently occurring patterns in business process models. Based on the results, they developed a pattern taxonomy summarizing the best practices in process modeling. Complementary to this work, Koschmider et al. [153] examined anti-patterns, i.e., patterns that signal inefficient or counterproductive patterns when modeling business processes and, therefore, to be avoided. Similarly, Becker et al. [154] elicited weakness patterns by examining 30 business process optimization projects in the banking sector using semantic business process modeling language (SBPML). Using these model patterns, analysts can analyze the models of their business processes and identify how these models can be redesigned to optimize business processes. However, process model patterns do not specifically address waiting time issues in business processes and require process models for analysis. In contrast, this thesis focuses on identifying redesign options that target the causes of waiting times, using event logs as the primary source for analysis.

Existing studies provide several collections of process redesign options that can assist analysts in process redesign efforts, extending beyond process models. Reijers and Mansar [45] introduce a framework capturing 29 redesign heuristics (i.e., generic rules to enable process changes) collected from literature and practical experiences. Such heuristics aim to transform a business process from an "as-is" state to a desired "to-be" state [155]. This framework categorizes redesign heuristics according to the business process components they target: customers, business process operation, business process behavior, organization, information, technology, and external environment. For each redesign heuristic, they specify how it impacts process performance (in terms of time, cost, quality, and flexibility), limits to and tool availability for its application, application examples, and references. For example, a redesign heuristic, "Extra resources" (see Figure 14), prescribes increasing the number of resources when current capacity is insufficient. This heuristic targets the resources that execute tasks in the process and, thus, belongs to the category of organization: population. Adding extra resources can reduce process time and increase flexibility, but it also leads to increased costs. The framework highlights that algorithms are available to assist in the optimal allocation of additional resources. Application examples of this heuristic include optimizing resource allocation in a telephone operator company.

Framework elements	Rule name	Impact on BP	Limits	Referred to by	Technique used	Tool availability	Application examples
Organization: population	Extra resources	↘ Time, ↗ flexibility, ↗ cost		Berg and Pottjewijd [38]	Increase capacity if possible, but not if it only moves the bottleneck	None	None
				Van Hee et al. [47]	Discussion of the optimality of several strategies to optimally allocate additional resources in a business process	Algorithms	Example of a telephone operator company

Figure 14. Excerpt from the business process redesign framework [45].

Fehrer [46] complemented the study of Reijers and Mansar [45] by presenting a collection of 50 redesign patterns for process improvement observed in the field (partially overlapping with the redesign heuristics presented by Reijers and Mansar [45]). Similarly, this collection categorizes the redesign patterns based on the business process components they are oriented towards (e.g., customer, organization, operation) and the motivation for redesign (e.g., sustainability, human resource productivity). For each redesign pattern, they provide its definition and detail the impact on process performance in terms of time, cost, quality, and flexibility (in some cases, also sustainability and customer centricity). Additionally, practical guidelines for implementing the patterns, examples of successful applications, and references to further resources are included. For instance, the redesign pattern "Establish standardized interfaces" prescribes using standardized interfaces with customers and partners to reduce errors, incomplete applications, and unintelligible information exchanges. The application of this pattern should ultimately result in faster processing times. In addition to the pattern description, they provide a related case study demonstrating how the check-in process was streamlined at the BPM Conference 2023.

Reijers and Mansar [45] and Fehrer [46] provide collections of diverse redesign options. While these studies focus on how processes can be changed, they do not necessarily detail when these redesign options are relevant to apply, i.e., which inefficiencies or issues can be tackled [156, 155]. Furthermore, several redesign heuristics can be used to address one specific issue [155]. In response to these challenges, Falk et al. introduced a catalog of 20 BPI patterns derived from the four existing collections of redesign patterns (namely, [45, 157, 158, 159]). The notion of a BPI pattern aims to connect a solution (i.e., process redesign) with a problem (i.e., process issue).

A metamodel of a BPI pattern is presented in Figure 15. The metamodel centers around the BPI pattern, identified by a unique name and illustrated with an example. Each BPI pattern is defined by a specific problem, context, and solution. The problem highlights the issues to be addressed, while the context outlines conditions for the pattern's applicability. The solution is linked to a mechanism detailing the instruction of how this solution can be applied (e.g., elimination of activities or modifications of the control flow). Optionally, the solution also provides building blocks, i.e., pre-built models for implementation that can be directly implemented within a process model. The goal of applying a BPI pattern is to achieve improvements, such as cost or time reductions (i.e., effect), which are measured using performance indicators to verify the effectiveness of the changes.

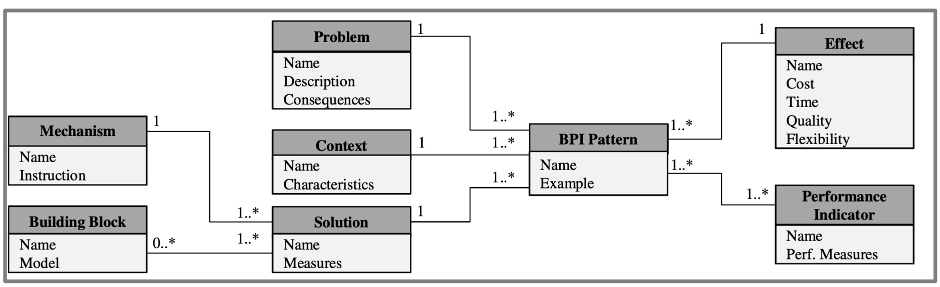


Figure 15. Metamodel of a BPI pattern [155].

According to the defined metamodel, 20 BPI patterns are described. For instance, BPI Pattern 2, named "Divide complex processes into smaller subprocesses," includes an example illustrating a problematic scenario (see Figure 16). The problem description identifies three key issues – a wide range of activities, complex control flow, and various inputs. This is followed by the consequences that these issues induce into the process, e.g., long waiting times. The context specifies which process components, such as activities and inputs, are suitable for redesign. The solution then details how to implement the redesign using the mechanism of control flow modification (i.e., by modifying the control flow). Finally, the expected effects (e.g., shorter waiting and cycle time) and performance indicators to measure (e.g., cycle time) are outlined.

Falk et al. [155] developed their collection of BPI patterns by drawing from four existing redesign pattern collections. They also highlighted that this collection can be expanded further by incorporating redesigns from the broader literature on business process improvement. Following this idea, we conducted an SLR where we reviewed 204 papers describing methods or cases of how improvement opportunities have been identified and how the processes have been redesigned, focusing on the case studies (i.e., describing practical cases of process analysis and redesign) [47]. From the review, 80 improvement opportunities and 64 process redesign patterns were identified. Based on the obtained results, we derived a catalog of defined improvement opportunities and the redesign patterns available to address each improvement opportunity. This collection complements the existing studies by providing a wider range of redesign patterns and bringing more focus on which improvement opportunities can be addressed with particular redesign options. The catalog is organized into four content parts:

1. *Improvement opportunity* part includes the improvement opportunity's name, definition, and an example from the reviewed papers.
2. *Redesign* part covers applicable redesign patterns for addressing the improvement opportunity, their definitions, and examples of their application.
3. *Redesign impact* specifies the process performance dimensions positively and negatively impacted by each redesign pattern, namely time, cost, quality, and flexibility (based on the Devil's Quadrangle framework presented in Section 2.1.2).
4. *References* part contains references to the reviewed papers describing this improvement opportunity and/or redesigns.

An excerpt of the catalog of improvement opportunities and redesign patterns is provided in Figure 17. For instance, the improvement opportunity "Similar process variants (in different facilities)" is defined as the occurrence of two or more process variants that are similar in how they execute cases across different facilities, departments, or locations. This opportunity is illustrated with two examples from case studies in manufacturing and retail company settings. To address this improvement opportunity, two redesign patterns can be applied. Process generalization recommends centralizing similar process variants in one location to reduce duplication of effort. Process standardization prescribes eliminating process variants to ensure uniform execution across different facilities. Further, examples of

BPI PATTERN 2: Divide complex processes into smaller sub-processes

Example:

Within the scope of an invoicing process, the additional tasks of billing working hours and material consumption as well as updating inventories are conducted. As numerous tasks are included in just one process, it is very difficult to manage, and many process instances that have been initialized are pending at a particular time. Since the invoicing process has to wait until all included tasks have been finished, its total cycle time from start to completion is quite long and the proportion of waiting time is extremely high.

PROBLEM:

Description:

Wide range of activities: One process deals with many different tasks and therefore contains a rather large number of activities that are only loosely connected regarding their underlying functions.
Complex control flow: The control flow is characterized by a large number of decision points, branches, and loops making it difficult to determine or follow the routing of a particular business case.
Various inputs: A plethora of inputs that have to be coordinated and provided in time, are required to correctly perform the activities of the process. Due to multiple dependencies, waiting times increase while the availability of necessary input is checked.

Consequences:

Since the process is inflated with activities that are merely loosely related to the process' central task, it becomes complex and difficult to manage. The flexibility of such a process, in the sense of its ability to adapt to changing requirements, is low, as simple change requests can escalate into unmanageable large projects. Because of the confusing process design and situations during a process run, error rates increase. Overall cycle times also increase because frequently making decisions or waiting for input that has to be provided by other activities delays the processing of a business case.

Cost	Time	Quality	Flexibility
neutral	negative	negative	negative
	Longer cycle time Long waiting time	High error rate	Low adaptability

CONTEXT:

Characteristics:

Activities: There are activities within the process that can be performed individually and independently of each other; i.e. an activity A does not need any of the outputs produced by other activities within the process as an input.
Input/Output: The interfaces, regarding input-output-relationships, between the task that is a candidate for extraction and preceding or subsequent tasks are well-defined.

SOLUTION:

Measures:

Divide one long and complex process into several sub-processes in such a way that …
… each sub-process is short and straightforward regarding its design.
… exchange of e.g. material, information, etc. between the resulting sub-processes becomes minimal.
… organizational disruptions within each sub-process become minimal.

Mechanism: Modify control flow
Building Block: -

EFFECTS AND PERFORMANCE INDICATORS:

Applying the pattern splits a complex and overloaded process into smaller logical segments that are far easier to understand and manage. Compared to the original process, the cumulated cycle time of the new process segments is shorter because of fewer waiting times. Error rates decrease due to a clear process specification and a smaller number of dependencies that have to be considered during a process run. Since the interfaces between the sub-processes are minimal and well-defined, each interface can be managed and changed individually which enhances flexibility in regard to change requests.

Cost	Time	Quality	Flexibility
neutral	positive	positive	positive
Cost per process run	Cycle time	Error rate	Reaction time to change requests

Figure 16. BPI pattern "Divide complex processes into smaller sub-processes" [155].

how these redesign patterns are applied in practice are provided, along with references to the reviewed papers.

Improvement Opportunity			Redesign			Redesign Impact		References
Name	**Definition**	**Example**	**Redesign Patterns**	**Definition**	**Example**	**Positive**	**Negative**	
Similar process variants (in different facilities)	Two or more process variants that are similar in how they execute the cases in different facilities/ departments/ locations.	A manufacturing company operates a factory, a headquarters, an international division, and six sales subsidiaries. Each subsidiary has separate but similar processes for logistics, financial, and personnel affairs. [15]	Process generalisation	Having similar processes in several facilities, centralise them into one place to reduce duplication of effort.	The logistics and organisational processes are centralised and transferred to the headquarters, whereas the company focuses only on sales and inventory management. This reduces the repetition of organisational functions and the excess of workers. [15]	Time, Cost, Quality	Flexibility	15, 151, 156, 178, 201
		A large retail organization operates three brands and has over 300 stores across these brands. Each of the brands executes a number of administrative processes differently, including the recruitment process. In the first brand, the recruitment process is done by the HR department at the Head-Office. However, the recruitment process for the second brand was conducted by individual store managers. [151]	Process standardisation	Eliminate process variants.	An optimised process variant is developed to be adopted at all three facilities. The process is also documented to maintain compliance with the organisation's policies. [151]	Time, Cost, Quality	Flexibility	
Similar process variants (for different products)	Two or more process variants that are similar in how they execute the case for different types of products.	In a SIM card ordering process, depending on the SIM card type, the procurement processes are different from each other (different procedures are executed, and different documents are required). [88]	Process standardisation	Eliminate process variants.	A SIM card ordering process is standardised to become product independent, allowing to include new products in the same process easily. Thus, when a new type of card is available, it can be ordered and serviced using the uniform process instead of designing a new process variant. [88]	Time, Cost, Quality	Flexibility	11, 88
		In the electronics retailer support service, the call centre receives calls regarding different types of products. The business machines service centre is responsible for managing business products such as copiers and fax machines. The computer service centre is responsible for servicing computer products such as personal computers, monitors and printers. The system products service centre is responsible for managing system products such as voicemail. Although all three centres provide customer support, each of them services customers only on specific products according to their specialization. [11]	Process generalisation	Having similar processes in several facilities, centralise them into one place to reduce duplication of effort.	The three centres are merged into one consolidated call centre to process customer calls independent of the product type. It implied that all call centre agents would need to be retrained to handle two other centres' calls. The re-engineered process flow has dramatically streamlined the operations and allowed customers of all product types to access the same hotline. [11]	Time, Cost, Quality	Flexibility	

Figure 17. Excerpt from the catalog of improvement opportunities and redesign patterns [47].

These studies provide a comprehensive overview of process redesigns and broad collections of redesign patterns for process improvement. However, in this thesis, we focus specifically on reducing waiting times. Therefore, we base our work on these studies to derive a collection of redesign options for addressing the causes of waiting times and, subsequently, reducing waiting times.

3.2.3. Semi-automated process redesign

Process redesign is predominantly a manual activity executed by process analysts [48]. Recent advancements in process mining have led to the development of semi-automated redesign approaches. This subsection reviews these approaches and the tools that implement them. This review is relevant to this thesis, as it seeks to offer a PM-based solution that can automatically recommend redesign options to reduce waiting times based on the analysis of their causes.

Existing semi-automated approaches use "as-is" process models as input and support analysts in creating and evaluating alternative models. For example, Kim et al. [159] introduced a tool designed to facilitate the redesign of process models. This tool enables the application of 16 workflow patterns, previously defined by van der Aalst et al. [151], to existing process models. Thus, instead of manually restructuring process models, analysts can select and apply patterns like "sequential split" or "activity parallelization," which are automatically implemented within the process model. This tool, therefore, streamlines the modeling of "to-be" processes and partially automates the modeling efforts of process analysts.

Another tool, the PrICE tool kit [49], takes an "as-is" process model (containing information on the control flow, data, resources and performance of the process) as input and highlights the possible redesign opportunities. Further, the tool allows analysts to select specific parts of the process model that can be redesigned with one or more of the applicable redesign operations. Analysts then can apply the available redesign operations (such as compose tasks or parallelize tasks) and, thus, create alternative ("to-be") process models. Finally, the "to-be" model can be simulated to predict its expected performance. By comparing the simulation results, analysts can make a quantitatively supported decision on the best alternative model. In this way, this tool facilitates the redesign of process models and the evaluation of alternative models. Figure 19 shows a screenshot of the PrICE tool's interface where a process part is selected for the application of the compose operation.

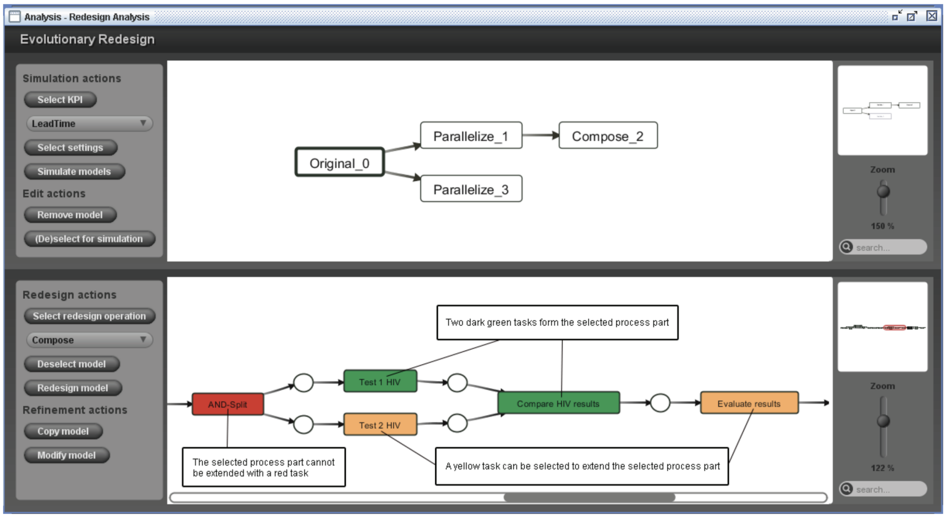


Figure 18. Interface of the PrICE tool [49].

Existing research offers several approaches that enable automated recommendations on process redesign. Thus, Mustansir et al. [48] propose an approach for the automatic extraction of redesign suggestions by classifying end-user feedback available in natural language. This approach utilizes natural language processing (NLP) techniques to analyze feedback and extract relevant redesign recommendations, thereby automating part of the redesign process. Another tool is called assisted business process redesign (aBPR) [50] that allows for automated suggestions on redesigns for "as-is" process models. For its analysis, the tool requires an "as-is" process model, but it also offers the capability to discover process models from event logs. Further, the tool employs algorithms to recommend redesign patterns based on the analysis of "as-is" model structure and performance data (such as "triage" and "parallelization"). The tool supports the 29 redesign patterns from Reijers and Mansar [45]. For instance, the "parallelization" pattern is suggested when the analyst selects the time dimension for improvement and the following conditions are met: activities are in sequence, have no data dependencies on each other, have similar duration times, are executed by resources from different roles (or when multiple resources with the same role are available), and there is no risk of overloading any role by placing the activities in parallel. In the same way, the specifications for all available redesign patterns are provided for the algorithm to consider. Analysts can then select and apply these redesign suggestions to create an alternative “to-be” process model. The simulation component further enables validation of the performance of the alternative model. Figure 19 provides a screenshot of the aBPR prototype’s interface where the redesign patterns of “Parallelism” and “Extra resource” are suggested.

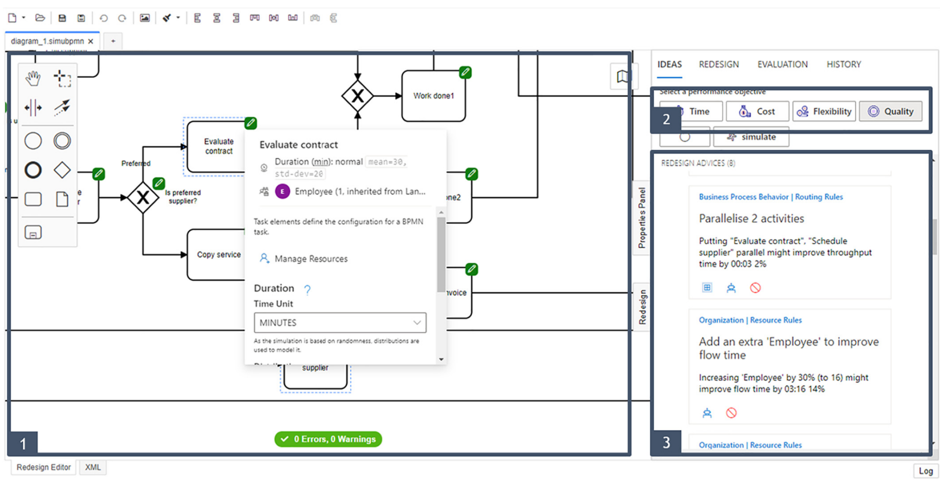


Figure 19. Interface of the aBRP prototype [50].

The proposed approaches and tools facilitate process redesign and partially automate activities conducted by process analysts. However, these approaches and tools still largely rely on inputs from process analysts (e.g., process models and decision interventions), use a limited number of predefined redesign patterns, and do not specifically target the causes of waiting times when offering redesign options. Therefore, this thesis aims to address these limitations by providing an automated solution capable of recommending redesign options to address the causes of waiting times, thereby aiding analysts in optimizing waiting times.

In order to test and validate the potential process redesigns (e.g., whether the redesign will give a desired effect, such as reduced waiting times), analysts can use BPS. Simulation allows for experimenting with various process parameters, such as altering the control flow, changing the number of resources, or adjusting work procedures (e.g., batch processing rules). For instance, Salimifard et al. [42] used PM-based simulation to conduct what-if analyses of different scenarios to reduce waiting times in the emergency department process. Their scenarios included increasing the number of resources (i.e., emergency specialists), replacing busy resources with alternative available ones (e.g., involving an emergency specialist instead of a general practitioner), and combining two activities executed by different resources into one executed by a single resource. These changes were shown to effectively reduce waiting times in process simulation. Similarly, Cho et al. [43] used simulation analysis to test several redesign options for reducing waiting times in an outpatient process. Their strategies included adding extra resources, distinguishing processes based on the case types (e.g., creating separate reservation timeslots for new patients and follow-up patients), and eliminating batch-processing and periodic activities. Simulation results indicated that these strategies were effective in reducing waiting times. Antunes et al. [44] combined process mining with discrete event simulation to optimize waiting times in an emergency department. More specifically, they applied a mathematical model to develop an optimal resource schedule (i.e., for physicians) aimed at reducing waiting times. Further, they used simulation to compare the original process flow and scheduling with the optimized solution. Although process simulation is a valuable tool for testing different redesign scenarios in a controlled environment and validating expected gains, analysts first need to identify which specific redesign options to apply. This thesis aims to address this need and propose a solution that aids in identifying relevant redesign options to address each cause of waiting times.

Additionally, a branch of research focuses on optimizing business processes by addressing waiting time-related inefficiencies associated with specific causes. This includes approaches that leverage advanced solvers using algorithms like logic programming, genetic algorithms, or machine learning-based optimization to find an optimal solution for problems such as resource allocation, scheduling, and workload balancing [160]. These solvers aim to allocate resources in the most efficient way possible to enhance process performance, in particular, minimize waiting times. For instance, Si et al. [161] propose a Generic Genetic Algorithm (GGA) framework that optimizes resource allocation in business processes modeled with Colored Petri Nets (CPN) (see Figure 20). Their approach allows to automate the search for optimal resource schemes by means of simulation, improving efficiency metrics, in particular, waiting times. EL-Rifai et al. [51] focus on solving the human resource availability problem in the emergency department to reduce patient waiting times. To achieve this, they offer a stochastic mixed-integer programming (MILP) model that considers the variability in patient arrivals and service times to create efficient personnel schedules. As a result, the model aims to balance the demand for services with the available human resources (physicians and nurses) to reduce waiting times. Similarly, Daldoul et al. [52] present a stochastic MILP model to reduce patient waiting times by optimizing resource allocation in an emergency department. Their model optimizes the allocation of both human resources (physicians and nurses) and material resources (beds) and allows to reduce total patient waiting time under uncertain demand conditions. While this research addresses waiting time inefficiencies related to a specific cause, focusing on optimal solutions through reallocation and rescheduling, this thesis examines multiple causes of waiting times and aims to offer a broader range of redesign options.

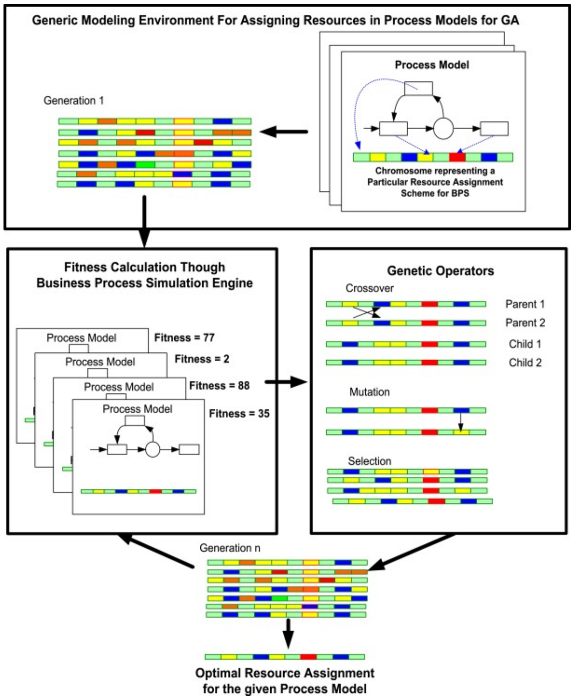


Figure 20. The Generic Genetic Algorithm (GGA) modeling framework for resource optimization in business process simulation [161].

Park and Song [162] propose a method for optimizing resource allocation in business processes through predictive process monitoring. Their two-phase approach involves offline construction of a prediction model with machine learning to estimate scheduling parameters, followed by online optimization using these predictions to improve process performance metrics, e.g., minimizing total completion time. Cals et al. [53] present a Deep Reinforcement Learning (DRL) approach for optimizing order batching and picking in warehouses, aiming to minimize tardy (late) orders, i.e., orders with increased waiting times. The approach enables real-time decisions on whether to pick orders individually or in batches and, if batched, determines the optimal grouping of orders. They introduce an agent that makes these batching and sequencing decisions, interacting with a simulated environment to assess the impact of such decisions. These research efforts concentrate on optimizing specific process patterns (such as batching, resource allocation, and scheduling) within the context of ongoing cases to reduce waiting times (i.e., addressing operational use cases [99]). This thesis, however, focuses on identifying redesign options that modify the process to achieve long-term temporal performance improvements (i.e., targeting tactical use cases [99]).

As a result, this review addressed the Q2 (“What approaches, frameworks and techniques have been used to identify redesigns to reduce waiting times?”), by examining the current solutions available to support analysts in redesigning business processes to reduce waiting times. Traditional methodologies like Six Sigma, Lean, and BPR provide valuable frameworks for process improvement. However, since these methodologies offer a universal approach to process improvement, they do not specifically target waiting time reduction. Domain-specific studies, particularly in healthcare, have effectively applied targeted redesigns such as resource optimization and workflow adjustments to minimize waiting times. However, these studies are limited to specific domains and offer a limited set of redesigns applicable in particular settings. Existing research also offers collections of redesign patterns, i.e., structured catalogs of heuristics that can aid analysts in identifying process redesign options. Yet, these collections encompass redesign options to improve various process performance dimensions and not necessarily to reduce waiting times. Additionally, the redesign options they include may not directly address the causes of waiting times. Recently, several semi-automated tools leveraging process mining and simulation have been developed to assist analysts in process redesign by suggesting and validating process changes. However, these tools often rely on inputs from analysts, support a limited set of predefined redesign patterns, and may not fully consider the causes of waiting times when offering redesign suggestions. This thesis seeks to complement existing research by addressing these gaps and proposing a solution specifically designed to offer automated redesign recommendations that target the causes of waiting times.

4. DISCOVERY AND ANALYSIS OF WAITING TIMES DUE TO BATCHING

Following the first research question RQ1, we aim to identify and quantify the causes of waiting times from event logs. Based on the review of the state-of-the-art, a research gap (RG) is identified in the area of data-driven analysis of waiting times and their causes:

RG1 Existing studies have not introduced an approach to attribute waiting times observed in business process event logs to their causes.

In response, this chapter explores batching as a cause of waiting time. Batching (a.k.a. batch processing) is frequently used to optimize processing times and costs by executing multiple activity instances together [163]. Batch processing is a common practice applied in different domains, such as manufacturing, logistics, healthcare, retail, business administration, and finance [163]. However, while batch processing reduces processing times, it often increases waiting times [116]. If this trade-off between processing and waiting times is not analyzed, batch processing can become a source of inefficiency [37]. Therefore, analyzing batch processing and its associated waiting times is relevant for a broad number of business processes.

Existing PM techniques enable the discovery of batch processing from event logs and its behavior (e.g., batch activation rules [25], frequency of batch processing, and durations of batch processing activity instances [35]). These techniques also allow for assessing the temporal performance of batch processing activities by measuring metrics, such as the waiting time of activity instances in a batch compared to those not in a batch [35], the time between arrivals of two cases in a batch, and the time between the start of two successive batches [116]. Additionally, by simulating and comparing different batch processing strategies, it is possible to examine how overall temporal process performance can be improved [26]. Nevertheless, existing process mining techniques do not specifically focus on examining the waiting times associated with batch processing.

By its definition, batch processing comprises two stages: activity instances are first accumulated to form a batch, after which they are batch processed [26]. Additionally, existing research outlines different types of batch processing regarding how cases are processed: in parallel, consecutively, or with a partial overlap in processing time [127]. Analyzing waiting times at each stage of batch processing, as well as how different batch processing types affect waiting times, can help identify waiting time-related inefficiencies. However, existing research has not yet examined the waiting times associated with each stage of batch processing or how waiting times for activity instances in a batch differ depending on the type of batch processing.

Therefore, this chapter addresses the identified research gap by exploring waiting times associated with batching processing, considering its stages and types.

First, this chapter outlines the research methodology we followed to develop the proposed approach. Second, it conceptualizes different types of waiting times caused at each stage of batch processing – waiting time for batch accumulation, waiting time of a ready batch, and waiting time for other activity instances (in the same batch) to be processed – and compares these waiting times across different batch processing types. Third, it introduces an approach for identifying these types of waiting times due to batching from event logs and discovering improvement opportunities based on the impact of each waiting time type on the CTE. Lastly, this section presents the evaluation of the proposed approach using synthetic event log data and demonstrates the approach's applicability on a real-life event log.

4.1. Research methodology

The objective of this research is to develop an approach to support analysts in analyzing waiting times associated with batching and identifying improvement opportunities related to these waiting times. To achieve this, we followed a DSR methodology [164] as it is suitable for designing and evaluating new artifacts in the field of information systems research [54]. An overview of the research process we followed is presented in Figure 21.

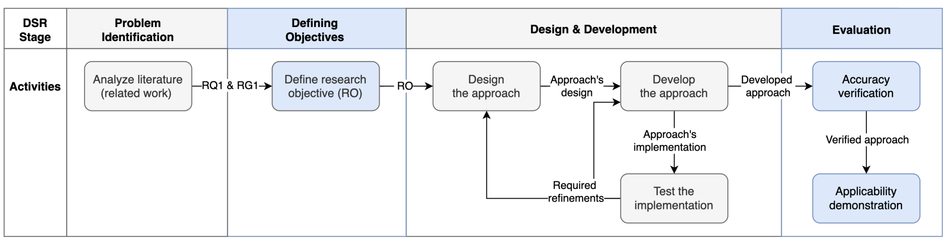


Figure 21. Research process to develop Contribution 1.

First, by reviewing the state-of-the-art related to RQ1, we identified and formulated the research gap (RG1) presented above. As stated in Chapter 3, this review revealed that while existing methods allow for analyzing temporal process performance and potential causes of waiting times, there is a gap in approaches specifically targeting the waiting times associated with batch processing. Based on this research gap, we defined a research objective (RO) to develop a data-driven approach for identifying and analyzing waiting times associated with batch processing from event logs.

Following this objective, we designed and developed the proposed approach. The Design & Development stage was an iterative process. We began by outlining the preliminary draft of the approach design, which defined the key phases of the approach and their components. This draft served as the foundation for our implementation efforts, during which we developed a preliminary version of the approach. When developing the approach, we continuously tested its implementation to identify areas requiring refinement. For example, this testing involved using synthetic event logs with different batch processing types to ensure the approach could identify and analyze the associated waiting times for each type. Similarly, we ensured that at each phase of the approach, all necessary analysis data was obtained to enable analysts to identify improvement opportunities, such as batch processing behavior and the impact of associated waiting times on the process performance. Feedback from these tests was translated into the refinements we made to both the approach's design and its implementation. Such iterations were repeated until no further refinements were identified.

After the Design & Development stage, we proceeded to the Evaluation stage. The evaluation of the proposed approach was conducted in two steps. First, we used synthetic data to validate the ability of the approach to accurately discover batch processing activities, their batching behavior, and different types of waiting times associated with batching (i.e., accuracy verification). Second, we applied the proposed approach to a real-life event log to demonstrate its applicability in practical scenarios (i.e., applicability demonstration).

4.2. Approach

This section presents the proposed approach for the discovery and analysis of waiting times due to batching from the event log data. As input, the proposed approach takes an activity instance log as defined in Section 2.2.1. The log requires to have a case identifier (ID), an activity label, a resource that executed the activity, a start time, and an end time (see Table 3). The start and end times are required to compute the waiting times between activity instances. The resource is required for detecting when an activity instance is being executed as part of a batch, implying that such instances are handled by the same resource [24]. Resources are considered separately, even when they share the same role with other resources. The enabled time of each activity instance is optional. If any of the required elements is missing in the activity instance log, the approach cannot be executed[1].

Table 3. Requirements to the input (activity instance log) of the approach.

Required	Optional	Discovered by the approach
Case ID	Enabled time	Enabled time (if absent)
Activity		Batch processing strategies
Start time		
End time		
Resource		

[1]Note that these are the minimum event log requirements for the approach to function, i.e., to identify batch processing and associated waiting times. However, certain batch activation rules, as discussed in Sec. 4.2.2, might require additional attributes for their discovery.

The approach consists of three phases (see Figure 22). First, given an activity instance log, the batch processing activities – i.e., the activities that are executed in batches – are discovered. The output of this stage is a report listing such activities and their batch processing frequencies (i.e., the proportion of cases processed in a batch w.r.t. their total executions). The second phase aims to analyze their batch processing strategies, i.e., type of batch processing, batch size distribution, and batch activation rules. The result of the stage is a report describing the batch processing strategy per activity. The final phase aims to analyze the time that the cases spend waiting before they are batch-processed. This includes analyzing how batch processing and their associated waiting times impact the CTE. The final output is a report presenting results on batch-processing activities that, based on CTE, can be used to identify improvement opportunities to reduce waiting time due to batching. The phases of the proposed approach are explained in the subsections below.

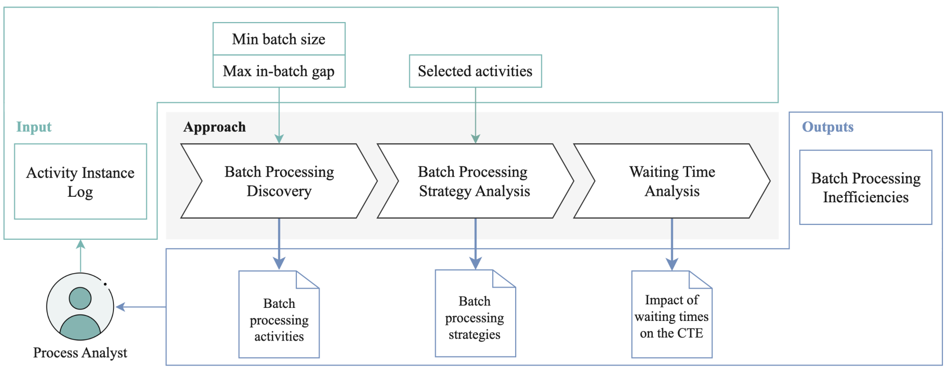


Figure 22. Overview of the proposed approach for the discovery and analysis of waiting times due to batching.

4.2.1. Phase 1: Batch processing discovery

The first stage focuses on discovering batch processing activities, i.e., when a set of instances of the same activity are accumulated to be processed together. For the discovery of batch processing activities, we built on the approach proposed by Martin et al. [24]. In their approach, they use event logs to identify different types of batch processing activities. To do so, they use the resource, start, and end time data to detect when an activity instance is being executed as part of a batch, i.e., by the same resource, and either in sequence, concurrently, or in parallel w.r.t. other activity instances in the batch. In this research, we focus on batches *intentionally* processed as such, i.e., the activity instances were not executed once they were enabled (available for processing) but accumulated and processed as a group. If an activity instance becomes available for processing after the start of the batch, it is excluded as it was not accumulated for batch processing. Accordingly, we extend the definition of a batch with the constraint that all cases that are part of a batch must be enabled before the batch starts. In this context, a batch is a set of activity instances $\mathscr{E}_b \subseteq L$, such that all $\varepsilon_i \in \mathscr{E}_b$ record the execution of the same activity, performed by the same resource, and processed as a batch (i.e. all of them were enabled before any of them started, and they were processed as a group).

To identify when cases become available for processing, we need the enabled time of each activity instance. If the enabled times are missing in the log, we estimate them and enrich the event log with the estimated enabled times. In a sequential process, each activity instance of a case is enabled by the completion of the preceding activity instance. However, concurrency is common in real-life processes. For example, Figure 23 shows the execution of a case with concurrency between two activities. The order of the activity instances is “Register invoice,” “Notify acceptance,” “Post invoice,” and “Pay invoice.” However, “Notify acceptance” and “Post invoice” are enabled when the activity instance “Register invoice” completes. In the same way, “Pay invoice” is enabled only when “Post invoice” is completed. In this way, we consider each activity instance to be enabled by its closest non-concurrent preceding activity instance. Since the first activity instance in a case does not have a predecessor, we assume that its enabled time is equal to its start time. We also assume that once an activity instance is enabled, it remains enabled until its processing starts.

