2026 | Faculty of Business Economics

UHASSELT
KNOWLEDGE IN ACTION

Doctoral dissertation submitted to obtain the degree of
Doctor of Philosophy, to be defended by

Marzie Hosseinpour

DOCTORAL DISSERTATION

Process Deviation Analysis for Continuous Auditing

Promoter: Prof. Dr. Mieke Jans | UHasselt

Co-promoter: Prof. Dr. Benoît Depaire | UHasselt

www.uhasselt.be
Hasselt University
Martelarenlaan 42 | BE-3500 Hasselt

UHASSELT
KNOWLEDGE IN ACTION

2026| Faculty of Business Economics

Doctoral dissertation submitted to obtain the degree of
Doctor of Philosophy, to be defended by

Marzie Hosseinpour

DOCTORAL DISSERTATION

Process Deviation Analysis for Continuous Auditing

Promoters: Prof. Dr. Mieke Jans | UHasselt

Co-promoter: Prof. Dr. Benoît Depaire | UHasselt

Hasselt University

Process Deviation Analysis for Continuous Auditing

Marzie Hosseinpour

Promoter: Prof. dr. Mieke Jans
Co-promoter: Prof. dr. Benoît Depaire

June 2026

To Maman,
still my measure of everything that matters.

Acknowledgements

Looking back, this dissertation feels less like a single project and more like a long journey that started years ago — in 2015 in Hasselt — and quietly continued across cities, countries, and phases of life. It is something I have carried with me for years, across different homes, different laptops, and different versions of myself. There were moments when it felt close, and others when it felt very far away. But through all of that, it remained there — unfinished, but never abandoned.

Along this journey, I have been fortunate to receive the support and guidance of many people.

I would like to first and foremost thank my promoter, Prof. dr. Mieke Jans. Over the years — and truly, over a decade — you have read and re-read versions of my work with remarkable patience and dedication. Your persistence and attention to detail have been invaluable, and through the years, when I felt like letting go of this thesis, you encouraged me to continue. I am deeply grateful for your support and for never giving up on this process.

My sincere thanks also go to my co-promoter, Prof. dr. Benoît Depaire. Your comments and advice have always been precise, insightful, and incredibly helpful. What I appreciate most is your structured and insightful approach to research, along with the thoughtfulness you bring to your interactions. I still remember moments where your words went far beyond academic feedback — thoughtful and unexpectedly supportive. This has stayed with me and made a lasting impression.

I would like to thank the members of my doctoral committee — dr. Nick van Beest, Prof. dr. Maarten Corten, and Prof. dr. Koen Vanhoof — for their time, feedback, and support throughout this journey.

A special thank you goes to dr. Nick van Beest for his support and contribution to the evaluation of the deviation classification framework in Chapter 6. Your help was a key foundation for the development of the tool.

I would like to thank Prof. dr. Maarten Corten for your valuable support. Your expertise in auditing was a strong anchor for me, especially in a topic I was not very

familiar with. I truly appreciated your insightful comments, which helped me better understand and strengthen this part of the work.

I would also like to thank Prof. dr. Koen Vanhoof for your support throughout the years. Beyond your helpful comments that contributed to improving this dissertation, I am grateful for the environment you helped create and sustain, which made this journey possible.

I am also grateful to my jury members, Prof. dr. Banu Aysolmaz and Prof. dr. Roger Meuwissen, for their time and consideration in evaluating this dissertation. I found your comments very insightful, and they contributed meaningfully to improving the quality of this work.

To my colleagues and friends — thank you for making this journey lighter. Frank, Mathijs, and Gert, my office mates, thank you for the many good moments, for the tea, and for the conversations that made even the difficult days more manageable. Marijke, thank you for always being supportive. Maggy, Mounes, Vincent, Corinne, and Serge, thank you for the fun, the laughter, the support, and the many good memories.

To my dearest brother Mehdi — thank you for your love, your constant support, and for always being there for me.

To my dear Stefan — although you are partly the reason this PhD thesis was not finished in 2019, thank you for your love and for standing by me through the later phases of this journey.

To Maman-e azizam — thank you for choosing to be my mother, and for being far more than one. You have always been my closest friend, my source of wisdom, my doctor, my therapist, and my refuge through every difficult season of life. Thank you for always caring more about my happiness than any achievement.

And to Baba — you are still with us, in the morning birdsong and in the spring breeze carrying cherry blossoms. I deeply wish I could have seen your smile as I handed you the final version of this dissertation.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

To illustrate the research context, this thesis begins with a fictitious but realistic scenario of a large financial organization operating in a highly regulated environment. Imagine a company whose daily operations span procurement, logistics, finance, and accounting, supported by integrated Enterprise Resource Planning (ERP) and Customer Relationship Management (CRM) systems. These systems continuously record transactions and activities, generating vast volumes of event data that capture every stage of business execution. In modern enterprises, such data has emerged as a valuable organizational resource that offers insights not only into individual transactions but also into the underlying business processes that determine performance and compliance Badakhshan et al. [2022], Oldenburg et al. [2025].

For this company, effective internal control is not only a compliance requirement but also a key mechanism for ensuring transparency, reliability, and operational efficiency. Over time, the company has invested substantial resources in its internal control and internal auditing functions to provide reasonable assurance to management, stakeholders, and regulatory authorities that its data and processes adhere to established policies. However, in line with broader industry developments, the organization has progressively shifted toward continuous auditing. This shift is motivated by several factors: First, traditional auditing relied heavily on sampling due to time and budget constraints, making it inherently incomplete and prone to sampling errors. In contrast, continuous auditing enables the examination of the entire population of process executions and controls based on systematically defined criteria [Groomer and Murthy, 1989, Vasarhelyi and Halper, 1991, Koch, 1981, Alles et al., 2006c]. Rather than offering only "reasonable" assurance through retrospective checks, continuous auditing can deliver near real-time assurance, ensuring that risks and irregularities are detected as processes unfold [Kuenkaikaew and Vasarhelyi, 2013]. While traditional

auditing is performed periodically, typically on an annual basis, continuous auditing can be conducted more frequently and systematically. Analyzing complete data sets at shorter intervals enables organizations to detect errors, inefficiencies, and potential fraud in a more timely manner. In addition to improving reliability, this approach can also reduce audit costs and enhance operational efficiency [Elliott, 1998, Menon and Williams, 2001]. Second, the complexity of organizational structures and business processes has increased significantly in recent decades [Vasarhelyi, 2010, Marcus et al., 2024]. This growing complexity introduces new challenges for traditional approaches to processing, analyzing, and verifying the correctness of business processes. Maintaining oversight across multiple systems and workflows now requires more systematic and automated approaches to ensure that activities are executed as intended and in accordance with both internal policies and external regulations. Third, the company's information systems infrastructure already provides extensive access to transactional and operational data, offering the necessary foundation for automation, enhanced coverage, and greater responsiveness. Leveraging this data environment, the company aims to strengthen its assurance mechanisms by integrating technology-driven and data-centric auditing practices.

Building upon this technological foundation, the company has implemented a continuous auditing framework that automates large parts of its internal control process. These automated systems execute a range of transactional checks designed to ensure compliance with business rules and regulatory standards. For instance, before a sales order is processed, the system verifies that the customer is registered and credit-approved; during billing, it ensures that the correct rate and percentage of value-added tax (VAT) is applied; and before a payment is executed, the system checks that the invoice corresponds to a valid purchase order and delivery confirmation.

While these automated controls are effective in detecting the inconsistencies in data perspective and ensuring compliance at the transactional level, they do not capture how activities are executed in sequence or how processes deviate from the intended sequence. In other words, they provide validation of data accuracy, but not assurance of process correctness. This limitation leaves a critical gap in the company's control environment that even if each transaction independently satisfies the compliance rules, the overall process may still diverge from the designed model.

To enhance its assurance coverage, the company has begun incorporating data analytics techniques into its internal auditing activities. Data analytics in auditing refers to the use of computational and statistical methods to analyze large data sets for patterns, anomalies, and risks Ditkaew and Suttipun [2023], Kim and Vasarhelyi [2012], Alles et al. [2006b], Issa [2013], Perols and Murthy [2012b], Jans et al. [2014]. Recent research also emphasizes full-population testing and machine-learning ana-

lytics that move beyond traditional sampling [Huang et al., 2022]. This also aligns with professional guidance, for example AICPA (2017), what defines data analytics as "the science and art of discovering and analyzing patterns, identifying anomalies, and extracting other useful information in data underlying or related to the subject matter of an audit, through analysis, modeling, and visualization."

While most of the data analytics techniques offer a powerful analytical foundation for continuous and internal auditing, their practical application reveals important limitations. In the company's current setup, automated controls are highly effective in identifying inconsistencies in transactional data and ensuring compliance with financial and regulatory requirements. These mechanisms form the backbone of the continuous auditing environment by providing rule-based assurance over transactions on a frequent basis. However, such controls operate at the data validation level and lack a process-oriented perspective meaning that they do not verify whether activities are executed in the correct sequence or adhere to the prescribed procedural model. This limitation introduces a significant gap in the organization's overall control framework. Even when each transaction satisfies the defined compliance rules, the end-to-end process may still deviate from the intended flow Duan et al. [2025], Föhr et al. [2025]. For example, a purchase order might be approved only after goods have been received, or a refund might be issued before the corresponding complaint is properly assessed. These deviations do not necessarily breach individual transaction-level rules but can introduce operational risks, inefficiencies, or potential compliance issues. To address this shortcoming, the company has started to integrate process mining techniques, particularly conformance checking, into its internal control system. This integration marks a shift in focus from verifying "Are individual transactions correct?" to asking "Are activities executed in the correct order and according to the defined process model?". Through this evolution, the company aims to transition from static, rule-based compliance toward a dynamic form of process assurance that reflects how processes are actually executed in practice rather than how they are ideally designed. In Chapter 3, more detailed background on process mining techniques is presented. The organization relies on a normative process model, provided by process owners who understand how key workflows, such as order-to-cash or purchase-to-pay, are ideally performed. This model serves as a reference benchmark against which real executions can be compared using event data. By aligning actual traces with the normative model, auditors can detect deviations and investigate their root causes. However, this approach is not without challenges: the normative model, while useful as a reference, represents an idealized version of operations. It omits many variations and exceptions that naturally occur in day-to-day operations. For example, urgent procurement orders may skip certain approval steps, or long-standing customers might

be granted accelerated payment processing. These variations are often permitted in practice but not explicitly captured in the model. The result is that deviations from the normative model are frequent, and not all of them are problematic. Some represent acceptable "exceptions", which are deviations justified by contextual knowledge, or operational constraints, while others are true "anomalies", indicating process failure, errors, process breakdowns, or non-compliance. The distinction between these two categories is inherently complex and often difficult to establish in practice. Part of the difficulty arises from the tacit and implicit knowledge that exists within the organization. Internal auditors, process owners, and experienced users understand many of the informal rules that govern daily operations, such as which steps may be skipped under specific conditions, or how the number of signatures may vary depending on the value of a transaction. Yet this knowledge is rarely documented. This knowledge is embedded in personal experience and organizational culture, transmitted verbally or through informal practices rather than documented in formal procedures. Sometimes when issues occur, they are usually addressed reactively through discussion or immediate action, without being first formally standardized as a written rule. When conformance checking is applied in this environment, it inevitably detects a high number of deviations. However, without access to the tacit rules that explain acceptable variations, the automated analysis is unable to distinguish between deviations of real significance and those that are justifiable. This creates a gap between the system's deviation detection capability and the auditor's judgment.

Consequently, the company faces what can be described as a classification challange. Each detected deviation must be interpreted and classified as either a legitimate exception, reflecting acceptable flexibility in process execution, or a problematic anomaly that requires attention or corrective action. The challenge lies not in detecting deviations per se, but in understanding their meaning within the operational context. This classification task is central to maintaining an effective balance between control and flexibility. Over-classifying deviations as anomalies can lead to excessive alarm flood and unnecessary investigations, and increasing audit workload. Under-classifying them, on the other hand, risks overlooking genuine control failures or compliance breaches. Achieving the right balance requires systematic methods that combine automated detection with human expertise and expert judgment. The challenge of bridging automated process conformance analysis with human interpretive insight, defines the core motivation of this research. It highlights a broader issue faced by many modern organizations: while transactional control systems ensure data-level accuracy, they leave process-level assurance under-monitored. As organizations increasingly adopt analytical and continuous auditing, the ability to interpret process deviations meaningfully becomes essential. This need forms the foundation for the

research motivation and problem statement discussed in the next section.

1.1 Research Motivation

While the company's adoption of conformance checking represents a step forward toward process-oriented assurance, several limitations remain that motivate this research. Process mining techniques, including conformance checking, have shown strong potential in revealing mismatches and discrepancies between designed and actual process executions. However, their practical application in internal auditing environments is still limited. In particular, auditors encounter significant difficulties related to the "interpretability of conformance checking techniques results" and "high volume of false-positive detections". These shortcomings restrict the extent to which process mining can be used as a reliable and efficient auditing instrument.

1.1.1 Non-meaningful nature of current outputs

One of the key challenges lies in the non-meaningful nature of current outputs. Auditing is a professional domain that requires expert judgment and nuanced interpretation of context. In practice, the outputs generated by conformance checking techniques are frequently expressed in highly technical terms, which can make their interpretation challenging for auditors and other non-technical users. In other words, the deviations are presented in a highly technical format, in a way that auditors who are supposed to analyze these deviations cannot get a comprehensive and meaningful overview of what the deviations actually imply for the process under review. This gap between technical output and professional judgment hinders auditors from fully leveraging the analytical potential of process mining tools.

1.1.2 Large number of deviations

The second challenge arises from the large number of deviations that conformance checking techniques typically identify. Because these algorithms compare observed executions with a simplified normative model, they tend to classify a substantial proportion of cases as deviations. Many of these, however, are not actual anomalies but acceptable exceptions. This results in an excessive number of false-positive detections, requiring auditors to manually reclassify the outcomes. Such manual effort is time-consuming, increases the risk of inconsistency and human error, and reduces the overall efficiency of the audit process. Meaningfully reducing these false positives requires the incorporation of additional contextual information such as data attributes, case conditions, or operational rules. In many real-world audit scenarios, understanding a deviation requires more than just knowing which activities occurred

in an unexpected order. Auditors often need contextual information, such as which employee performed an action, under what conditions it occurred, or which data attributes were affected. In settings such as the fictitious organization described above, the challenge lies not in the absence of integrated multi-perspective techniques, but in the fact that relevant contextual information is not formalized or integrated into these techniques. However, many of these attributes are implicit in practice and therefore difficult to formalize or integrate into traditional conformance checking techniques. Without such multi-perspective information, auditors cannot effectively distinguish legitimate exceptions from problematic anomalies.