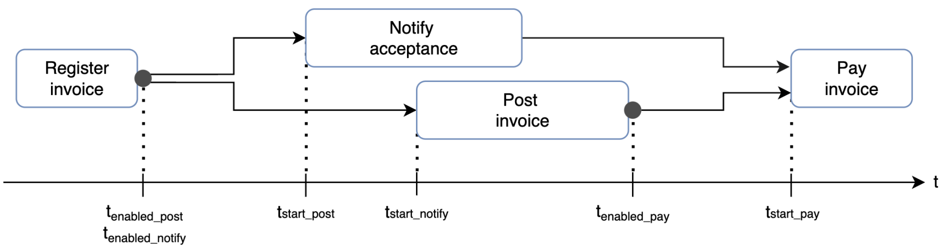


Figure 23. Waiting time in a case with concurrent activity instances.

To detect concurrent activities (e.g., if activities “Notify acceptance” and “Post invoice” are concurrent), in the absence of a user-provided model of the process, we use a concurrency oracle [165]. A concurrency oracle is a function that maps a pair of activities to true if these activities are independent of each other from a control-flow perspective (and thus may be performed concurrently), and false otherwise. A number of concurrency oracles have been proposed[2]. The concurrency oracle of the Heuristics Miner computes, for each pair of activities, related via a directly-follows relation, a coefficient indicating whether these activities are in a concurrent relation or not. This coefficient is computed based on the percentage of times the two activities directly follow each other in either order, e.g., “Post in-

[2]We note that other notions of concurrency have been proposed in the field of process mining. An in-depth treatment of concurrency notions in process mining is provided in Armas-Cervantes et al. [165]. We select the concurrency oracle of the Heuristics Miner [166] because it can be efficiently computed, and it gives results similar to those of more sophisticated concurrency oracles [165].

voice" followed by "Notify acceptance" versus "Notify acceptance" followed by "Post invoice". The concurrency oracle then determines which pairs of activities are concurrent based on certain thresholds.

Based on the outlined criteria, we retrieve, for each activity instance ε_i, its potential batch group ($\mathscr{E}_{b,i}$) composed of all other activity instances $\varepsilon_j \in L \setminus \{\varepsilon_i\}$ of the same activity ($\alpha(\varepsilon_i) = \alpha(\varepsilon_j)$), performed by the same resource ($\rho(\varepsilon_i) = \rho(\varepsilon_j)$), and enabled before its start ($\tau_e(\varepsilon_j) \leq \tau_s(\varepsilon_i)$); such that there are no non-working intervals of time in between any two activity instances in $\mathscr{E}_{b,i}$, i.e., they are all processed at the same time (sharing start and end timestamps), one after the other (the end timestamp of each activity instance coincides with the start of the next one), or one after the other with a slight overlap (the end timestamp of each activity instance succeeds the start of the next one)[3].

Additionally, we allow analysts to filter the results obtained from the batch processing discovery to incorporate their context knowledge about the processes. To perform such filtering, analysts can define a minimum batch size, i.e., the minimum number of cases that should be accumulated and processed as a group for it to be considered a batch (see Inputs in Figure 22). The analyst can also define a maximum time gap between the end of processing of a particular case and the start of the next one within one batch (max in-batch gap). This input allows for the discovery of batch processing when interruptions occur. Such interruptions are common for processes executed by human resources [130], e.g., when a resource needs a break or a set-up for particular cases is required [24].

The output of this step is a report listing the activities where the batch processing occurs. For these activities, we identify their case frequency (i.e., the number of cases in which an activity is processed w.r.t. the total number of cases of the process) and batch processing frequency (i.e., the number of times an activity is processed as part of a batch w.r.t. its total number of executions). The list of batch processing activities is sorted by the batch processing frequency in descending order. An analyst can decide which batch processing activities to analyze further based on the frequencies. For instance, the list might have an activity with only 60% batch-processed cases, but the case frequency might be close to 100%. Therefore, the improvement of such batch processing would target 60% of all cases. On the other hand, the improvement of the activity with batch processing frequency close to 100% (all cases are batch-processed) and case frequency of 5% would affect a relatively small number of cases and, thus, the overall impact would be negligible. Therefore, a combination of batch processing frequency and case frequency indicates the impact of the batch processing activity. Based on this information, the analyst can determine which batch processing activities to analyze further.

[3] We note that batch processing can be identified at the activity level (task-based batch processing) and the subprocess level (case-based batch processing) as in [24]. In this research, however, we consider only *task-based* batch processing since we focus on analyzing associated waiting times at the activity level. For a detailed discussion, refer to the Sec. 4.2.2.

4.2.2. Phase 2: Batch processing strategy analysis

At this stage, we discover the characteristics of how the activity instances are batch-processed in each identified batch processing activity, i.e., batch processing strategies. We describe batch processing strategies with batch processing frequency, size, type, and activation rule/-s.

Batch processing frequency indicates the proportion of cases of an activity that are processed in a batch w.r.t. its total executions.

Batch size illustrates the number of cases processed in a batch. The batch size can be constant (e.g., batch size = 5 cases in 100% of cases) or variable (e.g., batch size = 5 in 40% of cases, batch size = 6 in 60% of cases).

Batch processing type describes how the activity instances are batch-processed. In this research, we discover three batch processing types based on Martin et al. [24]:

- *Parallel batch processing* is when all activity instances included in a batch are processed at the same time.
- In a *sequential batch processing*, activity instances included in a batch are executed one after another.
- *Concurrent batch processing* is when activity instances included in a batch are executed with a partial overlap in processing.

In their research, Martin et al. [24] distinguished task-based and case-based batch processing for sequential and concurrent types. In task-based batch processing, activity instances of one activity are executed in a batch, while in case-based, activity instances of a set of multiple activities are executed for a batch (see Figure 24). In this research, we focus on discovering batch processing and associated waiting times on the activity level of granularity and, thus, consider task-based batch processing. Conversely, case-based batch processing involves batches at the subprocess level (i.e., a set of multiple activities executed as a group), and, therefore, we leave case-based batch processing out of scope. Detecting case-based batch processing would involve addressing two issues: i) identifying subprocesses (i.e., solving the granularity issue); and ii) detecting batches composed of single activities, treating the identified subprocesses as individual activities.

Batch activation rule/-s. We also identify batch activation rule/-s, i.e., the condition(s) that trigger the batch processing [25]. In the batch activation rule discovery, we follow the approach and batch activation rule types proposed by Martin et al. [25]. Thus, we discover the following types of batch activation rules:

- *Volume-based* rules trigger batch processing when a certain volume, weight, or number of cases are accumulated [167, 168, 169, 170, 171, 172].
- *Time-based* rules initiate batch processing based on time triggers, e.g., by scheduled date and time [167, 173], or when a case reaches a waiting time limit [167, 174].
- *Resource-based* rules initiate batch processing based on the resource-

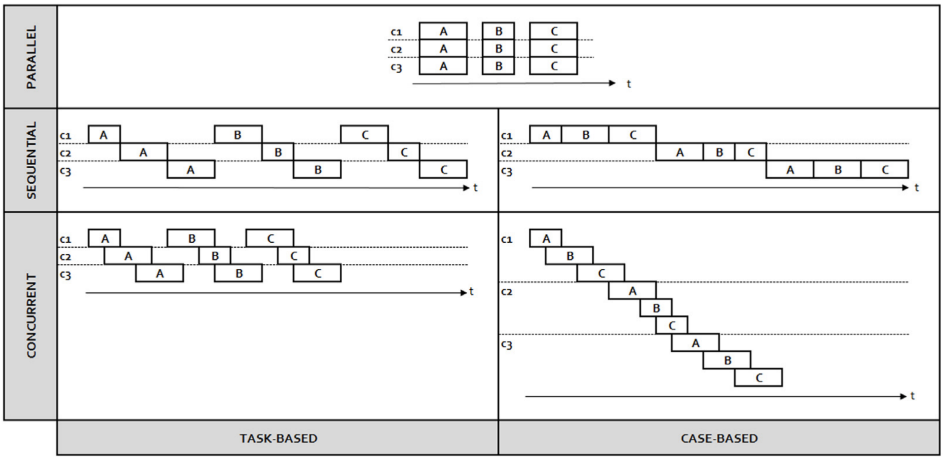


Figure 24. Batch processing types [24].

related attributes, e.g., the batch is processed when the resource workload allows processing this amount of work [25].

- *Case-based* rules trigger batch processing based on the case-related attributes, e.g., the batch is processed when a case of a particular type arrives, such as emergency or high-priority case [174].
- *Context-based* rules embrace other aspects out of the process scope, such as meteorological conditions [25].

Batch can be activated with a single rule, multiple rules, or on an ad-hoc basis. A single rule consists of one condition to be fulfilled to initiate batch processing, e.g., accumulated orders are shipped on scheduled date [167]. Multiple rules present a combination of different types of rules are used to activate a batch processing [175, 174], e.g., the combination of volume and time-based rules [167, 176, 177]: accumulated orders are shipped based either on the scheduled date or when certain weight, volume, or number of cases are collected [167]. However, when no specific pattern that confidently correlates with the start of batch processing can be identified (i.e., no activation rule is assigned [26]), it is assumed that the batch processing is determined by the resources, i.e., executed ad-hoc.

In our approach, the batch activation rules are elicited using the RIPPER algorithm [178]. RIPPER is a rule-based algorithm designed to discover decision rules for classification tasks. It is particularly effective and efficient at handling noisy data and imbalanced datasets and has been successfully applied to the event log data [179]. RIPPER starts by creating an initial rule set through a greedy algorithm, which selects conditions to cover as many instances of the target class as possible. After generating rules, RIPPER prunes them by removing parts that lead to overfitting, using error reduction to simplify the rules. This pruning process is repeated incrementally to refine the rule set. RIPPER handles noisy data by discarding rules that do not generalize well. It also addresses class imbalance by emphasizing rules that better classify the minority class. The algorithm iterates until a set of rules is found that balances accuracy and generalization [178, 180].

In the context of batch processing, RIPPER searches for rules that explain the activation of a batch, i.e., when will a given batch be activated? We construct a feature vector each time that a new activity instance is added to a batch. Each feature vector encodes the end timestamp of the newly added activity instance (day of the week, hour of the day, minute of the hour), the number of activity instances in the batch, and the value of each attribute of the process. We associate a boolean label to each such feature vector: True if the batch is activated (no more activity instances are in the batch), and False otherwise. As output, RIPPER produces a set of rules, such that if any of the rules is true, the batch is activated. In our approach, we use an existing implementation of the RIPPER algorithm[4]. The quality of the discovered batch activation rule is measured by support and confidence parameters.

Note that although the technique is designed to discover all of the outlined types of batch activation rules, their discovery depends on the availability of attributes, based on which these rules are formulated, in the event log. Some batch activation rules require additional attributes beyond the mandatory ones listed in Table 3. For example, with an event log containing attributes specified in Table 3, rules based on the batch size (volume-based rule), scheduled time (time-based rule), or resource workload (resource-based rule) can be identified. However, if batch activation is based on criteria like batch weight (another volume-based rule), arrival of a case of a specific type (case-based rule), or a particular product condition (context-based rule), additional attributes are needed for discovery. In situations when such attributes are missing, analysts are notified that no batch activation rules have been identified.

The output of this stage is a report detailing the batch processing strategies of each batch processing activity. This information serves as input for the next stage and can be used later by the analyst to identify what parameters to alter to improve batch processing.

4.2.3. Phase 3: Waiting time analysis of batch processing activities

In the final phase, we discover and analyze waiting times associated with batching processing activities to identify potential improvement opportunities. In this subsection, we describe the characteristics of the three batch processing types considered by the proposed approach and the waiting times they might induce. We determine three types of waiting times associated with batch processing – waiting time for batch accumulation, waiting time of a ready batch, and waiting time to process other instances in the batch.

Parallel batch processing. In parallel batch processing, a resource starts and completes processing all activity instances of a batch at the same time [24]. First,

[4] `https://github.com/imoscovitz/wittgenstein.git`

activity instances are accumulated, and once accumulated, they are processed. There are two waiting time types in parallel batch processing: *waiting time for batch accumulation* and *waiting time of a ready batch*.

Consider an example of the batch processing activity "Pay invoice" (see Figure 25) where the resource accumulates activity instances and then processes them simultaneously. *Waiting time for batch accumulation* is the time activity instances wait before a batch is accumulated and ready to be processed. A batch is accumulated when all the included activity instances are available for processing. Therefore, a batch is accumulated when the last-arriving activity instance in the batch is enabled. Until the enablement of the last activity instance, all earlier activity instances wait for the batch to be accumulated. Thereby, *waiting time for batch accumulation* ($WT_{accum11}$, $WT_{accum12}$ in Figure 25) is calculated as the difference between the enabled time of the activity instances in a batch and the enabled time of the last-arrived activity instance. In Figure 25, a batch is accumulated when three activity instances of the "Pay invoice" have been enabled. Activity instance $PayInvoice_{11}$ and $PayInvoice_{12}$ wait until the batch is ready to be processed (accumulated). Their waiting time is, therefore, from their enabled times (see $enabled_{11}$ = 10:00:00, $enabled_{12}$ = 10:20:00) until the enabled time of the last activity instance included in the batch, i.e., the third activity instance ($enabled_{13}$ = 11:00:00). The earlier an activity instance is enabled, the longer it waits. The last-enabled activity instance (see $PayInvoice_{13}$) has no waiting time for batch accumulation since its enablement marks the batch as ready to be processed.

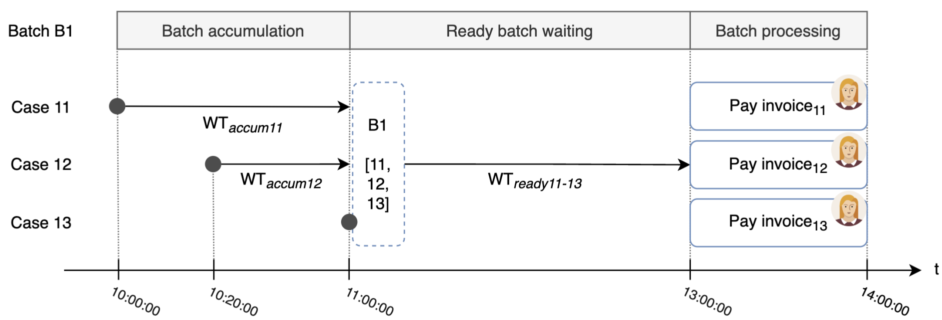


Figure 25. Waiting times in parallel batch processing.

However, a batch might not be processed as soon as it is ready. Thus, there is a second waiting time – *waiting time of a ready batch* ($WT_{ready11-13}$ in Figure 25). *Waiting time of a ready batch* is the time from when the activity instances are ready to be processed (batch accumulated) until they are processed (batch processing starts). Therefore, it is calculated as the difference between the enabled time of the last-arrived activity instance ($enabled_{13}$ = 11:00:00) and the start time of the batch processing ($start_{11-13}$ = 13:00:00).

Sequential batch processing. In sequential batch processing (Figure 26), a resource accumulates a group of activity instances of the same activity and processes them one after another.

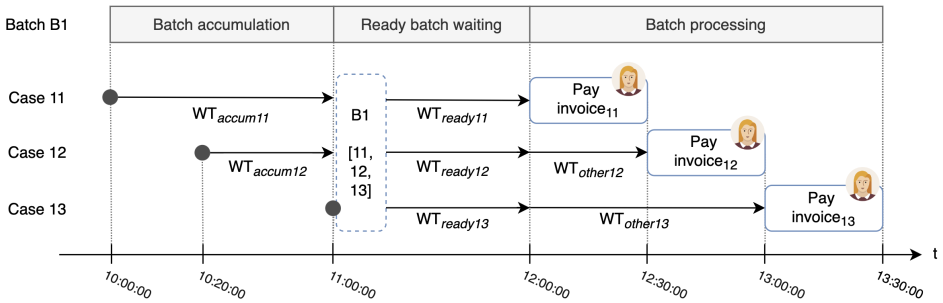


Figure 26. Waiting times in sequential batch processing.

Sequential batch processing has, similar to parallel batch processing, *waiting time for batch accumulation* and *waiting time of a ready batch*. But sequential batch processing also has *waiting time for other instances to be processed* within the batch processing, i.e., the time when an activity instance waits while work is done on other activity instances in the same batch. This waiting time occurs when the resource is processing some activity instance/-s in the batch (e.g., $PayInvoice_{11}$ in Figure 26), other activity instances have to wait for their turn ($WT_{other12}$ for $PayInvoice_{12}$ and $WT_{other13}$ for $PayInvoice_{other13}$ in Figure 26). *Waiting time for other instances to be processed* is calculated as the difference (interval) between the start time of the first activity instance/-s ($start_{11}$ = 12:00:00 in Figure 26) in a batch and the start time of the other instance/-s ($start_{12}$ = 12:30:00 and $start_{13}$ = 13:00:00 in Figure 26).

Concurrent batch processing. Concurrent batch processing (Figure 27) is when activity instances of the same activity are executed in the batch sequentially but with a partial overlap in processing time.

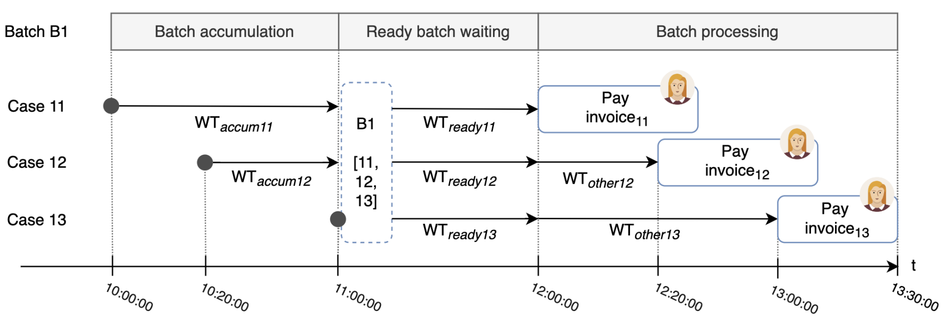


Figure 27. Waiting times in concurrent batch processing.

In concurrent batch processing, the waiting times are the same as in sequential batch processing. Although there is an overlap in processing time, the activity instances still have to wait for their turn to be processed. Therefore, *waiting time* *for other instances to be processed* occurs (e.g., $WT_{other12}$, $WT_{other13}$ in Figure 27). Given the same activity instance processing times, the difference between these batch processing types lies in a shorter *waiting time for other instances to be processed* due to processing time overlaps.

Waiting time analysis. We analyze discovered waiting times to identify potential improvement opportunities. First, we determine how long activity instances wait on average before they are batch-processed. Then, we examine the batch processing efficiency by calculating the CTE of batch processing activity/-ies, i.e., the ratio of the processing time to the cycle time of the batch processing activity/-ies. Finally, we measure the impact of each waiting time type on batch processing efficiency. We express the impact as the potential CTE improvement, i.e., how much improvement of the CTE can be obtained if a particular waiting time type is eliminated.

The first step is to examine the average waiting time per type. We measure, for each waiting time type, the *average waiting time for batch accumulation*, *average waiting time of a ready batch*, *average waiting time for other instances to be processed*, and *average total waiting time*. These metrics depict how long activity instances wait during batch accumulation and processing.

The analysis of waiting times provides the input required to assess the impact of discovered waiting time types on the batch processing CTE. It enables the discovery of batch processing inefficiencies from event logs. The impact of each waiting time (CTE_{im}) on the batch processing CTE is measured as the potential CTE that could be achieved if that specific waiting time is fully eliminated, i.e., the potential CTE improvement. For instance, if the current CTE of a batch processing activity is 50%, and eliminating its *waiting time for a ready batch* increases the CTE to 70%, the potential improvement opportunity is CTE_{im} = 70%.

The output of this step is a report that presents discovered waiting times per type and their impact on the batch processing activity CTE (CTE_{im}). Process analysts can use the obtained results to identify batch processing inefficiencies and from where (what particular waiting time type) the inefficiencies stem. Thus, the results can aid the analyst in identifying which batch processing activities to focus on in their improvement initiatives.

4.3. Computational evaluation

In this section, the computational evaluation of the proposed approach is presented. The proposed approach was implemented as a process-mining technique provided as a publicly available Python package[5]. The technique takes an activity instance log as input and produces a report containing the analysis results of batch processing activities and associated waiting times.

[5] `https://github.com/AutomatedProcessImprovement/batch-processing-discovery`

Using the approach's implementation, we conduct a computational evaluation to address the following evaluation question:

- To which extent is the technique able to detect batch processing activities, their characteristics, and associated waiting times?

To address this evaluation question, first, we use a synthetic event log with artificially injected batches. With this evaluation, we validate the ability of our technique to discover batch processing activities, their batch processing strategies, and different types of waiting times. Second, we apply the proposed approach to a real-life event log to demonstrate its applicability in real-life scenarios. The event logs used for evaluation and evaluation results are available on GitHub[6].

4.3.1. Accuracy verification

For the accuracy verification, we artificially added batches to a simulated event log of a loan application process with the specific purpose of evaluating if our technique was able to detect them. To do this, we first grouped the activity instances of three activities – "Assess loan risk," "Approve application," and "Cancel application" (see Table 4) — out of a total of 18 activities of the process. Then, we assigned a new resource to each group to ensure that the activity instances of a batch did not share their resource with other activity instances. For example, a group of resources called "Fake Clerk" was assigned to execute 78 injected batches in the "Cancel application" activity and included 78 resources where each resource executed only one batch, e.g., "Fake Clerk-5" executed only one batch "Cancel application-5." Finally, we delayed the start and end timestamps of the activity instances of each group (as well as the timestamps of succeeding activity instances) to force their execution as a batch.

To increase the variety in the synthetic evaluation, we added batches of different sizes (10, 12, and 14) and types (parallel, sequential, and concurrent) to three different activities (see Table 4). We left some activity instances unaltered, so not all their executions were processed in a batch (i.e., Batch processing type = none). The batch processing type for a given activity instance was randomly selected from the allowed options. For example, in the "Cancel application" activity, the type was chosen at random from parallel, concurrent, or no batching (i.e., cases were processed individually). Furthermore, for some batches, we designed the batch activation to be performed based on temporal rules, i.e., at a specific time on a specific day of the week. Thus, for the activity "Cancel application," the batch activation rule was set to $[day\_of\_week = 4,\ hour\_of\_day = 10]$. Finally, the waiting times were also altered to create batches with different characteristics, e.g., with the waiting time of a ready batch (WT_{ready} = True) or without it (WT_{ready} = False). The resulting simulated event log comprised 7,000 cases, capturing the execution of 18 activities in a total of 70,162 activity instances performed by 598 resources. In total, 583 batches were injected.

[6] `https://github.com/AutomatedProcessImprovement/batch-processing-analysis`

Table 4. Batch processing injected into the simulated event log of a loan application process.

Activity	Assess loan risk	Approve application	Cancel application
Resource group	Fake Loan Officer	Fake Tom	Fake Clerk
Number of resources	376 resources	129 resources	78 resources
Number of batches	376 batches	129 batches	78 batches
Batch size	14 cases	12 cases	10 cases
Batch processing type	Parallel, sequential, concurrent, none	Sequential, concurrent, none	Parallel, concurrent, none
Batch activation rule	Not assigned	Not assigned	$[day\_of\_week = 4,\ hour\_of\_day = 10]$
WT_{ready}	True	False	True
WT_{other}	True	True	False

We evaluated our approach with this event log, resulting in the discovery of all artificially added batches. Moreover, due to the displacement of the batch processing activities and their successors, batch processing was also detected in other activities. The discovered activation rules corresponded to the designed temporal constraints, indicating the day of the week and hour of activation. Finally, the performance analysis detected the correct waiting times and CTE, accurately reporting cases in which some waiting times were set to 0. The simulated event log and evaluation results are available on GitHub[7].

4.3.2. Applicability demonstration

This section demonstrates how waiting times due to batching can be analyzed by applying the proposed approach to a real-life (activity instance) event log of a manufacturing production process [181]. This log originates from an enterprise resource planning system recording the manufacturing process of metal parts. This process contains automated activities executed by production machines such as “Turning,” “Milling,” “Laser marking,” “Round Grinding” and manually executed work such as quality control-related activities, e.g., “Turning & Milling Q.C.,” “Round Q.C.” and “Final Inspection Q.C.”. The event log has 225 cases recording the execution of 24 activities in a total of 4,503 activity instances executed by 46 resources. This event log covers a span of 3 months of process execution from 02.01.2012 to 31.01.2012. The average case duration in this process equals 2.94 weeks, ranging from a minimum of 30 minutes to a maximum of 2.87 months. This event log was selected for evaluation since it fulfills the input requirements (refer to Table 3) and is sourced from real-life settings, which allows us to demonstrate the approach in a practical scenario.

Batch processing discovery. First, we discovered batch processing activities from the event log. We set the minimum batch size threshold to two cases with no in-batch time gaps (intervals between the activity instance processing within a batch) allowed. The output of this step is a report showing each batch processing

[7] `https://github.com/AutomatedProcessImprovement/batch-processing-analysis`

activity, their case frequency, and batch processing frequency (see Table 5). From the report, we see that, for this event log, 12 activities had batch processing. The activities are sorted in descending order based on batch processing frequency. For each activity, the report shows the case frequency and the batch processing frequency. From the report (Table 5), we note that activities "Lapping," "Packing," and "Turning Rework," have high batch processing frequencies (91.07%, 83.75%, and 66.67% respectively). These cases are predominantly processed in batches and, therefore, relevant for batch efficiency analysis. However, the "Turning Rework" activity is executed for a comparatively negligent amount of cases (for 1.33% of total cases) and, therefore, has little impact on the process efficiency and can be excluded from further analysis. Thus, the report shows that "Lapping" and "Packing" have high batch processing and case frequencies, and therefore, an analyst might decide to select these as input for the next step.

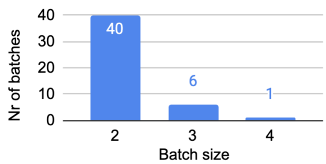
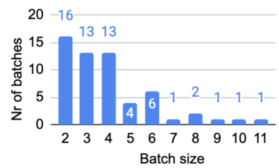
Table 5. Batch processing discovery results.

#	Activity	Case frequency	Batch processing frequency
1	Lapping	58.67%	91.07%
2	Packing	77.78%	83.75%
3	Turning Rework	1.33%	66.67%
4	Turning & Milling	72.00%	45.78%
5	Final Inspection Q.C.	78.22%	35.64%
6	Turning & Milling Q.C.	75.11%	27.78%
7	Grinding Rework	14.67%	23.71%
8	Laser Marking	74.22%	18.25%
9	Round Grinding	49.33%	14.08%
10	Turning Q.C.	12.00%	10.91%
11	Flat Grinding	26.22%	7.02%
12	Turning	10.67%	2.35%

Batch processing analysis. The next step is to analyze the strategies of the selected batch processing activities. The output of this step is a report that captures batch processing type, activation rules, and batch size distribution for each batch processing activity (see Table 6). As can be seen from Table 6, the batch processing activity named "Lapping" follows parallel, sequential, and concurrent batch processing types, i.e., in this activity, activity instances can be processed at the same time, sequentially one after another, with or without overlap in processing time. It indicates that there is no specific style of activity execution. The activity "Packing," on the contrary, follows only a parallel batch processing type, i.e., activity instances in a batch are processed simultaneously.

Next, we discover the batch activation rules. For instance, for "Lapping," batch processing starts from 4 to 5 hours. This rule has a confidence of 0.94 and a support of 0.12. Finally, the report (Table 6) also captures the discovered batch size distribution per activity. Batch size distribution describes the number of cases that are processed in one batch. In this case, for "Lapping," the batch size distribution is 2 to 4 cases. Most commonly, in 85% of batch processing, batches include 2 cases.

Table 6. Batch processing analysis results.

Batch processing activity	Lapping	Packing
Batch processing type	Parallel, sequential, concurrent	Parallel
Batch activation rules	$[hour = 4.0 - 5.0]$	$[hour \leq 5.0]$
Activation rules quality	$Conf. = 0.94,\ Supp. = 0.12$	$Conf. = 0.92,\ Supp. = 0.34$
Batch size distribution		

Waiting time analysis. Having discovered the batch processing strategies, we focus on quantitatively analyzing the waiting times. The output for this step is a report that measures average waiting times per type and their impact on batch processing CTE (Table 7). For each activity, the average processing time (PT in Table 7) and the average waiting times per type are calculated (WT_{accum}, WT_{ready}, WT_{other} in Table 7).

Table 7. Waiting time analysis results depicting the average processing time of a batch processing activity (PT); average waiting time for batch accumulation (WT_{accum}), of a ready batch (WT_{ready}) and for other instances to be processed (WT_{other}); current cycle time efficiency of the activity (CTE_b) and potential cycle time efficiency if the respective waiting time is eliminated (CTE_{im1}, CTE_{im2}, and CTE_{im3}).

Batch processing activity	PT	WT_{accum}	CTE_{im1}	WT_{ready}	CTE_{im2}	WT_{other}	CTE_{im3}	CTE_b
Lapping	0d 1h 50m	4d 4h 39m	2.8%	2d 15h 31m	1.8%	0d 0h 34m	1.1%	1.1%
Packing	0d 1h 0m	10d 8h 34m	0.3%	3d 17h 56m	1.1%	0d 0h 0m	0.4%	0.3%

The waiting times show how long, on average, activity instances wait for the batch to be accumulated and processed. For instance, activity instances wait on an average of 4d 4h 39m while the batch is accumulated for "Lapping." Then, once the batch is accumulated, the instances wait for 2d 15h 31m before the batch processing starts. When activity instances are processed sequentially or concurrently, they wait for an additional 34m on average while other instances are processed. The existing batch processing strategy, therefore, results in a CTE of 1.1% (CTE_b). The CTE value for this activity indicates a potential improvement opportunity. Table 7 shows the impact on the CTE if the waiting times are eliminated. For instance, for "Lapping", if the waiting time for other instances to be processed is eliminated, a slight improvement will be achieved, but the CTE will remain at 1.1% ($CTE_{im3} = CTE_b = 1.1\%$). However, if analysts focus on reducing the waiting time of a ready batch, the CTE could be increased up to 1.8%, corresponding to an improvement of CTE_{im2}. The most significant CTE improvement can be achieved if the efforts are focused on reducing the waiting time for batch accumulation (WT_{accum}) that could improve CTE up to 2.8% (CTE_{im1}).

The analysis demonstrates that attempts to reduce WT_{other} by changing the batch processing type (e.g., use only parallel batch processing in "Lapping") will not add much value. The analysis also shows the reason, i.e., that almost all the waiting time is related to the accumulation and waiting time of a ready batch. We also note from the analysis that the greatest improvements can be achieved by addressing the waiting times for accumulation for "Lapping" and "Packing." This is due to, as the analysis shows, activity instances being processed relatively fast w.r.t. the time they spend waiting for accumulation. Based on the analysis, the analyst can consider discarding batch processing in favor of processing activity instances individually when they are available for these two activities.

4.4. Summary

This chapter proposed an approach for analyzing event logs to identify improvement opportunities related to waiting times in processes with batch processing activities. For this, we extended existing research by defining how waiting times associated with batch processing can be identified from the event log. In addition, we defined three types of waiting times associated with batch processing, namely *waiting time for batch accumulation* and *waiting time of a ready batch* in parallel, sequential, and concurrent batch processing, and *waiting time for other instances to be processed* in sequential and concurrent batch processing. Then, we used these definitions to identify waiting times from event logs and analyze them by measuring the impact of batch processing waiting times on the activity CTE. This can enable analysts to identify where there are improvement opportunities and where to target the process changes.

The evaluation of the proposed approach has shown that its implementation can accurately identify batch processing activities, their characteristics, and associated waiting times. The evaluation with a real-life event log has demonstrated the approach's applicability in a practical scenario. The evaluation results suggest that the proposed approach can offer analysts insights into potential improvement opportunities within batch processing.

The proposed approach has several limitations. The approach uses all observations available per batch for the discovery of batch activation rules and reports on confidence and support. This process can be improved by establishing training/test partitions to discover and validate the rules. Furthermore, the applicability of the approach is limited to event logs containing resources, start and end timestamps. If any of these data are missing from the log, the approach cannot be executed, and the analyst is informed accordingly. In addition, the non-working periods of resources are part of the computed waiting times. We lift this limitation in the following chapter, where we add calendar information to improve the accuracy of identified waiting times.

The evaluation of the proposed approach also has several limitations. The evaluation could be expanded by incorporating additional event logs with varying characteristics. This limitation is addressed in the following chapters, where we increase the number of event logs used in the evaluation. Furthermore, the accuracy verification could be enhanced by exploring the impact of varying hyperparameters (i.e., minimum batch size and maximum in-batch gap) on batch identification accuracy. This limitation is addressed in the following chapter, where we perform a batch rediscovery experiment with different hyperparameter settings. Additionally, while the applicability of the proposed approach is demonstrated, it has not yet been evaluated by potential users, such as process analysts. This limitation is addressed in the following chapter through an interview study with process analysts experienced in process mining, assessing the usefulness and usability of the approach for discovering and analyzing multiple causes of waiting times, including batching.

5. DISCOVERY AND ANALYSIS OF OTHER WAITING TIME CAUSES

Contribution 1 presented in Chapter 4 partially addressed RQ1 by examining different types of waiting times associated with batch processing and proposing an approach to identify and analyze them from event logs. However, a review of the state-of-the-art reveals that other sources of waiting times exist beyond batching. Ali et al. [16] identify resource contention, prioritization, and resource unavailability as additional sources of waiting times that can potentially be detected from event log data.

In the existing research (reviewed in Chapter 3), several studies have applied a correlation-based approach to identify process aspects correlating with higher waiting times, e.g., associations between extended waiting times (i.e., delays) and certain involved resources [22]. Process mining techniques have also been proposed for identifying specific behavior patterns that could potentially cause waiting, such as queues [28], prioritization [32], and resource availability [128]). These techniques can also discover their descriptions (e.g., prioritization strategies [32] and resource working schedules [13]), and identify a single source of waiting times (e.g., due to queuing [28]). However, none of these techniques focuses on identifying waiting times directly associated with these behavior patterns (i.e., potential causes). Furthermore, these techniques do not consider multiple causes of waiting times or allow for their discovery from event logs. Therefore, this chapter continues addressing the research gap (RG) identified in the area of data-driven analysis of waiting times and their causes:

RG1 Existing studies have not introduced an approach to attribute waiting times observed in business process event logs to their causes.

Specifically, this chapter proposes *Contribution 2*: a process mining-based approach to decompose observed waiting times in each activity transition into multiple direct causes and to analyze the impact of each identified cause on the process CTE. First, this chapter outlines the research methodology we followed to develop the proposed approach. Second, it conceptualizes the causes of waiting times in activity transition that can be identified from an event log. Then, it introduces a process mining-based approach to (1) discover waiting times associated with activity transitions, (2) decompose observed waiting times in each activity transition into direct causes, and (3) identify improvement opportunities by analyzing the impact of each identified cause on the process CTE. Further, this chapter introduces the implementation of the proposed approach in the software tool called Kronos. Finally, it presents the approach's evaluation (using its implementation – Kronos) in terms of accuracy, applicability on real-life logs, usefulness, and usability. First, we have validated the accuracy of Kronos in quantifying waiting time causes using synthetic event log data. Second, we have demonstrated how Kronos can be applied on a real-life event log. Lastly, we have conducted a user study with process analysts to evaluate the usefulness and usability of Kronos when following the proposed approach of analyzing the waiting time causes.

5.1. Research methodology

The objective of this research is to develop an approach to support analysts in analyzing the causes of waiting times and identifying improvement opportunities related to these waiting times. To achieve this, we followed a DSR methodology [164]. An overview of the research process is presented in Figure 28.

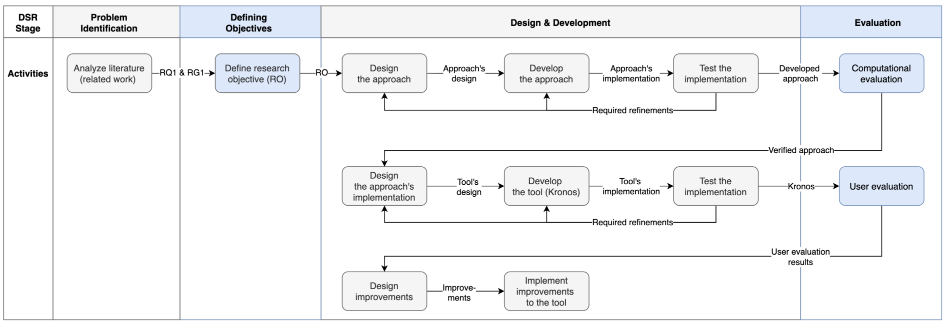


Figure 28. Research process to develop Contribution 2.

First, we reviewed the state-of-the-art related to RQ1 and the identified research gap (RG1) presented above. As stated in Chapter 3, this review revealed that while existing process mining techniques can discover specific patterns that can induce waiting and their characteristics, and in some cases, the waiting times associated with these patterns, these techniques do not specifically attribute waiting times in activity transitions to their causes and consider multiple causes of waiting times. To address this research gap, we formulated the research objective (RO): to develop a data-driven approach for decomposing the waiting times in activity transitions into their direct causes from event logs and analyze their impact on the temporal process performance.

Following this research objective, we designed and developed the proposed approach through an iterative Design & Development process. We began by outlining the initial design of the approach, and then created a draft implementation and identified necessary refinements. In subsequent iterations, these refinements were incorporated into the design and implementation. This process was repeated until no further refinements were identified. Afterward, we conducted a computational evaluation of the proposed approach to verify its accuracy and applicability. First, we assessed the accuracy of the approach in quantifying the causes of waiting times using a set of synthetic event logs. Second, to validate its applicability in practical scenarios, we applied the approach to a real-life event log.

After verifying the approach, we designed and implemented a software tool called Kronos to embody the proposed approach. This was conducted in the second phase of the Design & Development stage. The Design & Development process for Kronos was also iterative, involving cycles of identifying and incorporating refinements into the tool's design and implementation. To evaluate Kronos, we conducted an interview study with process analysts to assess its usefulness and usability in identifying improvement opportunities related to waiting times in a real-life event log. We analyzed the feedback from the user evaluation to identify potential improvements for the tool. These improvement ideas served as inputs for the third phase of the Design & Development stage. In this phase, we designed how the improvements could be integrated into the existing implementation and updated the implementation of Kronos accordingly.

5.2. Approach

This section describes the proposed approach to discover and analyze the causes of waiting times in a business process. As input, the approach requires an activity instance log as defined in Section 2.2.1. The log, as a minimum, must include a case identifier (ID), an activity label, a resource that executed the activity, a start time, and an end time (see Table 8). The start and end times are required to compute the waiting times between activity instances. The resource is required to detect when an activity instance is being executed as part of a batch and to discover prioritization and resource working schedules. Resources are considered separately, even when they share the same role with other resources. If any of the required elements is missing in the activity instance log, the approach cannot be executed. The enabled time of each activity instance is optional. If the enabled times are missing in the log, we estimate them as defined in Section 4.2.1 and then, enrich the log with the estimated enabled times.

Table 8. Requirements to the input (activity instance log) of the approach.

Required	Optional	Discovered by the approach
Case ID	Enabled time	Enabled time (if absent)
Activity		Batch processing strategies
Start time		Prioritization strategies
End time		Resource working schedules (calendars)
Resource		

Figure 29 depicts an overview of the three main steps of the approach. In the first step, we discover the transitions between activities and their characteristics – total frequency, case frequency, and total waiting time – from the activity instance log. In the second step, we identify the causes of waiting time in each transition. In the third and final step, we analyze the impact of each cause on the temporal efficiency of the process (by calculating the maximum improvement in the CTE when the specific cause is eliminated).

When identifying the waiting times and analyzing their potential causes, our proposal makes two assumptions:

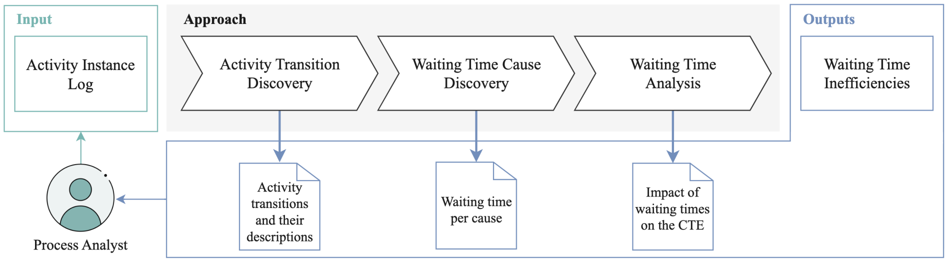


Figure 29. Overview of the proposed approach for the discovery and analysis of waiting time causes.

Assumption 1. *Activity instances are processed uninterruptedly.*

Although the processing of an activity instance is sometimes paused due to interruptions (e.g., the resource going temporarily off-duty), event logs typically store only the start and end timestamps of activity instances. Identifying interruptions in the processing of activity instances is orthogonal to the problem addressed by our proposal and, thus, it is out of the scope of this research. Accordingly, the proposed approach considers the entire interval from the start to the end of an activity instance as its processing time.

Assumption 2. *Resources can work on, at most, one activity at a time.*

This assumption establishes a maximum capacity per resource of one activity, i.e., there is no multitasking. Accordingly, our approach assumes resources cannot start processing a new activity instance if they are already working on one. This is typically assumed when analyzing the availability of a resource in a process [182, 110, 183]. However, our proposal could also be adapted to consider multitasking by discovering the maximum capacity per resource. Given this capacity, a resource is considered as busy when the number of activities they are performing equals their capacity. Discovering the capacity of resources is orthogonal to the contribution of this thesis and, thus, falls outside its scope.

The phases of the proposed approach are explained in the subsections below.

5.2.1. Phase 1: Activity transition discovery

The first step of our approach is to discover transitions between activities and associated waiting times. We define an *activity transition instance* as a pair of activity instances $\langle a_1, b_1 \rangle$ in a single case, such that the completion of a_1 enables b_1, i.e., b_1 cannot be executed before a_1 is completed. We call the first element of an activity transition instance the *source activity instance*, while the latter is the *target activity instance*. An *activity transition* is a set of activity transition instances with the same source and target activities, where the *source activity* is the activity executed in all its source activity instances, and the *target activity* is the activity executed in all its target activity instances. For example, the activity transition $\langle a, b \rangle$ is composed of the set of activity transition instances $\{\langle a_1, b_1 \rangle, \langle a_2, b_2 \rangle, \ldots \langle a_n, b_n \rangle\}$. For simplicity, we refer to activity transitions as *transitions* and to activity transition instances as *transition instances*.

As defined in Section 4.2.1, in a sequential process, each activity instance of a case is enabled by the completion of the preceding activity instance. When concurrency between activity instances occurs, each activity instance is enabled by its closest non-concurrent preceding activity instance. Referring to the previously provided example in Section 4.2.1 (provided again in Figure 30 for convenience), we can observe the concurrency between two activities – "Notify acceptance" and "Post invoice." "Notify acceptance" and "Post invoice" are enabled when the activity instance "Register invoice" completes, "Pay invoice" is enabled when "Post invoice" is completed. Accordingly, there are only three activity transitions in this example, namely $\langle RegisterInvoice, NotifyAcceptance\rangle$, $\langle RegisterInvoice, PostInvoice\rangle$, and $\langle PostInvoice, PayInvoice\rangle$.

Once the transition instances are discovered, we calculate their *duration*, i.e., the waiting times they induce. The waiting time of a target activity instance in a transition instance is the interval between its enablement and its start time. For example, in Figure 30, the waiting time in the transition $\langle PostInvoice, PayInvoice\rangle$ corresponds to the interval $(t_{enabled\_pay}, t_{start\_pay})$. In this way, we identify the waiting time of each target activity instance per transition.

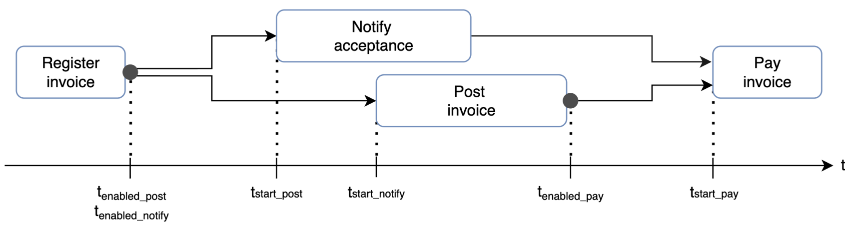


Figure 30. Waiting time in a case with concurrent activity instances.

Finally, we compute the following characteristics for each identified transition (composed of all its transition instances): case frequency, total frequency, and total waiting time. *Case frequency* illustrates the proportion of process cases from the total number of cases where this transition is observed. *Total frequency* indicates the number of occurrences of this transition in the process. *Total waiting time* is the sum of the waiting times of all instances of this transition. The output of this step is a report depicting all identified transitions and their characteristics, sorted by total waiting time in descending order. Based on this information, the analysts can see what transitions cause the highest waiting times and how frequently they are executed.

5.2.2. Phase 2: Waiting time cause discovery

Once the activity transitions and their characteristics are discovered, we analyze the waiting time of each transition instance and identify their causes.

Given the (activity instance) event log information, we consider that an enabled activity instance can wait for *i)* other activity instances to be enabled (so they are processed together) or *ii)* the assigned resource to become available. Below, we analyze each of these situations and relate them to the direct causes of waiting time. *i)* When an activity instance waits for another activity instance to be enabled, we observe a batch processing behavior, and thus, *waiting time due to batching*. *ii)* If an activity instance is not waiting for this reason, it might wait for the assigned resource. The assigned resource might be busy processing other activity instances, enabled before or after the waiting activity instance. Thus, we observe *waiting times due to resource contention* or *due to prioritization*, respectively. If the resource is not busy, they might be unavailable due to working schedules, causing *waiting time due to resource unavailability*. Finally, if there is waiting time that cannot be explained by any of the above causes, we consider the cause to be *due to extraneous factors*, i.e., causes that cannot be identified from the log. Accordingly, we propose to target five causes of waiting time: batching, resource contention, prioritization, resource unavailability, and extraneous factors.

The waiting time within a given transition instance may stem from one or multiple causes – e.g., for half the waiting time, the resource was busy with another activity, and for the remaining half, the resource was off-duty. If there are multiple waiting time causes for a given transition instance, we decompose this waiting time into non-overlapping time intervals and attribute each interval to one cause using the decision procedure in Figure 31. According to this decision procedure, we first identify if any intervals of the transition duration are caused by batching. Then, we look for intervals of waiting time caused by resource contention and prioritization, followed by resource unavailability and extraneous factors. This order is determined by the dominance relations between these causes. Batching dominates resource contention, prioritization, and unavailability, because regardless of the availability status of any given resource, an activity instance that is part of a batch is not ready to be assigned (and started) until the batch is ready. Resource contention and prioritization dominate resource availability, because if a resource has a work queue, they cannot start an activity instance until the latter reaches the front of the queue, or until this activity instance has the highest priority among all activity instances in the queue, regardless of the resource's availability status. Extraneous factors are dominated by all other causes, as they act as a "catch-all" cause for any waiting time that cannot be attributed to other causes.

Note that since each interval of waiting time is attributed to a single cause, this approach ensures the identified waiting times are additive, i.e., the sum of the waiting time causes is equal to the total waiting time of the transition instance.

The proposed approach considers both human and non-human resources. Thus, each waiting time cause can be associated with human and non-human resources. For instance, batch processing can be initiated by human resources or automated systems that cause waiting time due to batching. Similarly, prioritization may be carried out by human operators following procedural guidelines or

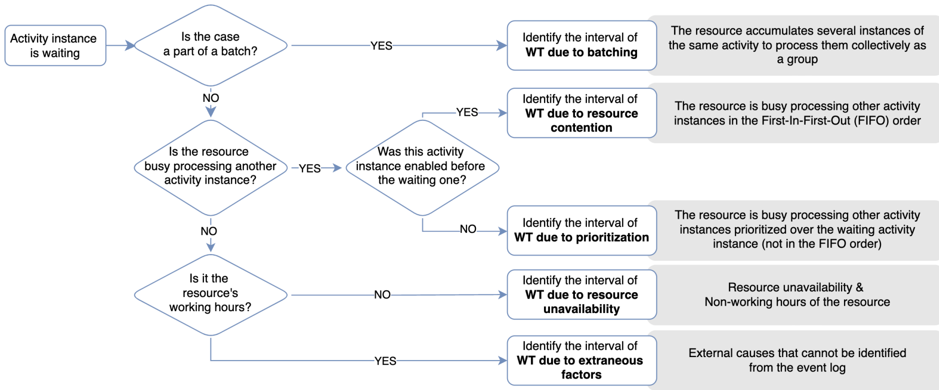


Figure 31. Overview of the waiting time cause discovery process and their definitions.

by programmed rules within the system. Queues of cases can form when human and non-human resources are busy processing other cases, potentially resulting in waiting time due to resource contention. Waiting time due to resource unavailability can also be associated with activities executed by human and non-human resources or a combination of both. Thus, human resource availability is determined by their working calendars. The availability of non-human resources operated by human resources, such as production machines, is dependent on the working calendars of the human operators. Finally, resources executing fully automated activities are considered constantly available (based on the discovered patterns of their constant availability). Occurrences, when resources are temporarily unavailable despite their expected availability (e.g., due to interruptions of human resources, machinery breakdowns, or service unavailability caused by server issues), are not systematic and do not fall into the discovered availability patterns. Therefore, such occurrences are attributed to waiting time due to extraneous factors.

When designing the proposed approach, we considered alternative choices and made several design choices. In particular, when defining the waiting time due to prioritization, we consider prioritization when an activity instance is not processed according to the FIFO order. Alternatively, prioritization could be defined based on other prioritization rules discovered from the event log, e.g., when the customer type is "premium" or high priority diagnosis in the emergency room [29]. In this thesis, we define the waiting time due to resource contention as the waiting time caused when an activity instance waits because other enabled activity instances are given priority according to the FIFO policy. The waiting time due to prioritization is then the waiting time arising when an activity instance a_1 waits for another activity instance a_2 to be processed, and a_2 became enabled after a_1 (hence this waiting time is caused by any prioritization policy other than FIFO). Extraneous causes for waiting time can be due to resources being shared with other processes and, therefore, not available for task execution within the process under analysis. Although object-centric process mining aims at addressing this issue [184], such information is absent in the event log under analysis and, thus, cannot be discovered as a cause of waiting time. Additionally, waiting time due to extraneous factors could be further decomposed into specific causes, such as waiting for a response from a customer [21] or a delivery [185]. Furthermore, waiting time could stem from scheduled tasks, i.e., task execution is triggered by, for instance, a scheduled appointment [43]. Waiting time can also be associated with regulatory delays, e.g., when a produced item needs a cooling down period before proceeding to the next step [186]. However, such information is typically not available in event logs and, thus, cannot be discovered as a direct cause of waiting time. In our proposed approach, all the mentioned situations are attributed to extraneous factors. A more detailed overview of the state of the art w.r.t. the possible waiting time causes is presented in [16].

Further, we describe how the waiting time causes can be identified from an (activity instance) event log: we define each waiting time cause in turn and specify how it is discovered within an activity instance log.

Waiting time due to batching. The first cause that we identify is *waiting time due to batching*. As we defined in Section 4.2.1, batch processing occurs when a set of instances of the same activity are accumulated to be processed together (either simultaneously or one after the other). To identify batch processing, we use the batch processing discovery technique presented in Section 4.2.1. Given an (activity instance) event log containing the resource, enabled time, start time, and end time information of each activity instance, it identifies the groups of activity instances that were first enabled, and then executed one after the other (i.e., executed as a batch). In batch processing, when an activity instance is enabled and ready to be processed, it can wait for other instances of the same activity until the batch is accumulated, i.e., until all instances that are part of a batch are collected. Accordingly, when the target activity instance of a transition instance is detected as part of a batch, the waiting time interval from its enablement time to the batch accumulation time is classified as *waiting time due to batching*. The waiting time due to batching of an activity instance is then defined as follows:

Definition 1 (Waiting Time Due to Batching). Given an activity instance log L, a batch $\mathscr{E}_b \subseteq L$, and an activity instance $\varepsilon_i \in \mathscr{E}_b$, the waiting time due to batching of ε_i is $\omega_{ba}(\varepsilon_i) = (\tau_e(\varepsilon_i), \tau_{bc}) \mid \tau_{bc} = max(\{\tau_e(\varepsilon_j) \mid \varepsilon_j \in \mathscr{E}_b\})$, i.e. the interval of time between the enablement of ε_i and the last enablement of the activities in the batch.

Consider the transition instance between "Post invoice" and "Pay invoice" of case `511` in the running example (Figure 32) with a waiting time between 10:14:15 and 17:55:40. The activity "Pay invoice" is a batch-processed activity where the resource accumulates instances and then processes them one by one (sequential batch processing) [24]. The batch is accumulated until the last activity instance of "Pay invoice" is enabled, i.e., case 513 is ready to be processed ($t_{enabled}$ = 12:00:00). Therefore, if we analyze the transition instance $\langle PostInvoice, PayInvoice\rangle$ of case `511`, its waiting time due to batching corresponds to the interval between 10:14:15 and 12:00:00.

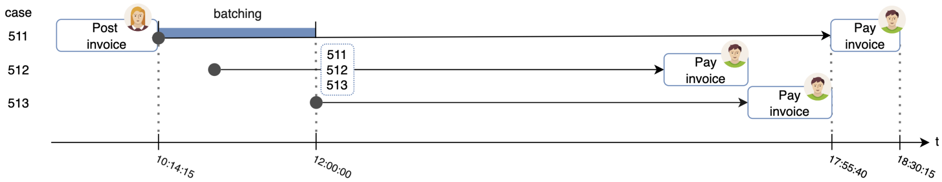


Figure 32. Waiting time due to batching.

In the context of *Contribution 1* (an approach for the discovery and analysis of waiting times due to batching), the *waiting time due to batching* defined here fully corresponds to the *waiting time for batch accumulation* defined in Section 4.2.3. Subsequently, *waiting time of a ready batch* and *waiting time for other instances to be processed* are further attributed to their direct causes, explaining why activity instances wait when the batch is accumulated.

Waiting time due to resource contention. There are situations where the resources that have to process a certain activity are busy processing other activity instances that were enabled earlier than the waiting one, and thus, it's understood that they start processing those before the current one (following a FIFO order)[1]. In such a case, if a resource is busy with another activity when it's needed, the waiting time counts as *waiting time due to resource contention*. Therefore, the waiting time due to resource contention of an activity instance is defined as follows:

Definition 2 (Waiting Time Due to Resource Contention). Given an activity instance log L, and an activity instance $\varepsilon_i \in L$, the waiting time due to resource contention of ε_i is $\Omega_{rc}(\varepsilon_i) = \{(\tau_i, \tau_j) \mid \tau_i = max(\tau_e(\varepsilon_i), \tau_s(\varepsilon_j)) \wedge \tau_j = min(\tau_s(\varepsilon_i), \tau_c(\varepsilon_j)) \wedge \varepsilon_j \in L \wedge \varepsilon_j \neq \varepsilon_i \wedge \rho(\varepsilon_j) = \rho(\varepsilon_i) \wedge \tau_e(\varepsilon_j) \leq \tau_e(\varepsilon_i) \wedge pt(\varepsilon_j) \perp \omega(\varepsilon_i)\}$, i.e. the set of intervals of processing time of all ε_j of L (executed by the same resource as ε_i, and enabled before it) overlapping with the waiting time of ε_i.

Coming back to the running example, during the transition instance between "Post invoice" and "Pay invoice" of case `511`, the resource works on case `514` that has an earlier enabled instance of "Post invoice" (see Figure 33). Therefore, there is waiting time due to resource contention between 12:00:00 and 13:00:00.

Waiting time due to prioritization. However, the resources might not always follow the FIFO policy. In some situations, the resources might give priority to certain activity instances over others. We call this behavior *prioritization*, meaning that an activity instance is processed out of turn w.r.t. a FIFO policy, thus causing other activity instances to wait longer. When this situation occurs, we

[1] As we assume that resources work only on one activity at a time (see Assumption 2), we foresee that the proposed estimation technique will not be suitable for event logs with a high proportion of multitask activity instances.

classify those intervals in which the resource that performed the activity instance was working on another activity instance enabled after it as *waiting time due to prioritization*. Therefore, the waiting time due to prioritization of an activity instance is defined as follows:

Definition 3 (Waiting Time Due to Prioritization). Given an activity instance log L, and an activity instance $\varepsilon_i \in L$, the waiting time due to prioritization of ε_i is $\Omega_{prior}(\varepsilon_i) = \{(\tau_i, \tau_j) \mid \tau_i = \tau_s(\varepsilon_j) \wedge \tau_j = min(\tau_s(\varepsilon_i), \tau_c(\varepsilon_j)) \wedge \varepsilon_j \in L \wedge \varepsilon_j \neq \varepsilon_i \wedge \rho(\varepsilon_j) = \rho(\varepsilon_i) \wedge \tau_e(\varepsilon_j) > \tau_e(\varepsilon_i) \wedge pt(\varepsilon_j) \perp \omega(\varepsilon_i)\}$, i.e. the set of intervals of processing time of all ε_j of L (executed by the same resource as ε_i, and enabled after it) overlapping with the waiting time of ε_i.

In Figure 33, when the resource starts processing the batch of cases `511-513`, instead of processing cases in their order of enablement, he prioritizes cases `512` and `513` over `511`. Waiting time due to prioritization then corresponds to the interval between 15:55:50 and 17:55:40.

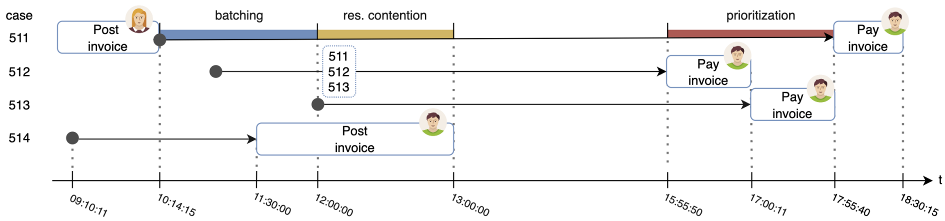


Figure 33. Waiting time due to resource contention and due to prioritization.

Waiting time due to resource unavailability. The fourth cause of waiting time we propose to consider is resource unavailability, which corresponds to the intervals in time in which the resource is not available for work due to their working schedules. To identify this waiting time, we need to first discover the working schedules of the resources. We propose to use the technique presented in [13] to discover calendars over time granules not fully described by the input data. This technique first retrieves, for each resource, the instants in time when each resource interacted with the system, i.e., the start and end timestamps of each activity instance. Then, following a circadian approach, it aggregates and counts the timestamps recorded in each hour of the day, for each day of the week (from Monday to Sunday). Finally, to avoid the detection of wrong working periods due to the presence of noise or outliers (e.g., delays in the recorded timestamps due to temporarily saturated systems), the technique receives one confidence and one support threshold as input, and retains only the weekly hours containing enough events to satisfy those values[2]. In this way, we obtain a weekly calendar defined on an hourly basis per resource, based on the evidence recorded in the event log[3].

[2]In this study, we used the default support (0.7) and confidence (0.1) values recommended in [13].

[3]Note that, although we propose to use a specific resource calendar discovery algorithm, this

Given the weekly calendars of each resource, we transform them into absolute time intervals to compare them with the waiting times observed in the log. Then, we classify those intervals where the resource is not available for work as *waiting time due to resource unavailability*. Therefore, the waiting time due to resource unavailability of an activity instance is defined as follows:

Definition 4 (Waiting Time Due to Resource Unavailability). Given an activity instance log L, an activity instance $\varepsilon_i \in L$, and being $cal_{av}(\rho) = \{(\tau_{avs}, \tau_{avc})\}$ a resource availability calendar with the set of time intervals in which the resource is available to work, the waiting time due to resource unavailability of ε_i is $\Omega_{unav}(\varepsilon_i) = \{(\tau_i, \tau_j) \mid (\tau_i, \tau_j) \in \omega(\varepsilon_i) \wedge \nexists(\tau_k, \tau_l) \in cal_{av}(\rho(\varepsilon_i)) \mid (\tau_i, \tau_j) \perp (\tau_k, \tau_l)\}$, i.e. the set of intervals of the waiting time of ε_i that do not overlap with the availability calendar of the resource $\rho(\varepsilon_i)$ that executed ε_i.

In the running example (Figure 34), the resource executing "Pay invoice" does not work between 13:00:00 and 15:00:00. Thus, in the transition instance of case 511, the respective waiting time interval is due to resource unavailability.

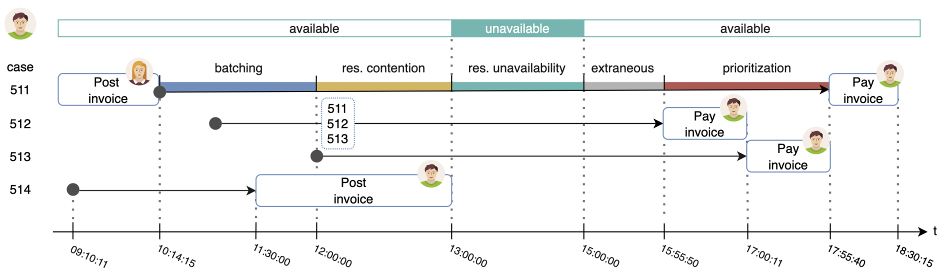


Figure 34. Waiting time due to resource unavailability and due to extraneous factors.

Waiting time due to extraneous factors. The last cause of waiting time that we propose to consider is "extraneous factors". We propose to classify as *waiting time due to extraneous factors* the waiting time intervals caused by the external effects that cannot be identified from the event log – e.g., the resource is working on another process, the activity instance cannot start until some unrecorded event has occurred, fatigue effects, or context switch. In Figure 34, the interval between 15:00:00 and 15:55:50 cannot be explained by the data available in the log and is considered due to extraneous factors. Thus, the waiting time of an activity instance due to extraneous factors is defined as follows:

Definition 5 (Waiting Time Due to Extraneous Factors). Given an activity instance log L and an activity instance $\varepsilon_i \in L$, the waiting time due to extraneous factors of ε_i is $\Omega_{extr}(\varepsilon_i) = \{(\tau_i, \tau_j) \mid (\tau_i, \tau_j) \in \omega(\varepsilon_i) \wedge \nexists(\tau_k, \tau_l) \in (\{\omega_{ba}(\varepsilon_i)\} \bigcup \Omega_{rc}(\varepsilon_i) \bigcup \Omega_{prior}(\varepsilon_i) \bigcup \Omega_{unav}(\varepsilon_i)) \mid (\tau_i, \tau_j) \perp (\tau_k, \tau_l)\}$, i.e. the set of intervals within the waiting time of ε_i that does not overlap with waiting times due to batching, resource contention, prioritization, or resource unavailability.

---

is modular to our proposal. As such, other calendar discovery algorithms and manually defined calendars can also be used with our approach.

In this way, each transition instance observed in an activity instance log is automatically decomposed in a set of intervals attributed to the defined causes. As a result, the approach generates a report that, for each transition execution, outlines: the activities between which the transition occurred (`source_activity` and `destination_activity` in Table 9), the resources that executed those activities (`source_resource` and `destination_resource`), the case for which the transition instance was executed (`case_id`), the total waiting time within the transition instance (`wt_total`), and a breakdown of the total waiting time into intervals based on the causes (i.e., `wt_batching`, `wt_contention`, `wt_prioritization`, `wt_unavailability`, and `wt_extraneous`). For example, activity transition instance #42 in Table 9 corresponds to the example depicted in Figure 34. It shows an activity instance transition of case `511` (`case_id`) between "Pay invoice" (`source_activity`) executed by Sarah (`source_resource`) and "Pay invoice" (`destination_activity`) executed by Jack (`destination_resource`). The total waiting time within this transition instance is 27,685.0 seconds (`wt_total`) (i.e., 7 hours, 41 minutes, and 25 seconds from 2012-03-25 10:14:15 until 2012-03-25 17:55:40). This total waiting time is further decomposed into intervals per cause. Thus, for example, waiting time due to batching (`wt_batching`) equals 6,345.0 seconds (i.e., 1 hour, 45 minutes, and 45 seconds) and corresponds to the interval between 10:14:15 and 12:00:00 in Figure 34. Similarly, waiting time due to resource contention (`wt_contention`) is 3,600.0 seconds (i.e., 1 hour) and corresponds to the interval between 12:00:00 and 13:00:00 in Figure 34. The same applies to the remaining causes of waiting time, i.e., waiting time due to prioritization (`wt_prioritization` = 7,190.0 seconds from 15:55:50 until 17:55:40), due to resource unavailability (`wt_unavailability` = 7,200.0 seconds from 13:00:00 until 15:00:00) and due to extraneous factors (`wt_extraneous` = 3,350.0 seconds from 15:00:00 until 15:55:50).

Table 9. Fragment of a sample report with the decomposition of waiting times into their causes (waiting times in seconds).

Transition instance	...	#42	#43	...
start_time	...	2012-03-25 10:14:15	2012-03-25 14:28:00	...
end_time	...	2012-03-25 17:55:40	2012-03-25 17:22:00	...
source_activity	...	Post invoice	Pay invoice	...
source_resource	...	Sarah	Jack	...
destination_activity	...	Pay invoice	Notify acceptance	...
destination_resource	...	Jack	Carolyn	...
case_id	...	511	517	...
wt_total	...	27,685.0	10,440.0	...
wt_batching	...	6,345.0	0.0	...
wt_contention	...	3,600.0	5,640.0	...
wt_prioritization	...	7,190.0	900.0	...
wt_unavailability	...	7,200.0	3,600.0	...
wt_extraneous	...	3,350.0	300.0	...

5.2.3. Phase 3: Waiting time cause analysis

This section describes how the waiting time causes can be analyzed to identify improvement opportunities in a business process. This corresponds to the final step of the approach (see Figure 29), where we analyze how much each waiting time cause contributes to the temporal performance of the process.

With that purpose, we propose to compute the percentage of time that each cause induces in the CTE of the process. In this context, we measure the CTE as the processing time divided by the sum of the processing and the waiting time (Processing Time (PT) + Waiting Time (WT)), where PT is the sum of the processing time of all activity instances in the process, and WT is the sum of their waiting times. The impact of waiting times per cause is calculated as the difference between the original process CTE and the CTE if a particular waiting time is eliminated. In this way, we can measure (1) the impact that each waiting time cause has on the process CTE, (2) the impact that each transition has on the process CTE, and (3) the impact that each waiting time cause has in each transition. These metrics can indicate the potential CTE improvement (i.e., the theoretical improvement) if a particular cause of waiting time is addressed.

As a result, we can analyze the discovered waiting time causes and their impact on CTE. With this information, process analysts can identify where the inefficiencies due to waiting times are localized, choose which transitions and/or waiting time causes to address, and which redesign alternatives to apply. In this way, causes associated with higher waiting times would have a higher impact on the process CTE and, thus, indicate where improvement efforts could be targeted.

This approach could facilitate identifying waiting time causes and their influence on the process CTE. While the analysis identifies waiting time causes, analysts use their process expertise to determine if these require improvement and what those improvements might be.

5.3. Implementation

In this section, we present the implementation of our proposed approach as a software tool called Kronos. Analysts can use Kronos to upload activity instance logs and obtain analysis results of waiting time causes that could help them identify improvement opportunities to increase temporal performance.

To design Kronos, we elicited requirements. The requirements are derived from the proposed approach. For instance, requirements relating to visualizations of analysis results were designed based on outputs generated at each step of the approach. Here, we considered the target users of the tool (process analysts familiar with process mining) and their goals (understanding the waiting time causes and identifying improvement opportunities related to waiting times) [187]. In addition, we created a mockup of the user interface using Figma[4]. These requirements

[4] `https://www.figma.com/`

constituted the specifications taken as input for the development of Kronos.

Kronos is developed as a React web application, publicly available at `http://kronos.cloud.ut.ee`. A set of event logs for testing is available on Owncloud[5]. The implementation of Kronos's logic is available in a GitHub repository[6], along with instructions for its installation and command-line usage.

Figure 35 gives an overview of Kronos's components. The components comply with the steps of our proposed approach described in Section 5.2. Thus, Kronos takes an activity instance log as input, identifies activity transitions, calculates their durations (waiting times), discovers and quantifies the causes of the waiting times, measures their impact on process CTE, and, finally, visualizes the results. In addition, we use the batch processing discovery algorithm[7] (proposed in Chapter 4) and prioritization discovery algorithm[8] to identify how batch processing and prioritization are executed if waiting time due to these causes is discovered (see Discovery of Batching and Prioritization Strategies in Figure 35). Thus, if waiting time due to batching is discovered, for each batch processing activity, we identify its batch processing strategy (case frequency, batch processing frequency and type, activation rule, batch size distribution, and distribution of waiting time due to batching per resource). In case of waiting time due to prioritization, for each activity with prioritization, we identify its prioritization strategy (case frequency, prioritization frequency, prioritization rule, and distribution of waiting time due to prioritization per resource). This information can help analysts redesign batching and prioritization strategies (e.g., change batch sizes or adjust prioritization rules) to reduce waiting times due to batching and prioritization.

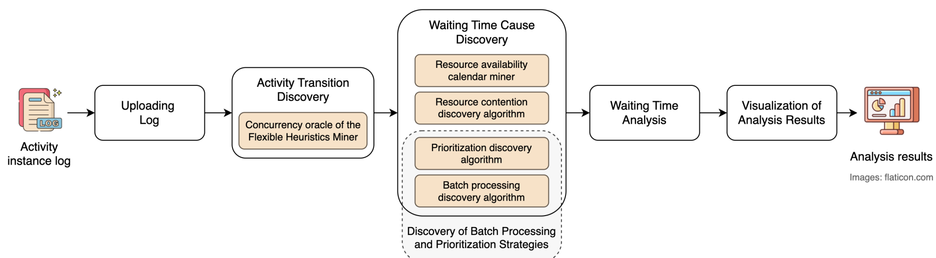


Figure 35. High-level architecture of Kronos.

The results are visualized as a dashboard. Figure 36 and Figure 37 show the Overview tab on the Kronos dashboard. The dashboard first shows key statistics of the process: (1) total number of cases, activities, and transitions; (2) the total/average cycle, processing, and waiting times; (3) the distribution of total/average waiting time by its causes and (4) over the timeframe that the event log captures (see Figure 36).

[5]`https://owncloud.ut.ee/owncloud/s/rZ4dSoTzwpwfpci`
[6]`https://github.com/AutomatedProcessImprovement/waiting-time-analysis`
[7]`https://github.com/AutomatedProcessImprovement/batch-processing-discovery`
[8]`https://github.com/AutomatedProcessImprovement/prioritization-discovery`

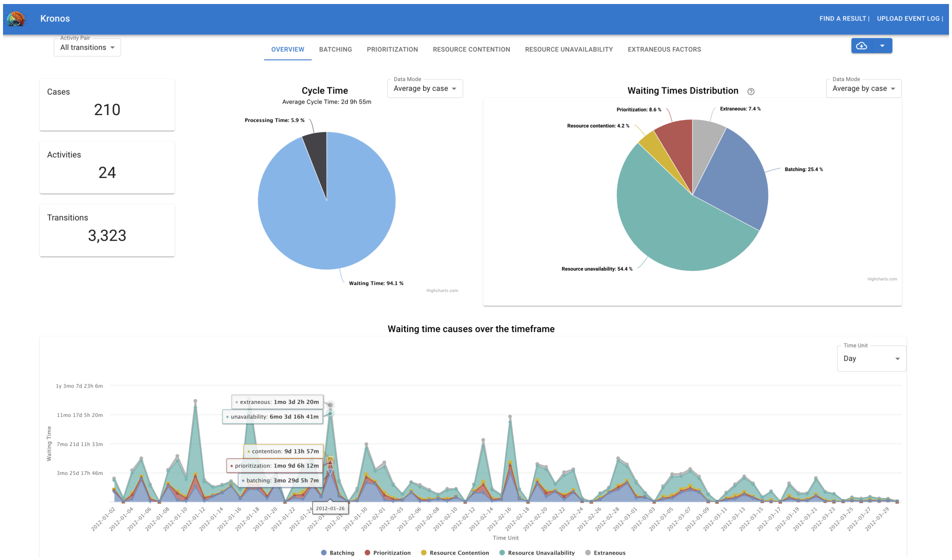


Figure 36. The Overview tab on the Kronos dashboard.

Then, the dashboard presents the waiting time causes per transition (see Figure 37). Finally, it depicts the potential CTE improvement if the waiting times are eliminated in all activity transitions of the process (e.g., if waiting time due to batching is completely eliminated in all transitions) and individually for each activity transition (e.g., if waiting time due to batching is eliminated in the transition between the activities “A” and “B”).

Further, analysts can focus on analyzing each waiting time cause in more detail. For that, they can switch the dashboard to view the analysis results focusing on a selected cause (e.g., waiting time due to batching) (see Figure 38). In addition to the aforementioned analysis results, analysts can see resources associated with the waiting time cause and information specific to that cause, e.g., batching and prioritization strategies. Analysts can also select to view the analysis results for all transitions or one particular transition. The analysis results can be downloaded in CSV and JSON formats.

The initial version of Kronos (before implementing improvements presented in Section 5.5.3) was demonstrated at the ICPM’23 conference [57].

5.4. Computational evaluation

In this section, we present the computational evaluation of the proposed approach. First, we use synthetic data to validate the ability of the technique to accurately discover waiting time causes we injected in the event logs. Second, we demonstrate the approach’s applicability on a real-life event log.