These challenges, namely lack of interpretability, a high number of detected deviations illustrate why process mining, despite its analytical potential, has not yet become a standard tool in internal auditing. For the company described above in this thesis, these challenges translate into a practical concern: While conformance checking adds a valuable process focus to existing transactional controls, its current application in the context of the examined environment generates substantial noise and limited actionable insight. These challenges stem from the underlying research assumptions, such as the use of incomplete process models and the lack of formalized contextual information, rather than from inherent limitations of the conformance checking techniques.

1.2 Problem Statement and Research Questions

The challenges described above converge into a central problem: how to design an algorithmic approach that can help auditors process and interpret the large number of deviations produced by conformance checking in a more efficient, meaningful, and context-aware manner. Current conformance checking techniques detect deviations reliably, but they do not provide the interpretive support needed to distinguish acceptable exceptions from problematic anomalies. As a result, auditors require analytical mechanisms that go beyond detection and offer actionable, context-sensitive insights that reduce noise and improve decision-making. Addressing this central problem requires both a methodological solution, i.e., an algorithmic framework, and a deeper understanding of the underlying issues that hinder effective deviation analysis. This logic is reflected in the structure of the research questions.

Algorithmic Design Question

RQ1: *How can a framework integrating process mining, data mining, and human-in-*

the-loop enhancement be designed to assist auditors in a continuous auditing practice?

This question drives the design of an algorithmic solution that can reduce the manual effort required from auditors, manage large volumes of deviations, and progressively learn from auditor feedback.

Problem-Driven Questions Supporting the Algorithmic Goal

RQ2: *How can the outputs of conformance checking techniques be made more interpretable and meaningful for auditors?*

This question addresses the interpretability problem by seeking ways to organize, structure, and represent deviations in a form aligned with auditors' reasoning. Existing conformance checking techniques typically present deviations in technical terms, such as skip, insert, which do not necessarily correspond to how auditors interpret process deviations. Therefore, this question also examines whether auditors rely on different, practice-oriented terminology and categorizations in their mental models, and how these can be captured to improve interpretability.

RQ3: *To what extent can incremental information provision assist auditors in classifying deviating instances more efficiently than traditional manual approaches?*

This question focuses on reducing the cognitive and analytical burden on auditors by supplying contextual information only when it is needed. In practice, manually classifying each detected deviation is cumbersome and highly time-consuming, especially when conformance checking produces substantial numbers of detected deviations. As a result, auditors often rely on sampling instead of reviewing the full population of deviations, which reduces assurance coverage. This question therefore examines whether a stepwise, targeted provision of information can make full-population analysis more feasible in practice.
In addition, incremental information provision naturally extends to incorporating multiple process perspectives such as data attributes, case conditions, and resource information, beyond the control-flow view. These additional perspectives are essential for distinguishing acceptable exceptions from true anomalies, and integrating them in an incremental manner ensures that auditors receive relevant contextual clues without being overwhelmed by unnecessary detail.
To address this problem in a structured and rigorous manner, the thesis adopts the algorithm engineering framework proposed by Mendling et al. [2025]. This framework

guides the development, design, and evaluation of algorithmic solutions to real-world challenges. The following section explains how it is applied within this research.

1.3 Methodology and knowledge contributions

This section positions the methodology of the thesis within the framework of algorithm engineering as proposed by Mendling et al. [2025]. Their framework discusses the ontology, epistemology and methodology of algorithm engineering and provides a structure to produce valid knowledge contributions regarding algorithms. Consistent with Herbert Simon's [Simon, 1996] view in The Sciences of the Artificial, this thesis adopts a design-oriented perspective in which scientific inquiry is not limited to describing how things are but also concerns the creation of artefacts intended to improve how things might be. Simon emphasizes that many fields—including engineering, business, and organizational decision-making—advance knowledge by developing artefacts that support and enhance human reasoning. This perspective is particularly relevant for research situated at the intersection of information systems, auditing, and analytics, where methods must be designed with practical use in mind and evaluated in relation to their intended purpose.

The algorithm engineering framework provides a structured view of how real-world problems are transformed into algorithmic solutions. As illustrated in Figure 1.1, the framework distinguishes between the real-world problem, the algorithmic task, the algorithm design, and the implementation.

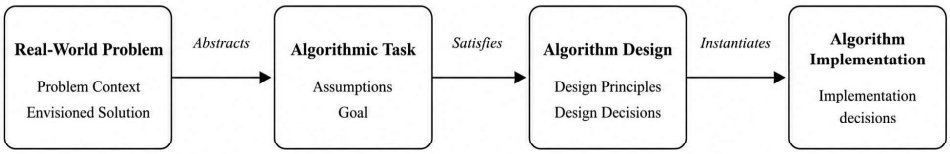


Figure 1.1: Algorithm engineering framework adapted from Mendling et al. [2025]

The following subsections describe the algorithm engineering framework in detail and apply it to the problem of process deviation analysis for continuous auditing.

1.3.1 The Algorithm Engineering Framework

To situate this thesis within a structured methodological perspective, this thesis adopts the framework of algorithm engineering as described by Mendling et al. [2025]. This framework provides a systematic way to connect real-world problems, algorithmic abstractions, conceptual designs, implementations, and knowledge contributions. Before applying it to the context of process deviation analysis, it is useful to briefly outline its main elements.

1.3.1.1 Ontological Concepts

The framework distinguishes between four ontological layers: Real-world problem: The real-world problem refers to the practical challenge that motivates algorithmic inquiry and originates from an application domain. It captures the organizational or system-level issue in its natural context, often involving inefficiencies, risks, or compliance concerns. At this level, the problem is domain-specific and not yet formally defined.

Algorithmic task: The algorithmic task is the formal abstraction of the real-world problem. It specifies inputs, outputs, and assumptions, and therefore it transforms a messy, domain-specific challenge into a computationally tractable problem. This abstraction defines the scope of what the algorithm should address.

Algorithm design: The algorithmic design represents the conceptual strategy developed to solve the task. It defines the underlying principles, models, and structural components of the approach, often expressed in pseudocode, diagrams, or blueprints. At this level, the focus is on the logic of the solution rather than its technical instantiation.

Algorithm implementation: The algorithmic implementation is the operational realization of the design in code or system form. It involves technical decisions such as the programming environment, data formats, and performance optimizations. They serve two complementary purposes: from a research perspective, they enable the empirical generation of knowledge about the behaviour and properties of the design; from a practical perspective, they provide the means to apply the design to real-world problems in the environment that motivated its development. Moreover, multiple implementations may exist for the same design.

This separation of entities allows for a clear distinction between what is being studied (ontology), how it can be known (epistemology), and how it is approached (methodology). Within this structure, algorithm design is the main focus of this thesis contribution, as it defines the conceptual principles and strategies for solving the task. Implementations are nevertheless essential for empirical research, as they bring the

design into an executable form and allow its behavior to be observed and analyzed. Thus, implementations serve an instrumental role, while the resulting knowledge contributions concern the properties and refinement of the underlying design.

1.3.1.2 Epistemology and Types of Knowledge Contributions

In the framework of Mendling et al. [2025], epistemology concerns the types of knowledge that can be generated about algorithmic tasks and designs. The framework differentiates between various kinds of knowledge, distinguishing between tasks and designs:

Knowledge of tasks: This refers to the formal definition and abstraction of a real-world problem into an algorithmic task. It specifies the scope of the task, its assumptions, and the mapping from inputs to outputs, thus making the problem tractable.

Knowledge of designs: This refers to the conceptual description of how an algorithm is structured to solve a task. It includes pseudocode, architectural blueprints, and design principles, providing a clear representation of the strategy before implementation.

Knowledge about tasks: This refers to insights into the properties and nature of tasks themselves. It includes identifying categories, understanding problem boundaries, and analyzing how variations of the task appear across contexts, thereby clarifying what the task involves in practice.

Knowledge about designs: This refers to empirical and theoretical insights into how a design behaves under different conditions. It encompasses performance knowledge (e.g., accuracy, efficiency), sensitivity knowledge (e.g., robustness to parameter changes), and explanatory knowledge (e.g., why certain outcomes occur).

Uncertainty knowledge: Uncertainty knowledge concerns the limitations of what can be known about a task or design. This includes understanding where incomplete information, variability in real-world environments, or inherent ambiguities restrict the certainty of conclusions. Uncertainty knowledge is particularly relevant in domains such as auditing, where incomplete models, tacit organizational rules, and context-dependent interpretations constrain the predictability and completeness of algorithmic behavior.

1.3.2 Algorithm Engineering Framework Application to This Thesis

1.3.2.1 Real-World Problem

The real-world problem addressed in this thesis arises from the practical challenges of implementing continuous auditing in the organization explained above. While automated controls ensure accuracy in the company's transactions, they fail to capture whether activities are executed in the correct order or align with the intended process model. Conformance checking has been introduced to address this limitation, yet its current implementation presents several issues: (1) outputs are technically complex and difficult for auditors to interpret; (2) the number of detected deviations is excessively high due to the lack of contextual differentiation between exceptions and anomalies; and (3) existing approaches offer a limited analytical perspective, restricted to control-flow information neglecting data and resource dimensions. Consequently, auditors face a growing gap between automated detection and meaningful interpretation. The need to classify deviations effectively, balancing control with operational flexibility, forms the real-world problem for this thesis. The main real-world problem can thus be summarized as the lack of interpretability, contextual understanding, and multi-perspective integration in current process-level auditing systems.

This practical challenge provides the basis for abstracting the problem into an algorithmic task.

1.3.2.2 Algorithmic Task

Building on the real-world challenge described above, the problem is abstracted into an algorithmic task that defines its computational focus and boundaries.

The *goal* of the algorithmic task in this thesis is to classify control-flow deviations from the normative model into two classes: exceptional cases and anomalous cases. Exceptional cases deviate from the model but can be explained by implicit organizational rules, representing acceptable process variations. Anomalous cases represent risks of non-compliance, fraud, or operational errors. The task is therefore to provide a systematic and repeatable way of separating these two types of deviations so that auditors can focus on the cases that require further investigation.

Three *assumptions* frame this task.

First, the available process model is a normative "happy path" model, describing the desired control-flow of the process. It reflects how the process is intended to operate under standard conditions but captures only idealized behavior, excluding many real-world variants and exceptions. In practice, such models are typically maintained by process owners to formalize the main procedural guidelines rather than to exhaustively

capture all process variability. This assumption is both realistic and methodologically appropriate. It is less restrictive than assuming the existence of a perfect normative model, which would require full coverage of all possible behaviors and exceptions. Many organizations document only their principal process flows for compliance purposes, while many other variations remain undocumented or regulated by formal or implicit rules. Therefore, starting from a normative model aligns with how process knowledge is actually represented in practice, while still providing a sufficient formal foundation for algorithmic analysis.

Second, there exists substantial tacit and implicit knowledge among internal auditors, process owners, and process users. This knowledge extends process understanding beyond what is captured in the normative model, reflecting experience-based judgments and contextual rules that remain undocumented. Acknowledging this assumption is crucial too, as it reflects the process executions are enacted in practice, through expertise and situational or contextual interpretation rather than exhaustive formalization.

Third, the task assumes the availability of event logs with a sufficiently detailed and reliable control-flow structure to support conformance checking. Event logs must contain well-defined case identifiers, activity labels, and timestamps to enable accurate reconstruction of process traces. While such logs are increasingly common in modern information systems, their completeness and quality vary across organizations. This assumption ensures that the computational analysis can be meaningfully applied, even though real-world logs may still omit contextual attributes needed for multi-perspective interpretation.

By defining this goal and these assumptions, the scope of this thesis is established: the focus lies on classifying cases based on the control-flow perspective, using the normative model as a baseline, event logs as the empirical basis for analysis, and human expertise as a complementary source of contextual knowledge.

1.4 Thesis Contributions

The contributions of this thesis are structured around a systematic approach to addressing the problem: improving the interpretability and practical usefulness of conformance checking in continuous auditing. Following the algorithm engineering framework, the research progresses from understanding the nature of the problem (knowledge of and about task) to developing, evaluating, and generalizing algorithmic solutions (knowledge of and about design). The approach adopted in this thesis

is structured around four main stages.

Understanding how auditors interpret process deviations (knowledge of task)
The research begins by analyzing process deviations to understand how auditors perceive and classify them in practice. This stage employs a field study to explore how auditors interpret detected deviations and how their mental models guide this interpretation. Of course, there exist some deviation patterns in the literature, but they all originated from theoretical approaches and therefore not by definition, aligned with the way auditors analyze deviations in real audit context. To address this gap, a field study is designed to study whether there is a set of deviation types for interpreting all possible deviations in a way as close as possible to the mental model of auditors. This categorization provides auditors with a high-level overview of deviations and their frequencies. Starting from this overview, the auditors can drill down in a targeted way. The insights gained from this analysis support the development of auditor-oriented deviation categories that make conformance checking results more interpretable for auditors. These results are presented in Chapter 4.

Applying deviation categories in automated deviation classification technique (knowledge of and about design)
Chapter 5 introduces a method using conformance checking and sequence mining techniques to reduce false positives and improve the efficiency of deviation analysis by providing targeted support for auditors. This approach integrates additional perspective into conformance checking to enhance its ability to assist auditors in classifying deviations. In other words, we provide a solution to assist auditors in analyzing more deviating instances in a shorter time compared with the current manual approach. Chapter 6 analyzes how the proposed classification technique can be applied in practice using a real world case study. These chapters are thereby generating knowledge of and about design and demonstrating how the deviation categories can be effectively used to support deviation analysis.

Developing and generalizing a framework for deviation analysis in continuous auditing (knowledge of design)
Building on the understanding developed in the first stage and the insights from the proposed classification technique, Chapter 7 introduces a novel framework that integrates process mining, data mining, and active learning to support and enhance continuous auditing. This framework is designed to manage alarm floods, integrate professional auditor judgment, and increase the utility of continuous auditing systems. It combines automated detection with auditor expertise to enable a more context-

aware analysis of process deviations. In addition, the framework incorporates control-flow perspectives and data perspectives to improve the applicability of conformance checking results. Through the integration of these components, the chapter presents a generalized framework for process deviation analysis that transforms conformance checking from a purely technical procedure into a decision-support tool for auditors. The conceptual foundation and algorithmic design of this framework are described in Chapter 7.

1.5 Knowledge Contributions in this Thesis

Within the epistemological framework proposed by Mendling et al. [2025], the contributions of this thesis generate different forms of algorithmic knowledge relating to both the underlying task and the proposed designs. First, the categorization of process deviations developed in Chapter 4 provides knowledge about the task. Supported by field study, this categorization enhances the understanding of how auditors interpret deviations in practice. Building on this task understanding, Chapter 7 contributes knowledge of design by introducing a conceptual and algorithmic framework for process deviation analysis. This framework integrates process mining, data mining, and active learning into a coherent design, specifying how these components can be combined to support continuous auditing. Chapter 5 contributes both knowledge of design and knowledge about design. It introduces a classification technique (knowledge of design) and examines its behavior using an academic case (knowledge about design), providing insights into how the method functions in practice and how the deviation categories can be applied effectively. Chapter 6 applies this technique to a real-world case study, generating further knowledge about design by illustrating its properties under realistic audit conditions. These analyses also refine aspects of the conceptual framework, thereby producing combined knowledge of and about design. Building on this understanding, Chapter 7 contributes knowledge of design by presenting a conceptual framework that integrates process mining, data mining, and active learning to support process-oriented continuous auditing. Together, these contributions advance both the understanding of the deviation classification task and the development of an algorithmic design to support process-oriented continuous auditing.