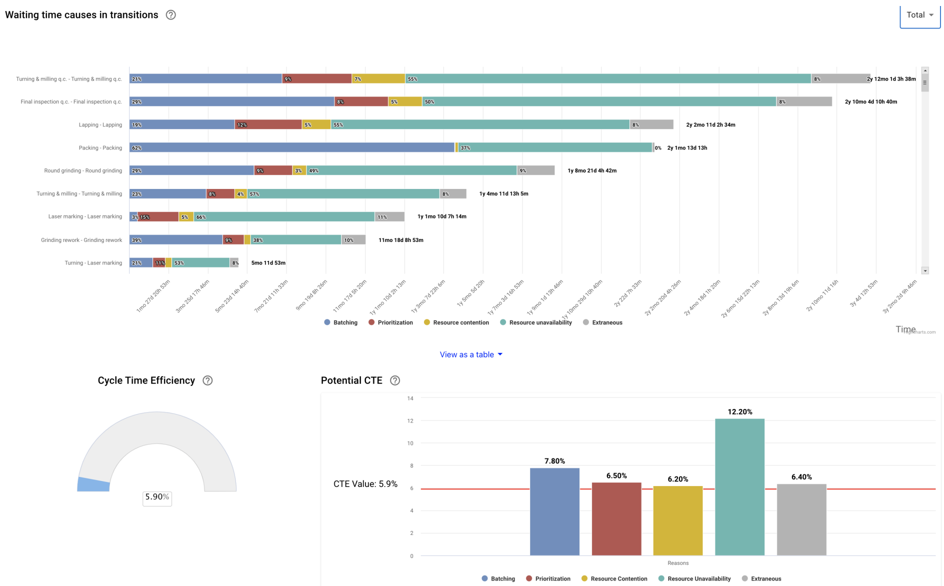


Figure 37. The Overview tab on the Kronos dashboard (continued).

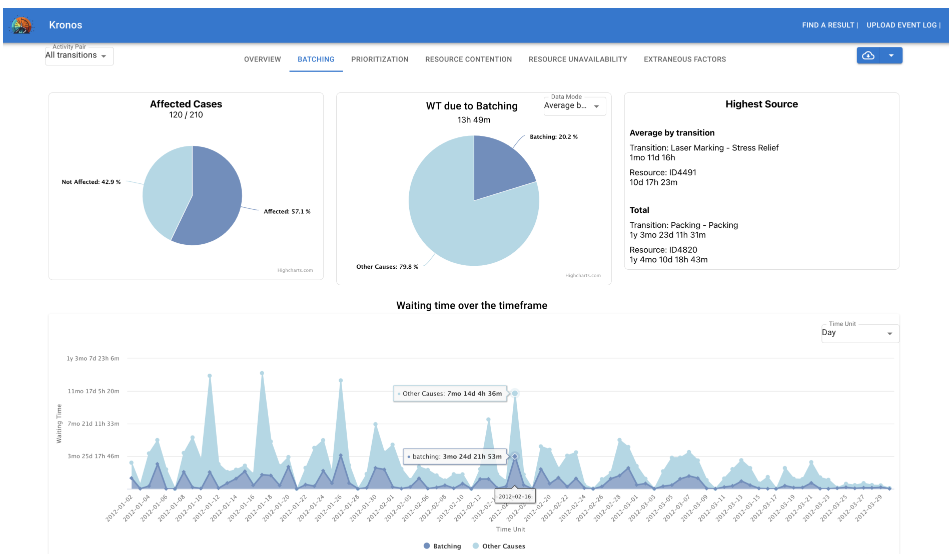


Figure 38. The Batching tab on the Kronos dashboard.

5.4.1. Accuracy verification

We evaluate the approach by addressing two evaluation questions (EQ):

EQ1 To which extent is the technique able to detect the presence or absence of certain waiting time causes?

EQ2 To what extent is the technique able to correctly quantify the amount of waiting time waste per each cause?

In this experimentation, we use synthetic data to validate the ability of the technique to accurately discover transitions, their waiting times, and the waiting time causes known to be present in the event logs. The approach implementation, the event logs, and experiment results are all available on GitHub[9].

To answer EQ1, we used a business process simulation model (BPS model) of a loan application process to simulate a set of event logs with different combinations of waiting time causes (see Table 10). To simulate waiting time due to resource contention, we set a low number of available resources in the BPS model, so that, in some cases, there were no resources available to process an enabled activity instance (available resources = `fewRes` in Table 10). In contrast, to ensure no resource contention, we set an extremely high number of available resources (available resources = `infRes`). To create waiting time due to resource unavailability, we set resource working calendars so that some resources worked from Monday to Wednesday, and others from Thursday to Friday and from 9 am to 5 pm (resource schedules = `8/5` in Table 10). Similarly, to ensure no resource unavailability, we set resources to be all-time available (resource schedules = `24/7`).

To simulate waiting time due to extraneous factors, we added timer events before some of the activities in the BPS model, thus delaying their start (`timers` in Table 10). Waiting time due to batching and prioritization cannot be injected by modifying the simulation parameters, as current BPS engines do not support them. Therefore, in batching, we added a set of new cases delaying the start of some activity instances so that they are processed as a batch (`batching`). To simulate prioritization, we added a set of new cases, changing the order of execution of some activity instances so they are processed following a prioritization order (`prioritization`). In some scenarios, no batching or prioritization was injected (batching and/or prioritization = –).

Combining these modifications, we simulated a set of 32 event logs with all the combinations of causes of waiting time and measured the performance using precision and recall. True positives and false positives stand for the discovery of a waiting time cause that, respectively, was and was not injected in the event log. True and false negatives denote an undiscovered waiting time cause that, respectively, was not and was injected.

Table 11 depicts the results for the simulated event logs used to evaluate EQ1. Our approach discovered the injected waiting time causes in all the event logs (no

[9] `https://github.com/AutomatedProcessImprovement/waiting-time-analysis/`

Table 10. Simulated event logs of a loan application process, showing injected patterns: high number of resources (`infRes`), limited resources (`fewRes`), 24/7 resource availability (`24/7`), weekday availability from 9 am to 5 pm (`8/5`), timers added (`timers`), no timers added (`no timers`), batching added (`batching`), prioritization added (`prioritization`), and no batching or prioritization added (`-`).

Simulation scenario	Available resources	Resource schedules	Timers	Batching and/or prioritization
S01	infRes	24/7	no timers	–
S02	infRes	24/7	no timers	batching
S03	infRes	24/7	no timers	prioritization
S04	fewRes	24/7	no timers	–
S05	infRes	8/5	no timers	–
S06	infRes	24/7	timers	–
S07	infRes	24/7	no timers	batching, prioritization
S08	fewRes	24/7	no timers	batching
S09	infRes	8/5	no timers	batching
S10	infRes	24/7	timers	batching
S11	fewRes	24/7	no timers	prioritization
S12	infRes	8/5	no timers	prioritization
S13	infRes	24/7	timers	prioritization
S14	fewRes	8/5	no timers	–
S15	fewRes	24/7	timers	–
S16	infRes	8/5	timers	–
S17	fewRes	24/7	no timers	batching, prioritization
S18	infRes	8/5	no timers	batching, prioritization
S19	infRes	24/7	timers	batching, prioritization
S20	fewRes	8/5	no timers	batching
S21	fewRes	24/7	timers	batching
S22	infRes	8/5	timers	batching
S23	fewRes	8/5	no timers	prioritization
S24	fewRes	24/7	timers	prioritization
S25	infRes	8/5	timers	prioritization
S26	fewRes	8/5	timers	–
S27	fewRes	8/5	no timers	batching, prioritization
S28	fewRes	24/7	timers	batching, prioritization
S29	infRes	8/5	timers	batching, prioritization
S30	fewRes	8/5	timers	batching
S31	fewRes	8/5	timers	prioritization
S32	fewRes	8/5	timers	batching, prioritization

false negatives), resulting in a recall of 100%. However, due to the influence that some waiting time causes have between them, the results contain 17 false positives (precision of 83%). False positives in waiting time due to resource unavailability (S06, S10, S13, S15, S19, S21, S24, and S28) are caused by the presence of extraneous waiting time, combined with limited data for the discovery of resources' working calendars. To simulate these logs, we set a 24/7 working calendar for all resources (high availability). However, when a resource has low occupation (executes few events), there is not enough data for the calendar discovery to identify a 24/7 calendar, and some intervals are interpreted as non-working time. When these non-working intervals overlap with extraneous waiting time, the tool classifies them as waiting time due to resource unavailability. This limitation is inherent to the discovery of the resource calendar, if there is no data showing that a resource was active during a period of time, it cannot be assumed that they were working.

Table 11. Results for the simulated event logs with all waiting time causes, depicting the true positives with '✓', the false positives with '✗', and the true negatives with an empty cell (there are no false negatives).

	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10	S11	S12	S13	S14	S15	S16
Batching		✓					✓	✓	✓	✓				✗		
Prioritization			✓				✓				✓	✓	✓	✗	✗	
Res. Contention				✓				✓			✓			✓	✓	
Res. Unavailability					✓	✗			✓	✗		✓	✗	✓	✗	✓
Extraneous factors						✓				✓			✓		✓	✓
	S17	S18	S19	S20	S21	S22	S23	S24	S25	S26	S27	S28	S29	S30	S31	S32
Batching	✓	✓	✓	✓	✓	✓	✗			✗	✓	✓	✓	✓	✗	✓
Prioritization	✓	✓	✓		✗		✓	✓	✓	✗	✓	✓	✓	✗	✓	✓
Res. Contention	✓			✓	✓		✓	✓		✓	✓	✓		✓	✓	✓
Res. Unavailability		✓	✗	✓	✗	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓
Extraneous factors			✓		✓	✓		✓	✓	✓		✓	✓	✓	✓	✓

The injection of waiting time due to extraneous factors also induced false positives of prioritization (S15, S21, S26, S30). When an activity instance is enabled but waiting due to extraneous factors, the resource might execute other activities enabled after the waiting one, being detected as waiting time due to prioritization. These false positives are due to the absence of an explicit indicator of the extraneous factors (i.e., extraneous waiting time is only detected when no other causes are identified). False positives due to batching (S14, S23, S26, S31) are caused by the appearance of a batch processing behavior that was not intentionally added. To create waiting time due to resource contention and unavailability, in some scenarios, we assigned working calendars with low resource availability. In such cases, while instances were waiting until the resource became available, they were collected and processed by the same resource. This resulted in an unassigned but correctly discovered batch processing behavior.

EQ2 aimed at assessing if our approach is able to correctly identify waiting time intervals and their causes. We could not use the set of event logs used for EQ1 to answer EQ2, as the logs were created through stochastic business process simulation – we know we have introduced a certain cause of waiting time, but we don't know to what extent. Therefore, we manually created a set of 5 event logs with different activity transitions and waiting time causes. Each event log contains a set of process cases where some of the transition instances present various waiting times in such a way that some of them overlap with others. In this way, we not only evaluate the ability to correctly detect the causes of waiting times, but also their precedence. Due to the low number of events per resource, automatically discovering the working calendars would introduce noise in the experiment. As we are interested in a controlled scenario for this evaluation question, we manually defined the resource working calendars of each resource. Then, we compared the discovered waiting time intervals and causes w.r.t. the ground truth, validating that all of them were correct.

Additionally, to address the limitation identified in the evaluation of Contribution 1 in Section 4.3.1, we conducted a batch rediscovery experiment. This experiment aimed to assess how changes in batch processing-related hyperparameters — specifically, minimum batch size and maximum in-batch gap — affect the accuracy of discovered batches. For that, we selected two simulated event logs of a loan application process with injected batch processing, namely S02 and S32. S02 represented a simpler process with only batching injected, while S32 represented a more complex process, including five causes of waiting times – batching, resource contention, prioritization, resource unavailability, and extraneous factors.

The batch processing discovery technique was then applied to each log using various hyperparameter settings (summarized in Table 12). For minimum batch size, options ranged from 2 to 10 cases per batch. A size of 2 is the most permissive and serves as the default setting (as in Section 4.3.1). A size of 10 represented the minimum size of the injected batches in these logs, which varied between 10, 12, and 14 cases. For the maximum in-batch gap, 0 seconds between processing cases within a batch is a default setting (as in Section 4.3.1) which we used as the lower limit. At the upper limit, the maximum in-batch gap was set to the average waiting time between activities in a simulated log, as this provided a meaningful reference point for possible gaps between case processing. This upper limit corresponded to 0 days 00:17:18 in S02, and 2 days 07:40:56 in S32. Additionally, three intermediate options were introduced, corresponding to 25%, 50%, and 75% of the average waiting time. For data analysis and visualization purposes, the maximum in-batch gap options were coded numerically from 0 to 4, as shown in Table 12.

Using these options, we executed all possible combinations of the hyperparameter settings for each event log. For example, we tested configurations such as minimum batch size = 2 and maximum in-batch gap = 0 (default), minimum batch size = 2 and maximum in-batch gap = 1, minimum batch size = 2 and maximum in-batch gap = 2, and so on. In total, 45 unique hyperparameter combinations were tested for each event log.

Table 12. Options for the hyperparameter settings.

Hyperparameter	Minimum batch size	Maximum in-batch gap
Options	from 2 to 10	0 means 0 seconds 1 means the 25% of the average time 2 means the 50% of the average time 3 means the 75% of the average time 4 means the 100% of the average time

Figure 39 and Figure 40 present the true and false positives in discovered batched activity instances with different hyperparameters in the simulated event logs S02 and S32, respectively. In both logs, all 29 injected batches and respective 348 batched activity instances were correctly re-discovered (see Batched Activity Instance True Positives). Additionally, the true positive values remain constant at 348 across all settings of minimum batch size and maximum in-batch gap. This suggests that changes in the hyperparameters do not affect the accuracy of identifying the true positives in these setups.

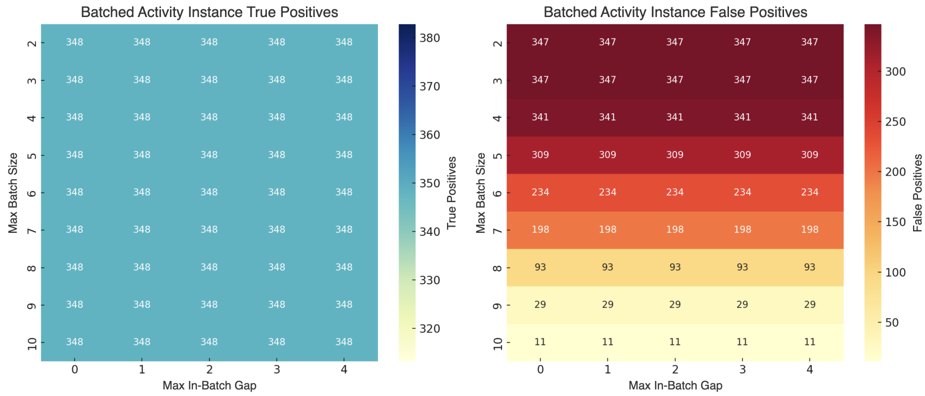


Figure 39. True positives and false positives in discovered batched activity instances with different hyperparameters in the simulated event log S02.

Moreover, batch processing was also detected in activities where batches were not injected. In these activities, activity instances accumulated for other reasons (e.g., long queues) and were subsequently processed together, i.e., demonstrated batch processing behavior. As a result, they were identified as batched (see Batched Activity Instance False Positives). False positives demonstrate a certain trend: as the minimum batch size increases, the number of false positives decreases. For example, in S02 (Figure 39), with a minimum batch size of 2 or 3, false positives are at their highest (347). As a minimum batch size goes up to 10, false positives reduce significantly to 11. Similarly, in S32 (Figure 40), as the minimum batch size increases, the number of false positives decreases. This relationship can be explained by the fact that a larger batch size reduces the probability of randomly accumulated activity instances being falsely detected as batches.

The maximum in-batch gap parameter does not impact false positives in S02 (Figure 39) – values remain the same across this axis. However, in S32 (Figure 40), as the maximum in-batch gap increases, the number of false positives also rises. This can be attributed to the fact that a larger in-batch gap increases the probability of misclassifying activity instances that are executed close together as part of the batch.

This experiment has demonstrated that the default settings in the current approach implementation (i.e., minimum batch size = 2, maximum in-batch gap = 0) might not be suitable for all processes, particularly for complex scenarios like S32, where multiple causes of waiting times are present. Therefore, it would be beneficial to allow analysts to adjust the default settings of the hyperparameters. Additionally, it could be useful to allow analysts to manually review and adjust the results of batch processing discovery using their contextual knowledge. Based on the adjusted batch processing results, the associated waiting times could be further identified and analyzed. This approach could enhance the accuracy of quantifying the causes of waiting times.

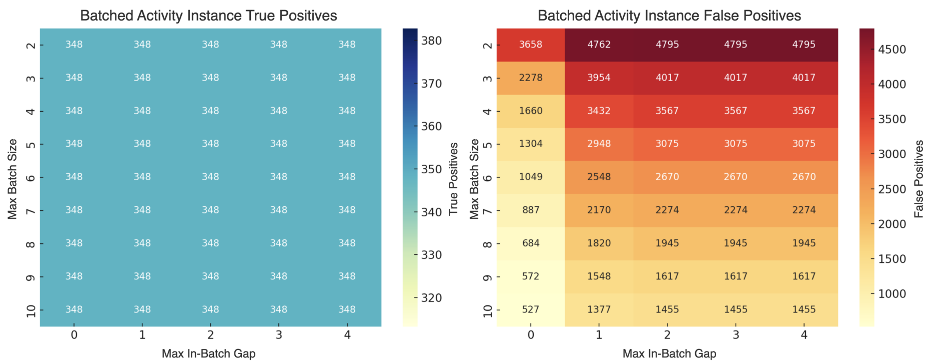


Figure 40. True positives and false positives in discovered batched activity instances with different hyperparameters in the simulated event log S32.

5.4.2. Applicability demonstration

To demonstrate the applicability of the proposed approach using Kronos, we use a real-life event log of a production process [181] (previously presented and used in Section 4.3.2). This event log was selected for evaluation since it fulfills the input requirements (refer to Table 8) and is sourced from real-life settings, which allows us to demonstrate the approach in a practical scenario.

First, we discovered the transition instances – 91 transitions with 3,323 transition instances (i.e., executions of each transition). In addition, their characteristics – case frequency, total frequency, and total waiting time – were identified.

Then, we discovered the waiting time causes, producing a report that captures to what extent each cause (per transition) contributes to the total waiting time of the process. The highest waiting times originated from the self-loop transitions, i.e. when the same activity is executed twice in a row. For instance, the self-loop of "Turning & Milling Q.C." induced a total waiting time of 2 years, 12 months, 1 day, 3 hours, and 38 minutes (see Figure 41), being the greatest contributor to the total waiting time. The largest portion of the waiting time in this transition (54.4%) was caused by resource unavailability (1 year, 7 months, 22 days, 19 hours, and 29 minutes).

The next largest causes of waiting were batching (25.4% of the total waiting time in this transition; 7 months, 15 days, 1 hour, and 5 minutes) and prioritization (8.6%; 3 months, 12 days, 5 hours, and 28 minutes). During the execution of the "Turning & Milling Q.C." activity, 31.4% of cases were processed in batches and followed the concurrent batch processing strategy (i.e., cases were processed one by one with a partial overlap in their processing times). The batch sizes ranged from 2 to 5 cases, where the majority of batches included 2 cases (78%). Approximately half of the batches were activated when the waiting time of a ready batch reached 12 hours and 49 minutes, i.e., followed the batch activation rule: WT_{ready} $\geq$ 12 hours 49 minutes (discovered rule confidence = 52.9%, discovered rule support = 49.3%). No prioritization strategies were identified, as the necessary case attributes for discovering such strategies are absent in this event log.

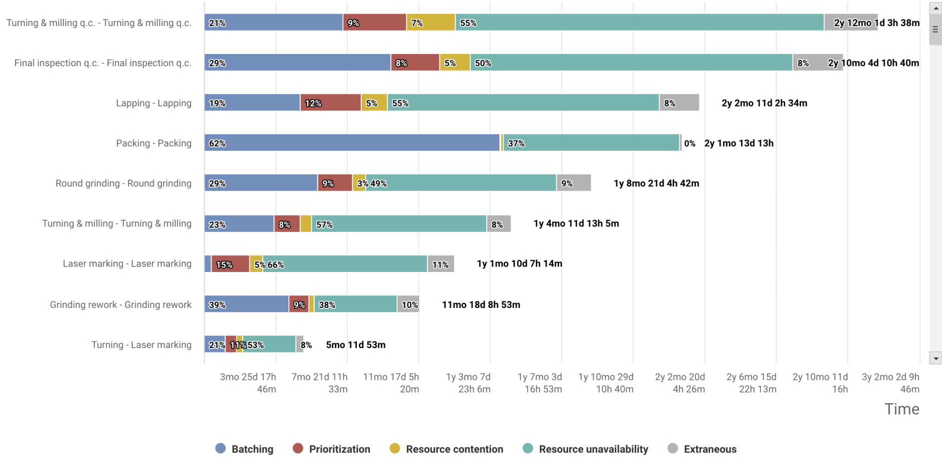


Figure 41. Waiting time causes in activity transitions of the production process.

Finally, we analyzed to what extent each waiting time cause contributes to the total waiting time of the process, and how they affect the CTE. Resource unavailability was the primary source of waiting time (as it caused 54.4% of the total waiting time), followed by batching (25.4%), prioritization (8.6%), extraneous (7.4%), and resource contention (4.2%). Then, we measured the impact of the waiting times by cause on the process CTE (CTE = 5.9%). The highest CTE increase (up to 12.2%) can be achieved if the waiting time due to resource unavailability is eliminated. Regarding individual transitions, addressing the "Turning & Milling Q.C." self-loop is the highest improvement opportunity. If the waiting time in this transition is addressed, the CTE could increase to 6.9%.

As a result, we demonstrated how the approach can be applied to a real-life log and how the outputs can be used to analyze the waiting time causes. This indicates the practical applicability of the proposed approach to a real-life log.

5.5. User evaluation

To evaluate usefulness and usability, we conducted an interview study with process analysts experienced in process mining. This section presents the design and findings of the user evaluation and discusses how the findings were used to improve the design of Kronos.

5.5.1. Interview study design

For the interview study, we set evaluation goals (EG) aiming to understand the process mining experts' perception of the tool's usefulness (EG1), ease of use (EG2) and identify how the tool could be improved (EG3). For this, we recruited ten process analysts (Table 13) who have used process mining for process improvement projects. The participants have been involved with process improvement projects in nine different domains, predominantly manufacturing. On average, the participants had seven years of experience with process analysis and four years with process mining.

Table 13. Interview participants.

Code	Domain	Experience (process analysis)	Experience (process mining)
P1	Consulting	6 years	4 years
P2	Energy	3 years	2,5 years
P3	Transportation, Logistics, Maintenance	15 years	3 years
P4	Manufacturing	5 years	5 years
P5	Consulting	10 years	10 years
P6	Manufacturing	8 years	3 years
P7	Manufacturing	10 years	1,5 years
P8	Manufacturing	5 years	5 years
P9	Banking, Energy, Retail	1 year	1 year
P10	Manufacturing, Finance, Logistics	7 years	5 years

For this evaluation, we combined semi-structured interviews with a survey study. The semi-structured interviews enabled an open conversation [188] suitable for providing participants the freedom to share their ideas about Kronos. The follow-up survey offered additional context for quantitatively analyzing the collected interview data [189].

We conducted individual interviews with the participants. During the interview, we first showed Kronos and introduced its key concepts (such as the waiting time causes). The version of Kronos we showed to the participants was minimalistic. In the starting screen, the user uploaded the event log and mapped each column to its respective type (e.g., activity or resource). On the first page of the dashboard, the tool displayed the descriptive statistics of the event log (e.g., number of activities, transitions, and cases), the grand total of the waiting time, and the waiting time attributable to each cause. The second page showed the decomposition of waiting time per cause in each transition, as shown in Figure 41. On the final page, Kronos illustrated the original CTE of the process and potential CTE improvement per waiting time cause. We decided to start the user study with a minimalistic design to avoid biasing the study participants and let their feedback determine the final design of the tool. The detailed description of the initial version is available as a demo video[10] produced for the ICPM'23 conference [57].

We then asked participants to use Kronos on a real-life event log of a production process (introduced in Section 4.3.2), analyze the results, and identify improvement opportunities related to waiting times. Afterward, we asked questions based on the evaluation goals. For instance, to evaluate the usefulness of

[10]https://youtu.be/vvOY_hbOOh4

the information presented in Kronos for identifying improvement opportunities, we asked *"Which data was useful when identifying improvement opportunities?"* (EG1). Furthermore, we sought to identify missing information crucial for this task that might not be readily available in Kronos. This was achieved by asking *"When identifying improvement opportunities, what data did you miss (if any)?"* (EG1). Then, to assess how easy it was for the participants to use the components of Kronos in their analysis, we asked *"Which part of Kronos was clear and easy to understand?"* and *"Which part of Kronos was the hardest to understand and why?"* (EG2). Finally, we collected suggestions on how Kronos could be improved with the question of *"What part or aspect of Kronos needs to be improved and how?"* (EG3).

After the interview, the participants filled out a survey. The survey consisted of three parts.

1. The first part was based on the Technology Acceptance Model (TAM) [189] and included questions on perceived usefulness (PU), perceived ease of use (PEOU), and intention to use (ItU). Questions on PU considered the usefulness of Kronos when identifying improvement opportunities related to waiting times. PEOU questions concerned the clarity and understandability of the information presented in Kronos and its ease of use. ItU questions focused on participants' intention to continue using Kronos for identifying waiting time-related improvement opportunities. For these questions, we reused a validated questionnaire proposed in [190, 189], adapting it to the Kronos context. We used a scale from 1 (strongly disagree) to 5 (strongly agree) for these questions.
2. The second part was based on the System Usability Scale (SUS) [191] and included questions focused on usability. SUS-based questions were added to complement TAM with user satisfaction of the tool [191]. Thus, for the SUS section, we asked questions concerning the usability of Kronos in terms of complexity, consistency, and ease of learning and use. For these questions, we reused a validated questionnaire proposed in [191], adapting it to the Kronos context. We used a scale from 1 (strongly disagree) to 5 (strongly agree) for these questions.
3. The third part comprised demographic questions. These questions captured data on occupation, years of experience, domains, and process mining activities participants performed.

The complete set of interview and survey questions can be found in the evaluation study protocol provided in the supplementary materials[11].

The interviews were conducted online and lasted from 28 to 45 minutes. We recorded the interviews, transcribed them using Otter.ai[12], and manually reviewed the transcripts. To analyze the interview data, we followed a thematic analysis ap-

[11] `https://owncloud.ut.ee/owncloud/s/kFbtYr7mE7Ln4Em`

[12] `https://otter.ai/`

proach. For the thematic analysis, the transcripts were coded iteratively. First, we created an initial set of codes based on the evaluation questions (e.g., "useful" and "not useful"). Then, we applied the codes to the transcripts and revised them as we analyzed the transcripts. For instance, we added new codes when none from the initial set fitted the observation. For example, we added "unexpected behavior" for situations when the tool's response to action did not match the user's expectations. As a result, we obtained a set of 26 codes that were clustered into 5 themes ("usefulness," "identification of improvement opportunities," "understandability of the content," "ease of use of the tool," and "improvements"). The themes of "usefulness" and "identification of improvement opportunities" relate to EG1, "understandability of the content" and "ease of use of the tool" to EG2, and "improvements" to EG3.

To analyze the TAM survey results, we aggregated the individual responses for each assessed criterion – perceived usefulness (PU), perceived ease of use (PEOU), and intention to use (ItU) – and calculated their mean and standard deviation values [192]. The mean values of PU, PEOU, and ItU provide insights into overall participant perceptions, highlighting the general acceptance and potential adoption of Kronos. The standard deviation values indicate the consistency or variability in the responses, revealing how uniformly participants perceive the usefulness, ease of use, and intention to use Kronos. High standard deviation values (≥ 1) would suggest diverse opinions among participants, while lower values (< 1) would indicate a more consistent perception across participants. To analyze the SUS survey results, we calculated the overall average SUS score as proposed in [191]. This average score provides a single, comprehensive measure of the system's usability (while individual item scores are not independently meaningful) [191].

5.5.2. Interview study findings

The evaluation results indicated that the participants found Kronos to be usable. This is evidenced by the average SUS score of 77.3 (a system is considered usable when the average SUS score is above 68) [193]. Participants were satisfied with using Kronos, saying that *"this is a great tool"* (P3), *"nice and clear"* (P6) and *"it's cool and it would perfectly make sense"* (P2). They also appreciated the novel contribution, mentioning that *"it's definitely something that's not available yet. And that makes it really cool"* (P4) and saying: *"I like this tool very much because this adds another dimension to the analysis that we currently do not have"* (P7).

EG1. Participants also generally perceived Kronos to be useful (EG1). This is supported by the survey responses on usefulness (M=4.27, SD=0.69)[13] and participants' comments, e.g., *"I think, especially for the core processes like the manufacturing process, this is a very helpful, powerful analysis"* (P4). Partic-

[13]"M" stands for mean value and "SD" for standard deviation.

ipants commonly commented on the usefulness of understanding the causes of waiting times, e.g., P5 expressed that *"this part of giving a cause or reason for a waiting time, I find extremely helpful."* In addition, they indicated the usefulness of identifying transitions in which waiting times occurred. For instance, one participant P6 shared that they *"could already also say where in the process it [waiting time] happens, that's for sure very helpful"* (P6). Participants also appreciated the resource view on waiting times. One interviewee said: *"I like that I can look into the specific resources that cause the highest waiting time. For instance, if it is due to resource unavailability, then we can maybe change their working hours or hire more resources"* (P1). Thus, it seems that understanding the waiting time causes could guide analysts to the root cause of the inefficiencies and ways to address them. Several participants highlighted the importance of seeing the CTE impact since *"it has a very strong message"* (P5). They appreciated seeing the potential CTE improvement in addition to the inefficiencies *"because besides knowing what the problem is, clients will want to know how to solve it and what results to expect"* (P9). Although some participants consider the CTE metric is generally useful, they did not exploit this metric in their analysis and, thus, assessed it as less applicable. One participant expressed that *"CTE impact, I think, is a great metric. It is true, though, I don't use it that much"* (P8). Overall, all Kronos's components were reported as useful in identifying improvement opportunities in waiting times since *"each part provides a different perspective on the issue"* (P1).

In addition, we explicitly asked the participants if they missed any information when identifying improvement opportunities with Kronos. Several participants reported that the provided information was sufficient. One participant commented that *"I think regarding the information presented in this tool, you have just enough to find any inefficiency in waiting times"* (P9). Although all participants managed to identify improvement opportunities, many also expressed the need for additional insights to better understand the causes of waiting times. For instance, in case of waiting time due to prioritization, they were interested in seeing which cases were prioritized, which resources decided on the prioritization, and how often. In addition, we observed the need to understand the trend of waiting times over the analyzed period to *"see if this waiting time cause is a constant problem or a one-time occurrence. Because if it's a one-time event, then probably it is not quite useful for an analyst to work on"* (P10). Finally, participants shared that they used average and median perspectives of waiting times more frequently than totals since *"average values might be more comprehensible for the user to understand and show how waiting happens case by case"* (P8).

EG2. Furthermore, participants perceived Kronos to be easy to use (EG2). This is supported by the survey responses (M =4.48, SD =0.64) and statements such as *"I like how the information is presented, it's structured and straight to the point, so you can easily see where your inefficiencies are"* (P7). However, several issues limited ease of use, such as *"inconvenient data formats"* (P9), *"unexpected behavior of the sorting control"* (P4), and *"no indication that the table* *lines are clickable"* (P2). The participants also found the content to be clear and understandable. For instance, one shared that *"I don't see any issues as all terms you use here are well-known in the process mining community"* (P7). However, we noted that while working with Kronos, participants asked to be reminded of how waiting time causes were defined and how CTE was calculated. Also, P1 suggested: *"If a new user comes in, who doesn't understand these terms, for instance, batching and whatnot, it would be better to have some kind of a legend or some hint about what each of these terms means"* (P1).

The survey showed a relatively lower score for intention to use (M=3.47, SD=0.68) as compared to the usefulness and ease of use. On the one hand, participants appreciated the tool's contribution and would like to exploit it. For instance, one participant expressed that *"this is something that I myself would want to look into when analyzing the waiting times"* (P8). On the other hand, several participants mentioned the limited availability of event logs with both start and end timestamps. This might be a factor limiting analysts from using Kronos. Thus, participant P4 shared that *"analysis of waiting times is, of course, a great deal in our work but, unfortunately, we do not often have processes with start and end timestamps."*

EG3. Further, incorporating feedback from participants, we have identified 49 potential improvements (Table 14) to Kronos (EG3). About half of the identified improvements – 25 improvements (51% of the total) – centered around providing additional insights into the waiting times and their causes. Participants conveyed that these insights could prove valuable for identifying improvement opportunities related to waiting times. The second most common category, consisting of 11 improvements (22%), concentrated on improving the visualization of analysis results. Another category of improvements, totaling 7 (14%), highlighted the need for conducting various types of analysis, e.g. variant analysis, and thus, increase the analysis scope. Lastly, 6 improvements (12%) were focused on enhancing annotation to improve understandability. Thus, improvements aimed at offering further insights and broadening the analysis scope (32 improvements) related to modifying the approach. Improvements in visualization and annotation (18 improvements) related to enhancing the usability of the tool.

5.5.3. Feedback loop back to Kronos

To improve the initial design of Kronos, we prioritized and implemented the improvements identified as a result of the analysis of the interview study data. We prioritized the identified improvements into 4 priority groups based on the following parameters: (i) the value of the improvement (the significance of the improvement in terms of its contribution to identifying improvement opportunities related to waiting times); (ii) the feasibility of implementation (the effort and time required for improvement implementation); and (iii) the frequency of mention (how many different participants discussed them). Thus, improvements adding high

Table 14. Categories of improvements for Kronos.

Category	Sample improvement	Number of improvements	Implemented (number and percentage of the total)
Additional insight	Identify and display which resources are associated with each waiting time cause and how much waiting time each resource induces.	25	13 (52.0%)
Visualization	Present the analysis results grouped by waiting time cause to allow analyzing each cause of waiting time separately.	11	11 (100.0%)
Analysis scope	Add the capability of comparing and analyzing waiting times among several process variants, e.g., based on departments, product types, locations, and time periods.	7	0 (0.0%)
Annotation	Clarify and unify all titles and labels to ensure clarity and consistency, promoting better understanding.	6	6 (100.0%)
Sum		**49**	**30 (61.2%)**

value, requiring low to medium effort/time to implement, and/or mentioned by multiple participants were assigned the highest (first) priority (14 improvements). Improvements adding limited value, requiring medium effort/time to implement, and/or mentioned by one participant scored second priority (16 improvements). Finally, improvements requiring high effort/time or out of scope got third (11 improvements) and fourth priority (8 improvements), respectively.

We implemented all improvements of the first and second priorities and left third-priority improvements for future releases of Kronos. Thus, we implemented 13 improvements focused on providing additional insights that are significant for analysts when analyzing waiting times. In particular, we provided more detailed information for each waiting time cause, such as information about the affected transitions and cases, and the distribution of waiting time by cause over the timeframe. Additionally, we enhanced the analysis from the resource perspective by providing insights into the resources and handovers associated with waiting times categorized by the waiting time cause. Lastly, we integrated functionality for identifying batching and prioritization strategies when the respective waiting time causes are identified, and the option to view all analysis results from an average and median perspective. With these improvements, Kronos provides a more comprehensive view of the waiting times and helps analysts identify improvement opportunities related to waiting times.

We also implemented all of the 11 improvements relating to enhancing the visualization. Specifically, we focused on presenting the analysis results in a user-friendly manner, making it easier for analysts to navigate and find relevant information. This includes grouping analysis results based on waiting time causes to provide users with easy access to all information about each specific cause in a single dashboard. In addition, we universally adopted the average perspective as the default setting, with the median provided as a second choice, recommended in cases where there might be outliers. This change was driven by the common preference among analysts for reviewing analysis results in terms of average values as opposed to displaying them as cumulative totals. Furthermore, we enhanced the usability of dynamic charts by implementing standardized, recognizable controls for data filtering and sorting and ensuring predictable responses to user actions. Lastly, we clarified visualizations that had previously caused issues during use among interview participants. As a result, we have attained a more coherent and consistent interface.

All improvements related to annotation were also successfully integrated, totaling 6 improvements. In particular, we reviewed and unified terms in all chart titles and labels. We also added tooltips to clarify the key concepts that might be new for an analyst (e.g., definitions of waiting time causes) and explained how the analysis results were obtained (e.g., how the values of potential CTE improvement are calculated). These changes allowed us to improve the consistency and understandability of the textual information.

Improvements related to expanding the scope of analysis encompassed including additional analysis types such as: comparing waiting times across different variants and periods, conducting milestone analysis to evaluate waiting times between subprocesses, analyzing against specified Key Performance Indicators (KPIs), providing recommendations for process redesign to reduce waiting times, and delivering real-time suggestions for interventions in ongoing cases to reduce potential waiting times. In its current form, Kronos is aimed to help analysts identify opportunities to reduce waiting times in activity transitions. However, these improvements pursued different analysis goals and, therefore, fell beyond the scope of our project and were not implemented. Incorporating these improvements can be considered for future Kronos extensions[14]. In summary, we implemented 30 out of 49 identified improvements (61% of the total): 13 improvements aimed at enhancing the approach and 18 – at increasing the usability of the tool.

The user study evaluation has some inherent limitations [188]. Our evaluation with 10 analysts poses a limitation on the generalizability of the results, i.e., the extent to which the results are applicable in other settings. This limitation was, to an extent, mitigated by interviewing analysts from different domains and backgrounds. In addition, when analyzing qualitative data, there's a threat of misinterpretation stemming from biases or subjectivity. To mitigate this threat, the collected and analyzed data was discussed within the research team. The evaluation study protocol is available in supplementary materials[15].

[14]For instance, we implement one of these improvements – providing recommendations for process redesign to reduce waiting times – as a part of *Contribution 3* (see Chapter 6).

[15]`https://owncloud.ut.ee/owncloud/s/kFbtYr7mE7Ln4Em`

5.6. Summary

This chapter proposed a process mining-based approach for discovering causes of waiting times and identifying improvement opportunities by assessing the impact of these causes on the process CTE. To achieve this objective, we proposed to attribute the waiting times of activity transitions to the five causes: *batching*, *resource contention*, *prioritization*, *resource unavailability*, and *extraneous factors*. Then, we outlined how these five causes of waiting time can be discovered from an activity instance log. Lastly, we described how the approach measures the impact of each waiting time cause on the CTE of a process to help analysts prioritize opportunities to reduce these waiting times.

The proposed approach has been implemented as a software tool called Kronos. Using this tool, we assessed the approach's ability to accurately detect and quantify each of the five causes of waiting times. In addition, we demonstrated the approach's applicability on a real-life log. We also conducted a user evaluation of Kronos with process analysts and applied the findings of this evaluation to refine the tool's design.

In its current form, the proposed approach has several limitations. First, the approach considers only waiting times in transitions between activity instances. Yet waiting times may also arise in at least two other settings: (i) between case creation and the start of the first activity instance; and (ii) within an activity instance due to interruptions (e.g., the resource interrupts their work and resumes it later). Another limitation of the approach is that it does not consider multitasking. Furthermore, when the calendar miner cannot accurately discover resource calendars — such as when resources are involved in multiple processes or work only a few minutes per day — waiting time due to resource unavailability and extraneous factors could be attributed inaccurately. Additionally, the current implementation of the approach does not allow analysts to adjust the default discovery settings (e.g., minimum batch size and maximum in-batch gap) or refine the discovered patterns based on their contextual knowledge, such as batch processing strategies or resource calendars. Finally, the values of potential CTE improvement depict a theoretical improvement achieved by eliminating specific waiting times, without accounting for any potential side effects resulting from a process redesign.

6. LLM-ASSISTED ANALYSIS AND REDESIGN OF WAITING TIMES

In Chapters 4 and 5, we proposed approaches for the discovery and analysis of waiting time causes, which addressed the first research question (RQ1). More specifically, Chapter 4 introduced *Contribution 1* – a process mining-based approach for discovering waiting time types associated with batching and analyzing their impact on the CTE of batch processing activities. Chapter 5 presented *Contribution 2*, which extended Contribution 1 by including other causes of waiting times apart from batching, namely resource contention, prioritization, and resource unavailability. Thus, Contribution 2 comprised a process mining-based approach for decomposing observed waiting times in each activity transition into direct causes and analyzing the impact of these causes on the process CTE. These proposed approaches enable process analysts to identify improvement opportunities related to waiting time causes.

Following the BPM cycle [1], once improvement opportunities are identified, the process is redesigned to address these improvement opportunities and enhance the (temporal) process performance. Therefore, in this chapter, we aim to examine how the redesign options can be identified to address the improvement opportunities to reduce waiting times (RQ2).

The review of the state-of-the-art (presented in Chapter 3) revealed that process redesign remains a predominantly manual activity executed by process analysts who rely on their expertise and experience when redesigning business processes [48]. To support process analysts in identifying applicable redesign options, existing research proposes several collections of redesign patterns. These collections comprise redesign patterns proven to be effective for process improvement and include works by Reijers and Mansar [45], Fehrer [46], and Lashkevich [47]. However, these collections target different performance perspectives (including cost, flexibility, quality, and customer centricity) and do not specifically indicate which redesign options are applicable to address the causes of waiting times. Additionally, analysts still need to manually examine these collections to identify relevant redesign options.

Existing research in process mining has also offered several semi-automated redesign approaches, such as the PrICE toolkit [49] and the assisted business process redesign (aBPR) [50]. While these tools facilitate process redesign and partially automate the tasks performed by process analysts, they require process models as inputs, use a limited number of predefined redesign patterns, and do not specifically relate to the causes of waiting times when recommending redesign options. Therefore, this chapter aims to address the following research gap (RG) in the field of process redesign:

RG2 Existing studies have not provided automated methods to recommend redesign options that reduce waiting times by targeting their causes.

Recent studies suggest that LLMs may assist business users in various process analysis tasks, particularly in conjunction with process mining techniques [11]. For instance, Berti et al. [194] use GPT4 LLM to convert natural language inquiries into SQL queries, then deliver the results to users in natural language. To improve the LLM performance in solving process mining-related tasks, specific prompting approaches have been applied. Grohs et al. [195] describe a prompting approach that uses GPT4 to discover process models from textual descriptions and to identify tasks that are suitable for automation. Jessen et al. [196] propose a prompting strategy to apply LLM for solving process mining tasks by building and executing SQL queries from natural language questions, e.g., for identifying bottlenecks. Meanwhile, other authors have explored the use of LLMs to derive process models from textual process documentation [197].

Thus, existing studies have demonstrated the applicability of LLMs across various process mining tasks. However, these studies focus on the use of LLMs for discovering process models and conducting analysis tasks. In this research, we aim to exploit the affordances LLMs to analyze the waiting time causes and recommend redesign options aimed at reducing waiting times.

Specifically, in this chapter, we propose *Contribution 3* – a prompting method for an LLM to enhance its ability to analyze the causes of waiting times and recommend redesign options to reduce those waiting times by targeting their causes. First, this chapter outlines the research methodology we followed to develop the prompting method. Second, it presents the formulated requirements for the prompting method. Further, this chapter introduces the proposed prompting method and its implementation as a conversational interface integrated into Kronos. Finally, this chapter presents the evaluation of the prompting method. We evaluate the proposed prompting method by (1) assessing the accuracy of the prompt-enhanced LLM for waiting time analysis questions, and (2) conducting a user study with process analysts to evaluate the quality of the redesign options provided by the prompt-enhanced LLM.

6.1. Research methodology

The objective of this research is to develop a prompting method to enable the LLM to support analysts to analyze waiting time causes and provide recommendations on redesign options to reduce waiting times. To achieve this, we followed a DSR methodology [164]. An overview of the research process we followed is presented in Figure 42.

First, we analyzed the state-of-the-art literature related to RQ2 and the identified research gap RG2 presented above. This review revealed the research opportunities in exploring the LLM affordances in solving process mining-related tasks. This led to the formulation of our research objective (RO) to develop a prompting method for an LLM to enable it to analyze waiting time causes and recommend redesign options to reduce waiting times. Based on the RO, we defined the re-

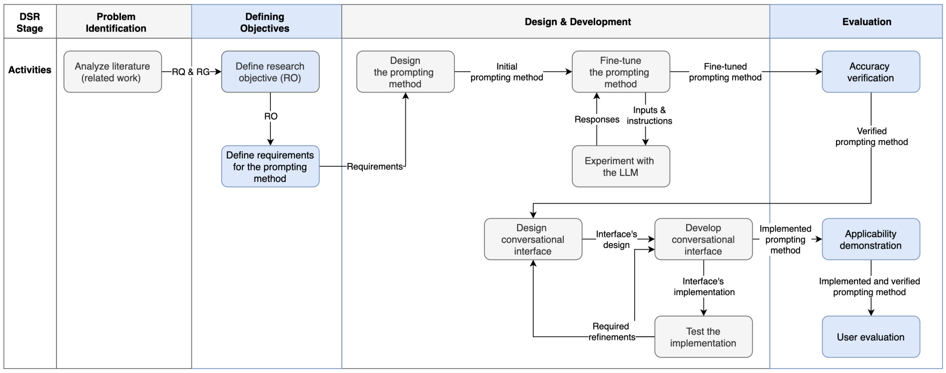


Figure 42. Research process for Contribution 3.

quirements for the prompting method.

The defined requirements guided the design of the proposed prompting method, which includes a descriptive analysis component (to analyze waiting times and their causes) and a prescriptive analysis component (to recommend redesign options). The Design & Development stage was iterative. We started by designing the descriptive component of the prompting method. We then applied the initial prompting method with a descriptive component to the LLM (GPT-4 Turbo LLM) and conducted testing experiments on the descriptive analysis tasks (e.g., to analyze waiting time causes). During these experiments, we fine-tuned the prompting method (descriptive component) according to the defined requirements. This involved modifying the inputs and instructions included in the prompt and analyzing the obtained responses to identify necessary refinements. This iterative process was repeated until no further fine-tuning was needed.

Similarly to the descriptive component, we designed the prescriptive component iteratively. This process resulted in three versions of the prescriptive component, differing in the type of redesign pattern data included in the prompt: baseline (no data on redesign patterns included), enhanced-unstructured (with unstructured redesign pattern data), and enhanced-structured (with structured redesign pattern data). We tested these versions with the LLM and fine-tuned them according to the defined requirements for the prompting method. After fine-tuning the prompting method, we proceeded to the Evaluation stage. We verified the accuracy of the descriptive analysis component in solving descriptive analysis tasks (e.g., in quantifying waiting time causes) using synthetic and real-life event logs.

After the accuracy verification, we designed and developed a conversational interface within the Kronos tool and integrated the proposed prompting method. This process was also iterative and repeated until no further refinements to the interface design and implementation were identified. Then, we applied the prompting method (both descriptive and prescriptive analysis components) to a real-life log to demonstrate its applicability in analyzing waiting time causes and recommending redesign options in a practical scenario. We then conducted a user study with process analysts to evaluate both the descriptive and prescriptive components of the prompting method. In the user study, we assessed the ability of the prompting method to support analysts in analyzing waiting time causes and identifying redesign options to reduce waiting times. Finally, we analyzed the results from the user evaluation and extracted findings that could be useful for further improving the proposed prompting method and its implementation.

6.2. Problem scoping and requirements

In this section, we define requirements for the prompting method. Given the prompting method is aimed at supporting analysts in process analysis and redesign, it should correspond to established procedures of process improvement. To improve business processes, analysts first identify and prioritize issues based on their impact on performance. When analyzing waiting times, analysts consider where the waiting time inefficiencies occur, how much of it is induced, their causes, and which resources contribute the most. Then, analysts identify changes that could address the issues [1]. Therefore, we formulated the following requirements (R) for the prompting method:

R1: The prompting method shall support analysts in analyzing waiting times w.r.t. the location of waiting times, their magnitude, causes, and resources associated with these waiting times.

R2: The prompting method shall support analysts in process redesign by recommending redesign options to address identified causes of waiting times.

To design the prompting method, we exploit LLM affordances. Therefore, we set the following requirements:

R3: The prompting method shall utilize the affordances of LLM technology.

R4: The prompting method shall deliver analysis results through interactive conversations with the user.

LLMs can be tuned to provide more relevant responses to user requests [198]. Responses to user requests should provide accurate results on, for instance, waiting time duration, and include useful and sufficient information to identify improvement opportunities. Therefore, the following requirements were added:

R5: The prompting method shall enable the LLM to provide accurate durations of waiting times and their causes.

R6: The prompting method shall enable the LLM to provide useful and sufficient information for analysts when analyzing waiting time causes for identifying improvement opportunities.

Shani et al. [199] have compiled a set of desirable properties of recommender systems. In our research, we focus on desirable properties related to the content of the recommendations, leaving aside properties related to user interface adequacy and perceived ease of use. On this basis, we compiled 8 desirable properties of recommender systems, namely recommendation relevance, novelty, diversity, serendipity, justification, information usefulness, information sufficiency, and confidence in the provided recommendations [200]. As we aim to have the proposed prompting method satisfy these properties, we define the following requirements:

R7: The prompting method shall enable the LLM to recommend redesign options that are relevant for addressing waiting time causes (recommendation relevance).

R8: The set of recommended redesign options shall be diverse, i.e., target the waiting time causes from multiple perspectives (recommendation diversity).

R9: The prompting method shall enable the LLM to provide understandable descriptions of recommended redesign options (recommendation understandability).

R10: The prompting method shall enable the LLM to provide useful and sufficient information for analysts when recommending redesign options (information usefulness & sufficiency).

We did not set requirements for novelty (i.e., recommending items that the user did not know) [199], serendipity (i.e., how surprising the successful recommendations are) [199] and confidence (i.e., recommender's ability to convince users of the recommended items) [200] since our key objective is to provide relevant and useful recommendations. However, we recognize the significance of these properties in enhancing user satisfaction and evaluate them to understand their impact on user satisfaction with the provided recommendations.

6.3. Prompting method

In this section, we present the proposed prompting method for an LLM to analyze waiting time causes and recommend redesign options to reduce waiting times. The prompting method was designed based on the requirements defined in Section 6.2. Using requirements R1 and R2, we created a prompting method with a descriptive analysis component (for analyzing waiting times and their causes) and a prescriptive analysis component (for recommending redesigns). Following requirements R3 and R4, we applied and tested this prompting method using the GPT-4 Turbo LLM[1], which supports external instructions and datasets and uses a conversational interface to communicate results to users.

The proposed prompting method consists of two phases (see Figure 43). First, given an activity instance log, the waiting time causes – batching, resource contention, prioritization, and resource unavailability – are discovered and quantified.

[1] `https://help.openai.com/en/articles/8555510-gpt-4-turbo-in-the-openai-api`

The output of this phase is 1) a report listing transition executions and their durations with decomposition per waiting time cause; 2) reports with discovered patterns describing the waiting time causes – batching and prioritization strategies, resource working calendars; and 3) case attributes. These reports are provided to the LLM as inputs in the form of datasets that the LLM can access when analyzing waiting times. Phase 2 involves prompting the LLM by providing it with a prompt template that includes descriptive and prescriptive analysis components, along with a user prompt (i.e., a specific question or task from the user). As a result, through the conversational interface, users can explore improvement opportunities related to waiting times and their causes, and receive redesign recommendations from the LLM to address identified improvement opportunities. Below we present the proposed prompting method in more detail.

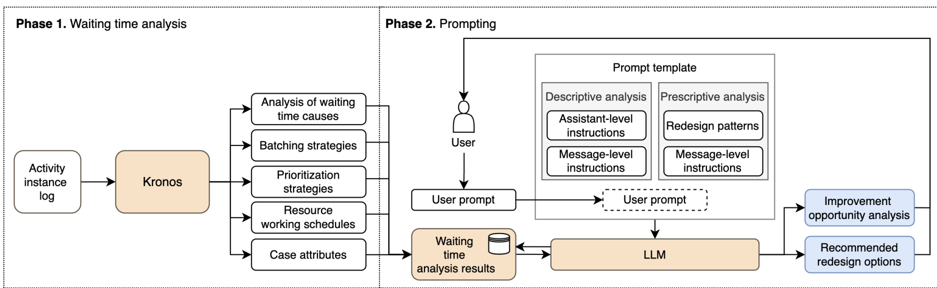


Figure 43. Overview of the proposed prompting method.

Phase 1. In the first phase, we discover and analyze the waiting time causes. We do so using Kronos (presented in Section 5.3) – the tool that embodies the approach[2] for the discovery and analysis of waiting time causes proposed in Chapter 5. Given an activity instance log, Kronos analyzes this log and produces a set of reports capturing the waiting time cause analysis results. These reports serve as input to the LLM, providing it with the necessary data for subsequent analysis. More specifically, Kronos produces five reports (see Table 15). *Analysis of waiting time causes* serves as a primary input for the LLM, providing a detailed analysis of waiting time causes. This report presents the decomposition of waiting times into their causes (batching, resource contention, prioritization, resource unavailability, and extraneous factors) per transition execution. *Batching strategies, prioritization strategies* and *resource working schedules* represent contextual information regarding waiting time causes to complement the analysis and enhance the LLM's capability to identify improvement opportunities. *Case attributes* are available for the LLM to identify correlations between case attributes and waiting times that could potentially explain waiting times. These reports (in the form of structured datasets) are fed into the LLM to enable its examination of root causes and identification of potential improvement opportunities.

[2]https://github.com/AutomatedProcessImprovement/waiting-time-analysis/

Table 15. Inputs for LLM (results of waiting time analysis).

Input	Content
Analysis of waiting time causes	Provides the waiting times of each transition execution and decomposes the waiting times into five causes: batching, resource contention, prioritization, resource unavailability, and extraneous factors. Each row includes the source and destination activities of a transition execution, source and destination resources executing the activities, total waiting time of the transition execution, and distribution of total waiting time per cause.
Batching strategies	Provides a list of activities where batch processing is detected and their associated batching policy: batch processing type (e.g., parallel batch processing), batch processing frequency, activation rule (e.g., fire when the batch size = 10), and batch size distribution.
Prioritization strategies	Provides a list of activities where prioritization is utilized, resources employing prioritization, and prioritization rules they follow (e.g., customer type = "premium").
Resource working schedules	Provides working schedules of resources in the form of their weekly calendars per resource.
Case attributes	Outlines unique case attributes for each case in the activity instance log.

Phase 2. In the second phase, we use a template-based prompting method to enable the LLM to answer questions about waiting times and redesign options on the basis of an event log. The prompt template consists of two components, covering the two types of tasks we expect the enhanced LLM to support: (i) a descriptive component that analyzes waiting time causes and contextual information, allowing users to identify improvement opportunities; and (ii) a prescriptive component aimed at recommending redesign options to reduce waiting times. Additionally, the template incorporates a placeholder for the user-provided prompt (placed at the end of the full prompt). Below, we discuss these components.

(i) Descriptive analysis component. To enable the LLM to analyze available waiting time analysis results, the prompt includes instructions. Instructions comprise two levels and contain the following information:

- **Assistant-level instructions** define the role of the LLM (*"You are an expert in process mining..."*), the goal (*"You are required to analyze the causes of waiting times..."*), definitions (e.g., waiting time causes), and descriptions of all inputs (e.g., what data the input "Analysis of waiting time causes" contains). These instructions are used to ensure the LLM comprehends the context and can read and retrieve the input data accurately.
- **Message-level instructions** specify preferred data format and instructions for analyzing and presenting improvement opportunities. In particular, instructions indicate what aspects must be included in the response: *"when analyzing waiting time causes in a transition, the response must include 1. where the waiting time is located (e.g., in which transition), 2. the waiting time duration, 3. the causes of waiting time and the duration of the causes of waiting time (how much waiting time is induced by each waiting time cause*

and percent from the transition's duration), 4. resources that contribute the most to the waiting time in this transition, and how much." These instructions are used to ensure that the LLM provides relevant, sufficient, and consistent results.

(ii) Prescriptive analysis component. Second, we use the LLM to recommend redesign options to address the identified causes of waiting times (e.g., resource unavailability). The prescriptive component guides the LLM in generating these redesign recommendations. For this component, we designed and tested three versions: (1) baseline ("zero-shot") method, (2) enhanced-unstructured method, and (3) enhanced-structured method. Below we describe these versions of the prompting method in detail.

(1) **Baseline ("zero-shot") method** includes the instructions on recommending redesign options:

- **Message-level instructions** specify how the recommendations on redesign patterns should be provided. In particular, they indicate what aspects must be included in the response: *"when recommending redesign options, the response must include: 1. where the redesign should be applied (e.g., to which transitions, activities, or resources), 2. what redesign option to implement (the name of the redesign option; definition and application, i.e., explanation of the redesign option and how the redesign option should be applied; benefits and impact, i.e., how the redesign option can impact the process performance and why this redesign is relevant."*

(2) **Enhanced-unstructured method.** LLMs can encounter challenges in managing domain-specific or highly specialized queries that require information outside of their training data corpus [201]. To overcome this limitation, the Retrieval-Augmented Generation (RAG) approach prescribes providing external data sources directly to LLMs, which can be used as contextual information for retrieving data and generating more accurate and relevant responses [202]. Therefore, to enhance the accuracy and relevance of LLM responses [202] in process redesign tasks, we provided the LLM with research papers proposing the collections of redesign patterns, as external data sources: Reijers & Mansar [45], Lashkevich [47], and Fehrer [46]. Accordingly, the LLM can access these documents to retrieve relevant data when recommending redesign options.

In this way, the enhanced-unstructured method includes the same instructions as the baseline ("zero-shot") method, three documents containing collections of redesign patterns (namely, Reijers & Mansar [45], Lashkevich [47], and Fehrer [46]), and instructions asking the LLM to use these documents when recommending redesign options.

(3) **Enhanced-structured method**. To further fine-tune the LLM for recommending redesign options to reduce waiting times, we created a structured dataset with redesign options aimed at improving the temporal performance of business processes. We incorporated this dataset into the prompt instead of the three doc-

uments included in the enhanced-unstructured method. Providing a specialized dataset tailored to our redesign aims offers several advantages. First, the structured dataset can be easier and more efficiently queried due to its organized format. Second, it reduces the search space that the LLM needs to consider when identifying redesign options, making the search process more efficient. Third, it creates a focus on temporal improvement, thereby guiding the LLM to concentrate its reasoning based on the provided data [203]. This can result in the LLM more effectively and accurately identifying the relevant redesign options to reduce waiting times.

To design the dataset of redesign patterns, we use the previously conducted SLR [47] (presented in Chapter 3). In this SLR, we reviewed 204 papers describing methods or cases of how improvement opportunities have been identified and how the processes have been redesigned, focusing on the case studies (i.e., describing practical cases of process analysis and redesign). From the review, 80 improvement opportunities and 64 process redesign patterns were identified. Based on the obtained results, we derived a catalog of defined improvement opportunities and the redesign patterns to address each improvement opportunity.

In this research, we aim to identify redesign patterns that can reduce waiting times and improve the temporal performance of business processes. Therefore, we reviewed the catalog of improvement opportunities and redesign patterns to extract those redesign patterns that were reported to positively impact the temporal performance of business processes, i.e., when "Redesign impact" -> "Positive" contained "Time," This review yielded a list of 41 redesign patterns, such as "Customer scheduling" and "Extra resources."

However, since the papers included in the SLR captured only case studies, they might not cover all possible redesign patterns known to be applicable for the temporal improvement of business processes. Therefore, to enrich the obtained list of redesign patterns, we also reviewed the following studies: Reijers & Mansar (2005) [45] that presents 29 best practices of process redesign derived from a literature survey and supplemented with author experiences, and Fehrer (2023) [46] that presents a collection of 50 redesign patterns for process improvement observed in the field. This review added another 16 redesign patterns applicable for the temporal improvement, such as "Order assignment" (i.e., let workers perform as many steps as possible for single orders) [45] and "Workload-based task assignment" (i.e., allocate tasks based on individuals' incomplete workload) [46]). In addition, we included 3 redesign patterns (namely, "Batch strategy optimization," "Prioritization strategy optimization," and "Resource schedule optimization") that could be applied to reduce waiting times due to respective causes (i.e., due to batching, prioritization, and resource unavailability). As a result, we obtained a dataset containing 60 redesign patterns. The overview of the collected redesign patterns and their sources are presented in Table 16.

The dataset of redesign patterns was designed in a structured format. For each redesign pattern, it provides a redesign pattern name, its definition, explanation

Table 16. Identified redesign patterns and their sources.

Redesign patterns	**Quantity**	**Source**
Arrival time incentives, Capture data at source, Case buffering, Case reassignment, Case-based work, Contact reduction, Customer integration, Customer scheduling, Customer teams, Data composition, Data elimination, Data standardization, Empower, Exception, Extra resources, Fixed assignment, Flexible assignment, Follow-up, Fragment automation, Fragment elimination, IT system integration, IT system interfacing, IT system upgrade, Integral technology, Inventory buffering, Order types (Case types), Parallelism, Periodic action, Process automation, Process decomposition, Process generalization, Process standardization, Resequencing, Resource scheduling, Role independence, Specialist, Task automation, Task composition (combination), Task decomposition, Task elimination, Triage.	41 patterns	Lashkevich (2020) [47]
Buffer information, Establish standardized interfaces, Information buffering, Order assignment, Resource centralization, Trusted party.	6 patterns	Reijers & Mansar (2005) [45]
Automate for environmental impact, Department-based assignment, Experience-based task assignment, Expertise-based task assignment, First-contact problem resolution, Performance-based task assignment, Role-based task assignment, Task delegation, Teamwork-based assignment, Workload-based task assignment.	10 patterns	Fehrer (2023) [46]
Batch strategy optimization, Prioritization strategy optimization, Resource schedule optimization	3 patterns	Author
Total	**60 patterns**	

(i.e., a detailed description of the redesign pattern), examples of its applications, positive and negative impacts (w.r.t. time, cost, quality, flexibility), and references. An excerpt of the obtained dataset is provided in Figure 44. A complete dataset of redesign patterns is available in supplementary materials[3].

Redesign patterns	Redesign pattern definition	Explanations	Examples	Positive Impact	Negative Impact	References
Customer integration	Assign an activity execution to a customer to reduce the employees' workload.	Companies should offer customers the possibility to perform tasks included in business processes that have previously been performed by employees. When implementing this heuristic, companies should consider whether customers can choose between self-service and the traditional process. Companies must also consider that not all tasks included in business processes can be performed by customers. Performance considerations: Empowered interaction capability, concerted interaction capability. By applying this redesign pattern, the employees' workload could be reduced and employees could focus on the narrower and more specialized scope of tasks. Additionally, when a substantial part of the process can be delegated to customers themselves, it has the potential to free up resource capacity, thereby reducing the need for additional resources, e.g., self-service options in grocery stores. Furthermore, customers can experience reduced waiting times as they no longer have to queue for employee assistance. However, it's crucial to underline that customers should reap tangible benefits from this integration, such as shorter waiting times. If customers do not perceive significant advantages (e.g., customers perform part of the process but waiting times remain high), customer satisfaction may decrease.	Example 1 In an outpatient process at a hospital, long waiting times and patient queues occurred due to a lack of personnel to serve all the patients in time. The most extended delay occurred in the payment procedure where the patients needed to wait in a queue for a long time to be served by the cashiers. To overcome delays in the payment procedure, self-payment devices were introduced. Thus, some of the patients are served by the cashiers, and others use self-service devices to make payments. [107] Example 2 In an incident reporting process at an IT Help Desk, only 30% of the total number of incidents are recorded. Minor issues are not recorded at all. Hence, the same or similar issues are repeatedly solved from scratch, which increases the cycle time of the process. Client self-service can be used to resolve minor issues by clients themselves. With access to the Knowledge Base System, clients can review the recorded minor incidents and fix them without contacting customer service. [73] Example 3 A registrar's office at a university deals with several processes, one of which is the matriculation renewal process, which takes place twice a year. During the weeks when the renewal of matriculation	Time, Cost	Flexibility	73, 81, 107, 133, 155

Figure 44. Excerpt from the dataset of redesign patterns.

In this way, the enhanced-structured method includes the same instructions as the baseline ("zero-shot") method, the structured dataset of 60 redesign patterns aimed to improve the temporal performance, and instructions asking to use this

[3] https://doi.org/10.6084/m9.figshare.25421107

dataset when recommending redesign options. Table 17 summarizes the data included in the three versions of the proposed prompting method, highlighting the differences between them.

Table 17. Three versions of the proposed prompting method.

<table>
<tr><th>Version</th><th>Prescriptive component</th><th>Descriptive component</th></tr>
<tr><td>Baseline (“zero-shot”) method</td><td>- Message-level instructions</td><td rowspan="3">- Assistant-level instructions
- Message-level instructions</td></tr>
<tr><td>Enhanced-unstructured method</td><td>- Message-level instructions
- 3 research papers with redesign pattern collections</td></tr>
<tr><td>Enhanced-structured method</td><td>- Message-level instructions
- Structured dataset of redesign patterns</td></tr>
</table>

During the design and development of the proposed prompting method, we used 2 event logs (cf. Procure-to-Pay and Production logs in previously referenced supplementary materials) to test and fine-tune the 3 versions of the prompting method for recommending redesign options to reduce waiting times. We conducted two rounds of testing. In the first exploratory testing round, we asked the LLM to propose redesign options to reduce waiting times in the transition with the highest waiting time. We asked this question using each of the 3 versions of the prompting method above, and using each of the two logs. We examined the responses in search of visible deficiencies. For example, in the descriptive component, we found that the LLM answered the question “Identify the transition with the longest duration” by returning the transition with the largest maximum duration rather than the largest mean duration, as one would expect. In the prescriptive component, we found that the LLM was not reading through the patterns provided in the input (both for the enhanced-structured and enhanced-unstructured methods). We fine-tuned the instructions to clarify the terminology and to instruct the LLM to read the inputs thoroughly.

In the second round, we repeated the experiment on the same logs to test if the 3 prompting methods yield distinct sets of redesign options. During these tests, we noted that with the enhanced-unstructured method, the LLM mostly ignored the redesign patterns provided in the input documents. Out of 40 recommended redesign options, only 15% of them were based on the documents provided as inputs. In most cases, the unstructured-enhanced method recommended the same redesign options as the “zero-shot” baseline. Meanwhile, the enhanced-structured method recommended redesign options based on the prompt content in 68% of cases. Given that the outputs of the enhanced-unstructured method were largely similar to the baseline, we excluded the enhanced-unstructured method from the subsequent evaluation.

The detailed results of version testing and the full text of the prompts (instructions for LLM and the redesign patterns knowledge-base) are given in the previously referenced supplementary materials.

6.4. Implementation

The enhanced-structured version of the prompting method is implemented as a conversational interface integrated with Kronos (introduced in Section 5.3). Analysts can use this interface to conversationally query waiting time analysis results and obtain redesign suggestions for reducing waiting times.

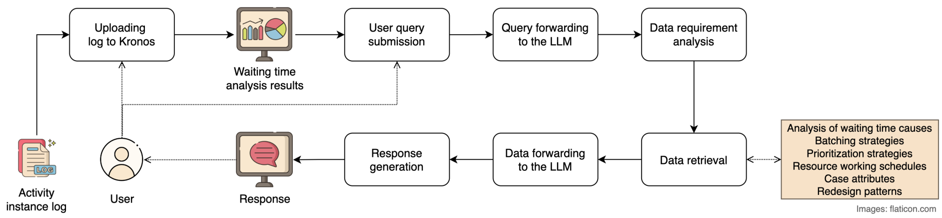


Figure 45. Overview of the conversational interface workflow.

The flow of the conversational interface is depicted in Figure 45. Users must first upload an activity instance log to Kronos and obtain the waiting time analysis results visualized on the dashboard. Then, the conversational interface becomes available on the main page of the Kronos dashboard. The interface presents a text input box where users can type their questions (see Figure 46). When a user submits a question, it is then forwarded to the LLM. The LLM performs a data requirement analysis to determine the specific data needed to generate a response (e.g., for a question about the efficiency of batching in a given transition, the data on batching strategies and waiting time cause decomposition in this transition are required). Next, relevant data is retrieved from one or several available datasets (e.g., from "analysis of waiting time causes" and "batching strategies" to analyze the efficiency of batching) and sent back to the LLM. Finally, the LLM processes the retrieved data, generates a response, and presents the response to the user. Afterward, a user can continue with follow-up questions. The interface tracks the conversation history and builds on previous interactions for coherent and relevant dialogue. Alternatively, a user can proceed with questions on a different issue. A screencast that describes the application of the conversational interface is available on YouTube[4].

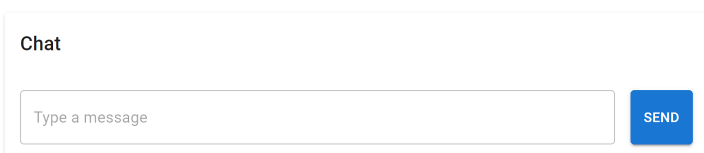


Figure 46. Chat interface for user question submission.

[4] `https://youtu.be/ScxjMnjqPc8`

The conversational interface is publicly available at `http://kronos.cloud.ut.ee/`. The implementation of the interface logic is available in GitHub repositories: one for the frontend[5] and one for the WebAPI[6], together with instructions on usage. An API key from OpenAI is needed to start the tool and generate answers. The implementation of the conversational interface is presented in more detail in [60].

6.5. Computational evaluation

In this section, we present the computational evaluation of the proposed approach. First, we use synthetic and real-life event logs to validate the ability of the prompt-enhanced LLM to provide accurate responses when analyzing waiting time causes. Second, we demonstrate the applicability of the prompt-enhanced LLM for analyzing waiting times and recommending redesign options to reduce waiting times on a real-life event log.

6.5.1. Accuracy verification

In this section, we present the computational evaluation of the proposed prompting method. To verify that the prompt-enhanced LLM (descriptive analysis component) correctly identifies waiting time durations and their causes, we conducted a computational experiment aimed at addressing the following evaluation question:

EQ1 To which extent does the prompting method enable LLM to provide accurate responses when analyzing waiting time causes? (cf. requirement R5)

For this evaluation, we compiled a list of questions derived from [196], focused on descriptive analysis of waiting times. We filtered out questions not related to waiting times, removed duplicates, and, when needed, adapted them to our scope. For instance, we modified "How can I filter only the slowest cases?" to "Which cases had the longest waiting time?" as the focus is on identifying improvement opportunities and not specifying instructions. As a result, we obtained a list of 28 questions. We supplemented this list with another 27 questions related to possible causes of waiting times (specifically batching, resource contention, prioritization, and resource unavailability) and questions related to these causes (batching and prioritization strategies, and resource working calendars). As a result, the list consisted of 55 questions that targeted the analysis of waiting time from multiple perspectives, such as waiting time durations (e.g., "Where is the highest waiting time?"), causes (e.g., "What is the main cause of waiting time and how much?"), resources (e.g., "Which resources cause the most waiting?"), cases (e.g., "What cases are affected by waiting time the most?") and cause-related information (e.g., "What are the batching strategies in this process?"). For the list of questions used for accuracy verification, see the previously referenced supplementary materials.

[5] `https://github.com/AutomatedProcessImprovement/waiting-time-frontend`
[6] `https://github.com/AutomatedProcessImprovement/waiting-time-backend`

To evaluate the enhanced-structured method, we submitted these 55 questions in the context of four event logs: procure-to-pay, claims management, device repair, and production. Not all questions were applicable to every event log. For instance, the device repair process did not have any waiting times due to prioritization, so the questions associated with prioritization were skipped for this log. After a manual assessment of the applicability of each question on each log, we obtained 175 relevant question-answer pairs out of 220 possible question-log combinations. To mitigate stochastic effects, each question was submitted twice. Hence, a total of 350 responses were obtained. For each relevant question-log pair, we manually built a ground truth using the waiting time analysis results produced by Kronos. Using this ground truth, we assessed the accuracy of the generated responses produced by the LLM on a score scale from "0 (Total error)" to "5 (Fully correct)" (the complete score scheme is provided in Table 18). This score scheme was developed based on the established 5-point Likert scale [204]. The Likert scale is commonly used to assess respondents' attitudes or levels of agreement on a particular subject, offering a range of 5 options from "strongly disagree" to "strongly agree" [204]. We adapted the Likert scale statements to define levels of alignment between the generated responses and the ground truth. Thus, the score of "5 (Fully correct)" corresponds to the highest point on the Likert scale, "strongly agree," as both represent the highest level of alignment with the intended response. Similarly, the score of "1 (Unrelated/error)" corresponds to the lowest point on the Likert scale, "strongly disagree," as both indicate a complete lack of alignment with the intended response. Additionally, we included a score of "0 (Total error)" to represent instances of no response or a complete failure to address the question, as in such cases, there is no content to assess for meaningful alignment using the 5-point scale.

Table 18. Score scheme for assessing the accuracy of the prompted LLM responses.

Score	Description
0 (Total error)	No response or complete failure to address the question.
1 (Unrelated/error)	Response provided but unrelated to the question or acknowledges an error.
2 (Slightly related)	Mostly unrelated with only a vague connection to the question or contains significant mistakes.
3 (Related but flawed)	Related to the question but with major inaccuracies or very minimal relevant content.
4 (Mostly correct)	Related and mostly accurate with minor errors or slight irrelevance.
5 (Fully correct)	Fully relevant and accurate with no or negligible errors.

First, we identified responses where no output was provided or a system error occurred and assigned them a score of 0. Next, to evaluate the remaining responses, we used a separate GPT-4 Turbo LLM session to act as a referee and assess the responses provided by the first LLM. The referee LLM was instructed to evaluate and rate responses from the first LLM session against the ground truth on a scale from “1 (Unrelated/error) to “5 (Fully correct)” as provided in Table 18.