1.6 Scope and Limitations

This thesis focuses on the problem of process deviation analysis within the broader context of internal auditing, as motivated by the fictitious organization described

earlier. The work is concerned with how detected deviations are interpreted and classified when applying process-oriented analytical techniques in an internal audit environment. The thesis concentrates on conformance checking as a type of process mining technique for identifying control-flow deviations. Among the various analytical perspectives available in process mining, the research is initially restricted to the control-flow perspective when defining deviation categories disregarding supplementary data and resource information (e.g., the value of a transaction or the identity of the person executing a specific activity). This limitation is deliberate: the normative model used in the examined environment is specified at the control-flow level, and most of the conformance checking algorithms operate and emphasize on the sequence of activity execution. Focusing on this perspective ensures coherence between the organizational setting, the available artefacts, and the assumptions of the algorithmic task.

Consequently, the deviation categorization developed in Chapter 4 does not incorporate additional data and resource attributes. It does not incorporate contextual factors such as transaction amounts, user roles, or operational conditions, elements that influence auditors' judgments but are not formally captured in the normative model available in this environment. In the subsequent stages of the thesis, particularly in the automated classification techniques presented in Chapters 5 and 6, the scope expands to include the data perspective. This extension is introduced not to detect additional deviations, but to enrich the interpretation of already detected control-flow deviations by providing contextual information that supports the classification process. Integrating selected data attributes enables the classification approach to better reflect auditors' reasoning while remaining consistent within a controlled algorithm engineering framework and with the algorithmic task defined earlier in the introduction.

Despite this extension, the thesis does not aim to develop a fully multi-perspective conformance checking approach that simultaneously integrates control-flow, data, and resource information. Resource perspective information such as who performs a given activity, remains outside the scope because the examined environment lacks sufficiently formalized resource information and contextual rules needed to model them reliably.

The thesis relies on a single real-world case study to evaluate the proposed classification technique, which allows for an in-depth analysis grounded in authentic operational data. However, using one organizational setting means that the findings cannot be generalized broadly without further empirical validation across additional contexts. Overall, the scope reflects the methodological choices necessary to study the central problem: bridging the gap between automated deviation detection and

meaningful interpretation in internal auditing. The main limitations concern:
(1) the exclusive focus on control-flow deviations as the deviation type of interest; (2) the reliance on a normative process model rather than exhaustive process descriptions, i.e, a fully representative model of actual operations; and (3) the use of selected cases rather than broad generalization across organizational settings. These limitations follow naturally from the epistemological positioning of the thesis and do not diminish the validity or relevance of the knowledge contributions it provides.

1.7 Thesis Overview and Structure

This thesis is structured as follows. Chapter 2 introduces the real-world data set that serves as the case study throughout the thesis. Chapter 3 provides background on continuous auditing within internal auditing, process mining, and the conformance checking techniques relevant to this research. Chapter 4 proposes a set of deviation categories for interpreting process deviations. Chapter 5 presents a framework and technique designed to assist auditors in classifying control-flow deviations more effectively. Chapter 6 applies this technique to a real-world case study for evaluation and illustrate its practical use. Chapter 7 introduces a framework for integrating conformance checking with active learning to support continuous auditing. Finally, Chapter 8 concludes the thesis. An overview of the thesis structure is provided in Figure 1.2.

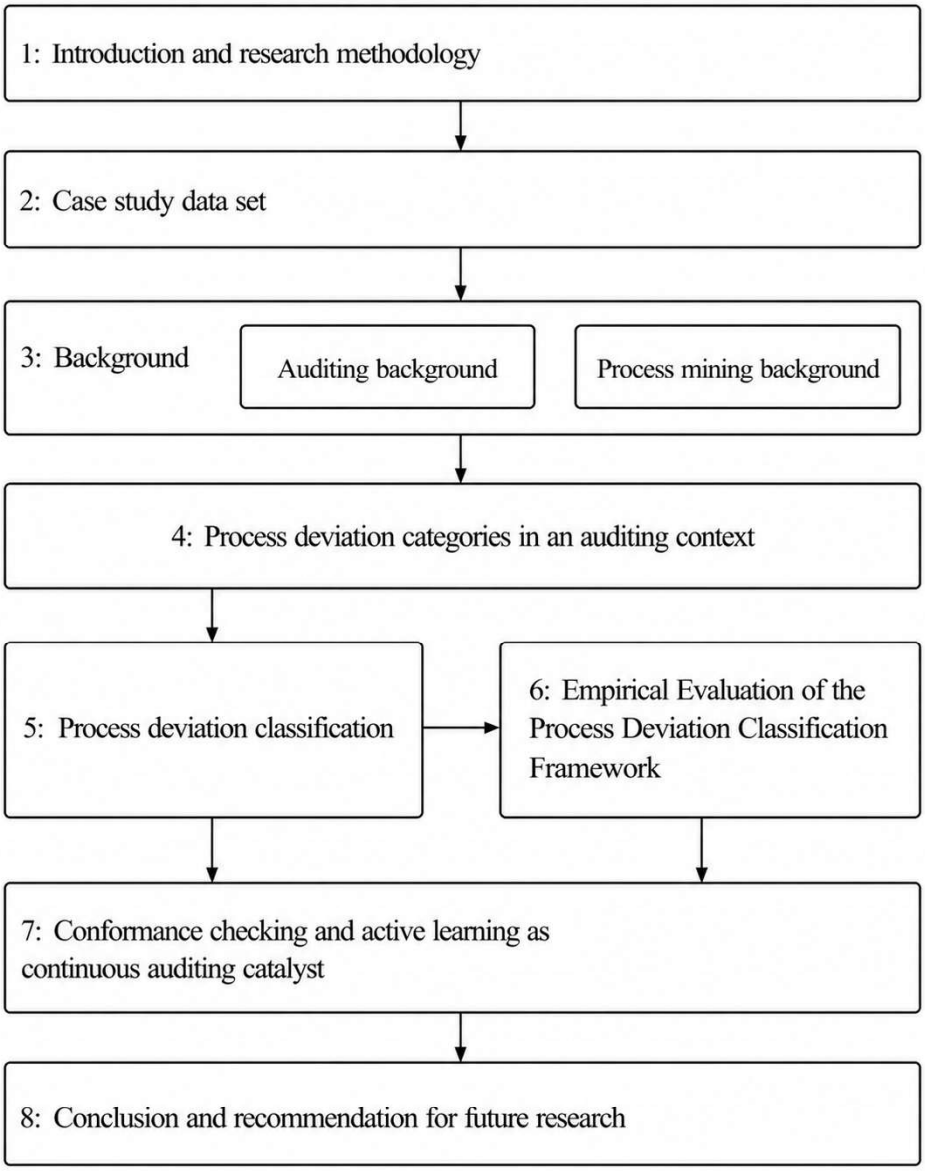


Figure 1.2: Outline of the thesis

Chapter 2

Thesis Case Study Data Set

In this chapter, we introduce the data set of the case company that is used throughout the thesis. Presenting the data set and organizational context at this stage helps to further clarify and ground the research problem addressed in this thesis. The case study is introduced early in the thesis to provide a practical context for understanding the theoretical concepts and techniques discussed throughout the research. Positioning the case study at this stage allows readers to familiarize themselves with the specific processes, data, and challenges that the research addresses. By grounding the thesis in a concrete example, it becomes easier to illustrate the relevance and applicability of the proposed methods as they are developed in subsequent chapters.

Furthermore, presenting the case study at the beginning ensures that the alignment between theoretical frameworks and real-world auditing scenarios is evident from the outset. This structure highlights the practical implications of the research and helps establish a strong connection between the foundational concepts in process mining and their application in auditing practices.

A real-world procurement (Purchase-to-Pay) process is introduced in this chapter as our case study data. This case study's data is used throughout the following chapters of this thesis. The data set was firstly used in Jans et al. [2014]. The data transition from procurement cycle was collected from the SAP system to generate the event log based on the domain knowledge.

The event log belongs to a period between 02.01.2007 00:00:00 and 25.01.2008 00:00:00 and contains 26,185 process instances. There are, in total, seven distinct activities. The activities are as follows: Create Purchase Order, Sign, Release, Goods Receipt, Invoice Receipt, Pay, and Change Line. Aside from the timestamps and originator, the other attributes of the process are the following: purchase order creator, purchase order quantity (the number of ordered units on this line), unit (the

unit in which the quantity is expressed), purchase order value (in Euros), supplier, purchasing group, document type, goods receipt indicator (whether or not the GR indicator was flagged. If this indicator is flagged, the input of a GR is mandatory for the payment of the invoice. If it is not, the invoice can be paid without a GR), goods receipt total quantity, goods receipt total value (total value of all goods received for this purchase order), invoice receipt total quantity, invoice receipt total value (total value of all invoices received for this process instance), pay total value (total value of all payments, that are associated with this process instance).

The process instance ID is a unique 10-digit number, starting with 45, such as '45xxxxxxx' for each of the purchase orders. Two examples of the process instance IDs are shown in Table 2.2 together with their attributes.

The directly-follow graph[1] of this process log has revealed that the medium case duration is 20 days and the mean case duration is 46.2 days. In total, there are 980 variants. The directly-follow graph in Figure 2.1 shows the actual flow of all process instances of this purchase-to-pay data set. This process map is automatically discovered by a process mining tool. The boxes represent the activities (tasks), and an arrow between two boxes visualizes the process flow between two activities. The absolute frequencies of activities or flows are displayed in the activity boxes or on the arcs. The thickness of the arrows and the darkness of the activities visually support the frequency numbers on the map. In the map, PO, GR, and IR are abbreviations for Purchase Order, Goods Receipt, and Invoice Receipt, respectively.

The activity that takes place fewer times is Change Line, with 15,468 occurrences. The most frequent activity is Pay which occurred 31,817 times. Table 2.1 illustrates the frequency of each activity. As it is illustrated in both Figure 2.1 and Table 2.1, the activity Create PO only occurred 26,185 times, showing each process instance has only one Create PO, and it is the starting activity of the process. Activities Pay, Invoice Receipt, and Release have happened more than the number of process instances, meaning that there are cases that have more than one execution of these activities. On the other hand, activities Sign, Goods Receipt, and Change Line have a lower frequency compared with the total number of process executions, simply meaning that there are cases that lack these activities.

[1]The academic version of Disco, one of the process mining tools in the market, is used for generating this graph. The tool is available on www.fluxicon.com

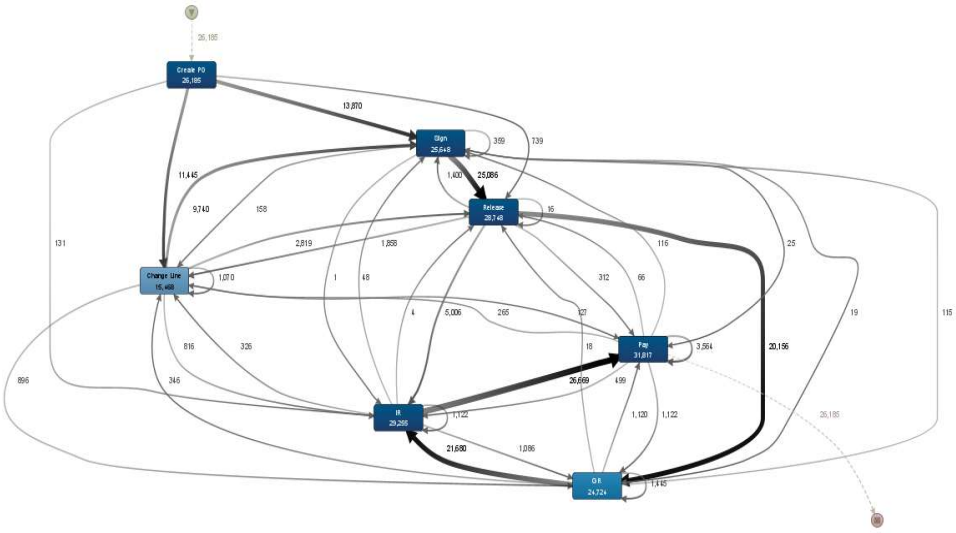


Figure 2.1: The actual directly-follows graph of all process instances in the purchase to pay example

Activity	Frequency	Relative frequency
Pay	31,817	17.5%
Invoice Receipt	29,255	16.09%
Release	28,748	15.81%
Create Purchase Order	26,185	14.4%
Sign	25,648	14.1%
Goods Receipt	24,724	13.6%
Change line	15,486	8.51%

Table 2.1: Frequency of all activities in the event log and their relative frequency in relation to the number of process instances

Table 2.2 shows an example of data attributes taken from Jans et al. [2014]

Process Instance ID	Attribute Name	Value
450000000190	Doc Type	DI
450000000190	PG	B01
450000000190	Supplier	12345
450000000190	Net Value	10:000
450000000190	Unit	EA
450000000190	Quantity PO	1
450000000190	GR Ind	X
450000000190	GR Total Quantity	1
450000000190	GR Total Value	10:000
450000000190	IR Total Quantity	1
450000000190	IR Total Value	10:000
450000000190	Pay Total Value	10:000
450000000210	Doc Type	FO
...		

Table 2.2: Example of data attributes of the event log

Besides the attributes mentioned above, the following event attributes are related to activities: Concept name (for all activities), Resource (for all activities), Lifecycle (for all activities), Timestamps (for all activities), Reference GR (IR), Reference Pay (IR), Quantity IR (IR), Value IR (IR), Reference IR (GR), Quantity GR (GR), Value GR (GR), Reference IR (Pay), Value (Pay), Modification (Change Line) and Relative Modification (Change Line). Table 2.3 describes all the event attributes which exist per activity in this event log. The event attribute description and the type of attributes are also provided in the table.

Table 2.4 shows four out of 26,185 instances from this event log. For each process instance, the process variants, case attributes, and (for the sake of space) some of the event attributes are provided.

Event	Event attribute	Event attribute description	Type
Create PO	Concept name	Name of the event	Categorical
	Resource	The originator who performed the event	Numerical
	Lifecycle	An indicator about the completeness of the event	Binary
	Timestamp	The date-time that the event is completed	DateTime
GR	Concept name	Name of the event	Categorical
	Resource	The originator who performed the event	Numerical
	Lifecycle	An indicator about the completeness of the event	Binary
	Timestamp	The date-time that the event is completed	DateTime
	GR-Quantity	Quantity of Goods Receipt	Numerical
	GR-LFBNR	SAP field with the reference to the invoice doc.	Numerical
	GR-Value	Value of Goods Receipt	Numerical
IR	Concept name	Name of the event	Categorical
	Resource	The originator who performed the event	Numerical
	Lifecycle	An indicator about the completeness of the event	Binary
	Timestamp	The date-time that the event is completed	DateTime
	IR-Objectkey	SAP field with the reference to the payment doc.	Numerical
	IR-Value	Value of Invoice Receipt	Numerical
	IR-Quantity	Quantity of Invoice Receipt	Numerical
	IR-LFBNR	SAP field with the reference to the goods doc.	Numerical
Pay	Concept name	Name of the event	Categorical
	Resource	The originator who performed the event	Numerical
	Lifecycle	An indicator about the completeness of the event	Binary
	Timestamp	The date-time that the event is completed	DateTime
	Pay-Value	Value of Pay	Numerical
	Pay-Objectkey	SAP field with the reference to the invoice doc.	Numerical
Change Line	Concept name	Name of the event	Categorical
	Resource	The originator who performed the event	Numerical
	Lifecycle	An indicator about the completeness of the event	Binary
	Timestamp	The date-time that the event is completed	DateTime
	Rel-Modification	Relative difference between old - and new values	Numerical
	Modification	Absolute difference between old - and new values	Numerical

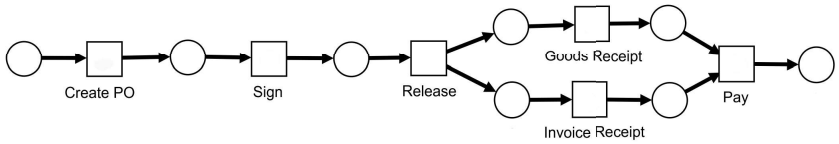
Table 2.3: Event attributes of events in the case study

No.	Process instance	Attributes
	Variant	Create PO, Sign, Release, IR, Pay
1: 450040304610	**Case Attributes**	Value-PO:128250, Quantity-PO:30, Conceptname:450040304610, Doc-Type:FO, PG:H01, Value-Pay:128200, Supplier:82706, Quantity-IR:30, Value-IR:128250, Unit:BL
	Event Attributes	Conceptname:Create PO, resource:U70162, lifecycle:complete, timestamp:2007-01-30T00:00:00 Conceptname:Sign, resource:G22364, lifecycle:complete, timestamp:2007-01-31T00:00:00 Conceptname:Release, resource:G40152, lifecycle:complete, timestamp:2007-02-06T00:00:00 Conceptname:IR, IR-Value:128250, resource:G10849, IR-Quantity:30, IR-LFBNR:No value defined, IR-Objectkey:5105739598200 7, lifecycle:complete, timestamp:2007-03-05T00:00:00 No info for GR Conceptname:Pay, resource:G10849, Pay-Value:128250, Pay-Objectkey:5105739598200 7, lifecycle:complete, timestamp:2007-03-23T00:00:00
	Variant	Create PO, Sign, Release, Change Line, Sign, Release, GR, Pay
2: 450040096380	**Case Attributes**	Value-PO:1316276, Quantity-PO:1, Quantity-GR:1, Value-Pay:1592600, Quantity-IR:1, Unit:EA, Conceptname:450040096380, Value-GR:1592694, Doc-Type:FO, PG:I17, Supplier:2343, Value-IR:1592694, GR-Ind:X
	Event Attributes	Conceptname:Create PO, resource:G24237, lifecycle:complete, timestamp:2007-01-23T00:00:00 Conceptname:Sign, resource:U66052, lifecycle:complete, timestamp:2007-01-23T00:00:00 Conceptname:Release, resource:U77555, lifecycle:complete, timestamp:2007-01-26T00:00:00 Conceptname:Change Line, Modification:0, resource:G30708, Rel-Modif:0, lifecycle:complete, timestamp:2007-07-13T00:00:00 Conceptname:Sign, resource:G38262, lifecycle:complete, timestamp:2007-07-16T00:00:00 Conceptname:Release, resource:U66052, lifecycle:complete, timestamp:2007-07-16T00:00:00 Conceptname:GR, resource:G24237, GR-Quantity:1, GR-Value:1592694, GR-LFBNR:No value defined, lifecycle:complete, timestamp:2007-07-26T00:00:00 Conceptname:Pay, resource:U71980, Pay-Value:1592694, Pay-Objectkey:5105787307 2007, lifecycle:complete, timestamp:2007-07-31T00:00:00

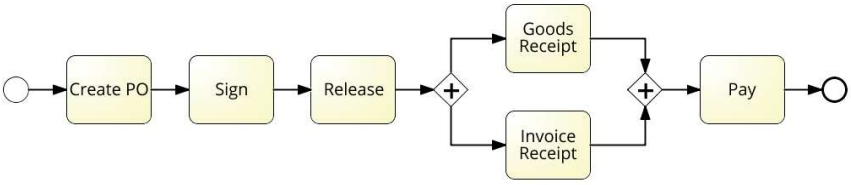
3: 450039661580	**Variant**	Create PO, Change Line, Sign, Release, GR, IR, Pay
	Case Attributes	Value-PO:1074, Quantity-PO:1, Quantity-GR:1, Value-Pay:1000, Quantity-IR:1, Unit:EA, Conceptname:450039661580, Value-GR:1074, Doc-Type:FO, PG:L07, Supplier:1816, Value-IR:1074, GR-Ind:X
	Event Attributes	Conceptname:Create PO, resource:U45859, lifecycle:complete, timestamp:2007-01-08T00:00:00 Conceptname:Change Line, Modification:0, resource:U45859, Rel-Modif:0, lifecycle:complete, timestamp:2007-01-08T00:00:00 Conceptname:Sign, resource:G16977, lifecycle:complete, timestamp:2007-01-09T00:00:00 Conceptname:Release, resource:U29598, lifecycle:complete, timestamp:2007-01-09T00:00:00 Conceptname:GR, resource:U45859, GR-Quantity:1, GR-Value:1074, GR-LFBNR:No value defined, lifecycle:complete, timestamp:2007-01-12T00:00:00 Conceptname:Pay, resource:G15330, Pay-Value:1074, Pay-Objectkey:5105724255200 7, lifecycle:complete, timestamp:2007-01-24T00:00:00
4: 450039860620	**Variant**	Create PO, Sign, Change Line, Sign, Sign, Release, GR, IR, Pay
	Case Attributes	Value-PO:1328, Quantity-PO:1, Quantity-GR:1, Value-Pay:1600, Quantity-IR:1, Unit:EA, Conceptname:450039860620, Value-GR:1607, Doc-Type:DI, PG:L14, Supplier:6508, Value-IR:1607, GR-Ind:X
	Event Attributes	Conceptname:Create PO, resource:U21356, lifecycle:complete, timestamp:2007-01-15T00:00:00 Conceptname:Sign, resource:G20186, lifecycle:complete, timestamp:2007-01-15T00:00:00 Conceptname:Change Line, Modification:0, resource:U21356, Rel-Modif:0, lifecycle:complete, timestamp:2007-01-16T00:00:00 Conceptname:Sign, resource:G20186, lifecycle:complete, timestamp:2007-01-16T00:00:00 Conceptname:Sign, resource:G20186, lifecycle:complete, timestamp:2007-01-16T00:00:00 Conceptname:Release, resource:G56639, lifecycle:complete, timestamp:2007-01-17T00:00:00 Conceptname:GR, resource:U50978, GR-Quantity:1, GR-Value:1607, GR-LFBNR:No value defined, lifecycle:complete, timestamp:2007-01-23T00:00:00 Conceptname:Pay, resource:G55584, Pay-Value:1607, Pay-Objectke:5105730080200 7, lifecycle:complete, timestamp:2007-02-14T00:00:00

Table 2.4: Process instances examples together with their event attributes. Only four out of 15,056 deviating instances are shown here.

The normative model of the procurement process in this event log is depicted in Figure 2.2. Figure 2.2a shows the model in Petri-net notation. Petri-net is a modeling language using directed graph in which the nodes represent transitions (rectangles) which are activities and places (circles). Figure 2.2b illustrates the same process model in BPMN (Business Process Model and Notation) format. BPMN is another graphical representation for business processes using some standard objects like events (circles), activities (boxes), gateways and connectors between them. As it is clear in both representations of the model, only two sequences of tasks are allowed by the model: ⟨Create PO, *Sign*, *Release*, *Goods Receipt*, *Invoice Receipt*, *Pay*⟩ and ⟨*Create PO*, *Sign*, *Release*, *Invoice Receipt*, *Goods Receipt*, *Pay*⟩.



(a) Normative model for the purchase to pay process in Petri-net



(b) The same normative model for the purchase to pay process in BPMN

Figure 2.2: Normative model for the purchase to pay process that is used in this thesis as the case study data set

The two variants that follow these two sequences of tasks are the most frequent variant with 10,943 cases (41% of all process instances) and the 12th variant with 186 cases (0.71% of all process instances). Figure 2.3 shows these two normal variants.
Among 26,185 process instances (980 variants) in process instances, there are 15,056 instances (978 variants) that are deviating from the normative process model. Figure 2.4 shows the map of all deviating cases from this normative model.
There are, in total, 272 roles who were responsible and performers of all the activities throughout the period that this event log is selected.

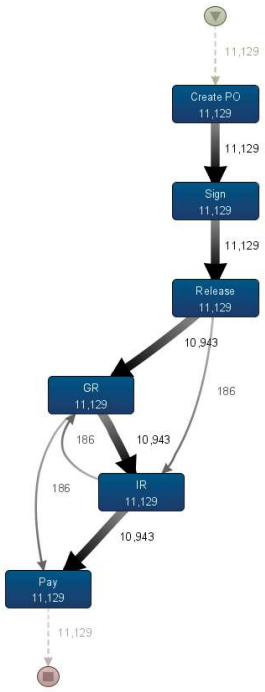


Figure 2.3: The process map of the normal variants. The sequences which follow the normative process model

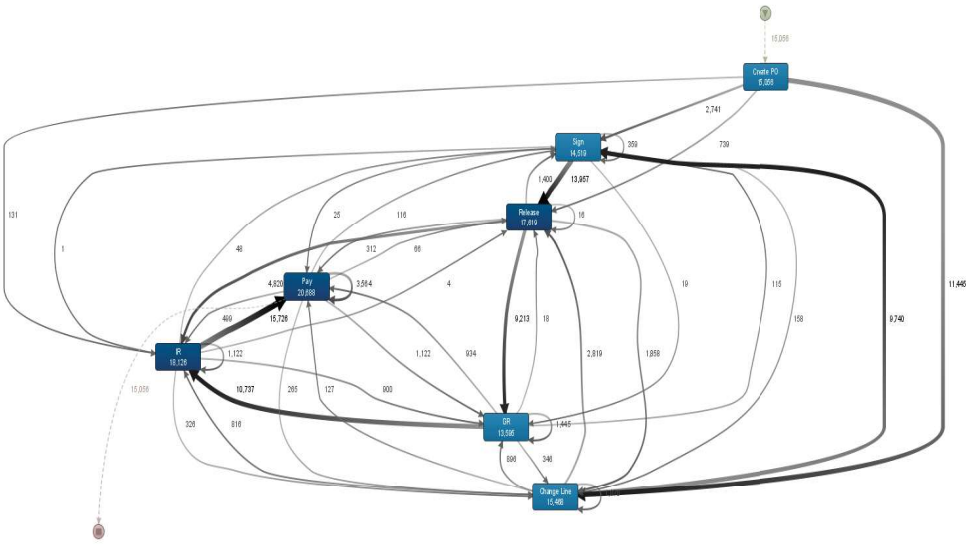


Figure 2.4: The process map of the deviating cases from the normative model

Chapter 3

Background

This chapter establishes the conceptual foundation of the thesis by examining the evolution of auditing practices in the context of digital transformation. It highlights how the emergence of Continuous Internal Auditing and process mining has improved the way assurance is performed within modern organizations. Continuous auditing extends internal auditing toward an automated and data-driven form of continuous assurance, while process mining provides analytical techniques to evaluate how business processes are executed in practice. Together, these developments form the technological and methodological basis for this research, which aims to enhance process-level assurance in continuous auditing environments, while acknowledging the challenges associated with their practical adoption.

Recent studies highlight that, although emerging digital technologies such as artificial intelligence, robotic process automation, and process mining are increasingly discussed in auditing, their adoption in practice remains uneven and their impact on audit processes and organizational structures is still evolving [Vitali and Giuliani, 2024].

3.1 Internal Auditing in a Changing Environment

Internal auditing is an independent and objective assurance activity designed to evaluate and improve an organization's governance, risk management, and internal control processes. Traditionally, internal auditing relied on periodic testing, manual inspection, and sample-based procedures to assess control effectiveness. However, in recent years, the internal audit function has undergone a profound transformation, driven by advances in information technology, stronger regulatory demands, and the increas-

ing speed and complexity of business operations. A key development in this transformation is the rapid digitization of organizational processes. Modern enterprises increasingly operate on integrated ERP, CRM, and workflow management systems that record large volumes of transactional and event-level data. This data-rich environment provides internal auditors with unprecedented opportunities to shift from a sampling-based approach and periodic reviews toward more frequent, data-driven, and automated forms of assurance activities Eulerich et al. [2025], Marcus et al. [2024], Vasarhelyi and Halper [1991], Abdolmohammadi and Sharbatouglie [2005]. As highlighted by the PCAOB, "advancements in technology continue to affect the nature, timing, and preparation of financial information … and the planning and performance of audits" [PCAOB, 2021], underscoring the extent to which auditing practices are reshaped by digital transformation.
The conceptual shift toward more automated and data-driven assurance is not entirely new. In 2004, four levels of audit objectives for continuous assurance were proposed: transactional verification, compliance verification, estimate verification, and judgment verification. Among these, transactional verification is most closely aligned with business processes and provides the strongest foundation for the practical adoption of continuous auditing principles.

These principles, as articulated by Chan and Vasarhelyi [2011a], encompass several key dimensions:

1. Conducting audits more frequently to provide timely assurance.

2. Adopting proactive approaches to identify and address risks in real time.

3. Utilizing automated procedures to enhance efficiency and reduce manual intervention.

4. Focusing on exception-based work and applying human judgment selectively, with the auditor acting as a certifier.

5. Implementing continuous control monitoring and data assurance, enabling simultaneous control monitoring and detailed testing across entire data populations.

6. Leveraging data modeling and analytics for monitoring and testing to improve accuracy and insight.

7. Increasing the frequency of reporting to align with the dynamic needs of stakeholders.

These dimensions represent a shift from periodic, retrospective audits toward a dynamic, technology-enabled, process-integrated assurance framework, within which transactional verification plays the central operational role.