To cross-check the automated accuracy assessment of the referee LLM, we validated its assessments on a random sample of question-log pairs. Following a simple random sampling approach [205], we calculated the necessary sample size for a statistically significant subset – 183 out of 350 question-log pairs. We randomly selected 183 question-log pairs, and we manually assigned an accuracy score between 1 to 5 to the responses provided by the first LLM. As expected, sometimes the referee LLM assigned accuracy scores different from our manual assessments. However, these deviations were minimal, occurring only between scores 1 and 2 (i.e., the referee LLM assigned a score of 1 while we assigned a 2, or vice versa), and between scores 3 and 4 (i.e., the referee LLM assigned a score of 3 while we assigned a 4, or vice versa). In all cases where the referee LLM assigned a perfect score (5), our manual assessment matched this score.

The enhanced-structured method achieved a mean accuracy rating of 4.3/5.0. This mean score excludes 12 responses that scored 0 due to no output or system errors because such responses lack substance for accuracy comparison. The most frequent error occurred when the LLM did not generate a response due to the anticipated lengthiness and, instead, asked the user to proceed or not (cf. supplementary materials). Out of the remaining responses, 66% were rated the highest score of 5 (see Figure 47).

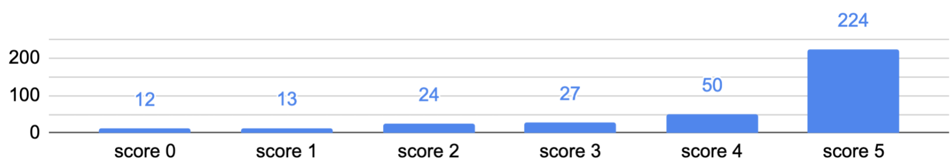


Figure 47. Accuracy verification results.

6.5.2. Applicability demonstration

To illustrate the application of a conversational interface, we use a real-life event log of a production process [181] (previously presented in Sec. 4.3.2 and used in Sec. 5.4.2). The first question aims to identify improvement opportunities by focusing on the transition that caused the most waiting time in the process. Therefore, we submit the following question through the text input box: “Identify and analyze the transition with the highest waiting time in total (for the whole dataset).” Upon submission, the LLM examines the available datasets to retrieve relevant data. Then, it generates a response specifying the transition with the highest total waiting time and explaining potential causes (see Figure 48). The response indicates that the “Turning & Milling Q.C.” loop has the highest waiting time, totaling around 3 years. The most significant causes of this waiting time are resource contention (about 55% of the total waiting time) and batching (21%). The primary resource contributing to the waiting time in this transition is “ID4618” (see Figure 49). These results highlight the potential improvement opportunities. The analysis results provided in the response are consistent with those produced by Kronos and depicted in the Kronos dashboard (see Sec. 5.3).

You
Identify and analyze the transition with the highest waiting time in total (for the whole dataset)

Assistant
Highest Waiting Time Transition Analysis

Transition: "Turning & Milling Q.C." -> "Turning & Milling Q.C."
Total Waiting Time: 94275480 seconds (approx. 3 years)

Breakdown of Waiting Times by Cause:
- **Unavailability:**
- **Duration:** 51650940 seconds (approx. 1 year 7 months)
- **Percentage:** 54.80%

- **Batching:**
- **Duration:** 19443900 seconds (approx. 7 months)
- **Percentage:** 20.63%

- **Prioritization:**
- **Duration:** 8832480 seconds (approx. 3 months)
- **Percentage:** 9.37%

- **Contention:**
- **Duration:** 6818940 seconds (approx. 2 months)
- **Percentage:** 7.23%

- **Extraneous:**
- **Duration:** 7529220 seconds (approx. 2 months)
- **Percentage:** 7.97%

Figure 48. Response to the question on waiting time analysis (part 1).

Once improvement opportunities are identified, the next step is process redesign. Therefore, we submit the second question, focusing on addressing these opportunities: “Based on the analysis results of this transition, what specific process redesigns would you suggest to reduce waiting time?” In this case, the LLM uses data from the previous question-answer pair and a dataset of redesign patterns useful for temporal improvements. As a result, it suggests the redesign patterns that can be applied to reduce the identified waiting times by targeting their causes (see Figure 50). Specifically, to address the waiting time due to resource unavailability, the LLM recommends adding extra resources and adjusting the resource schedules to ensure better availability during high-demand periods. Additionally, to reduce waiting time due to batching, the LLM suggests reviewing and adjusting the current batching strategy applied in the “Turning & Milling Q.C.” activity. Changing the prioritization strategy (i.e., adjusting prioritization rules or frequency) is suggested to reduce waiting time due to prioritization.

As a result, this demonstration showcased how the proposed solution can be applied to a real-life log and how the outputs can be used to identify improvement opportunities and possible redesign options. This indicates the practical applicability of the proposed solution to a real-life log.

Destination Resources Contributing to Most of the Waiting Time:

1. **Resource ID4618**
- **Waiting Time:** 50428440 seconds (approx. 1 year 7 months)

2. **Resource ID4287**
- **Waiting Time:** 16972020 seconds (approx. 6 months)

3. **Resource ID4163**
- **Waiting Time:** 12812880 seconds (approx. 5 months)

4. **Resource ID4493**
- **Waiting Time:** 9057060 seconds (approx. 3 months)

5. **Resource ID0420**
- **Waiting Time:** 2881380 seconds (approx. 1 month)

6. **Resource ID4955**
- **Waiting Time:** 1305660 seconds (approx. 15 days)

7. **Resource ID0937**
- **Waiting Time:** 681000 seconds (approx. 8 days)

8. **Resource ID4529**
- **Waiting Time:** 126600 seconds (approx. 1.5 days)

9. **Resource ID4167**
- **Waiting Time:** 10440 seconds (approx. 3 hours)

Summary

- The main cause of waiting time is **Unavailability**, contributing approximately 54.80% of the total waiting time.
- Resource **ID4618** is the largest contributor to the waiting time in this transition, accounting for around 1 year and 7 months.

Figure 49. Response to the question on waiting time analysis (part 2).

Assistant
Recommendations for Redesign

Main Cause: Resource Unavailability (54.80% of Total Waiting Time)

1. **Redesign Option 1: Resource Schedule Optimization**
- **Definition:** Adjust the working schedules of the resources to cover the periods when waiting times due to resource unavailability occur.
- **Explanation:** To reduce waiting time due to resource unavailability, resource schedules should be optimized. For example, more resources should be available during peak hours of demand and fewer during low-demand hours.
- **Application:**
- Analyze the current availability and demand patterns for the task "Turning & Milling Q.C."
- Adjust the schedules of key resources like ID4618 to ensure their availability aligns better with peak demand periods.
- **Benefits:** Reduced waiting times by having resources available when needed, leading to a smoother workflow and decreased cycle times.
- **Impact:** Significant reduction in waiting time due to unavailability, potentially improving resource utilization and cycle time by around 55%.

2. **Redesign Option 2: Extra Resources**
- **Definition:** Increase the number of resources if their capacity is not sufficient.
- **Explanation:** Allocating additional resources increases capacity, thus handling more tasks concurrently and reducing queue times.
- **Application:**
- In situations where Resource ID4618 is a major bottleneck, additional similar resources should be allocated.
- Ensure that the additional resources are trained and capable of handling the same tasks.
- **Benefits:** Increased capacity to handle tasks, reduced waiting time, and improved process efficiencies.
- **Impact:** Comprehensive reduction in waiting time due to unavailability, with a significant impact on cycle times, particularly in peak demand situations.

Secondary Cause: Batching (20.63% of Total Waiting Time)

1. **Redesign Option 3: Batch Strategy Optimization**
- **Definition:** Adjust batching strategy to reduce the waiting time and/or processing time of the batch-processed cases.
- **Explanation:** Adjust batching strategies by changing batch processing rules, size, type, or frequency to optimize the flow.
- **Application:**
- Reduce the batch size to decrease the waiting time for collecting the batch.
- Increase the batching frequency to ensure tasks are processed more regularly, thus reducing waiting times.
- **Benefits:** Reduced batching-related waiting times, leading to more continuous and less intermittent workflows.
- **Impact:** Potential reduction of around 20% in waiting time, leading to smoother task progression and decreased overall process time.

Tertiary Cause: Prioritization (9.37% of Total Waiting Time)

1. **Redesign Option 4: Prioritization Strategy Optimization**
- **Definition:** Adapt the prioritization strategy to minimize the waiting time for cases that are not receiving priority.
- **Explanation:** Change the prioritization rule or frequency to ensure a more balanced and equitable handling of tasks.
- **Application:**
- Adjust the prioritization rules to be less exclusive, enabling a more balanced workload distribution.
- Reduce the frequency of prioritization interventions to decrease waiting time due to prioritization.
- **Benefits:** More balanced workload distribution, reduced wait times for lower-priority tasks.
- **Impact:** Possible reduction of around 9% in waiting times, with improvements in the equity of task handling.

Figure 50. Response to the question on process redesign to reduce waiting times.

6.6. User evaluation

To evaluate how the prompt-enhanced LLM aids analysts in analyzing and redesigning waiting times, we conducted an interview study with process analysts experienced in process mining. This section presents the design of the user evaluation and discusses the obtained findings.

6.6.1. Interview study design

We conducted the user evaluation with two aims: (1) to validate the effectiveness of the proposed prompting method in assisting analysts with analyzing waiting time causes, and (2) to determine which version of the prompting method analysts prefer for identifying redesign options. Thus, the evaluation focused on the following questions:

EQ2 In what way can a prompt-enhanced LLM aid analysts in analyzing waiting times causes? This EQ aims to evaluate the usefulness and sufficiency of the information provided in the LLM responses relative to the task of analyzing causes of waiting time (descriptive analysis component) (cf. requirement R6).

EQ3 How efficient is the proposed prompt-enhanced LLM relative to the "zero-shot" baseline, when it comes to recommending redesign options to reduce waiting times? This EQ aims to validate if the enhanced prompting method improves the LLM's ability to provide useful redesign recommendations (prescriptive analysis component) (cf. requirements R7-10).

We recruited 12 process analysts (Table 19) with experience in process mining and process improvement projects. On average, the participants had 6.5 years of experience with process analysis and 4.5 years with process mining.

Table 19. Interview participants.

Code	Domain	Experience (process analysis)	Experience (process mining)
P1	Manufacturing, logistics	5 years	5 years
P2	Academic	4 years	3 years
P3	Various	5 years	5 years
P4	Academic	6 years	6 years
P5	Various	10 years	2 years
P6	Research	1 year	3 years
P7	Software development	6 years	6 years
P8	Aviation, manufacturing, logistics	20 years	10 years
P9	Production, distribution	5 years	3 years
P10	Manufacturing	10 years	6 years
P11	Accountability	3 years	3 years
P12	Information technology	4 years	3 years

We combined semi-structured interviews with a survey study. Semi-structured interviews facilitated open dialogues [188], allowing participants to articulate their feedback regarding LLM responses. The survey allowed for quantitative analysis to complement the qualitative insights obtained from the interview data [189].

We conducted individual interviews online. First, we introduced the key definitions of waiting time analysis (such as the waiting time causes). Then, we showed an application of a prompt-enhanced LLM analysis of a procure-to-pay process and asked evaluation questions.

In the first part, we showed the response of the LLM to "Where is the highest waiting time in this process on average? Analyze it." The response included the waiting time location (i.e., the transition), its average duration, waiting time causes and their durations, and resource pairs that contributed the most on average. Participants reviewed the response, and then we discussed it. The questions covered information usefulness and sufficiency (EQ2). For instance, to evaluate usefulness, we asked *"Which data is useful when analyzing waiting time causes? Which data is not useful, and why?"* After the discussion, the participants filled out a survey. The survey included three statements for measuring information usefulness, sufficiency, and overall satisfaction. We used a scale from 1 (strongly disagree) to 5 (strongly agree) for these statements.

In the second part, we showed the LLM responses to "How to reduce waiting time in this transition?" – i.e., the transition analyzed in the first part of the interview. Here, we examined if an LLM with an enhanced prompt will provide more meaningful and useful recommendations on process redesigns compared to a "zero-shot" baseline (EQ3). Therefore, we used a simultaneous within-subjects design, applicable when assessing two conditions of recommending systems [206]. It implies that participants interact with both experimental conditions at the same time [206] – in our case, participants review two responses – one generated by the LLM with a minimalistic prompt ("zero-shot" baseline) and one provided by the LLM with an enhanced prompt. Using the simultaneous within-subjects design is relevant as it allows participants to effectively compare the responses, identify differences between them, and select the preferred one [206].

Each response included 10 redesign options – 2 redesign options for each waiting time cause (batching, prioritization, resource contention, resource unavailability, and extraneous factors). We displayed both responses side by side. To mitigate the potential location bias (e.g., when the response on the left may be perceived as superior) [199], their positions (left or right side) were randomized [206]. The participants reviewed the responses and answered questions related to recommender system properties (8 properties described in Section 6.2): recommendation relevance, novelty, diversity, serendipity, understandability, information usefulness, sufficiency, and confidence. For instance, to evaluate the relevance of the recommended redesigns, we asked *"Which redesign options do you consider relevant? Which are not relevant, and why?"*

After the discussion, the participants filled out a survey. The survey questions are based on an existing scale for evaluating recommender systems [200]. Since we aim to compare two lists of recommended redesign options, we adapted the questions to be comparative instead of absolute [200]. In each question, 5 options were available, with the extremes labeled as "Much more A than B" and "Much more B than A." Thus, to evaluate users' preferences related to recommendation relevance, the survey asked *"Which response has more relevant redesign options for addressing the identified improvement opportunities?"* Similarly, participants rated their preferences based on all 8 properties. Finally, the participants answered which response they were more satisfied with.

The interviews were conducted online and lasted between 36 and 70 minutes. We recorded the interviews, transcribed them using Otter.ai[7], and manually reviewed the transcripts. To analyze the interview data, we followed a thematic analysis method and coded the transcripts iteratively. First, we created an initial set of codes based on the evaluation questions (e.g., "relevant" and "not relevant"). Then, we applied these codes and refined them, e.g., we added the code "undefined" for situations when participants could not assess the recommendation relevance. As a result, we obtained a set of 47 codes clustered into 21 themes.

For the user study protocol and coding scheme, see the previously referenced supplementary materials.

6.6.2. Interview study findings

EQ2. First, the participants evaluated how the prompt-enhanced LLM aids in analyzing waiting time causes. Results indicated that participants found the LLM response information useful. This is supported by the survey responses on usefulness (M=4.42 (out of 5), SD=0.51)[8] (see Figure 51). Specifically, participants indicated that the prompt-enhanced LLM *"detects the location and resources responsible for that, and it detects the causes, so they are definitely useful"* (P5). The causes of waiting time and associated resources were perceived as the most useful information by all participants. They also appreciated that the prompting method *"can quickly summarize the analysis results"* (P9) and saves time by automatically obtaining analysis results that analysts typically acquire manually.

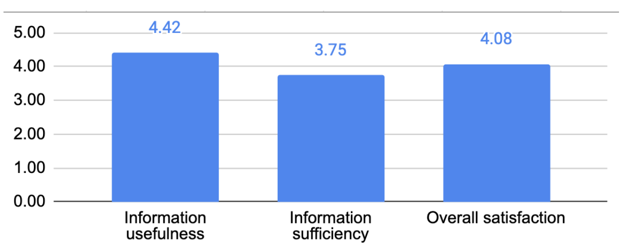


Figure 51. Survey results (EQ2).

[7] `https://otter.ai/`

[8] "M" stands for mean value and "SD" for standard deviation.

We explicitly asked the participants if they missed any information in the response for identifying improvement opportunities. Several participants found the information sufficient but noted the need for more insights to better understand factors contributing to waiting time. This is supported by the survey responses on sufficiency (M=3.75/5, SD=0.62) (see Figure 51). Participants were particularly interested in how resources contributed to each waiting time cause, the percentage of cases affected by waiting time, and case attributes correlating with waiting time. They also noted that *"if some data is not immediately here in the response, some additional questions can be asked to provide this type of information"* (P2). Overall, participants were satisfied with how the prompt-enhanced LLM helps analyze waiting time causes (overall satisfaction is rated as M=4.08/5, SD=0.67).

EQ3. Second, the participants compared the recommended redesign options obtained by two prompting methods, a baseline and an enhanced one. For consistency, hereinafter, we call the responses provided by a baseline method "Response B" and provided by an enhanced method "Response E."

Recommendation Relevance. In general, participants found relevant redesign options in both responses. However, 9 participants (P1, P3, P4, P6, P8-12) mentioned that response E provided specific recommendations, whereas response B provided high-level suggestions. Thus, recommendations in response E were described as *"more action-oriented and practical"* (P8), *"pragmatic"* (P4), *"low-level, technical suggestions"* (P3), providing *"concrete actions and simple solutions"* (P10). Recommendations in response B were characterized as *"more general"* (P6, P9), *"high-level"* (P10), *"strategic"* (P3), that are to be *"discussed at the leadership level"* (P3). In addition, participants (P3, P4) noted that both responses are relevant depending on the target audience for recommendations.

Recommendation Novelty and Serendipity. Overall, participants were familiar with the redesign options provided in both responses. Nevertheless, they noted that response B was able to provide *"more creative"* (P4, P11) and *"interesting"* (P2, P7) recommendations. They also mentioned that creative suggestions might not always be relevant, e.g., *"dynamic batching adjustment, job rotation [response B] – it's something that is more creative. And since we think it's not practical, we never proposed that to any clients"* (P4).

Recommendation Diversity. In general, participants found both responses to be diverse in their recommendations. Several participants indicated that response B is slightly more diverse since it *"provides a better overview from different angles on the issue"* (P10) and *"includes more quality-based approaches"* (P4). Others found response E slightly more diverse because it *"has fewer similar options"* (P3), whereas response B *"speaks about the same things in two different ways"* (P11). P12 also observed that they didn't view diversity as an advantage since diverse suggestions can be challenging to implement simultaneously, whereas similar redesign options are complementary and can be easily integrated.

Recommendation Understandability. Participants generally indicated no major difference in understandability between the two responses. However, several participants preferred response E along this aspect. Particularly, they noted that *"the explanations are more natural and clear. For me, they are easier to understand, more straightforward, and future-oriented"* (P8). Several participants (P11, P12) pointed out that the terminology employed in response E was more aligned with the field of process improvement, resulting in clearer explanations. Moreover, participants (P6, P8, P11, P12) valued the structure and conciseness of response E and its employment of clearer language (P6, P8).

Information Usefulness. Most participants (P2-4, P7-9, P11) found all provided information useful for understanding the recommended redesign options. However, some participants identified overlapping information as less useful, e.g., they suggested merging benefits and impact into one category (P1, P6, P10, P12). Participants also noted that response E contained more useful information as it was *"more detailed"* (P8) and *"more specific"* (P6).

Information Sufficiency. Since both responses covered the same aspects (redesign option definition, application, benefits, and impact), they were evaluated as equally sufficient. We explicitly asked what information participants missed when exploring recommended redesign options. Thus, participants generally inquired for more specific information connected to the analyzed process. For instance, they suggested including detailed justifications for why a redesign option is recommended (P2, P4, P5), concrete instantiations of redesigns (e.g., how many resources to add) (P4, P6), and estimates of how the waiting times will decrease (P3, P6). While these responses lacked specificity, participants agreed that they are a valuable source of ideas, e.g., *"a good starting point"* (P8). Thus, P9 noted that *"sometimes we didn't think of that particular option, it simply didn't come to our mind. But here it provides possible solutions that you can get some ideas from, and it's fantastic"* (P9).

Confidence and Overall Satisfaction. Based on the previous reflections, participants generally concluded that they were more confident and satisfied with response E because of the higher relevance of the recommendations. For instance, P2 noted: *"I would be more confident with [response E], actually significantly more confident simply because [response B] gave some suggestions that I found questionable"* (P2). Similarly, P4 said: *"I'm more satisfied with [response E] because my view is more pragmatic. So the suggestions should be quantifiable and action-based as in [response E]"* (P4).

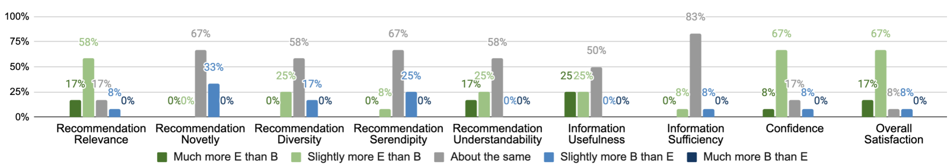


Figure 52. Survey results (EQ3).

The participants' statements were supported by the survey results (Figure 52).

The enhanced prompting method (response E) scored higher on recommendation relevance, confidence, and overall satisfaction. Half of the participants saw no significant difference in information usefulness, while the other half preferred the enhanced method (response E) w.r.t. this property. Both prompting methods were similar in diversity, understandability, and information sufficiency. While both prompting methods were comparable in recommendation novelty and serendipity, the baseline method (response B) was preferred slightly more often than the enhanced one (response E).

6.7. Summary

This chapter proposed a prompting method to enhance the LLM's capacity to analyze waiting time causes and recommend redesign options to reduce waiting times by targeting their causes. The crux of the prompting method is to include a structured list of redesign patterns, curated w.r.t. the target improvement objective, to enable LLMs to offer recommendations on process redesign. The computation evaluation verified the ability of the prompt-enhanced LLM to provide accurate responses when analyzing waiting time causes and its practical applicability to real-life logs. The evaluation, conducted with process analysts, demonstrated the prompting method's effectiveness in analyzing waiting time causes and suggesting relevant redesign options. Moreover, comparison with a baseline method revealed that the enhanced prompting method excels at recommending actionable redesign options tailored for process analysts, while the baseline method leans towards high-level solutions suited for managerial decision-making.

This research has several limitations. First, there are limitations inherent to the DSR methodology. The user evaluation with 12 analysts poses a limitation on the generalizability of the results, i.e., the extent to which the results are applicable in other settings. This limitation was partially mitigated by interviewing analysts from different domains and backgrounds. Additionally, selecting a different LLM, processes, and LLM responses might have also yielded different results.

We adopted a simultaneous within-subject approach to compare two prompting methods, which helped minimize between-participant variability and avoid order effects [206]. To mitigate position bias, we randomized the placement of responses [206]. We acknowledge that presenting two conditions simultaneously deviates from a realistic usage scenario, constituting a limitation. However, our goal was to evaluate the response content instead of tool usability.

Second, there are limitations related to the LLM performance. Thus, LLM applications face challenges with prompting and output correctness [207]. Prompt brittleness occurs when slight changes in input phrasing lead to significantly different outputs, reducing reliability and consistency. Additionally, LLMs may generate plausible-sounding but incorrect answers (hallucinations). Hence, it is useful to test designed prompts for accuracy and incorporate human-in-the-loop reviews [207] to ensure reliable and correct LLM outputs in practice.

7. CONCLUSION

7.1. Summary of contributions

This thesis investigated how the causes of waiting times can be discovered and analyzed using the event logs data and how processes can be redesigned to reduce these waiting times. Specifically, the thesis addressed two overarching research questions:

RQ1 How to identify causes of waiting times from event logs?

RQ2 How to identify redesign options to address improvement opportunities to reduce waiting times?

Following the research question RQ1, we reviewed the state-of-the-art in the field of business process analysis. Specifically, we examined existing process mining-based approaches and techniques for identifying and quantifying waiting times from event logs, analyzing these waiting times, and discovering and analyzing their potential causes. As a result of this review, we identified the following research gap in the current state-of-the-art:

RG1 Existing studies have not introduced an approach to attribute waiting times observed in business process event logs to their causes.

To address this research gap RG1, as *Contribution 1*, we proposed a process mining-based approach for the discovery and analysis of waiting times due to batching and identification of related improvement opportunities. We first conceptualized three types of waiting times that can be observed when different batch processing strategies are applied: waiting time due to batch accumulation, waiting time of a ready batch, and waiting time for other instances (of the same batch) to be processed. We then introduced an approach that allows for discovering these types of waiting times associated with batch processing from event logs and identifying improvement opportunities based on the impact of each waiting time type on the activity CTE. This approach takes an activity instance log as input and produces a report comprising discovered batch processing activities, their characteristics (batch processing behavior), associated waiting times by type, and the impact of these waiting times on the activity CTE in terms of maximum CTE improvement if those waiting time types are eliminated. Using the proposed approach, analysts can diagnose batch processing activities in their processes and identify waiting times due to batching at different stages of batch processing, which could facilitate the identification of improvement opportunities.

The proposed approach was implemented as a process-mining technique and provided as a publicly available Python package. The computational evaluation of this process-mining technique indicated that it can accurately discover batch processing activities and their characteristics and quantify waiting times due to batching by type. The demonstration of the proposed approach on a real-life event log illustrated how process analysts can apply it in practical scenarios.

A limitation of the proposed approach is that it considers batching as the sole cause of waiting times, excluding other potential causes. Another limitation is that the approach focuses exclusively on quantifying waiting times in activity transitions involving batch processing activities, leaving out waiting times in activity transitions where batch processing is not observed. Therefore, Contribution 1 partially addressed the research gap RG1.

In our subsequent research, we addressed these limitations by examining other causes of waiting times in addition to batching that can be identified from event logs, considering each activity transition. As a result, in *Contribution 2*, we proposed a process mining-based approach to decompose observed waiting times in each activity transition into multiple direct causes. Given an activity instance log, the approach allows for discovering five direct causes of waiting times in activity transitions: waiting time due to batching, resource contention, prioritization, resource unavailability, and extraneous factors. Then, the approach highlights improvement opportunities by calculating the maximum potential improvement in the process CTE if the causes of waiting times in each activity transition are eliminated. Using the proposed approach, analysts can diagnose the waiting times in their processes for the direct causes and their impact on the temporal process performance, thereby facilitating the identification of improvement opportunities.

The proposed approach was implemented as an open-source software tool, Kronos, which is freely available for process analysts to use. Kronos was evaluated using both synthetic and real-life event logs. The computational evaluation on synthetic event log data showed that Kronos can accurately quantify the causes of waiting times. The demonstration using a real-life event log illustrated how analysts can apply Kronos to analyze the causes of waiting times and identify related improvement opportunities in practical scenarios. The user study conducted with process analysts indicated that Kronos is useful and easy to use for analyzing waiting time causes and identifying improvement opportunities. Based on the findings from the user study, we identified several improvements for Kronos to further enhance its usefulness and ease of use, which were subsequently analyzed and partially implemented. This resulted in an enhanced version of Kronos, which allows analysts to upload their event logs and obtain an analysis of waiting time causes (where waiting times are attributed to their direct causes and their impact on the temporal performance is assessed), thereby supporting the identification of improvement opportunities. In this way, Contribution 2 addressed the RG1.

Building on the research on identifying waiting time causes and related improvement opportunities, the research questions RQ2 focused on determining how these improvement opportunities can be addressed through process redesign. To address research question RQ2, we first reviewed the state-of-the-art in business process redesign. Specifically, we examined existing frameworks that support analysts in process redesign, as well as process mining-based approaches and tools that assist analysts in identifying redesign options. This review revealed the following research gap in the current state-of-the-art:

RG2 Existing studies have not provided automated methods to recommend redesign options that reduce waiting times by targeting their causes.

To address this gap, as *Contribution 3*, we proposed a prompting method for an LLM that tunes it to analyze the causes of waiting times and recommend redesign options to reduce these waiting times by targeting their causes. The proposed method utilizes Contribution 2 (i.e., analysis of waiting time causes produced by Kronos) for input data sources and includes a tailored prompt for analyzing these sources. The prompting method comprises a descriptive component that enables the LLM to analyze the causes of waiting times and a prescriptive component that enhances the LLM's ability to recommend relevant redesign options. The central part of the prescriptive component is a structured dataset of redesign patterns aimed at improving the temporal performance of business processes.

The proposed prompting method was implemented as a conversational interface and integrated with Kronos. Analysts can use this conversational interface to query the analysis results of waiting time causes and obtain tailored recommendations of redesign options to reduce these waiting times.

The prompting method was evaluated using synthetic and real-life event logs. Computational experiments with the prompt-enhanced LLM verified its accuracy in identifying waiting time durations by its causes. The applicability of the prompt-enhanced LLM in practical scenarios was demonstrated on a real-life log. A user study with process analysts showed that the proposed prompting method is effective in assisting analysts with analyzing waiting time causes and identifying relevant redesign options. Additionally, the research compared different versions of the proposed prompting method – a baseline ("zero-shot") method with a minimalistic prompt and an enhanced method incorporating redesign patterns. Process analysts evaluated these versions based on desirable properties of recommendation systems, such as relevance, usefulness, and diversity of the recommended redesigns. As a result, process analysts indicated a preference for the enhanced prompting method regarding recommendation relevance, confidence, and overall satisfaction. The evaluation indicated that the proposed prompting method enables the LLM to automatically generate relevant recommendations of redesign options to reduce waiting times by targeting their causes, thereby assisting analysts in their process redesign initiatives. In this way, Contribution 3 addressed the research gap RG2.

Kronos, as the implementation of the proposed contributions, can serve as a valuable tool for discovering and analyzing the causes of waiting times in business processes and identifying the redesign options to reduce waiting times. Kronos can be applied to diagnose processes across various industries, from manufacturing to services, as demonstrated in the evaluation of the contributions. Kronos primarily targets process analysts focused on analyzing and improving the temporal performance of business processes, particularly by reducing waiting times. For redesign recommendations, an LLM-based assistant with an enhanced prompt provides process analysts with practical, actionable redesign options. By contrast, an LLM-based assistant with a baseline prompt offers high-level recommendations, making it more suitable for managerial decision-making at the departmental, divisional, or company level.

Furthermore, the proposed contributions can offer inputs for other analysis methods. For instance, the results of the analysis of waiting time causes can serve as input for Failure Modes and Effects Analysis (FMEA) [208], helping to identify, prioritize, and mitigate risks associated with delays in business processes. Additionally, the results of the analysis of waiting time causes can be used for predictive process monitoring [209]. By analyzing historical data on waiting time occurrences, their dynamics, and causes, the predictive process monitoring algorithms could identify recurring patterns, allowing for the early forecasting of potential waiting time inefficiencies and their causes before they arise. Similarly, the results of the analysis of waiting time causes, combined with the history of operational interventions, can be used for prescriptive process monitoring [210]. By analyzing these inputs, patterns related to waiting times and their causes can be identified, along with the impact of applied interventions on waiting times. This could allow prescriptive algorithms to generate actionable recommendations for interventions, facilitating the timely proactive mitigation of potential waiting time inefficiencies.

In summary, this thesis proposed three contributions to address the research questions on discovering causes of waiting times from event logs and identifying redesign options to reduce these waiting times. First, to answer RQ1, it proposed a process mining-based approach for the discovery and analysis of waiting times due to batching (*Contribution 1*). This approach conceptualized three types of waiting times and allowed for their identification and analysis from event logs, providing insights on improvement opportunities. The approach was implemented as a Python package. Second, addressing the limitations of Contribution 1, we expanded our research to include other causes of waiting times, resulting in a process mining-based approach to decompose observed waiting times into multiple direct causes (*Contribution 2*). This approach considers five causes of waiting times – batching, resource contention, prioritization, resource unavailability, and extraneous factors – and attributes waiting times in each activity transition to these causes. This approach is implemented in a software tool called Kronos. Finally, to answer RQ2, the thesis proposed a prompting method for an LLM to enable it to analyze waiting time causes and recommend redesign options to reduce waiting times (*Contribution 3*). This method, integrated as a conversational interface into Kronos, utilizes the waiting time analysis results produced by Kronos and employs a structured dataset of redesign patterns to provide targeted redesign recommendations. Kronos can be applied to diagnose processes in various industries, primarily targeting process analysts who focus on analyzing and improving the temporal performance of business processes.

7.2. Limitations and future work

The contributions presented in this thesis have several limitations, which open up future research directions, as described below.

Enhancing the discovery of behavior patterns related to waiting time causes. As a result of the experimentation with Contribution 1 and Contribution 2 (related to the discovery of waiting times causes), we detected potential improvements in the proposed approaches. The first one related to the technique for the discovery of batching processing behavior, in particular, batch activation rules (used in both Contribution 1 and Contribution 2). In its current form, the technique uses all observations available per batch for the discovery of batch activation rules and reports on confidence and support. The discovery of batch activation rules can be improved by establishing training/test partitions to discover and validate the rules.

Another limitation pertains to the discovery of prioritization patterns (related to Contribution 2). Currently, the approach identifies prioritization in the context of the FIFO policy, detecting prioritization when an activity instance is processed out of turn based on this policy. However, existing literature outlines other prioritization strategies (e.g., LIFO) and offers process mining techniques that can discover these strategies from event logs. Therefore, the proposed approach can be further enhanced by incorporating the discovery of multiple prioritization strategies. This enhancement would enable more accurate quantification of waiting time causes in processes that use prioritization policies other than FIFO. In addition, similar to batch activation rules, the current approach uses all available observations per activity with prioritization to discover prioritization rules and reports on confidence and support. Therefore, the discovery of prioritization rules can be enhanced by creating training and test partitions to identify and validate these rules.

Furthermore, a limitation exists concerning the discovery of resource availability calendars (related to Contribution 2). Currently, the technique uses the confidence and support thresholds to ensure the accuracy of the discovered resource availability calendars (default values for confidence (0.1) and support (0.7) as recommended by [13]). However, there might be cases when discovered calendars fall below these thresholds, such as when resources participate in multiple processes or work only briefly (e.g., a few minutes per day), resulting in limited data to accurately assess their true availability in the process. When the non-working intervals overlap with extraneous waiting time, the technique classifies them as waiting time due to resource unavailability. This limitation is inherent to the discovery of the resource calendar – if there is no data showing that a resource was active during a period of time, it cannot be assumed that they were available for work. Therefore, the proposed approach can be further enhanced by displaying to analysts the discovered resource working calendars along with their quality, allowing analysts to manually adjust them if needed. This enhancement would enable a more accurate quantification of waiting times due to resource unavailability and extraneous factors. Additionally, when the confidence and support of the discovered calendars fall below the thresholds, analysts are not warned about it in the current implementation of the approach, i.e., in the Kronos dashboard. Adding a warning message to the Kronos dashboard in future updates could help address this limitation.

Extending the analyzed waiting time intervals. In its current form, the proposed approach for discovering waiting time causes (Contribution 2) considers waiting times in each transition between activity instances. However, waiting times can also occur in at least two other settings: (i) between case creation and the start of the first activity instance; and (ii) within an activity instance due to interruptions (e.g., when a resource pauses work and resumes it later). The first type of waiting time could be analyzed using methods that estimate the inter-arrival time of each case [211]. The second type requires developing new methods for modeling and inferring interruptions. Therefore, as future work, the approach can be further extended to detect waiting times in these two settings and identify the causes of waiting times.

Incorporating multitasking behavior. Another limitation of the proposed approach for discovering waiting time causes (Contribution 2) is that it does not consider multitasking. The approach assumes that resources cannot start processing a new activity instance (of a different activity) if they are already working on one. To address this, the approach could be extended to incorporate multitasking behavior. This could involve discovering the maximum capacity per resource from event logs. Given this capacity, a resource would be considered busy when the number of activities they are performing equals their capacity. Additionally, by analyzing the past multitasking behavior of a resource, it could be possible to estimate when a resource would typically start an activity instance that is waiting. Extending the proposed approach to include these aspects is a direction for future work.

Validating process redesigns via simulation. In this thesis, we measure how the identified causes of waiting time impact the temporal process performance by measuring the maximum CTE improvement when these causes are fully eliminated (Contribution 1 and Contribution 2). The values of potential CTE improvement depict a theoretical improvement achieved by eliminating waiting time causes, without accounting for any potential side effects resulting from a process redesign. This shortcoming could be addressed by employing a simulation-based analysis to explore various redesign options considering associated side effects. This approach would enable analysts to apply selected redesign options to their processes and use simulation to observe how these changes contribute to the reduction of waiting times and the improvement of process CTE.

Following this direction, another option for future work is to develop a method for discovering business process simulation models from event logs that explicitly account for the five causes of waiting times examined in this thesis. Such simulation models could assist analysts in identifying combinations of redesign options to optimize CTE, while also considering other performance dimensions (e.g., cost).

Extending the evaluation of the proposed approaches. We evaluated the proposed contributions through interview studies with process analysts experienced in process mining. In the evaluation of Contributions 1 and 2, participants completed several tasks using the deployed software tool, Kronos, and were interviewed about their experiences with the tool. In contrast, the evaluation of Contribution 3 involved participants assessing the responses generated by the LLM-based assistant beforehand, without the opportunity to test the deployed assistant in action. This approach was taken because our focus for this evaluation was on the content of the responses rather than the tool itself. However, we acknowledge this as a limitation. To address this limitation, the evaluation of Contribution 3 can be expanded by conducting additional interview studies that allow analysts the flexibility to use the assistant for solving process optimization tasks. Based on their experience, analysts could evaluate the assistant's usefulness, usability, and intention for further use by employing frameworks such as the Technology Acceptance Model (TAM) [189] and the System Usability Scale (SUS) [191], following the same methodology used in the evaluations for Contributions 1 and 2.

Furthermore, the interview studies conducted for all proposed contributions did not specifically evaluate the efficiency and effectiveness of the approaches and their implementations in addressing process analysis and redesign tasks, which we also acknowledge as a limitation. Therefore, to address this limitation, further evaluation can be conducted through controlled experiments, where analysts perform specific tasks using Kronos compared to baseline approaches, such as alternative methods to analyze waiting times and identify redesign options they currently employ.