Building on these principles, continuous auditing can be defined as the systematic, automated, and frequent, or even real-time, assessment of organizational activities, controls, and transactions, leveraging event-level data and analytical techniques to deliver more timely assurance Gonzalez et al. [2012]. Although advancements in information systems have enabled the automation of repetitive and routine tasks Eulerich et al. [2025], the auditing profession increasingly depends on sophisticated computational techniques capable of processing large and complex data sets. The increased use of structured and unstructured data, more advanced algorithms and analytical capabilities is expected to increase audit efficiency and enhance analytical depth Yoon et al. [2015], Duan et al. [2025]. These developments enable auditors to identify anomalies, patterns, and risks beyond the limits of traditional rule-based or sampling-based testing Laghmouch et al. [2024]. Moreover, it allows the monitoring of controls, transactions, and risks at much shorter intervals than the traditional audit cycle Lenz and Jeppesen [2022]. Rather than relying solely on sampling or periodic testing, auditors increasingly use automated checks, dashboards, rule-based monitoring, anomaly detection, and even predictive models to detect irregularities and control deviations as they occur. These data-enabled audit processes enhance transparency, enable more timely interventions, and provide more comprehensive assurance coverage Chan and Vasarhelyi [2011b], Barros and Marques [2022]. Reflecting this shift, the PCAOB observes that technology allows auditors to "analyze complete populations of transactions in new or unexpected ways" [PCAOB, 2021]. Such capabilities significantly expand the scope of the auditor's analysis.

Recent audit research increasingly positions data analytics, artificial intelligence, machine learning, full-population testing techniques, robotic process automation, and process analytics as central technologies for improving audit efficiency, quality, and effectiveness (e.g., Perdana et al. [2023], Huang and Vasarhelyi [2019], Kim and Vasarhelyi [2012], Alles et al. [2006b], Issa [2013], Jans et al. [2014]). However, despite their potential, these technologies are not yet fully embedded in audit practice. Key challenges include data ingestion, explainability, false positives, auditor adoption, required analytical skills, and integration into audit standards and workflows [Black and Gerard, 2025, Vitali and Giuliani, 2024].

According to the AICPA [of Certified Public Accountants, 2017], data analytics for auditing is "the science and art of discovering and analyzing patterns, identifying anomalies, and extracting other useful information in data underlying or related to

the subject matter of an audit, through analysis, modeling, and visualization". In line with this shift, the PCAOB emphasizes that new technologies enable auditors to identify emerging risks, refine their procedures iteratively, and adjust responses as new information becomes available [PCAOB, 2021]. Taken together, these developments underscore the need for internal audit functions to expand their analytical scope beyond transaction-level validation toward a deeper understanding of business processes in their operational context.

3.2 Audit Analytics in Internal Auditing

Recent developments in auditing reflect a shift from sample-based procedures toward data-driven and technology-supported approaches, enabling full-population analysis and more continuous forms of assurance. This section discusses two strands of audit analytics that are relevant to auditing supported by data analytics: textual analysis, which focuses on unstructured data relevant to auditing, and process-oriented analytics, which focuses on structured event log data and process execution.

3.2.1 Textual Analysis in Auditing

Within this broader development, textual analysis has gained importance because many audit-relevant sources are not purely numerical. Documents such as journal entries, invoices, contracts, emails, and reports contain textual information that may provide evidence about risks, explanations, and unusual behavior. Textual analysis, text mining, and natural language processing techniques enable the extraction of patterns, keywords, and risk-related signals from such unstructured data [Sun and Vasarhelyi, 2018, Loughran and McDonald, 2016].

Textual analysis complements quantitative methods by capturing qualitative signals such as sentiment and risk indicators, but transforming text into quantitative representations introduces imprecision and context dependency. Consequently, domain-specific modeling and careful interpretation are essential when applying textual analysis in auditing contexts [Loughran and McDonald, 2016].

In auditing research, textual analysis has been applied in several ways. Sun and Vasarhelyi [2018] discuss the potential of textual data analytics and deep learning for extracting audit-relevant information from textual sources. Lee et al. [2022] demonstrate that integrating textual analysis with visualization enables auditors to identify anomalous journal entries more effectively by transforming unstructured descriptions into interpretable patterns, thereby showing how textual information in accounting systems can complement numerical transaction analysis. These studies suggest that

textual data can enrich audit analytics by capturing qualitative explanations and contextual signals that are not available in structured numerical data alone.

Literature reviews show that text mining techniques have been applied to extract, classify, and analyze textual data in accounting contexts, supporting tasks such as risk identification, disclosure analysis, and the interpretation of qualitative information [Senave et al., 2023].

More recently, large language models (LLMs) have advanced textual analysis by enabling the extraction of context-specific information from unstructured data sources. Unlike traditional text mining approaches, LLMs can retrieve structured financial information from complex documents more effectively, thereby improving the efficiency of audit data extraction and analysis [Li et al., 2025, Dong et al., 2024].

However, the literature also highlights important limitations. Textual data is often noisy, domain-specific, and difficult to interpret without contextual knowledge. Moreover, advanced techniques such as deep learning may improve classification or prediction performance but can reduce transparency, which is problematic in auditing where evidence, professional judgment, and explainability are critical. Therefore, the integration of textual analysis into auditing should not be viewed purely as a technical automation task, but as part of a broader audit analytics environment in which analytical outputs must remain interpretable, auditable, and useful for human decision-making.

In this context, the challenge is not only to extract information from textual data, but to integrate it into decision-support structures that align with auditors' reasoning and workflow.

3.2.2 Process-Oriented Audit Analytics

While textual analysis illustrates the growing role of unstructured information in audit analytics, the present thesis focuses on structured event log data and process-level assurance. Building on the broader shift toward audit analytics, process mining techniques can be regarded as a suitable and promising analytical approach. By reconstructing and evaluating actual process executions from event logs, process mining enables auditors to identify inefficiencies, deviations from intended behavior, and violations of prescribed procedures [Duan et al., 2025, Föhr et al., 2025]. This process-oriented perspective is essential in modern audit environments, where transactions may individually comply with rules but the overall process may still diverge from regulatory or organizational expectations. Modern data-enabled internal auditing differentiates itself from traditional internal auditing along at least three key dimensions: frequency, automation, and coverage.

First, with respect to frequency, traditional internal audit operates on cyclical

intervals, i.e., annually, semi-annually, or quarterly, whereas data-driven auditing enables much more frequent, often continuous or near real-time monitoring [Eulerich et al., 2025].

Second, with respect to automation, rule-based checks, dashboards, data mining, and anomaly detection complement or replace manual sampling and retrospective testing [Chan and Vasarhelyi, 2011b, Marcus et al., 2024].

Third, analyzing the full population or near full population of transactions reduces sampling risk and provides broader insight into operational process and control performance.

The increasing importance of data-driven assurance practices can be explained by three contextual developments:

First, the digitization of enterprise systems along with the integration of ERP and CRM systems, has created extensive event logs and data infrastructures necessary for continuous auditing [Gonzalez et al., 2012].

Second, increasing regulatory demands and stakeholder expectations for transparency require more timely and reliable assurance [Lenz and Jeppesen, 2022].

Third, organizational structures and business processes have become increasingly complex and interdependent, generating risk exposures that are not easily detected through static or retrospective auditing approaches [Vasarhelyi, 2010].

In practice, these developments are often implemented through components such as continuous controls monitoring (CCM), continuous data assurance (CDA), and continuous risk assessment (CRA) [Barros and Marques, 2022]. These capabilities embed assurance processes into day-to-day operations, enabling more proactive and responsive internal audit functions. Together, these components establish a continuous assurance infrastructure that delivers audit evidence efficiently, supports management decisions, and embeds assurance into day-to-day operations rather than conducting it as an isolated, retrospective exercise.

Empirical studies demonstrate that many large organizations recognize the strategic potential of data-driven and technology-enabled auditing but continue to encounter challenges in implementation. For instance, González et al. [Gonzalez et al., 2012] identified IT capability, managerial support, and perceived relative advantage as key factors influencing the adoption of data-analytic approaches within internal auditing. More recent research similarly highlights persistent barriers. Eulerich and Kalinichenko [Eulerich et al., 2025] report that many internal audit functions struggle with limited data quality, insufficient analytical skills, and organizational resistance, all of which hinder the effective adoption of advanced audit analytics. Overall, the move toward more automated, continuous, and data-driven assurance represents both a technological and an organizational evolution. It reflects the shifting nature

of business operations, which have become faster, more integrated, and more complex, thereby increasing the demand for timely, reliable, and transparent assurance. Within the context of this thesis, data-driven auditing forms the operational and conceptual foundation upon which process-oriented assurance builds. As organizations increasingly rely on digital audit techniques to validate transactional accuracy across large data populations, ensuring process correctness through analytical methods such as process mining becomes the next essential step toward achieving comprehensive assurance.

In summary, the internal audit profession is undergoing a shift toward continuous, data-driven, and process-aware assurance, supported by advanced analytics and the availability of detailed event data. Against this backdrop, process mining plays a critical role by enabling internal auditors to move beyond transactional accuracy and toward evaluating whether business processes are executed as designed. This evolving context constitutes the conceptual backdrop for the research in this thesis.

3.3 Limitations of Traditional Auditing

Traditional internal auditing is typically conducted on an annual or semi-annual basis, relying heavily on manual procedures, retrospective review, and sampling. While historically sufficient for static business environments, this periodic approach is increasingly inadequate in modern organizations where processes evolve rapidly and risks materialize continuously. One key limitation concerns the delayed detection of fraudulent or erroneous practices. Recent global evidence shows that occupational fraud schemes remain undetected for extended periods: the ACFE's 2024 *Report to the Nations* reports a median fraud duration of 18 months, with losses increasing substantially the longer a scheme continues [of Certified Fraud Examiners, 2024]. Similarly, regulators emphasize that periodic audit procedures "may not identify issues on a timely basis," particularly in environments characterized by fast-changing data and risks [PCAOB, 2021]. These findings highlight the temporal insufficiency of traditional audit cycles in providing timely assurance.

A second limitation arises from the inherent constraints of sampling-based testing. Traditional audits typically apply sampling techniques (as prescribed in ISA 530) to infer conclusions about the entire population. However, sampling cannot guarantee the detection of rare but critical deviations in large-scale digital environments. As noted in recent auditing research, reliance on samples increases the likelihood of overlooking infrequent but high-risk anomalies, especially in organizations with extensive ERP-driven data populations [Eulerich et al., 2025, Singh et al., 2013]. This sampling

risk compromises the completeness and reliability of audit conclusions, particularly when deviations occur in small clusters or deviate from routine execution patterns.

Traditional auditing also faces limitations in its ability to identify and evaluate internal control performance objectively. Assessments often depend on interviews, workshops, and self-reported control descriptions, which offer a subjective and potentially biased representation of actual process execution. Modern regulatory insights further stress that traditional manual or paper-based techniques are insufficient for evaluating controls embedded in increasingly complex and automated information systems [PCAOB, 2021]. As organizations adopt advanced digital platforms, the need for objective, data-driven verification of control operation grows correspondingly. Recent academic work also points to the challenges traditional audits face in detecting process-level deviations. While a transaction may individually comply with formal rules, it may still be part of a broader process violation, such as incorrect activity sequencing or unauthorized bypassing of controls, that traditional audits are not designed to detect [Duan et al., 2025]. This limitation results from the absence of end-to-end process visibility in conventional audit procedures.

In response to these structural limitations, research over the past decade has increasingly focused on analytic techniques capable of evaluating entire populations of events rather than relying on samples. Modern studies demonstrate the feasibility and advantages of full-population deviation detection using statistical analysis, clustering, anomaly detection, and process mining [Laghmouch et al., 2024, Duan et al., 2025]. For instance, Kim and Vasarhelyi [2012] have used frequency tests based on potential fraud/anomaly indicators for detecting deviations. In their approach, scores were assigned to each indicator as to their level of significance. The deviations can be distinguished from the normal cases by aggregating all the scores for each case.

Thiprungsri and Vasarhelyi [2011] employed a similar idea of frequency tests methodology, by applying a cluster-based technique to label individual observations and small clusters as the deviations. This approach was further refined by Jans et al. [2011a], who demonstrated its effectiveness in identifying anomalies through enhanced clustering methods.

Moreover, process mining-based conformance checking enables a systematic comparison of real process executions against normative models for the identification of deviating cases from normal cases [Jans et al., 2014, 2013]. These developments show clear pathways for overcoming the inherent temporal, procedural, and analytical constraints of traditional auditing.

Overall, the limitations of traditional auditing, i.e., delayed detection, sampling risk, subjective control evaluation, and lack of process-level insight, highlight the importance of transitioning toward continuous, data-driven, and process-oriented as-

surance approaches. Modern audit analytics provide the technological foundation for this transition, enabling organizations to evaluate controls and processes based on complete, objective, and timely data rather than periodic snapshots.

3.4 Benefits, Challenges, and the Need for Process-Oriented Assurance in Data-Driven Auditing

Adopting data-driven and technology-enabled auditing practices brings a number of potential benefits to organizations and audit functions. From a benefits perspective, higher audit coverage (via full-population tests), more timely detection of control failures or anomalies, and improved alignment of audit activity with business execution all feature prominently [Chan and Vasarhelyi, 2011b]. These approaches support enhanced risk mitigation and monitoring by enabling management and internal audit to respond more quickly to emerging issues, for instance, detecting duplicate transactions, unusual patterns, or control deviations as they occur rather than months later [Gonzalez et al., 2012]. Furthermore, when automated and data-analytic routines are embedded into business systems, the internal audit function may shift from purely retrospective assurance toward more proactive risk monitoring, thereby supporting decision-making, improving control effectiveness, and potentially reducing audit costs [Eulerich et al., 2025].

However, the transition to such data-driven modes of assurance is not without significant challenges. Three key challenges stand out: (i) interpretability and meaningfulness of outputs, (ii) the volume and noise of alerts and deviations, and (iii) the limited analytical perspective of many existing techniques that fail to capture the full scope of process execution. Each of these challenges directly affects the utility of data-driven auditing within practice.

Interpretability and meaningfulness of outputs

While advanced analytical techniques enable the identification of complex patterns and anomalies, their outputs are often in a technical format (e.g., scripts, dashboards, or performance metrics) that auditors may find difficult to interpret in operational and business terms. This concern is particularly relevant for AI- and machine-learning-based audit tools, where limited explainability can restrict auditors' ability to evaluate, document, and rely on analytical outputs as audit evidence [Zhang et al., 2022a].

Without effective translation into audit language and contextual meaning, these alerts may lead to "alert fatigue," low response rates, or missed opportunities for intervention [Tronto and Killingsworth, 2021]. In essence, while automated and data-

driven systems can produce large quantities of information, transforming these outputs into actionable assurance insights remains a core challenge.

Volume of data and false positives

As data-driven auditing shifts from sampling to full-population or high-frequency testing, the number of identified deviations or exceptions tends to increase substantially. Many of these may represent benign process variations (exceptions permitted by policy or practice), rather than true control failures or anomalies. This creates a significant burden for auditors who must triage, investigate, and classify each case appropriately. The risk of false positives (and the resulting workload), can reduce the efficiency gains associated with automation [Gonzalez et al., 2012].

Limited analytical perspective and need for process-oriented assurance

A further limitation is that many data-analytic auditing approaches focus primarily on transactional accuracy or control adherence (e.g., "Was the VAT percentage applied correctly?"), while neglecting broader process views such as sequence of activities, temporal dependencies, resource assignments, cross-system flows or variants in execution. From the perspective of process assurance (which asks whether business processes are executed as intended, end-to-end?) this focus is too narrow. In practice, a transaction may individually comply with rules yet still be part of a process deviation (e.g., goods received before PO approval), which raises operational or compliance risk [Duan et al., 2025, Föhr et al., 2025]. Many data-driven auditing setups fail to capture these deviations.

In view of these limitations, the role of process-oriented assurance becomes crucial. Integrating data-driven auditing with methods that can analyze not only transaction-level compliance but also full process flows, deviations from normative models, and multiple process perspectives (control-flow, resource, temporal, and data) provides richer diagnostic insight. For example, process mining and conformance checking enable internal audit functions to detect when an activity sequence diverges from its intended path, even if each individual step met compliance criteria [Duan et al., 2025]. This strengthens data-driven auditing by adding the assurance that business processes execute as intended, not merely that individual transactions comply.

3.5 Hindrances and Constraints in Data-Driven and Continuous Auditing

Despite the substantial benefits associated with data-driven and continuous auditing, organizations frequently encounter significant practical barriers when attempting to implement these approaches effectively. These challenges arise from organizational maturity constraints, technological limitations, financial considerations, system instability, and the inherent complexity of interpreting automated audit signals. A fundamental prerequisite for reliable continuous auditing is a sufficiently mature internal control environment. As emphasized by Rikhardsson and Dull [2016], organizations require well-documented processes, stable control structures, and consistent governance practices before automated monitoring can function effectively. Recent studies further confirm that continuous assurance depends on the availability of clean, integrated, and well-governed enterprise data, i.e., conditions that remain difficult to achieve in many organizations [Marcus et al., 2024, Barros and Marques, 2022]. Even when such structures exist, implementation depends heavily on robust IT and data infrastructures, including integrated information systems, high-quality master data, and scalable storage capable of handling event-level process information. Continuous auditing further requires automation frameworks that can execute rules, generate alerts, and integrate with ERP environments which are capabilities that many organizations lack or must invest heavily to develop [Chiu et al., 2014, Eulerich et al., 2025].

For small and mid-sized entities, these investments in technology, staffing, and system adaptation may represent a prohibitive financial burden. Complicating matters further, although investors increasingly recognize the value of continuous assurance, recent evidence suggests that they remain reluctant to commit the resources necessary to support full-scale adoption [Farkas and Murthy, 2014, Davidson et al., 2013].

Beyond structural and financial constraints, human and organizational factors also play a significant role. Audit teams must acquire new competencies in data management, analytics, and process interpretation; yet many internal audit functions continue to report skill shortages, insufficient analytical maturity, and resistance to adopting new technologies [Eulerich et al., 2025]. Organizational resistance, including cultural inertia, concerns about transparency, or limited executive sponsorship, may further hinder implementation efforts. Even when technological and organizational conditions are supportive, early deployments of continuous auditing frequently generate an overwhelming number of alerts. This “alarm flood”, documented extensively in earlier work [Alles et al., 2006b, 2008b] and confirmed in more recent digital auditing

studies [Tronto and Killingsworth, 2021], burdens audit teams with large volumes of false positives or contextually irrelevant deviations. Without advanced prioritization mechanisms, iterative rule refinement, and close collaboration with process owners, auditors may struggle to distinguish genuine anomalies from legitimate business exceptions. The resulting information overload can significantly reduce the effectiveness of monitoring supported by analytics.

Continuous auditing systems also require ongoing maintenance to remain effective. Changes to ERP configurations, process designs, workflow rules, or data structures can break monitoring algorithms or trigger spurious alerts, causing operational disruptions and eroding trust in automated controls [Barros and Marques, 2022, Marcus et al., 2024].

A deeper limitation, however, stems from the nature of business knowledge itself. Even with high-quality event data, not all information required to interpret deviations is explicitly documented or captured in systems. In many organizations, some business rules may be informal or only partially documented, and certain exceptional cases may rely heavily on tacit knowledge rather than written procedures. Local workarounds and context-specific variations often remain undocumented or not reflected in formal process models or process documentation, and some decision rationales or temporal constraints may similarly be left undocumented.

As a result, automated analytics may detect deviations, however auditors cannot always determine whether these indicate risk, routine flexibility, or legitimate business logic. Recent empirical and review studies suggest that data-driven auditing, although powerful, cannot by itself fully replace auditor judgment, because key control-relevant information is often undocumented or implicitly embedded in business practices [Föhr et al., 2025]. This gap between system-recorded data and implicit organizational knowledge presents one of the fundamental limits of continuous auditing, demonstrating that human insight remains essential for meaningful interpretation. Taken together, these challenges reveal that the transition to data-driven or continuous auditing is not purely technological. Successful implementation requires organizational readiness, analytical capability, stable infrastructures, and sustained commitment to system calibration. Moreover, certain aspects of process interpretation, particularly those relying on undocumented or tacit knowledge, cannot be fully automated. Without addressing these constraints holistically, the promise of continuous auditing may not be fully realized.

3.6 Process Mining

The field of process mining emerged around that same period of time as continuous auditing, and due to increasing the amount of business processes data stored in information systems. To date, in many organizations, it is possible to store and trace user activities from databases. This information can be extracted from information systems into *event log*. Starting from the event logs, process mining is defined as a discipline that develops techniques to gain insightful knowledge to discover, monitor, and improve real processes [Van der Aalst, 2016].

3.6.1 Event log

The basic assumption for process mining is that it is possible to have a set of recorded events that contains information such that "(i) each event refers to a task [...], (ii) each event refers to a case [...], and (iii) events are totally ordered." [Weijters and van der Aalst, 2001]. The event log is assumed to contain only data from one business process, such as Purchase-to-Pay or Order-to-Cash process. Each occurrence of activity is a unique event which refers to a *process instance*, or a *case*, as it is called in the process mining domain and this thesis. An event log includes more information than only transactional data; It covers *what* has been done *when* by *whom* and in relation to which process instance [Van der Aalst, 2016]. For example, in purchase to pay process, instead of only knowing that goods have been received, it is possible to retrieve when and how many times goods are delivered to which employee. Therefore, we could state that an event log minimally contains: a process instance identifier (case identifier), and timestamps which provide the ordering information on the events within a case.

Table 3.1 is an example of a simplified event log from the case study explained in Chapter 2.

In our case study, the event log contains 181,445 events. These events are performed by 272 distinct performers. Below in the following definitions adapted from [Van der Aalst, 2016], the concepts of an event, its attributes, a case, and an event log are formalized:

An *event log* consists of a set of *traces*. Each trace contains a sequence of *events* produced by one execution of a process. Each event in a trace refers to a task in the process, and all events are related via a total order induced by their timestamps. An event log is defined formally as follows [Van der Aalst, 2016]:

Definition 1 (Event Log and Trace)**.** *Let $\mathcal{E}$ be the universe of all possible events. An* event *$e \in \mathcal{E}$ is a tuple containing attributes such as* case identifier*,* activity*,* timestamp*, and possibly other information.*

Purch. Order	Event 1	Event 2	Event 3	Event 4	Event 5	Event 6	Event 7	...
1	Create PO	Sign	Release	Goods Receipt	Invoice Receipt	Pay		
2	Create PO	Sign	Release	Goods Receipt	Invoice Receipt	Pay		
3	Create PO	Release	Invoice Receipt	Pay				
4	Create PO	Sign	Release	Goods Receipt	Invoice Receipt	Pay	Pay	
5	Create PO	Release	Invoice Receipt	Pay				
6	Create PO	Sign	Release	Invoice Receipt	Pay			
7	Create PO	Sign	Invoice Receipt	Goods Receipt	Pay			
8	Create PO	Change Line	Sign	Release	Goods Receipt	Invoice Receipt	Pay	
9	Create PO	Sign	Release	Invoice Receipt	Goods Receipt	Pay		
10	Create PO	Sign	Release	Goods Receipt	Invoice Receipt	Pay		

Table 3.1: A selection of first 10 process executions of a procurement process

A trace σ *is a finite sequence of events, i.e.,* $\sigma = \langle e_1, e_2, \ldots, e_n \rangle$ *with* $e_i \in \mathcal{E}$ *for all* $1 \leq i \leq n$*, where events are ordered based on their timestamps.*

An event log L *is a multiset of traces, i.e.,* $L \in \mathbb{B}(\mathcal{E}^*)$*, where* $\mathcal{E}^*$ *denotes the set of all possible finite sequences over* $\mathcal{E}$*, and* $\mathbb{B}(X)$ *represents the bag (multiset) of elements from set* X*.*

An event log is a set of traces where multiple cases can have the same trace. An example of event log is $L_1 = \{[a, b, c, d, f, g]^{10}, [a, c, d, f, g]^5, [a, b, c, e, d, g]^5\}$, where a, b, c, d, f, and g are the events.

Starting from an event log, it is possible to have a better view of process traces. A trace can appear multiple times in an event log. That means there may be several traces that represent the same order of events. Therefore, we refer to a collection of identical traces in an event log as a *variant*. So, a trace is a unique sequence of activities that are executed to complete the process. In our example, in Table 3.1, a trace consists of all events that are mentioned on one row. Purchase orders 1, 2, and 10 share the same trace, just like purchase orders 3 and 5 do.

Note that this simplified event log contains six activities, but 56 unique occurrences of the activities (6 + 6 + 4 + 7 + 4 + 5 + 5 + 7+ 6 + 6), called events. This example log only uses the ordering of the events in time, making abstraction of other characteristics such as value, supplier, document number, etc.

Definition 2 (Events and Event Attributes)**.** *Let* $\mathcal{E}$ *be the universe of all possible events. Each event* $e \in \mathcal{E}$ *is characterized by a set of attributes. An event* e *is a tuple of the form:*

$$e = (c, a, t, \lambda)$$

where:

- c *is the* case identifier*, linking the event to a specific process instance (case).*
- a *is the* activity *associated with the event.*
- t *is the* timestamp*, indicating when the event occurred.*
- λ *is a set of additional attributes, such as the resource executing the activity, costs, lifecycle transitions, or other contextual information.*

In an auditing context, the events in an event log typically refer to transactions on documents. In order to structure transactional data into an event log, a timestamp for each event must be available, along with a link to a common case. On top of these minimal requirements, an event log may store additional information. This can refer to the resource (a particular person or system) that executed the activity or other

data elements, like the transaction value. Although this additional information is not part of the minimal process mining requirements from a technical point of view, they are most probably from an auditing point of view. Discovering real process executions in terms of event sequences, without details on involved persons and invoice numbers and transactions, for example, will not provide any substantial audit evidence.

3.6.2 Three types of process mining

Process mining techniques can be categorized into 'three types of process mining': process discovery, conformance, and enhancement [Van der Aalst, 2016].

3.6.2.1 Business Process Discovery

The first type of process mining, *process discovery*, starts from an event log and reveals the underlying process behavior in the form of a process model. This visual representation definitely has been the key to success for the process mining field. As it is depicted in Figure 3.1, process discovery is a method to extract the baseline (as-is view) of business processes based on historical data.

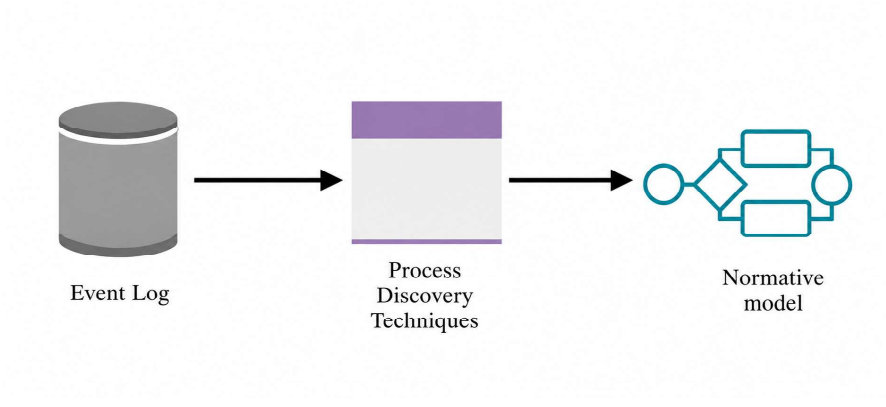


Figure 3.1: Process Discovery technique

However, to visualize real-life event logs in a representative and understandable map is not straightforward. Different algorithms have been developed to address this task. The α-algorithm of [van der Aalst et al., 2004a] is considered as the substantial start of process discovering algorithms [De Weerdt et al., 2012b]. This algorithm has been improved in the following years. Enormous improvement steps followed by some other process discovery algorithms such as the Heuristics Miner, the Fuzzy Miner, the Genetic Miner, Split Miner, Integer Linear Programming (ILP), and the Inductive Miner [Weijters et al., 2006a, Günther and Van Der Aalst, 2007a, De Medeiros et al.,

2007, Augusto et al., 2019, van Zelst et al., 2015, Leemans et al., 2013]. It is important to note that you might get different results depending on which algorithm you use.

In order to check the control-flow perspective of the financial statements, a standard model is needed. This model could be a normative model designed by the company, or it can be generated based on historical data. Where no predefined model exists, auditors need to get their understanding of each process by interviewing the process managers and observing employees. An appropriate understanding can be gained in audit data analytics by generating a model using process discovery techniques. For the tests of controls, auditors need to check a sample set of process controls against the organization's designed control measures to check whether they are implemented correctly. If the normative model does not exist, auditors have to analyze the (disorganized) documents and perform top-down structured interviews with employees to get a perception of the organization's processes based on process managers' expectations. However, this information might be biased or inaccurate because domain experts are mostly familiar with how in each special case activities should be executed but not how the process exactly works in general [Dumas et al., 2013]. Generating process models automatically based on historical data would help have an accurate, unbiased, and complete model representing the organization's executed behavior. The model can contain all control-flow, data, time and resource, or social network perspectives.

Process discovery in the process mining domain is a collection of tools and techniques that map business processes within an organization into a model with no prior information.

The alpha-algorithm [van der Aalst et al., 2004b], for instance, is getting an event log as input, can generate a Petri net model. The Alpha-algorithm is considered the substantial start of process discovering algorithms [De Weerdt et al., 2012a]. It scans the process traces in an event log, takes the sequence of activities into account, and visualizes the pattern it finds.

Heuristic miner [Weijters et al., 2006b] is another tool that creates a model. It is a robust tool that uses Causal Nets (C-Nets)[1].

The Heuristics miner is an improvement of the α-miner, which takes the frequency of process executions into account. Considering the frequency of process executions helps generate a model that filters out noise and infrequent behavior. The user can define a dependency threshold, then the model is constructed based on the activities'

[1]Causal Nets (C-Nets) is a modeling language introduced by van der Aalst [2011], Van Der Aalst et al. [2011] for process discovery. They claim that C-Nets are more representative than other design-oriented languages such as Petri nets, BPMN, BPEL, EPCs, YAWL, and UML activity diagrams for process discovery techniques

relations whose frequencies are higher than this predefined threshold.