Moreover, a case study could be conducted to analyze the temporal performance of a real-life business process using Kronos to assess its applicability and effectiveness in practical settings. Additionally, by collaborating with process analysts, improvement opportunities and potential redesign options could be identified, selected, and implemented in the real-life process. After a suitable period, Kronos could then be used to re-evaluate the temporal performance of the redesigned process, comparing it to the original performance to assess the actual impact of the process redesigns recommended by Kronos. Such evaluation studies could allow us to assess the proposed contributions from additional performance perspectives and help identify how the contributions can be further improved. We consider these evaluation studies as valuable directions for future work.

Additionally, several future research directions are identified as extensions to existing work and are presented below.

Customizing prompts to align with the analysts' needs. As Contribution 3, the thesis proposed a prompting method for enabling the LLM to analyze waiting time causes and recommend redesign options to reduce waiting times. The comparison of the two versions of the prompting method (baseline and enhanced) with process analysts revealed differences in the recommendations: while the enhanced method provided more specific and actionable recommendations, the baseline method offered more high-level suggestions. This indicated that the application of the proposed method can be tailored depending on the specific goal and audience. When there is a need for more detailed and actionable redesign recommendations (e.g., preferred by process analysts for practical application), employing the enhanced prompting method proves beneficial. On the other hand, if the goal is to obtain high-level, business model-oriented recommendations (e.g., for managerial decision-making), using the baseline prompting method may be more suitable. Therefore, the conversational interface in Kronos can be improved by providing analysts the flexibility to choose the most suitable prompting method based on their specific goals.

During user evaluation, several participants mentioned that there is an opportunity to combine the advantages of the two prompting methods (baseline and enhanced), such as the higher relevance of the enhanced prompting method and the creativity of the baseline one. For instance, by adjusting LLM parameters, such as the temperature setting, we can increase the creativity of the enhanced prompting method. However, this increase in creativity could come at the cost of relevance, leading to more arbitrary and impractical suggestions. Thus, as a direction for future work, customizable LLM settings can be added to Kronos, enabling analysts to manipulate these settings to suit their specific needs. For instance, incorporating a temperature setting would allow users to adjust creativity levels, ensuring that recommendations are both innovative and contextually appropriate.

Prompting LLMs to facilitate process optimization. The prompting method proposed in this thesis focuses on the optimization of waiting times in business processes. Our findings indicate that current LLM technology can be used to support process analysis and redesign. It also revealed that when designing prompting methods for LLMs for process optimization, it is more useful to create structured data inputs (e.g., a structured dataset on redesign patterns) instead of using multiple inputs with unstructured data (e.g., articles about process redesigns). By offering structured data inputs, LLMs are better equipped to integrate external information into processing user requests, resulting in more relevant responses. Moreover, our findings demonstrate that the proposed prompting method empowered the LLM to more effectively address the requirements of analysts during the optimization of waiting times. Thus, a possible direction for future research is the construction of repositories of validated prompts to facilitate process optimization across other KPIs, such as cost reduction.

BIBLIOGRAPHY

[1] M. Dumas, L. M. Rosa, J. Mendling, and A. H. Reijers. *Fundamentals of business process management*. Springer, 2018.

[2] T. R. Rohleder and E. A. Silver. "A tutorial on business process improvement". In: *Journal of Operations Management* 15.2 (1997), pp. 139–154.

[3] T. Melton. "The benefits of lean manufacturing: what lean thinking has to offer the process industries". In: *Chemical engineering research and design* 83.6 (2005), pp. 662–673.

[4] M. Jansen-Vullers and H. Reijers. "Business process redesign in healthcare: towards a structured approach". In: *INFOR: Information Systems and Operational Research* 43.4 (2005), pp. 321–339.

[5] M. O. George. *The lean six sigma guide to doing more with less: cut costs, reduce waste, and lower your overhead*. John Wiley & Sons, 2010.

[6] W. M. Van Der Aalst, H. A. Reijers, A. J. Weijters, B. F. van Dongen, A. A. De Medeiros, M. Song, and H. Verbeek. "Business process mining: An industrial application". In: *Information Systems* 32.5 (2007), pp. 713–732.

[7] S. Ramakrishnan, S. Kumaran, H. Chang, N. Kulkarni, and K. Srihari. "Defining and categorizing handoff points for the service domain". In: *Proceedings of the 29th Annual Conference of ASEM*. 2008, pp. 12–15.

[8] J. L. Rummel, Z. Walter, R. Dewan, and A. Seidmann. "Activity consolidation to improve responsiveness". In: *European Journal of Operational Research* 161.3 (2005), pp. 683–703.

[9] W. van der Aalst. *Process Mining - Data Science in Action, Second Edition*. Springer, 2016. ISBN: 978-3-662-49850-7.

[10] F. Milani, K. Lashkevich, F. M. Maggi, and C. Di Francescomarino. "Process mining: a guide for practitioners". In: *International Conference on Research Challenges in Information Science*. Springer. 2022, pp. 265–282.

[11] M. Vidgof, S. Bachhofner, and J. Mendling. "Large language models for business process management: Opportunities and challenges". In: *Business Process Management Forum*. Springer. 2023, pp. 107–123.

[12] W. M. Van Der Aalst, H. A. Reijers, and M. Song. "Discovering social networks from event logs". In: *Computer Supported Cooperative Work (CSCW)* 14 (2005), pp. 549–593.

[13] O. López-Pintado and M. Dumas. "Business process simulation with differentiated resources: Does it make a difference?" In: *International Conference on Business Process Management*. Springer. 2022, pp. 361–378.

[14] F. Milani and F. M. Maggi. "A comparative evaluation of log-based process performance analysis techniques". In: *Business Information Systems: 21st International Conference, BIS 2018*. Springer. 2018, pp. 371–383.

[15] K. Kubrak, F. Milani, and A. Nolte. “A visual approach to support process analysts in working with process improvement opportunities”. In: *Business Process Management Journal* 29.8 (2023), pp. 101–132.

[16] M. A. Ali, F. Milani, and M. Dumas. “Data-Driven Identification and Analysis of Waiting Times in Business Processes: A Systematic Literature Review”. In: *Business & Information Systems Engineering* (2024), pp. 1–18.

[17] M. S. Uysal, S. J. van Zelst, T. Brockhoff, A. F. Ghahfarokhi, M. Pourbafrani, R. Schumacher, S. Junglas, G. Schuh, and W. van der Aalst. “Process mining for production processes in the automotive industry”. In: *Industry Forum at BPM*. Vol. 20. 2020.

[18] T. G. Erdogan and A. K. Tarhan. “Multi-perspective process mining for emergency process”. In: *Health Informatics Journal* 28.1 (2022), p. 14604582221077195.

[19] S. Singh, R. Verma, and S. Koul. “A collaborative method for simultaneous operations: case of an eye clinic”. In: *Opsearch* 59.2 (2022), pp. 711–731.

[20] F. A. Tridalestari, B. Warsito, A. Wibowo, and H. Prasetyo. “Analysis of E-commerce process in the downstream section of supply chain management based on process and data mining”. In: *Ingénierie des Systèmes d’Information* (2022).

[21] D. R. Ferreira and E. Vasilyev. “Using logical decision trees to discover the cause of process delays from event logs”. In: *Computers in Industry* 70 (2015), pp. 194–207.

[22] Y. Pan and L. Zhang. “Automated process discovery from event logs in BIM construction projects”. In: *Automation in Construction* 127 (2021), p. 103713.

[23] M. De Leoni, W. M. Van der Aalst, and M. Dees. “A general framework for correlating business process characteristics”. In: *International Conference on Business Process Management*. Springer. 2014, pp. 250–266.

[24] N. Martin, L. Pufahl, and F. Mannhardt. “Detection of batch activities from event logs”. In: *Information Systems* 95 (2021), p. 101642.

[25] N. Martin, A. Solti, J. Mendling, B. Depaire, and A. Caris. “Mining batch activation rules from event logs”. In: *IEEE Transactions on Services Computing* 14.6 (2019), pp. 1908–1919.

[26] L. Pufahl and M. Weske. “Batch activity: enhancing business process modeling and enactment with batch processing”. In: *Computing* 101.12 (2019), pp. 1909–1933.

[27] T. Yampaka and P. Chongstitvatana. “An application of process mining for queueing system in health service”. In: *13th International Joint Conference on Computer Science and Software Engineering (JCSSE)*. IEEE. 2016, pp. 1–6.

[28] H. Nguyen, M. Dumas, A. H. ter Hofstede, M. La Rosa, and F. M. Maggi. “Business process performance mining with staged process flows”. In: *Advanced Information Systems Engineering: 28th International Conference, CAiSE 2016*. Springer. 2016, pp. 167–185.

[29] E. Rojas, A. Cifuentes, A. Burattin, J. Munoz-Gama, M. Sepúlveda, and D. Capurro. “Analysis of emergency room episodes duration through process mining”. In: *Business Process Management Workshops*. Springer. 2019, pp. 251–263.

[30] R. Thullner, S. Rozsnyai, J. Schiefer, H. Obweger, and M. Suntinger. “Proactive business process compliance monitoring with event-based systems”. In: *2011 IEEE 15th International Enterprise Distributed Object Computing Conference Workshops*. IEEE. 2011, pp. 429–437.

[31] Y. Jäger and C. Roser. “Effect of Prioritization on the Waiting Time”. In: *Advances in Production Management Systems. Production Management for Data-Driven, Intelligent, Collaborative, and Sustainable Manufacturing*. Springer. 2018, pp. 21–26.

[32] S. Suriadi, M. T. Wynn, J. Xu, W. M. van der Aalst, and A. H. ter Hofstede. “Discovering work prioritisation patterns from event logs”. In: *Decision Support Systems* 100 (2017), pp. 77–92.

[33] N. O. Aissaoui, H. Ben Mbarek, S. B. Layeb, and A. B. Hadj-Alouane. “A BPMN-VSM based process analysis to improve the efficiency of multidisciplinary outpatient clinics”. In: *Production Planning & Control* 35.5 (2024), pp. 461–491.

[34] Z. Toosinezhad, D. Fahland, Ö. Köroğlu, and W. M. Van Der Aalst. “Detecting system-level behavior leading to dynamic bottlenecks”. In: *2020 2nd International Conference on Process Mining (ICPM)*. IEEE. 2020, pp. 17–24.

[35] N. Martin, B. Depaire, A. Caris, and D. Schepers. “Retrieving the resource availability calendars of a process from an event log”. In: *Information Systems* 88 (2020), p. 101463.

[36] R. Banuelas and J. Antony. “Six sigma or design for six sigma?” In: *The TQM Magazine* 16.4 (2004), pp. 250–263.

[37] J. Kilpatrick. “Lean principles”. In: *Utah Manufacturing Extension Partnership* 68.1 (2003), pp. 1–5.

[38] M. Hammer and J. Champy. *Reengineering the Corporation: A Manifesto for Business Revolution*. Zondervan, 2009.

[39] L. T. Lot, A. Sarantopoulos, L. L. Min, S. R. Perales, I. d. F. S. F. Boin, and E. C. de Ataide. “Using Lean tools to reduce patient waiting time”. In: *Leadership in Health Services* 31.3 (2018), pp. 343–351.

[40] U. Naiker, G. FitzGerald, J. M. Dulhunty, and M. Rosemann. “Time to wait: a systematic review of strategies that affect out-patient waiting times”. In: *Australian Health Review* 42.3 (2017), pp. 286–293.

[41] W. M. van der Aalst. “Business process simulation revisited”. In: *Workshop on enterprise and organizational modeling and simulation*. Springer. 2010, pp. 1–14.

[42] K. Salimifard, S. Y. Hosseini, and M. S. Moradi. “Improving Emergency Department Processes Using Coloured Petri Nets”. In: *Joint Proceedings of PNSE+ ModPE*. 2013, pp. 335–349.

[43] M. Cho, M. Song, S. Yoo, and H. A. Reijers. “An evidence-based decision support framework for clinician medical scheduling”. In: *IEEE Access* 7 (2019), pp. 15239–15249.

[44] B. B. Antunes, A. Manresa, L. S. Bastos, J. F. Marchesi, and S. Hamacher. “A solution framework based on process mining, optimization, and discrete-event simulation to improve queue performance in an emergency department”. In: *Business Process Management Workshops*. Springer. 2019, pp. 583–594.

[45] H. A. Reijers and S. L. Mansar. “Best practices in business process redesign: an overview and qualitative evaluation of successful redesign heuristics”. In: *Omega* 33.4 (2005).

[46] T. Fehrer. “Process-pattern. app-A collection of business process redesign patterns.” In: *BPM (Demos/Resources Forum)*. 2023, pp. 117–121.

[47] K. Lashkevich. “Business process improvement opportunities: a framework to support business process redesign”. MA thesis. University of Tartu, 2020.

[48] A. Mustansir, K. Shahzad, and M. K. Malik. “Towards automatic business process redesign: an NLP based approach to extract redesign suggestions”. In: *Automated software engineering* 29.1 (2022), p. 12.

[49] M. Netjes, H. A. Reijers, and W. M. van der Aalst. “The PrICE tool kit: tool support for process improvement”. In: *International Conference on Business Process Management (BPM)*. Springer. 2010, pp. 58–63.

[50] T. Fehrer, D. A. Fischer, S. J. Leemans, M. Röglinger, and M. T. Wynn. “An assisted approach to business process redesign”. In: *Decision Support Systems* 156 (2022).

[51] O. El-Rifai, T. Garaix, V. Augusto, and X. Xie. “A stochastic optimization model for shift scheduling in emergency departments”. In: *Health care management science* 18 (2015), pp. 289–302.

[52] D. Daldoul, I. Nouaouri, H. Bouchriha, and H. Allaoui. “A stochastic model to minimize patient waiting time in an emergency department”. In: *Operations Research for Health Care* 18 (2018), pp. 16–25.

[53] B. Cals, Y. Zhang, R. Dijkman, and C. van Dorst. “Solving the online batching problem using deep reinforcement learning”. In: *Computers & Industrial Engineering* 156 (2021), p. 107221.

[54] A. R. Hevner, S. T. March, J. Park, and S. Ram. “Design science in information systems research”. In: *MIS Quarterly* (2004), pp. 75–105.

[55] K. Lashkevich, F. Milani, D. Chapela-Campa, and M. Dumas. “Data-driven analysis of batch processing inefficiencies in business processes”. In: *International Conference on Research Challenges in Information Science*. Springer. 2022, pp. 231–247.

[56] K. Lashkevich, F. Milani, D. Chapela-Campa, I. Suvorau, and M. Dumas. “Why am I waiting? Data-driven analysis of waiting times in business processes”. In: *International Conference on Advanced Information Systems Engineering*. Springer. 2023, pp. 174–190.

[57] K. Lashkevich, F. Milani, D. Chapela-Campa, I. Suvorau, and M. Dumas. “Kronos: Discovery and analysis of waiting time causes”. In: *5th International Conference on Process Mining (Doctoral Consortium/Demo)*. 2023.

[58] K. Lashkevich, F. Milani, D. Chapela-Campa, I. Suvorau, and M. Dumas. “Unveiling the causes of waiting time in business processes from event logs”. In: *Information Systems* (2024), p. 102434.

[59] K. Lashkevich, F. Milani, M. Avramenko, and M. Dumas. “LLM-assisted optimization of waiting time in business processes: a prompting method”. In: *International Conference on Business Process Management*. To appear, preprint available at: `https://owncloud.ut.ee/owncloud/s/zy66MaTxQpK7wbk`. 2024.

[60] K. Lashkevich, F. Milani, M. Avramenko, and M. Dumas. “Redesigning business processes to reduce waiting times using large language models”. In: *International Conference on Business Process Management (Demos/Resources Forum)*. To appear, preprint available at: `https://owncloud.ut.ee/owncloud/s/gqN2tRXXgC47AoR`. 2024.

[61] T. H. Davenport. *Process innovation: reengineering work through information technology*. Harvard Business School Press, 1993.

[62] R. Macedo de Morais, S. Kazan, S. Inês Dallavalle de Pádua, and A. Lucirton Costa. “An analysis of BPM lifecycles: from a literature review to a framework proposal”. In: *Business Process Management Journal* 20.3 (2014), pp. 412–432.

[63] J. Vom Brocke and M. Rosemann. *Handbook on business process management 2*. Springer, 2010.

[64] A. Van Looy and A. Shafagatova. “Business process performance measurement: a structured literature review of indicators, measures and metrics”. In: *SpringerPlus* 5.1 (2016), p. 1797.

[65] V. Bosilj-Vuksic, L. Milanovic, M. Indihar-Stemberger, et al. “Organizational performance measures for business process management: A performance measurement guideline”. In: *Tenth International Conference on Computer Modeling and Simulation (uksim 2008)*. IEEE. 2008, pp. 94–99.

[66] H.-Y. Wu. “Constructing a strategy map for banking institutions with key performance indicators of the balanced scorecard”. In: *Evaluation and program planning* 35.3 (2012), pp. 303–320.

[67] K. Kutucuoglu, J. Hamali, J. Sharp, and Z. Irani. “Enabling BPR in maintenance through a performance measurement system framework”. In: *International Journal of Flexible Manufacturing Systems* 14 (2002), pp. 33–52.

[68] H. Jagdev, P. Bradley, and O. Molloy. “A QFD based performance measurement tool”. In: *Computers in Industry* 33.2-3 (1997), pp. 357–366.

[69] M. Lehnert, A. Linhart, and M. Röglinger. “Chopping down trees vs. sharpening the axe–Balancing the Development of BPM Capabilities with Process Improvement”. In: *Business Process Management: 12th International Conference, BPM 2014, Haifa, Israel, September 7-11, 2014. Proceedings 12*. Springer. 2014, pp. 151–167.

[70] A. Nowak, F. Leymann, D. Schleicher, D. Schumm, and S. Wagner. “Green business process patterns”. In: *Proceedings of the 18th conference on pattern languages of programs*. 2011, pp. 1–10.

[71] A. Longo and G. Motta. “Design processes for sustainable performances: a model and a method”. In: *International Conference on Business Process Management*. Springer. 2005, pp. 399–407.

[72] R. D. Banker, H. Chang, S. N. Janakiraman, and C. Konstans. “A balanced scorecard analysis of performance metrics”. In: *European journal of operational research* 154.2 (2004), pp. 423–436.

[73] M. Bliemel and K. Hassanein. “E-health: applying business process reengineering principles to healthcare in Canada”. In: *International journal of electronic business* 2.6 (2004), pp. 625–643.

[74] S. Limam Mansar and H. A. Reijers. “Best practices in business process redesign: use and impact”. In: *Business Process Management Journal* 13.2 (2007), pp. 193–213.

[75] I. Kis, S. Bachhofner, C. Di Ciccio, and J. Mendling. “Towards a data-driven framework for measuring process performance”. In: *Enterprise, Business-Process and Information Systems Modeling: 18th International Conference, BPMDS 2017*. Springer. 2017, pp. 3–18.

[76] M. Cho, M. Song, J. Park, S.-R. Yeom, I.-J. Wang, and B.-K. Choi. “Process mining-supported emergency room process performance indicators”. In: *International Journal of Environmental Research and Public Health* 17.17 (2020), p. 6290.

[77] B. Estrada-Torres, A. del-Río-Ortega, and M. Resinas. “Defining process performance measures in an object-centric context”. In: *International Conference on Business Process Management*. Springer. 2022, pp. 210–222.

[78] D. Okes. *Performance metrics: The levers for process management*. Quality Press, 2013.

[79] S. Kavanagh and D. Krings. “The 8 Sources of Waste and How to Eliminate Them”. In: *Government Finance Review* 27.6s 18 (2011).

[80] R. Mans, H. Reijers, M. van Genuchten, and D. Wismeijer. “Mining processes in dentistry”. In: *Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium*. 2012, pp. 379–388.

[81] P. Jaisook and W. Premchaiswadi. “Time performance analysis of medical treatment processes by using disco”. In: *13th International Conference on ICT and Knowledge Engineering*. IEEE. 2015, pp. 110–115.

[82] A. Arumbasari, P. Rhamadani, A. C. Sembiring, and I. Budiman. “Developing Standard Operating Procedure for Production in North Sumatra Construction Companies”. In: *Journal Knowledge Industrial Engineering* 8.2 (2021), pp. 82–89.

[83] B. van Dongen. *BPI Challenge 2020*. 2020. URL: `https://data.4tu.nl/collections/_/5065541/1`.

[84] D. W. Fogarty. “Work in process: performance measures”. In: *International Journal of Production Economics* 26.1-3 (1992), pp. 169–172.

[85] L. Grubb. “Using total cycle time for quick response manufacturing”. In: *Assembly Automation* 18.1 (1998), pp. 20–24.

[86] I. Nday and H. Thomas. “Optimization of the cycle time to increase productivity at Ruashi Mining”. In: *Journal of the Southern African Institute of Mining and Metallurgy* 119.7 (2019), pp. 631–638.

[87] L. Kren and T. Tyson. “Using cycle time to measure performance and control costs in focused factories”. In: *Journal of Cost Management* 16.6 (2002), pp. 18–23.

[88] J. P. Ignizio. “The impact of operation-to-tool dedications on factory stability”. In: *Proceedings of the 2010 Winter Simulation Conference*. IEEE. 2010, pp. 2606–2613.

[89] S. Venkatraman and R. Venkatraman. “Process innovation and improvement using business object-oriented process modelling (BOOPM) framework”. In: *Applied System Innovation* 2.3 (2019), p. 23.

[90] F. H. A. Rahma, E. Chumaidiyah, and W. Tripiawan. “Business Process Design of the Proposed PCR Examination at the PCR Laboratory of Pertamina Balikpapan Hospital Using the Business Process Improvement (BPI) Method”. In: *2021 IEEE 12th International Conference on Mechanical and Intelligent Manufacturing Technologies (ICMIMT)*. IEEE. 2021, pp. 310–314.

[91] N. Novita, A. D. Percunda, and D. Chalidyanto. “Outpatient Service Business Development in an Effort to Reduce Service Time”. In: *Jurnal Kesehatan Masyarakat* 19.1 (2023), pp. 76–86.

[92] B. Waspodo and I. N. Rizki. “Business Sub-Process Analysis and Redesign of Counceling Week (Pekan Konseling) at Career Center Services using Business Process Improvement (BPI)”. In: *2023 11th International Conference on Cyber and IT Service Management (CITSM)*. IEEE. 2023, pp. 1–7.

[93] N. D. S. Darmayanti, A. D. Nastiti, and H. T. Karsanti. “Transactional Business Process Redesign of Procurement Planning System using Heuristic Method”. In: *2024 2nd International Conference on Software Engineering and Information Technology (ICoSEIT)*. IEEE. 2024, pp. 192–197.

[94] J. P. Womack and D. T. Jones. “Lean thinking—banish waste and create wealth in your corporation”. In: *Journal of the Operational Research Society* 48.11 (1997), pp. 1148–1148.

[95] F. Voehl, H. J. Harrington, C. Mignosa, and R. Charron. *The lean six sigma black belt handbook: tools and methods for process acceleration*. CRC Press, 2013.

[96] S. Gross, M. Malinova, and J. Mendling. “Navigating through the maze of business process change methods”. In: *52nd Hawaii International Conference on System Sciences*. 2019.

[97] A. K. Vadera, M. G. Pratt, and P. Mishra. “Constructive deviance in organizations: Integrating and moving forward”. In: *Journal of Management* 39.5 (2013), pp. 1221–1276.

[98] P. Delias. “A positive deviance approach to eliminate wastes in business processes: The case of a public organization”. In: *Industrial Management & Data Systems* 117.7 (2017), pp. 1323–1339.

[99] D. Chapela-Campa and M. Dumas. “From process mining to augmented process execution”. In: *Software and Systems Modeling* 22.6 (2023), pp. 1977–1986.

[100] M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer Berlin, Heidelberg, 2014. DOI: 10.1007/978-3-662-69518-0.

[101] G. Janssenswillen and B. Depaire. “bupaR: Business Process Analysis in R”. In: *BPM (Demos)*. 2017.

[102] W. M. van der Aalst. “Process mining: a 360 degree overview”. In: *Process Mining Handbook*. Springer, 2022, pp. 3–34.

[103] M. L. Van Eck, X. Lu, S. J. Leemans, and W. M. Van Der Aalst. “PM^2: a process mining project methodology”. In: *International Conference on Advanced Information Systems Engineering*. Springer. 2015, pp. 297–313.

[104] K. I. Masani, P. Oza, and S. Agrawal. “Predictive maintenance and monitoring of industrial machine using machine learning”. In: *Scalable Computing: Practice and Experience* 20.4 (2019), pp. 663–668.

[105] R. Wolniak and W. Grebski. “The five stages of business analytics”. In: *Silesian University of Technology Scientific Papers. Organization and Management Series* 178 (2023), pp. 735–752.

[106] L. Pufahl, A. Meyer, and M. Weske. “Batch regions: process instance synchronization based on data”. In: *2014 IEEE 18th International Enterprise Distributed Object Computing Conference*. IEEE Computer Society. 2014, pp. 150–159.

[107] B. F. Hompes, A. Maaradji, M. La Rosa, M. Dumas, J. C. Buijs, and W. M. van der Aalst. “Discovering causal factors explaining business process performance variation”. In: *Advanced Information Systems Engineering: 29th International Conference, CAiSE 2017*. Springer. 2017, pp. 177–192.

[108] V. Denisov, D. Fahland, and W. M. van der Aalst. “Unbiased, fine-grained description of processes performance from event data”. In: *International Conference on Business Process Management*. Springer. 2018, pp. 139–157.

[109] G. Park and M. Song. “Predicting performances in business processes using deep neural networks”. In: *Decision Support Systems* 129 (2020), p. 113191.

[110] C. Fracca, M. de Leoni, F. Asnicar, and A. Turco. “Estimating activity start timestamps in the presence of waiting times via process simulation”. In: *International Conference on Advanced Information Systems Engineering*. Springer. 2022, pp. 287–303.

[111] A. Pika, W. M. Van Der Aalst, C. J. Fidge, A. H. Ter Hofstede, and M. T. Wynn. “Profiling event logs to configure risk indicators for process delays”. In: *Advanced Information Systems Engineering: 25th International Conference, CAiSE 2013*. Springer. 2013, pp. 465–481.

[112] A. Senderovich, A. Shleyfman, M. Weidlich, A. Gal, and A. Mandelbaum. “P-folder: optimal model simplification for improving accuracy in process performance prediction”. In: *International Conference on Business Process Management*. Springer. 2016, pp. 418–436.

[113] M. Leemans, W. M. Van Der Aalst, and M. G. Van Den Brand. “Hierarchical performance analysis for process mining”. In: *Proceedings of the 2018 International Conference on Software and System Process*. 2018, pp. 96–105.

[114] A. Senderovich, M. Weidlich, and A. Gal. “Context-aware temporal network representation of event logs: Model and methods for process performance analysis”. In: *Information Systems* 84 (2019), pp. 240–254.

[115] W. Z. Low, J. De Weerdt, M. T. Wynn, A. H. ter Hofstede, W. M. van der Aalst, and S. vanden Broucke. “Perturbing event logs to identify cost reduction opportunities: A genetic algorithm-based approach”. In: *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2014, pp. 2428–2435.

[116] E. L. Klijn and D. Fahland. “Performance mining for batch processing using the performance spectrum”. In: *Business Process Management Workshops: BPM 2019 International Workshops, Vienna, Austria, September 1–6, 2019, Revised Selected Papers 17*. Springer. 2019, pp. 172–185.

[117] G. A. dos Santos, L. F. P. Southier, and E. E. Scalabrin. “Method to reduce lead-time of business process discovered”. In: *2020 13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*. IEEE. 2020, pp. 840–845.

[118] I. Drosouli, G. Theodoropoulou, G. Miaoulis, and A. Voulodimos. “A process mining approach for resource allocation management in a bike sharing system”. In: *Proceedings of the 24th Pan-Hellenic Conference on Informatics*. 2020, pp. 327–333.

[119] R. Andrews and M. Wynn. “Shelf time analysis in CTP insurance claims processing”. In: *Trends and Applications in Knowledge Discovery and Data Mining: PAKDD 2017 Workshops, MLSDA, BDM, DM-BPM*. Springer. 2017, pp. 151–162.

[120] K. Ganesha, S. Dhanush, and S. R. SM. “An approach to fuzzy process mining to reduce patient waiting time in a hospital”. In: *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*. IEEE. 2017, pp. 1–6.

[121] K. Goel, S. J. Leemans, N. Martin, and M. T. Wynn. “Quality-informed process mining: A case for standardised data quality annotations”. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 16.5 (2022), pp. 1–47.

[122] A. Stefanini, D. Aloini, E. Benevento, R. Dulmin, and V. Mininno. “Performance analysis in emergency departments: a data-driven approach”. In: *Measuring Business Excellence* 22.2 (2018), pp. 130–145.

[123] F. Mannhardt and A. D. Landmark. “Mining railway traffic control logs”. In: *Transportation Research Procedia* 37 (2019), pp. 227–234.

[124] E. Selvarajah and G. Steiner. “Approximation algorithms for the supplier’s supply chain scheduling problem to minimize delivery and inventory holding costs”. In: *Operations Research* 57.2 (2009), pp. 426–438.

[125] G. Cachon and C. Terwiesch. *Matching Supply With Demand: An Introduction To Operations Management*. McGraw Hill, 2012.

[126] S. Henn, S. Koch, and G. Wäscher. “Order batching in order picking warehouses: a survey of solution approaches”. In: *Warehousing in the global supply chain*. Springer, 2012, pp. 105–137.

[127] N. Martin, M. Swennen, B. Depaire, M. Jans, A. Caris, and K. Vanhoof. “Retrieving batch organisation of work insights from event logs”. In: *Decision Support Systems* 100 (2017), pp. 119–128.

[128] J. Nakatumba. “Resource-aware business process management: analysis and support”. Phd Thesis 1 (Research TU/e / Graduation TU/e). Mathematics and Computer Science, 2013.

[129] Y. Wen, Z. Chen, J. Liu, and J. Chen. “Mining batch processing workflow models from event logs”. In: *Concurrency and Computation: Practice and Experience* 25.13 (2013), pp. 1928–1942.

[130] A. Pika, C. Ouyang, and A. H. ter Hofstede. “Configurable batch-processing discovery from event logs”. In: *ACM Transactions on Management Information Systems (TMIS)* 13.3 (2022), pp. 1–25.

[131] P. Waibel, C. Novak, S. Bala, K. Revoredo, and J. Mendling. “Analysis of business process batching using causal event models”. In: *International Conference on Process Mining*. Springer. 2020, pp. 17–29.

[132] E. Petitdemange, E. Lamine, F. Fontanili, and M. Lauras. “Enhancing Emergency Call Centers’ Performance Through a Data-driven Simulation Approach”. In: *ISCRAM*. 2020, pp. 218–227.

[133] L. de Smet. “Queue mining: Combining process mining and queueing analysis to understand bottlenecks, to predict delays, and to suggest process improvements”. MA thesis. Eindhoven University of Technology, 2014.

[134] W. Whitt. “Predicting queueing delays”. In: *Management Science* 45.6 (1999), pp. 870–888.

[135] A. Senderovich, M. Weidlich, A. Gal, and A. Mandelbaum. “Queue mining for delay prediction in multi-class service processes”. In: *Information Systems* 53 (2015), pp. 278–295.

[136] A. I. Kyritsis and M. Deriaz. “A machine learning approach to waiting time prediction in queueing scenarios”. In: *2019 Second International Conference on Artificial Intelligence for Industries (AI4I)*. IEEE. 2019, pp. 17–21.

[137] W. Premchaiswadi and P. Porouhan. “Process modeling and bottleneck mining in online peer-review systems”. In: *SpringerPlus* 4 (2015), pp. 1–18.

[138] Y. Alotaibi and F. Liu. “Average waiting time of customers in a new queue system with different classes”. In: *Business Process Management Journal* 19.1 (2013), pp. 146–168.

[139] J. Patrick, M. L. Puterman, and M. Queyranne. “Dynamic multipriority patient scheduling for a diagnostic resource”. In: *Operations Research* 56.6 (2008), pp. 1507–1525.

[140] S. A. Rahimi, A. Jamshidi, D. Ait-kadi, and A. R. Bartolome. “Applied methods in prioritization of patients in surgery waiting lists”. In: *IIE Annual Conference. Proceedings*. Institute of Industrial and Systems Engineers (IISE). 2014, p. 1857.

[141] F. Silva-Aravena, E. Álvarez-Miranda, C. A. Astudillo, L. González-Martínez, and J. G. Ledezma. "Patients' prioritization on surgical waiting lists: A decision support system". In: *Mathematics* 9.10 (2021), p. 1097.

[142] A. Alman, C. Di Ciccio, D. Haas, F. M. Maggi, and A. Nolte. "Rule mining with RuM". In: *2020 2nd International Conference on Process Mining (ICPM)*. IEEE. 2020, pp. 121–128.

[143] F. Mannhardt, P. Arnesen, and A. D. Landmark. "Estimating the impact of incidents on process delay". In: *2019 International Conference on Process Mining (ICPM)*. IEEE. 2019, pp. 49–56.

[144] M. De Leoni, W. M. van der Aalst, and M. Dees. "A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs". In: *Information Systems* 56 (2016), pp. 235–257.

[145] T. Wireman. *Total productive maintenance*. Industrial Press Inc., 2004.

[146] S.-u. Rahman. "Theory of constraints: a review of the philosophy and its applications". In: *International journal of operations & production management* 18.4 (1998), pp. 336–355.

[147] A. S. Pertiwi and T. N. Rochmah. "Implementation of theory of constraint on waiting time of prescription service". In: *Jurnal Administrasi Kesehatan Indonesia* 7.1 (2019), pp. 1–8.

[148] E. Gijo, J. Antony, J. Hernandez, and J. Scaria. "Reducing patient waiting time in a pathology department using the Six Sigma methodology". In: *Leadership in Health Services* 26.4 (2013), pp. 253–267.

[149] C. Stephenson and W. Bandara. "Enhancing best practices in public health: using process patterns for business process management". In: *Proceedings of the 15th European Conference on Information Systems*. University of St Gallen, Switzerland. 2007, pp. 2123–2134.

[150] H. N. Tran, B. Coulette, and B. T. Dong. "A UML-based process metamodel integrating a rigorous process patterns definition". In: *International Conference on Product Focused Software Process Improvement*. Springer. 2006, pp. 429–434.

[151] W. M. van Der Aalst, A. H. Ter Hofstede, B. Kiepuszewski, and A. P. Barros. "Workflow patterns". In: *Distributed and Parallel Databases* 14 (2003), pp. 5–51.

[152] M. Fellmann, A. Koschmider, R. Laue, A. Schoknecht, and A. Vetter. "Business process model patterns: state-of-the-art, research classification and taxonomy". In: *Business Process Management Journal* 25.5 (2019), pp. 972–994.

[153] A. Koschmider, R. Laue, and M. Fellmann. "Business process model anti-patterns: a bibliography and taxonomy of published work". In: *Proceedings of the 27th European Conference on Information Systems (ECIS)*. 2019.

[154] J. Becker, P. Bergener, D. Breuker, and M. Räckers. “An empirical assessment of the usefulness of weakness patterns in business process redesign”. In: *European Conference on Information Systems*. 2012.

[155] T. Falk, P. Griesberger, and S. Leist. “Patterns as an artifact for business process improvement-insights from a case study”. In: *Design Science at the Intersection of Physical and Virtual Design: 8th International Conference, DESRIST 2013*. Springer. 2013, pp. 88–104.

[156] G. D. Bhatt and M. D. Troutt. “Examining the relationship between business process improvement initiatives, information systems integration and customer focus: an empirical study”. In: *Business Process Management Journal* 11.5 (2005), pp. 532–558.

[157] F. Forster. “The idea behind business process improvement: toward a business process improvement pattern framework”. In: *BP Trends* (2006).

[158] O. Barros. “Business process patterns and frameworks: Reusing knowledge in process innovation”. In: *Business Process Management Journal* 13.1 (2007), pp. 47–69.

[159] D. Kim, M. Kim, and H. Kim. “Dynamic business process management based on process change patterns”. In: *2007 International Conference on Convergence Information Technology (ICCIT 2007)*. IEEE. 2007, pp. 1154–1161.

[160] L. Pufahl, S. Ihde, F. Stiehle, M. Weske, and I. Weber. “Automatic resource allocation in business processes: A systematic literature survey”. In: *arXiv preprint arXiv:2107.07264* (2021).

[161] Y.-W. Si, V.-I. Chan, M. Dumas, and D. Zhang. “A Petri nets based generic genetic algorithm framework for resource optimization in business processes”. In: *Simulation Modelling Practice and Theory* 86 (2018), pp. 72–101.

[162] G. Park and M. Song. “Optimizing resource allocation based on predictive process monitoring”. In: *IEEE Access* 11 (2023), pp. 38309–38323.