Genetic process mining by Medeiros et al. [2007] is a tool that generates a random model based on an event log and gradually improves its quality in many iterations. The Genetic miner is also developed to tackle noise in the data and infrequent. This miner is designed in a way to mimic the process of evolution in biological systems. It applies genetic operators, iteratively, to find the fittest model to the log amongst all possible generated process models.

Fuzzy miner [Günther and Van Der Aalst, 2007b] is another process discovery tool that generates a model with meaningful abstractions of structured and unstructured processes. The Fuzzy miner is developed using the concept of a road map as a metaphor. Like road maps, the fuzzy process map is constructed in such a way that it 1) aggregates information that has lots of details, 2) makes abstraction of less important behavior, 3) puts a visual emphasis on information that may be most important, and 4) allows the user to customize the map. As a result, this miner is suitable to deal with complex, unstructured, and large event logs.

Inductive mining [Leemans et al., 2014] generates a model from a decomposed event log. Getting an event log as an input, it creates a process tree by repeatedly splitting the event log based on the likelihood of its events. Then the process trees can then be used to construct the process model using the divide and conquer technique. The process model is then generated based on the process tree. The Inductive miner is an improvement of the α-miner and Heuristics miner, which guarantees to generate a logical process model.

All the process discovery techniques mentioned above generate a model with a focus on control-flow, hence, they only consider the relation and order of the activities within cases. For auditing purposes, though, other perspectives such as data, time, and resource perspectives are also important. For instance, some activities have to happen after each other within a specific time, and exceeding the time is not acceptable (time perspective), or for testing the segregation of duties, the resources should be considered (resource perspective), or for specific activities, the amount of invoice or loan is needed to be analyzed as well (data perspective).

In both (traditional and automated) methods, after generating the model, it should be guaranteed that the model meets certain quality criteria. In process mining literature there are four quality dimensions that a sound generated model should have [van der Aalst, 2011]: fitness, simplicity, generalization, and appropriateness (precision) [Muñoz-Gama and Carmona, 2010, Munoz-Gama and Carmona, 2011, Adriansyah et al., 2012].

Fitness relates to the question of whether the observed process behavior complies with the control-flow described by the process model. In other words, it evaluates how

well the model is able to replay all the possible behavior in the event log completely. It is measured by computing the fraction of process instances that can be fitted on the process model [van der Aalst, 2011, van der Aalst et al., 2012, Rozinat and van der Aalst, 2008, Adriansyah et al., 2011].

Simplicity evaluates how complex/simple the process model is to be easily understood by the users.

Precision is the extent of the behavior allowed by the process model that is not observed in the event log (i.e., the model should not be under-fitting).

The generalization metric measures the ability of the process model to describe possible behavior which is not yet seen in the event log (i.e., the model should not be over-fitting). Generalization assesses to what extent the generated model can reproduce the behavior that was **not** in the event log but might occur in the future.

Although in the process mining context, it is said that a sound model has to have a good balance among these four dimensions, for discovering a process model for auditing purposes, the focus is just on fitness and precision. Also, note that one might get different results depending on which algorithm they use. The differences between the algorithms are related to different challenges. For example 'Which paths are important enough to visualize', 'To what extent do we need to generalize the observed behavior (captured in the event log) for possible future process executions?' and 'How to deal with noise (i.e., outliers and infrequent behavior that does not represent typical behavior)?'. These questions are addressed in the light of other, more technical constraints beyond the scope of this thesis. However, note that dealing with outliers and infrequent behavior is an important characteristic of the process discovery algorithms. The origin can be found in the primary goal of process discovery algorithms: representing the process as it is executed in an understandable manner. However, representing all observed behavior makes it very complex and challenging to understand. Therefore, process discovery algorithms often abstract the less frequent behavior and represent only the typical and mainstream behavior [Van der Aalst, 2016, Conforti et al., 2016]. However, as looking into the infrequent behavior plays an essential role for auditors to investigate the possibility of fraudulent or erroneous behavior, this might hinder its adoption in the later stages in the auditing practice.

3.6.2.2 Business process conformance checking

Another type of process mining, conformance checking, deals with comparing an event log with their associated normative process model (or with a set of business rules) [Rozinat and van der Aalst, 2008]. The original purpose is to check whether the real process (discovered from the event log) is aligned with the designed process or

whether a process model is a valid representation of reality. The former question stems from a compliance focus, whereas the latter concern stems from a quality check and evaluation of the applied process discovery algorithm. The output of these techniques is a set of complying cases (called normal cases in this thesis) and a set of deviating cases. Figure 3.2 illustrates the input and output of these techniques.

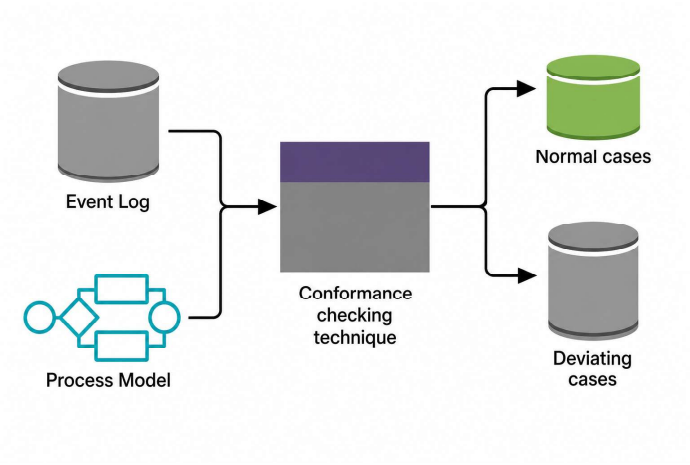


Figure 3.2: Conformance checking technique

The procedural process model is often a graphical process map where all accepted behavior is explicitly modeled. The goal is to find the degree of conformity between process executions and a model and to detect the differences between them (e.g., De Weerdt et al. [2012a] and Mendling et al. [2007]). Conformance checking results can also be used to improve or repair the process models in the enhancement phase (e.g., Fahland and van der Aalst [2015]). However, it can also discover the process executions that do not conform to the designed model, i.e., deviating cases. For auditing and internal control purposes, only this latter application of conformance checking techniques is relevant. Different dimensions are used to qualify whether the model and real behavior are aligned: fitness, precision, generalization, and simplicity. Given the purpose of this thesis, however, only the two mostly used dimensions will be addressed here: fitness and precision. The fitness dimension expresses how much of the observed behavior (i.e., data in the event log) is captured by -or fits- the process model. This dimension is therefore highly related to compliance. For example, does the trace $< CreatePO, Sign, Release, GoodsReceipt, InvoiceReceipt, Pay >$ fit the procurement model in Figure 3.3. If all traces fit the model, fitness is said to be 100%, and all transactions comply with the model. To check the alignment between both observations in the log and the model, also the opposite side of conformance might be

taken into account: how precise is the model? In other words: how much additional behavior that is not present in the event log does the model allow for? Consider as an example a process model that only specifies the tasks that need to be executed but does not articulate any fixed order between them.

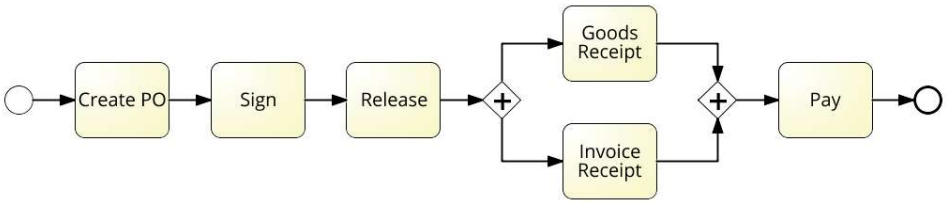


Figure 3.3: Normative model for the purchase to pay process in BPMN format

As mentioned before, whether or not a model is precise is mainly of interest when checking the output quality of a discovery algorithm. Therefore, the precision dimension starts from the log and then checks upon the model. The precision of a model is said to be 100% if all possible paths, according to the model, are actually observed. If this is the case, there exists at least one trace in the event log for every possible execution according to the model. In contrast, the precision of a model is said to be low if it allows much more behavior than observed in the event log.

For the purpose of auditing and internal control testing, conformance checking is mainly approached from the fitness perspective and not from the precision perspective.

The algorithms that check the fitness of logs with regard to models can be grouped in two main approaches: One is the replay-based approach ([Rozinat and van der Aalst, 2008]), and the other is the alignment-based approach (e.g., Adriansyah et al. [2012] and de Leoni and van der Aalst [2013b]). The replay-based approach replays each process execution (trace) individually on the (Petri-net) process model to check its conformance. The technique is called token-based because while process instances are replayed on the Petri-net, the Petri-net places get marked by tokens. Whenever the process execution deviates from the model, the algorithm keeps track of this and calculates a fitness measure by counting the number of *produced*, *consumed*, *missing* and *remaining* tokens. The fitness measure from each trace is calculated, and then a weighted mean is provided as a general fitness measure. This overall number indicates how well the behavior in the event log can be replayed on the process model. In our running example, considering the model in Figure 2.2a and the event log in Table 3.1, for process execution 1, 2, 4, 9, and 10, the fitness is 1 (i.e., 100%) because they conform to the model. However, for process execution 3 and 5, the fitness will be 0.67 (67%) because two out of the six required events (i.e., 'Sign' and 'Goods Receipt')

are missing. Aside from giving one general fitness measure, also a general overview of how many times and at what places the log deviates from the model is presented in the replay-based approach. So the output is on the level of the process model, with aggregated information on the mismatches between model and executions. An example output could be: "The activity 'Goods Receipt' has been executed 402 times, while, according to the prescribed model, this was not allowed at that phase of the process execution".

The alignment-based approach is a more robust approach than the replay-based approach [Adriansyah et al., 2012]. This approach also relies on a comparison of individual process executions with the model, but it first narrows down 'the model' to 'the path of the model that is closest to the process execution.' In the alignment-based approach, for each trace that exists in the event log (T_l), one possible trace in the model (T_m) is selected: The trace that is closest to trace T_l in the log.

The alignment starts from the beginning of both traces, and step by step compares the activities in trace T_l with the activities in trace T_m. The output of this approach states for each process execution, whether it contains extra or missing events, compared to the model. The technique investigates whether there is a mismatch between these two traces. Ideally, there should be no mismatch between trace T_l and trace T_m. In this case, we say both traces fit each other completely, and trace T_l is considered as a normal (i.e., compliant) trace.
If there is an extra event in trace T_l (and not in the model trace T_m), it means an unexpected event has been executed, or as it is called, an event is *inserted*.
If there is an extra event in trace T_m (and not in the log trace T_l), it indicates that an activity that should have been executed did not take place or it is *skipped*.

For the comparison between a process model and an event log, first of all, the model's closest path to the process execution is considered. Linking the recorded process execution with this specific trace of the model is called the optimal alignment. Therefore, the optimal alignment is the comparison of two traces, process execution and its closest path in the model, with each other. For example, consider the normative model in Figure 3.4 as the reference model. There are two possibilities to trace this model, which are provided in the figure as well. Now take the following trace:
$< A \rightarrow B \rightarrow E \rightarrow G \rightarrow F \rightarrow H >$.

The closest path to this trace in the model is *ABCEGFH*. Next, the activities from both traces are compared with each other based on their sequence, activity by activity as explained before.

So this approach yields detailed output (on the level of the process execution), as opposed to the "replay-based approach" that yields aggregated output (on the level

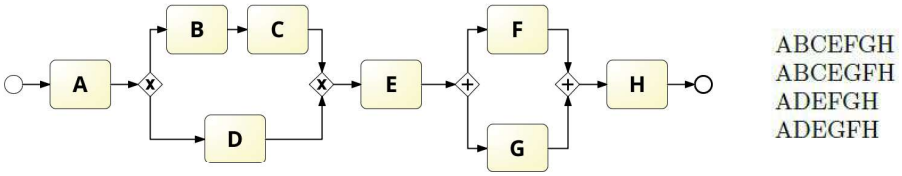


Figure 3.4: An example of a business process model in BPMN language and all of its possible paths

of the model).

The conformance technique proposed by García-Bañuelos et al. [2018] uses event structures [Nielsen et al., 1981] to pinpoint deviations and interpret them as a set of natural language statements. The conformance of the model or log against each other is checked by comparing two event structures that one is created by folding the event log and the other by unfolding the process model. The compression is performed via a partially synchronized product (PSP), and if there is a difference between event structures, it will be captured in a node of the PSP.

All these approaches focus on the name and order of process events and ignore other attributes. However, for auditing purposes, not only the control-flow but also other perspectives such as data- and resource perspectives have significant value. de Leoni and van der Aalst [2013b] propose a process mining technique to consider data associated with the cases. They first compute the alignments between the model and the log and then discover the data-flow using decision-tree learning algorithms read and write and guard functions.

Where the previously mentioned conformance checking approaches compare a log with a model, the conformance of a log can also be tested against a set of rules. If the process at hand is very flexible or no procedural model is present, a set of predefined rules can be used to check compliance. In the context of internal control testing, there are often plenty of business rules in place. An example of a rule can be 'The value of the purchase order must equal the value of the invoice,' or 'If the value of a purchase is below 1000$, no signature is required'. The work of van der Aalst et al. [2005] introduced the first 'rule tester' to check event logs against the rules: the LTL Checker, referring to the Linear Temporal Logic that the rules need to be formulated in. Other interesting work on this topic is the work of Caron et al. [2013] and Ramezani et al. [2012]. A last relevant dimension when discussing conformance checking is the applied process mining perspective. In general, four perspectives are specified: the control-flow, the time, the organizational, and the case perspective

[Van der Aalst, 2016]. When a control-flow perspective is applied, the focus is on the order of the activities. For example: is an invoice first booked before it is paid? When a time perspective is applied, the focus is on time and duration related aspects. For example, how long does it take to pay an invoice after being booked on average? When an organizational perspective is applied, the focus is on the persons, roles, functions, or departments that execute the different activities. For example, which roles are involved in booking invoices, and are they expected to be involved? At last, when a case perspective is applied, the focus is on the remaining characteristics of a case. For example, which suppliers are involved in cases related to purchase orders on material number x? In the light of conformance checking, the techniques mentioned above that compare a log with a model apply a control-flow perspective. This implies that a case conforms to the model if it follows the prescribed order of activities, making abstraction of all other characteristics of the case. The conformance checking techniques that compare a log against the rules mostly also include the other perspectives.