[163] L. Pufahl and M. Weske. “Requirements framework for batch processing in business processes”. In: *Enterprise, Business-Process and Information Systems Modeling: 18th International Conference, BPMDS 2017, 22nd International Conference, EMMSAD 2017*. Springer. 2017, pp. 85–100.

[164] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee. “A design science research methodology for information systems research”. In: *Journal of Management Information Systems* 24.3 (2007), pp. 45–77.

[165] A. Armas-Cervantes, M. Dumas, M. L. Rosa, and A. Maaradji. “Local concurrency detection in business process event logs”. In: *ACM Transactions on Internet Technology (TOIT)* 19.1 (2019), pp. 1–23.

[166] A. Weijters and J. T. S. Ribeiro. “Flexible heuristics miner (FHM)”. In: *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, CIDM*. IEEE. 2011, pp. 310–317.

[167] J. Higginson and J. H. Bookbinder. “Policy recommendations for a shipment-consolidation program”. In: *Journal of Business Logistics* 15.1 (1994).

[168] Y. Baba. “A bulk service GI/M/1 queue with service rates depending on service batch size”. In: *Journal of the Operations Research Society of Japan* 39.1 (1996), pp. 25–35.

[169] D. Claeys, J. Walraevens, K. Laevens, and H. Bruneel. “A queueing model for general group screening policies and dynamic item arrivals”. In: *European Journal of Operational Research* 207.2 (2010), pp. 827–835.

[170] L. Pufahl and M. Weske. “Batch activities in process modeling and execution”. In: *Service-Oriented Computing: 11th International Conference, ICSOC 2013, Berlin, Germany, December 2013, Proceedings*. Springer. 2013, pp. 283–297.

[171] L. Pufahl, E. Bazhenova, and M. Weske. “Evaluating the performance of a batch activity in process models”. In: *Business Process Management Workshops: BPM 2014 International Workshops*. Springer. 2015, pp. 277–290.

[172] A. Maity and U. Gupta. “Analysis and optimal control of a queue with infinite buffer under batch-size dependent versatile bulk-service rule”. In: *Opsearch* 52 (2015), pp. 472–489.

[173] J. V. Simons Jr and G. R. Russell. “A case study of batching in a mass service operation”. In: *Journal of Operations Management* 20.5 (2002), pp. 577–592.

[174] L. Pufahl. “Modeling and executing batch activities in business processes”. doctoralthesis. Universität Potsdam, 2018, pp. xix, 163.

[175] D. Sha, S.-Y. Hsu, and X. Lai. “Design of due-date oriented look-ahead batching rule in wafer fabrication”. In: *The International Journal of Advanced Manufacturing Technology* 35 (2007), pp. 596–609.

[176] S. Çetinkaya. “Coordination of inventory and shipment consolidation decisions: A review of premises, models, and justification”. In: *Applications of supply chain management and e-commerce research* (2005), pp. 3–51.

[177] R. Cigolini, M. Perona, A. Portioli, and T. Zambelli. “A new dynamic look-ahead scheduling procedure for batching machines”. In: *Journal of Scheduling* 5.2 (2002), pp. 185–204.

[178] W. W. Cohen. “Fast effective rule induction”. In: *Machine Learning Proceedings 1995*. Elsevier, 1995, pp. 115–123.

[179] L. Măruşter, A. J. Weijters, W. M. Van Der Aalst, and A. Van Den Bosch. “A rule-based approach for process discovery: Dealing with noise and imbalance in process logs”. In: *Data Mining and Knowledge Discovery* 13 (2006), pp. 67–87.

[180] V. Margot and G. Luta. “A new method to compare the interpretability of rule-based algorithms”. In: *AI* 2.4 (2021), pp. 621–635.

[181] D. Levy. *Production Analysis with Process Mining Technology*. 2014. DOI: 10.4121/uuid:68726926-5ac5-4fab-b873-ee76ea412399.

[182] D. Chapela-Campa and M. Dumas. "Enhancing business process simulation models with extraneous activity delays". In: *Information Systems* 122 (2024), p. 102346.

[183] A. Wombacher and M. Iacob. "Start time and duration distribution estimation in semi-structured processes". In: *Proceedings of the 28th annual ACM symposium on applied computing, SAC 2013*. Association for Computing Machinery (ACM), 2013, pp. 1403–1409.

[184] W. M. van der Aalst. "Object-centric process mining: Dealing with divergence and convergence in event data". In: *Software Engineering and Formal Methods: 17th International Conference, SEFM 2019*. Springer. 2019, pp. 3–25.

[185] H. Ding, C. Ren, W. Wang, and J. Dong. "Applying simulation in a supply chain transformation case". In: *Proceedings of the 2006 Winter Simulation Conference*. IEEE. 2006, pp. 614–620.

[186] C. Satitcharoenmuang, P. Porouhan, A. Nammakhunt, N. Saguansakiyotin, and W. Premchaiswadi. "Benchmarking efficiency of children's garment production process using alpha and ILP replayer techniques". In: *2017 15th International Conference on ICT and Knowledge Engineering (ICT&KE)*. IEEE. 2017, pp. 1–7.

[187] J. Patton. "Understanding user centricity". In: *IEEE Software* 24.6 (2007), pp. 9–11.

[188] R. Edwards and J. Holland. *What is qualitative interviewing?* Bloomsbury Academic, 2013.

[189] W. Mertens, A. Pugliese, and J. Recker. *Quantitative Data Analysis – A Companion for Accounting and Information Systems Research*. Springer, 2017.

[190] F. D. Davis. "Perceived usefulness, perceived ease of use, and user acceptance of information technology". In: *MIS Quarterly* (1989), pp. 319–340.

[191] J. Brooke. "SUS: A 'Quick and Dirty' Usability Scale". In: *Usability evaluation in industry* 189.3 (1996), pp. 189–194.

[192] J. Recker. "Reasoning about discontinuance of information system use". In: *Journal of Information Technology Theory and Application (JITTA)* 17.1 (2016), p. 3.

[193] J. R. Lewis and J. Sauro. "Item benchmarks for the system usability scale". In: *Journal of Usability Studies* 13.3 (2018).

[194] A. Berti, D. Schuster, and W. M. van der Aalst. "Abstractions, scenarios, and prompt definitions for process mining with LLMs: a case study". In: *International Conference on Business Process Management*. Springer. 2023, pp. 427–439.

[195] M. Grohs, L. Abb, N. Elsayed, and J.-R. Rehse. “Large language models can accomplish business process management tasks”. In: *Business Process Management Workshops*. Springer. 2023, pp. 453–465.

[196] U. Jessen, M. Sroka, and D. Fahland. “Chit-chat or deep talk: prompt engineering for process mining”. In: *arXiv preprint arXiv:2307.09909* (2023).

[197] P. Bellan, M. Dragoni, and C. Ghidini. “Leveraging pre-trained language models for conversational information seeking from text”. In: *arXiv preprint arXiv:2204.03542* (2022).

[198] J. Zamfirescu-Pereira, R. Y. Wong, B. Hartmann, and Q. Yang. “Why Johnny can’t prompt: how non-AI experts try (and fail) to design LLM prompts”. In: *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 2023, pp. 1–21.

[199] G. Shani and A. Gunawardana. “Evaluating recommendation systems”. In: *Recommender systems handbook* (2011), pp. 257–297.

[200] P. Pu, L. Chen, and R. Hu. “A user-centric evaluation framework for recommender systems”. In: *Proceedings of the fifth ACM conference on Recommender systems*. 2011, pp. 157–164.

[201] M. Fatehkia, J. K. Lucas, and S. Chawla. “T-RAG: lessons from the LLM trenches”. In: *arXiv preprint arXiv:2402.07483* (2024).

[202] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. “Retrieval-augmented generation for knowledge-intensive NLP tasks”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 9459–9474.

[203] J. Jiang, K. Zhou, Z. Dong, K. Ye, W. X. Zhao, and J.-R. Wen. “Structgpt: A general framework for large language model to reason over structured data”. In: *arXiv preprint arXiv:2305.09645* (2023).

[204] A. Joshi, S. Kale, S. Chandel, and D. K. Pal. “Likert scale: Explored and explained”. In: *British Journal of Applied Science & Technology* 7.4 (2015), pp. 396–403.

[205] S. Noor, O. Tajik, and J. Golzar. “Simple random sampling”. In: *International Journal of Education & Language Studies* 1.2 (2022), pp. 78–82.

[206] B. P. Knijnenburg and M. C. Willemsen. “Evaluating recommender systems with user experiments”. In: *Recommender systems handbook*. Springer, 2015, pp. 309–352.

[207] J. Kaddour, J. Harris, M. Mozes, H. Bradley, R. Raileanu, and R. McHardy. “Challenges and applications of large language models”. In: *arXiv preprint arXiv:2307.10169* (2023).

[208] D. H. Stamatis. *Failure mode and effect analysis*. Quality Press, 2003.

[209] C. Di Francescomarino and C. Ghidini. “Predictive process monitoring”. In: *Process Mining Handbook*. Springer International Publishing Cham, 2022, pp. 320–346.

[210] K. Kubrak, F. Milani, A. Nolte, and M. Dumas. “Prescriptive process monitoring: Quo vadis?” In: *PeerJ Computer Science* 8 (2022), e1097.

[211] N. Martin, B. Depaire, and A. Caris. “Using event logs to model interarrival times in business process simulation”. In: *Business Process Management Workshops*. Springer. 2016, pp. 255–267.

[212] K. Lashkevich, L. M. Mediavilla Ponce, M. Camargo, F. Milani, and M. Dumas. “Discovery of improvement opportunities in knock-out checks of business processes”. In: *International Conference on Research Challenges in Information Science*. Springer. 2023, pp. 381–397.

[213] K. Lashkevich, F. Milani, and N. Danylyshyn. “Analysis templates for identifying improvement opportunities with process mining”. In: *European Conference on Information Systems*. 2023.

[214] M. Camargo, M. Dumas, and O. González-Rojas. “Automated discovery of business process simulation models from event logs”. In: *Decision Support Systems* 134 (2020), p. 113284.

[215] A. Delgado, B. Weber, F. Ruiz, I. G.-R. de Guzmán, and M. Piattini. “Continuous improvement of business processes realized by services based on execution measurement”. In: *Evaluation of Novel Approaches to Software Engineering, ENASE 2011*. Springer. 2013, pp. 64–81.

[216] W. van der Waal, I. van de Weerd, I. Beerepoot, X. Lu, T. Kappen, S. Haitjema, and H. A. Reijers. “Putting the SWORD to the Test: Finding Workarounds with Process Mining”. In: *Business & Information Systems Engineering* (2024), pp. 1–20.

[217] J. J. Koorn, X. Lu, H. Leopold, and H. A. Reijers. “From action to response to effect: Mining statistical relations in work processes”. In: *Information Systems* 109 (2022), p. 102035.

[218] S. K. vanden Broucke and J. De Weerdt. “Fodina: A robust and flexible heuristic process discovery technique”. In: *Decision Support Systems* 100 (2017), pp. 109–118.

[219] J. Hurwitz, M. Kaufman, A. Bowles, A. Nugent, J. G. Kobielus, and M. D. Kowolenko. *Cognitive computing and big data analytics*. Vol. 288. Wiley Online Library, 2015.

[220] C. D. Ittner and D. F. Larcker. “Product development cycle time and organizational performance”. In: *Journal of Marketing Research* 34.1 (1997), pp. 13–23.

[221] M. White, J. García, J. A. Hernandez, and J. Meza. “Cycle time improvement by a Six Sigma project for the increase of new business accounts”. In: *International Journal of Industrial Engineering* 16.3 (2009), pp. 191–205.

[222] A. K. Gupta, V. K. Ganesan, and A. I. Sivakumar. “Cycle time variance minimization in dynamic scheduling of single machine systems”. In: *The*

International Journal of Advanced Manufacturing Technology 42 (2009), pp. 544–552.

[223] R. S. Kaplan and D. P. Norton. *The strategy-focused organization: How balanced scorecard companies thrive in the new business environment*. Vol. 2. Harvard Business school press Boston, MA, 2001.

ACKNOWLEDGEMENTS

First, I would like to express my deepest gratitude to my supervisors, Associate Professor Fredrik Milani and Professor Marlon Dumas, for their careful guidance and invaluable support throughout my research journey. From both of you, I have learned lessons that extend far beyond academia, and I truly appreciate that.

I would like to thank David Chapela-Campa for his continuous support in developing the contributions to this thesis. Your expertise and dedication made this work not only possible but also highly fulfilling.

My sincere thanks also go to Anti Alman for assisting with the translation of thesis sections into Estonian and for the insightful discussions we have had. Your willingness to lend a helping hand whenever needed has meant a great deal to me.

I would like to extend my gratitude to the University of Tartu for providing me with a fresh perspective on academia and helping me find my place within it. I am deeply thankful to the University not only for the knowledge I have gained but also for connecting me with some of my closest friends.

And to Estonia, my new home, I am grateful for welcoming me with open arms and offering a place where I can continue to grow and find inspiration.

I would also like to offer my sincere thanks to my family and friends for their love and support. Words cannot fully convey my gratitude for the countless ways you have stood by me and enriched my life.

Finally, and most importantly, I would like to express my heartfelt gratitude to my husband, Kiryl. Your endless love, unwavering belief in me, and constant encouragement have been my greatest source of strength and motivation. This milestone would not have been possible without you by my side, and for that, I am forever grateful.

SISUKOKKUVÕTE

Ooteaegade andmepõhine analüüs ja optimeerimine äriprotsessides

Järjepidev äriprotsesside analüüs ja parendamine on operatiivset tipptaset saavutada soovivate organisatsioonide jaoks võtmetähtsusega tegevused. Äriprotsessid määravad, kuidas tööd tehakse, ja sellest tulenevalt, mõjutavad märkimisväärselt kogu organisatsiooni tulemuslikkust. Äriprotsesside juhtimise kriitiline aspekt on äriprotsessides ebaefektiivsuste tuvastamine, analüüs ja vähendamine. Üheks sellistest ebaefektiivsustest on ooteajad, mis tähistavad perioode, mille jooksul tööd ei tehta, ja mis seega üldjuhul ei lisa protsessile väärtust vaid pärsivad protsessi ajalist jõudlust.

Protsessikaeve võimaldab analüüsida äriprotsesside täitmisel tekkinud andmeid ehk sündmuslogisid. Olemasolevad protsessikaeve meetodid võimaldavad tuvastada ja kvantiteerida ooteaegu, sealjuures tõstes esile pikenenud ooteaegu ja viivitusi. Samas ei suuda olemasolevad meetodid neid ooteaegu konkreetsetele põhjustele omistada ning see piirab analüütikute võimekust teha sihitud parendusi äriprotsessides. Käesolev väitekiri käsitleb seda puudujääki, uurides, kuidas tuvastada sündmuslogidest ooteaegade põhjuseid ning kuidas seda teavet kasutades soovitada selliseid protsessi ümberkorraldusi, mis oleksid tõhusad ooteaegadest tulenevate ebaefektiivsuste vähendamisel.

Doktoritöö keskendub kahele põhiküsimusele: (1) Kuidas tuvastada sündmuslogidest ooteaegade põhjuseid? ja (2) Kuidas tuvastada protsessi parendamise ja ümberkorraldamise võimalusi, mis vähendaksid protsessi ooteaegu? Nendele küsimustele vastamiseks esitab väitekiri kolm peamist panust.

Esiteks uurib väitekiri rühmitamist kui võimalikku ooteaegade põhjust äriprotsessides. Täpsemalt pakub väitekiri protsessikaeve meetodi, mis on mõeldud rühmtöötlusest tulenevate ooteaegadega seonduvate ebaefektiivsuste avastamiseks ja analüüsimiseks. Selleks määratleme kolm ooteaja tüüpi, mis esinevad erinevate rühmtöötluse strateegiate rakendamisel. Seejärel pakume välja protsessikaeve meetodi, mis võimaldab nendele tüüpidele vastavaid parendusvõimalusi sündmuslogidest avastada, lähtudes iga tüübi mõjust protsessi tsükliaja efektiivsusele (Cycle Time Efficiency, CTE). Pakutud meetod on teostatud protsessikaeve tehnikana ja hinnatud sünteetilistele sündmuslogidele tuginevate katsetega. Lisaks on pakutud meetodi rakendatavust demonstreeritud kasutades reaalseid andmeid sisaldavat sündmuslogi.

Teiseks laiendab väitekiri eelnevat analüüsi teistele võimalikele ooteaegade põhjusele, arvestades kõiki äriprotsessis sisalduvaid tegevuste üleminekuid. Täpsemalt pakub väitekiri protsessikaeve meetodi, mis jagab iga tegevuse üleminekuga seotud ooteajad mitmeks otseseks põhjuseks. Tuginedes sündmuslogile võimaldab see meetod avastada 5 otsest ooteaegade põhjust: rühmitamine, ressurssside hõivatus, prioriseerimine, ressursside puudumine ja välised tegurid. Tuvastatud ooteaegade põhjuseid kvantiteeritakse ja analüüsitakse, et määrata kindlaks parendamisvõimalused, hinnates iga põhjuse mõju protsessi tsükliaja efektiivsusele. Pakutud meetodi täpsust on hinnatud sünteetilistele sündmuslogidele tuginevate katsetega ning meetodi rakendatavust on demonstreeritud kasutades reaalseid andmeid sisaldavat sündmuslogi. Lisaks on pakutud meetod teostatud tarkvaratööriistana Kronos. Ooteaegadega seotud parendamisvõimaluste tuvastamisel on Kronose kasulikkust ja kasutatavust hinnatud protsessianalüütikutega läbiviidud kasutajauuringu kaudu.

Kolmandaks käsitleb väitekiri protsessi ooteaegade vähendamisele suunatud ümberkorralduste soovitamise väljakutset. Täpsemalt pakub väitekiri suure keelemudeli (Large Language Model, LLM) jaoks viipamismeetodi, mis kohandab mudeli analüüsima ooteaegade põhjuseid ja soovitama nendest põhjustest lähtuvaid ümberkorraldusi ooteaegade vähendamiseks. Pakutud meetod kasutab sisendandmetena selle väitekirja teise panusega saadavaid tulemusi ning sisaldab viipa, mis on kohandatud nende tulemuste analüüsimiseks ja asjakohaste ooteaegade põhjustele suunatud ümberkorraldamise võimaluste soovitamiseks. Pakutud viipamismeetod on rakendatud vestlusliidesena ja integreeritud Kronosega. Viipamismeetodiga täiustatud keelemudeli täpsust ooteaegade kestuste ja põhjuste tuvastamisel on katsetes kontrollitud nii sünteetilisi kui ka reaalseid andmeid sisaldavate sündmuslogidega. Meetodi tõhusust analüütikute abistamisel ooteaegade põhjuste analüüsis ja ümberkujundamise võimaluste tuvastamisel on hinnatud kasutajauuringuga. Eelkõige võrreldi kasutajauuringus kahte lähenemisviisi: (1) baastaseme lähenemisviis ("zero-shot"), mis hõlmab minimalistlikku viipa; ja (2) täiustatud lähenemisviis, kus viip sisaldab väiksemate ooteaegadega protsessideni jõudmiseks loodud ümberkorraldusmustrite kirjeldusi. Analüütikud võrdlesid soovitatud ümberkorraldusi soovitussüsteemidelt oodatavate omaduste, näiteks asjakohasuse, kasulikkuse ja mitmekesisuse osas.

Kronos, kui esitatud panuste teostus, on rakendatav väärtusliku tööriistana ooteaegade põhjuste avastamiseks ja analüüsimiseks äriprotsessides ning ooteaegade vähendamiseks sobivate ümberkorraldusvõimaluste tuvastamiseks. Kronost saab rakendada protsesside diagnoosimiseks erinevates tööstusharudes, alates tootmisest kuni teeninduseni, nagu demonstreeritud selle väitekirja panuste hindamiseks läbi viidud katsetes. Kronos on peamiselt suunatud protsessianalüütikutele, kes keskenduvad äriprotsesside ajalise jõudluse analüüsimisele ja parendamisele, eriti kui fookus on ooteaegade vähendamisel. Ümberkorraldussoovituste osas pakub täiustatud viibaga LLM-põhine assistent protsessianalüütikutele konkreetseid ümberkujundamisvõimalusi, samas kui baastaseme viip pakub üldisemaid soovitusi, mis sobivad kõrgema tasandi juhtimisotsuste tegemiseks osakondade, divisjonide või ettevõtete tasanditel.

CURRICULUM VITAE

Personal data

Name:	Katsiaryna Lashkevich
Date of Birth:	12.10.1992
Citizenship:	Belarusian
Contact:	katsiaryna.lashkevich@gmail.com

Education

2020–2025	University of Tartu, Faculty of Science and Technology, doctoral studies, specialty: Computer Science
2018–2020	University of Tartu, Faculty of Social Sciences, master's studies, specialty: Innovation and Technology Management
2010–2015	Belarusian State Economic University, Faculty of International Economic Relations, bachelor's studies, specialty: Management of Foreign Economic Activity

Employment

2020–2025	University of Tartu – Tartu, Estonia Junior Research Fellow
2019–2020	University of Tartu – Tartu, Estonia Business Analysis Specialist
2018–2019	CactusSoft – Minsk, Belarus Business Analyst
2017–2018	Texode – Minsk, Belarus Business Analyst
2016	Bosch – Minsk, Belarus Foreign Economic Activity Specialist

Scientific work

Main fields of interest:

- Process Mining
- Business Process Optimization

ELULOOKIRJELDUS

Isikuandmed

Nimi:	Katsiaryna Lashkevich
Sünniaeg:	12.10.1992
Kodakondsus:	Valgevene
Kontakt:	katsiaryna.lashkevich@gmail.com

Haridus

2020–2025	Tartu Ülikool, Loodus- ja täppisteaduste valdkond, doktoriõpe, eriala: Informaatika
2018–2020	Tartu Ülikool, Sotsiaalteaduste valdkond, magistriõpe, eriala: Innovatsiooni ja tehnoloogia juhtimine
2010–2015	Valgevene Riiklik Majandusülikool, Rahvusvaheliste majandussuhete teaduskond, bakalaureuseõpe, eriala: Välismajandustegevuse juhtimine

Teenistuskäik

2020–2025	Tartu Ülikool – Tartu, Eesti Nooremteadur
2019–2020	Tartu Ülikool – Tartu, Eesti Ärianalüüsi spetsialist
2018–2019	CactusSoft – Minsk, Valgevene Ärianalüütik
2017–2018	Texode – Minsk, Valgevene Ärianalüütik
2016	Bosch – Minsk, Valgevene Välismajandustegevuse spetsialist

Teadustegevus

Peamised uurimisvaldkonnad:

- Protsessikaeve
- Äriprotsessi optimeerimine

DISSERTATIONES INFORMATICAE PREVIOUSLY PUBLISHED IN DISSERTATIONES MATHEMATICAE UNIVERSITATIS TARTUENSIS

19. **Helger Lipmaa.** Secure and efficient time-stamping systems. Tartu, 1999, 56 p.
22. **Kaili Müürisep.** Eesti keele arvutigrammatika: süntaks. Tartu, 2000, 107 lk.
23. **Varmo Vene.** Categorical programming with inductive and coinductive types. Tartu, 2000, 116 p.
24. **Olga Sokratova.** Ω-rings, their flat and projective acts with some applications. Tartu, 2000, 120 p.
27. **Tiina Puolakainen.** Eesti keele arvutigrammatika: morfoloogiline ühestamine. Tartu, 2001, 138 lk.
29. **Jan Villemson.** Size-efficient interval time stamps. Tartu, 2002, 82 p.
45. **Kristo Heero.** Path planning and learning strategies for mobile robots in dynamic partially unknown environments. Tartu 2006, 123 p.
49. **Härmel Nestra.** Iteratively defined transfinite trace semantics and program slicing with respect to them. Tartu 2006, 116 p.
53. **Marina Issakova.** Solving of linear equations, linear inequalities and systems of linear equations in interactive learning environment. Tartu 2007, 170 p.
55. **Kaarel Kaljurand.** Attempto controlled English as a Semantic Web language. Tartu 2007, 162 p.
56. **Mart Anton.** Mechanical modeling of IPMC actuators at large deformations. Tartu 2008, 123 p.
59. **Reimo Palm.** Numerical Comparison of Regularization Algorithms for Solving Ill-Posed Problems. Tartu 2010, 105 p.
61. **Jüri Reimand.** Functional analysis of gene lists, networks and regulatory systems. Tartu 2010, 153 p.
62. **Ahti Peder.** Superpositional Graphs and Finding the Description of Structure by Counting Method. Tartu 2010, 87 p.
64. **Vesal Vojdani.** Static Data Race Analysis of Heap-Manipulating C Programs. Tartu 2010, 137 p.
66. **Mark Fišel.** Optimizing Statistical Machine Translation via Input Modification. Tartu 2011, 104 p.
67. **Margus Niitsoo**. Black-box Oracle Separation Techniques with Applications in Time-stamping. Tartu 2011, 174 p.
71. **Siim Karus.** Maintainability of XML Transformations. Tartu 2011, 142 p.
72. **Margus Treumuth.** A Framework for Asynchronous Dialogue Systems: Concepts, Issues and Design Aspects. Tartu 2011, 95 p.
73. **Dmitri Lepp.** Solving simplification problems in the domain of exponents, monomials and polynomials in interactive learning environment T-algebra. Tartu 2011, 202 p.

74. **Meelis Kull.** Statistical enrichment analysis in algorithms for studying gene regulation. Tartu 2011, 151 p.
77. **Bingsheng Zhang.** Efficient cryptographic protocols for secure and private remote databases. Tartu 2011, 206 p.
78. **Reina Uba.** Merging business process models. Tartu 2011, 166 p.
79. **Uuno Puus.** Structural performance as a success factor in software development projects – Estonian experience. Tartu 2012, 106 p.
81. **Georg Singer.** Web search engines and complex information needs. Tartu 2012, 218 p.
83. **Dan Bogdanov.** Sharemind: programmable secure computations with practical applications. Tartu 2013, 191 p.
84. **Jevgeni Kabanov.** Towards a more productive Java EE ecosystem. Tartu 2013, 151 p.
87. **Margus Freudenthal.** Simpl: A toolkit for Domain-Specific Language development in enterprise information systems. Tartu, 2013, 151 p.
90. **Raivo Kolde.** Methods for re-using public gene expression data. Tartu, 2014, 121 p.
91. **Vladimir Šor.** Statistical Approach for Memory Leak Detection in Java Applications. Tartu, 2014, 155 p.
92. **Naved Ahmed.** Deriving Security Requirements from Business Process Models. Tartu, 2014, 171 p.
94. **Liina Kamm.** Privacy-preserving statistical analysis using secure multiparty computation. Tartu, 2015, 201 p.
100. **Abel Armas Cervantes.** Diagnosing Behavioral Differences between Business Process Models. Tartu, 2015, 193 p.
101. **Fredrik Milani.** On Sub-Processes, Process Variation and their Interplay: An Integrated Divide-and-Conquer Method for Modeling Business Processes with Variation. Tartu, 2015, 164 p.
102. **Huber Raul Flores Macario.** Service-Oriented and Evidence-aware Mobile Cloud Computing. Tartu, 2015, 163 p.
103. **Tauno Metsalu.** Statistical analysis of multivariate data in bioinformatics. Tartu, 2016, 197 p.
104. **Riivo Talviste.** Applying Secure Multi-party Computation in Practice. Tartu, 2016, 144 p.
108. **Siim Orasmaa.** Explorations of the Problem of Broad-coverage and General Domain Event Analysis: The Estonian Experience. Tartu, 2016, 186 p.
109. **Prastudy Mungkas Fauzi.** Efficient Non-interactive Zero-knowledge Protocols in the CRS Model. Tartu, 2017, 193 p.
110. **Pelle Jakovits.** Adapting Scientific Computing Algorithms to Distributed Computing Frameworks. Tartu, 2017, 168 p.
111. **Anna Leontjeva.** Using Generative Models to Combine Static and Sequential Features for Classification. Tartu, 2017, 167 p.
112. **Mozhgan Pourmoradnasseri.** Some Problems Related to Extensions of Polytopes. Tartu, 2017, 168 p.

113. **Jaak Randmets.** Programming Languages for Secure Multi-party Computation Application Development. Tartu, 2017, 172 p.
114. **Alisa Pankova.** Efficient Multiparty Computation Secure against Covert and Active Adversaries. Tartu, 2017, 316 p.
116. **Toomas Saarsen.** On the Structure and Use of Process Models and Their Interplay. Tartu, 2017, 123 p.
121. **Kristjan Korjus.** Analyzing EEG Data and Improving Data Partitioning for Machine Learning Algorithms. Tartu, 2017, 106 p.
122. **Eno Tõnisson.** Differences between Expected Answers and the Answers Offered by Computer Algebra Systems to School Mathematics Equations. Tartu, 2017, 195 p.

DISSERTATIONES INFORMATICAE UNIVERSITATIS TARTUENSIS

1. **Abdullah Makkeh**. Applications of Optimization in Some Complex Systems. Tartu 2018, 179 p.
2. **Riivo Kikas**. Analysis of Issue and Dependency Management in Open-Source Software Projects. Tartu 2018, 115 p.
3. **Ehsan Ebrahimi**. Post-Quantum Security in the Presence of Superposition Queries. Tartu 2018, 200 p.
4. **Ilya Verenich**. Explainable Predictive Monitoring of Temporal Measures of Business Processes. Tartu 2019, 151 p.
5. **Yauhen Yakimenka**. Failure Structures of Message-Passing Algorithms in Erasure Decoding and Compressed Sensing. Tartu 2019, 134 p.
6. **Irene Teinemaa**. Predictive and Prescriptive Monitoring of Business Process Outcomes. Tartu 2019, 196 p.
7. **Mohan Liyanage.** A Framework for Mobile Web of Things. Tartu 2019, 131 p.
8. **Toomas Krips.** Improving performance of secure real-number operations. Tartu 2019, 146 p.
9. **Vijayachitra Modhukur.** Profiling of DNA methylation patterns as biomarkers of human disease. Tartu 2019, 134 p.
10. **Elena Sügis.** Integration Methods for Heterogeneous Biological Data. Tartu 2019, 250 p.
11. **Tõnis Tasa.** Bioinformatics Approaches in Personalised Pharmacotherapy. Tartu 2019, 150 p.
12. **Sulev Reisberg.** Developing Computational Solutions for Personalized Medicine. Tartu 2019, 126 p.
13. **Huishi Yin.** Using a Kano-like Model to Facilitate Open Innovation in Requirements Engineering. Tartu 2019, 129 p.
14. **Faiz Ali Shah.** Extracting Information from App Reviews to Facilitate Software Development Activities. Tartu 2020, 149 p.
15. **Adriano Augusto**. Accurate and Efficient Discovery of Process Models from Event Logs. Tartu 2020, 194 p.
16. **Karim Baghery.** Reducing Trust and Improving Security in zk-SNARKs and Commitments. Tartu 2020, 245 p.
17. **Behzad Abdolmaleki.** On Succinct Non-Interactive Zero-Knowledge Protocols Under Weaker Trust Assumptions. Tartu 2020, 209 p.
18. **Janno Siim.** Non-Interactive Shuffle Arguments. Tartu 2020, 154 p.
19. **Ilya Kuzovkin.** Understanding Information Processing in Human Brain by Interpreting Machine Learning Models. Tartu 2020, 149 p.
20. **Orlenys López Pintado.** Collaborative Business Process Execution on the Blockchain: The Caterpillar System. Tartu 2020, 170 p.
21. **Ardi Tampuu.** Neural Networks for Analyzing Biological Data. Tartu 2020, 152 p.

22. **Madis Vasser.** Testing a Computational Theory of Brain Functioning with Virtual Reality. Tartu 2020, 106 p.
23. **Ljubov Jaanuska.** Haar Wavelet Method for Vibration Analysis of Beams and Parameter Quantification. Tartu 2021, 192 p.
24. **Arnis Parsovs.** Estonian Electronic Identity Card and its Security Challenges. Tartu 2021, 214 p.
25. **Kaido Lepik.** Inferring causality between transcriptome and complex traits. Tartu 2021, 224 p.
26. **Tauno Palts.** A Model for Assessing Computational Thinking Skills. Tartu 2021, 134 p.
27. **Liis Kolberg.** Developing and applying bioinformatics tools for gene expression data interpretation. Tartu 2021, 195 p.
28. **Dmytro Fishman.** Developing a data analysis pipeline for automated protein profiling in immunology. Tartu 2021, 155 p.
29. **Ivo Kubjas.** Algebraic Approaches to Problems Arising in Decentralized Systems. Tartu 2021, 120 p.
30. **Hina Anwar.** Towards Greener Software Engineering Using Software Analytics. Tartu 2021, 186 p.
31. **Veronika Plotnikova.** FIN-DM: A Data Mining Process for the Financial Services. Tartu 2021, 197 p.
32. **Manuel Camargo.** Automated Discovery of Business Process Simulation Models From Event Logs: A Hybrid Process Mining and Deep Learning Approach. Tartu 2021, 130 p.
33. **Volodymyr Leno.** Robotic Process Mining: Accelerating the Adoption of Robotic Process Automation. Tartu 2021, 119 p.
34. **Kristjan Krips.** Privacy and Coercion-Resistance in Voting. Tartu 2022, 173 p.
35. **Elizaveta Yankovskaya.** Quality Estimation through Attention. Tartu 2022, 115 p.
36. **Mubashar Iqbal.** Reference Framework for Managing Security Risks Using Blockchain. Tartu 2022, 203 p.
37. **Jakob Mass.** Process Management for Internet of Mobile Things. Tartu 2022, 151 p.
38. **Gamal Elkoumy.** Privacy-Enhancing Technologies for Business Process Mining. Tartu 2022, 135 p.
39. **Lidia Feklistova.** Learners of an Introductory Programming MOOC: Background Variables, Engagement Patterns and Performance. Tartu 2022, 151 p.
40. **Mohamed Ragab.** Bench-Ranking: A Prescriptive Analysis Approach for Large Knowledge Graphs Query Workloads. Tartu 2022, 158 p.
41. **Mohammad Anagreh.** Privacy-Preserving Parallel Computations for Graph Problems. Tartu 2023, 181 p.
42. **Rahul Goel.** Mining Social Well-being Using Mobile Data. Tartu 2023, 104 p.

43. **Anti Ingel.** Algorithms using information theory: classification in brain-computer interfaces and characterising reinforcement-learning agents. Tartu 2023, 142 p.
44. **Shakshi Sharma.** Fighting Misinformation in the Digital Age: A Comprehensive Strategy for Characterizing, Identifying, and Mitigating Misinformation on Online Social Media Platforms. Tartu 2023, 158 p.
45. **Kristiina Rahkema.** Quality Analysis of iOS Applications with Focus on Maintainability and Security Aspects. Tartu 2023, 182 p.
46. **Ivan Slobozhan.** Studying Online Social Media Engagement in CIS Countries during Protests, Mass Demonstrations and War. Tartu 2023, 81 p.
47. **Nurlan Kerimov.** Building a catalogue of molecular quantitative trait loci to interpret complex trait associations. Tartu 2023, 248 p.
48. **Pavlo Tertychnyi.** Machine Learning Methods for Anti-Money Laundering Monitoring. Tartu 2023, 117 p.
49. **Abasi-amefon Obot Affia.** A Framework and Teaching Approach for IoT Security Risk Management. Tartu 2023, 180 p.
50. **Raimond-Hendrik Tunnel.** Video Game Design and Development Bachelor's Curriculum for Estonia. Tartu 2024, 137 p.
51. **Ahto Salumets.** Bioinformatics analysis of various aspects in immunology. Tartu 2024, 198 p.
52. **Mohammed Abdulhameed Shaif Ali.** Deep Learning Methods for Cell Microscopy Image Analysis. Tartu 2024, 143 p.
53. **Pille Pullonen-Raudvere**. Foundations of Efficient and Secure Algorithm Development for Secure Multiparty Computation. Tartu 2024, 265 p.
54. **Marili Rõõm.** Multiple approaches to learners' success and factors affecting it in computer programming MOOCs. Tartu 2024, 170 p.
55. **Shivananda Rangappa Poojara.** Design and Orchestration of Scalable, Event-Driven Serverless Data Pipelines for Internet of Things (IoT) Applications. Tartu 2024, 172 p.
56. **Hassan Abdulgaleel Hassan Salim Eldeeb.** Empowering Machine Learning Pipelines with Automated Feature Engineering. Tartu 2024, 121 p.
57. **Muhammad Uzair.** Soft decision making for agri-food 4.0. Tartu 2024, 158 p.
58. **Kirill Milintsevich.** Estimation of Depression Level from Text: Symptom-Based Approach, External Knowledge, Dataset Validity. Tartu 2024, 130 p.
59. **Maksym Del.** Multilingual and Multi-Domain Representational Patterns Across Trpansformer-Based Models. Tartu 2024, 131 p.
60. **Kristo Raun.** Adaptive Out-of-order Handling in Streaming Conformance Checking. Tartu 2024, 118 p.
61. **Toivo Vajakas.** Towards integration of mobile network data into analyzing human mobility. Tartu 2024, 103 p.