Recent advancements have introduced novel techniques that extend beyond these traditional approaches. For example, Both Goulart Rocha et al. [2024] and Mehr et al. [2023] contribute to advancing conformance checking by moving beyond mere deviation detection toward a more insight-driven approach. Goulart Rocha et al. [2024] employ pattern mining techniques to extract recurring behavioral structures from event logs, enabling the classification of deviations and facilitating root cause analysis. In the same context, Mehr et al. [2023] incorporate explainable AI (XAI) techniques and pattern recognition to identify and explain patterns of deviations in process executions. Instead of simply detecting misalignments, the approach analyzes behavioral patterns to understand why deviations occur.

In addition, Peeperkorn et al. [2023] proposes global conformance checking measures by combining shallow representations and deep learning techniques to assess the alignment between event logs and process models. The approach enhances conformance analysis by leveraging machine learning to capture complex patterns and deviations more effectively. They introduce two data-driven conformance checking techniques: shallow representation learning using Word2Vec to compare process trace embeddings and deep learning with recurrent neural network (RNN) (more specifically a long short-term memory (LSTM) networks) to model sequential behaviors and detect deviations. These methods assess process alignment without requiring explicit process models. The techniques were validated on real-world and synthetic datasets, offering an alternative to traditional conformance checking approaches. In another research, Burigana et al. [2024] propose a “glocal” conformance checking framework that evaluates alignment in multi-agent systems without a central global

process model. Their approach integrates local views represented as Data Petri Nets and leverages an alignment-based method to ensure global coherence. Their techniques involve computing local alignments for each agent's perspective and ensuring their compatibility on a global scale. To address the complexity of this task, they introduce regular expression-based cost functions that consider the context of activities within the global trace. In their work, Polyvyanyy and Kalenkova [2022] introduce an entropy-based approach for a more flexible conformance checking of processes that only partially match, meaning the processes which share similarities but are not entirely identical. It leverages entropy measures to quantify deviations, providing a more flexible and informative analysis of process alignment.

Several conformance checking techniques incorporate multiple process perspectives beyond just the control-flow. For example, Zhang et al. [2022b] present a fuzzy multi-perspective approach to conformance checking, such as control-flow, data, and resources, allowing for flexible alignment between event logs and process models. Their technique leverages fuzzy logic to handle uncertainty and partial compliance, allowing deviations to be assessed on a continuum rather than as strict matches or mismatches. Moreover, Felli et al. [2021] introduce a Satisfiability Modulo Theories (SMT)-based approach for conformance checking of multi-perspective processes, considering control-flow, data, and resources. By encoding process execution constraints into SMT formulas, it efficiently detects deviations between event logs and process models. This technique enhances conformance checking by providing a precise and scalable solution for handling complex process dependencies. Building on this, the authors extend their work in [Felli et al., 2023] by addressing uncertainty in control-flow, data, and resources using the same SMT-based approach. Their method encodes uncertainty as logical constraints and uses an SMT solver to determine compliance efficiently.

Regarding the uncertainty in event data, i.e., when event logs contain uncertain, missing or imprecise data, Pegoraro et al. [2021] introduce a probabilistic framework that integrates uncertainty into the conformance analysis. Instead of assuming fixed event data, it models uncertainty using Dempster-Shafer theory and three way decision rule, to assign belief values to alignments. This enables partial compliance scoring, making conformance checking more robust and adaptable to real-world, uncertain event data.

3.6.2.3 Process enhancement

Another type of process mining, process enhancement, is concerned with improving an existing model, using the information from the actual behavior or predicting future behavior, and providing some recommendations based on the past experiences

[Van der Aalst, 2016]. When a violation against the normative process model is detected, the process owner might decide that the violation is typical or expected in the real-life, therefore an improvement in the business model is needed. Figure 3.5 depicts the process enhancement procedure. Moreover, some predictions can also be made on the completion time or the attributes of an activity or a sub-process by using a comparison between a current situation and the historical log. Another operational application of process mining is the recommendation. Process mining can assist the user by suggesting the next activity that should be performed based on its frequency in previous, its occurrence, or based on minimizing the flow time of the process or its related costs [Van der Aalst, 2016]. Since it lends itself more to making recommendations in the area of operational efficiency and less to providing assurance, this type is considered outside the scope of this thesis [Jans and Hosseinpour, 2019].

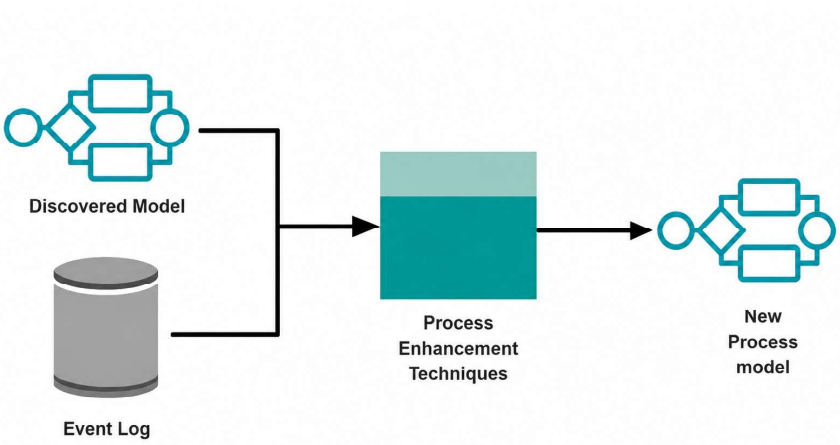


Figure 3.5: Process Enhancement technique

3.7 Process Mining in Auditing

Although process mining research has gained substantial momentum within the Business Process Management (BPM) community, it has received comparatively limited attention in accounting and auditing research. Existing contributions are largely confined to a small set of studies [Huang et al., 2009, van der Aalst et al., 2010, Jans et al., 2011b, 2013, 2014, Werner, 2017, Jans and Hosseinpour, 2019, Chiu and Jans, 2019, Werner and Gehrke, 2015, Föhr et al., 2025, Duan et al., 2025, Bäßler and Eu-

lerich, 2022], of which only a few have appeared in accounting journals. This relative scarcity stands in contrast to the growing interest and efforts observed in practice and among standard-setting and professional bodies.

On the practice and policy side, the Rutgers AICPA Data Analytics Research (RADAR) initiative has devoted considerable research effort to the use of process mining in auditing applications [AICPA, 2015]. At the same time, researchers from the BPM and information systems domains increasingly address accounting and auditing challenges from a process-oriented perspective (e.g., [Schultz, 2013, Sonnenberg and Brocke, 2014]), further underlining the perceived potential of process mining as an audit analytic.

More recently, several studies have begun to explore the systematic integration of process mining into audit and assurance tasks. Föhr et al. [Föhr et al., 2025] propose an Audit Process Mining (APM) framework that structures how process mining can be embedded into different phases of the audit, highlighting both opportunities for enhanced assurance and practical barriers to adoption. In parallel, research on advanced internal control evaluation has combined process mining with machine learning techniques to assess control design and operation across large event populations, illustrating how process-oriented analytics can extend traditional control testing and exception analysis [Duan et al., 2025]. From a complementary perspective, Baessler et al. [Bäßler and Eulerich, 2022] demonstrate how predictive process monitoring can support internal audit by forecasting risk-relevant outcomes (e.g., payment punctuality), thereby moving from retrospective anomaly detection toward forward-looking risk assessment. Broader survey evidence on technology usage in internal auditing further indicates that internal audit functions are gradually increasing their reliance on data analytics and process-oriented tools, although maturity levels and implementation depth still vary considerably across organizations [Eulerich et al., 2025].

Despite these developments, several limitations must be addressed before process mining techniques can be fully integrated into mainstream auditing practice. From a technical perspective, process mining algorithms continue to face challenges such as handling noisy and incomplete event data, coping with model complexity and variant explosion, and managing the often large numbers of detected deviations in a way that remains interpretable and actionable for auditors. From an organizational and methodological perspective, open questions persist regarding how to align process mining outcomes with audit objectives, how to integrate process-level findings into established audit methodologies, and how to combine automated detection with auditor judgment and domain knowledge. These shortcomings, and their implications for process-level assurance, are further discussed and addressed in the subsequent chapters of this thesis.

In summary, this chapter has positioned continuous internal auditing and process mining as complementary developments in the evolution of data-driven assurance. However, existing approaches still face important limitations in auditing contexts, particularly regarding the interpretation of deviations, the handling of large numbers of alerts, and the integration of auditor judgment and implicit business knowledge. These limitations motivate the focus of this thesis on the classification and interpretation of control-flow deviations in process executions, as developed in the following chapters.

Chapter 4

Process Deviation Categories*

Previous chapters explained the need to complement traditional auditing with continuous auditing. A couple of data analytic methods have been proposed so far to obtain the features of continuous auditing (e.g., Kim and Vasarhelyi [2012], Alles et al. [2006b], Issa [2013], Perols and Murthy [2012b], Jans et al. [2014]). Conformance checking techniques are one type of suggested techniques among these methods [Jans et al., 2013, 2014]. They are able to discover deviations in the underlying business process execution from the control-flow perspective. For instance, conformance checking techniques can detect when a change has happened in a procurement execution or when an approval is skipped.

However, one of the key problems of these techniques is the immense number of deviations detected by them, which makes the follow-up cumbersome for auditors. This can be attributed to the complexity of the information systems (e.g., ERP systems) [Alles et al., 2008b], or the dynamic flexibility required by process users. The other problem is that the identified deviations by these techniques are presented in a non-meaningful and difficult language for human users, i.e., in a too technical or on a too detailed level format.

One way to address these problems is to map the identified deviations into a set of deviation categories. Mapping the deviations into certain categories gives better insight into the occurred types of deviations and their frequency on a high level. Starting from this overview, the auditors can drill down in a targeted way. Of course, there are some deviation patterns in the literature, but they are all originated from theoretical approaches and therefore, by definition not aligned with the way how auditors classify deviations.

*This chapter is based on a study previously published as the journal article "Auditors' Categorization of Process Deviations" in Journal of Information Systems (JIS).

From the algorithm engineering perspective adopted in this thesis, this chapter contributes to knowledge about the task by investigating how auditors conceptualize and interpret process deviations. These insights provide the empirical basis for the deviation categories used in the classification framework developed in Chapter 5.
This chapter explains the study that is performed to discover whether there is a set of deviation types for the interpretation of all possible deviations in a way as close as possible to the mental model of auditors. As it is explained before this study is limited to the control-flow perspective. Therefore, the deviating sequences in process executions were analyzed independently of data and resource information, such as transaction values or the identity of the executing resource. The goal of this chapter is to facilitate future research in process deviation analysis by providing a set of deviation types that are aligned with how auditors perceive deviating process executions. This chapter answers the following research question:

RQ: Which process deviation categories, used in the field of business process management, are applicable in an auditing context from a control-flow perspective?

This research question is firstly split into two secondary research questions:

SRQ1: Which deviation types do exist in the business process management literature?

SRQ2: To what extent are the deviation types from literature applicable as deviation categories by auditors?

To answer the first secondary research question, a literature review in the field of business process management and process mining was conducted and for the second research question, a field research was conducted to map the process deviation categories from literature to deviation categories that are applicable by auditors.

4.1 Literature Review

The first type of considerable mismatch is the *existence* issue, i.e., an activity is not present in the process execution (while it was expected in the model), or an activity is executed while this was not prescribed in the process model. This existence mismatch is comparable with the InDel (Insertion, Deletion) mismatch of the Needleman–Wunsch algorithm [Needleman and Wunsch, 1970] in sequence alignments

in bioinformatics. The second type is the *sequence* or *order* issue, meaning that some of the activities are re-positioned and are not in the same order as in the reference trace.

These two intuitive mismatch types are at an abstract level, and perhaps do not contain meaningful information for auditors. To find suitable deviation categories a literature study is conducted in process mining and business process management domain. This section reports on a literature study to answer the first secondary research question (SRQ 1).

4.1.0.1 Process deviation patterns in conformance checking literature

In Section 3.6.2.2 a background on conformance checking techniques is given. This section provides more detail on the application of conformance checking to identify deviating cases. Here, only the techniques which can present deviations, their benefits, and shortcomings are discussed.

The alignment-based technique by Adriansyah et al. is discussed before. This technique compares each process execution in the event log with its closest match in the process model. The activities from both traces are compared with each other based on their order, activity by activity as trace in the model. That means the frequency of the traces is not considered in finding the optimal alignment. This technique also fails to deal with large and noisy event logs [Reißner et al., 2017].

García-Bañuelos et al. [2018] verbalized the detected deviations in a set of natural language statements. They propose nine deviations patterns, i.e., *immediate causality-concurrency, immediate concurrency-conflict, task skipping, unmatched repetition, task substitution, task relocation, task absence/insertion and unobserved acyclic interval, unobserved cyclic interval.* To date, their approach is the only technique that provides deviation patterns at a higher level in natural language. Some of these patterns, potentially, might give auditors an insight into what kind of deviations exist in a set of transactions. Nevertheless, their approach is not completely suitable for auditing purposes. Firstly, the approach creates an 'event structure' from the event log and compares it with an 'event structure' created from the normative model. Consequently, the process execution and traces in the event log cannot be observed anymore as standalone sequences, but the event log is seen as an abstraction. Therefore, finding the deviating process executions or traces, or even the frequency of each deviation type is not possible.

Among the nine suggested patterns, five of them might be in accordance with auditing purposes: *task skipping*, *unmatched repetition*, *task substitution*, *task relocation*, *task absence/ insertion* since these patterns also compare real executions with

the prescribed model. The remaining patterns are not suitable for auditing purposes since they relate to characteristics of causality and parallelism in the model. Causality and parallelism are not observable in the sequential traces which auditors have to evaluate, hence they are not suitable for auditing purposes. Causality and parallelism are often proposed for additional model behavior that are not observed in the event log, while auditing practice is based on the historical data, i.e., the event log.

4.1.0.2 Process deviation patterns in business process management literature

Beyond the conformance checking techniques, in process management literature, Weber et al. [2008] proposed a comprehensive set of change patterns for comparing business processes. The authors suggested 18 'change patterns' from a control-flow perspective[1]. Among these change patterns, only six patterns seem relevant for the observable deviations in event log traces. These are: *insert process fragment*, *delete process fragment*, *replace process fragment*, *swap process fragment*, *copy process fragment*, *embed process fragment in loop*. Other change patterns such as *parallelize activities*, *embed process fragment in conditional branch*, cannot be observed in transaction sequences. The reason is that traces of process executions in an event log are always in a sequential format. Therefore, the parallel activities or conditional branches cannot be observed there. The rest of their proposed patterns are not suitable either for comparison of sequences with a model, because in sequential traces, those patterns are observable as other change patterns such as *swap process fragment* or *delete process fragment*. The pattern *move process fragment* sounds like a combination of insert and delete process fragment patterns. However, it cannot be considered either, because of its definition in their approach. Move process fragment is the relocation of a task to somewhere further in the trace, not sequentially, but the moved task is embedded in a branch which requires the application of parallelism, which does not exist in an event log.

Regardless of the different terminology that is used in the three studies, there is a general overlap in meaning. Table 4.1 presents an overview of all relevant deviation categories as suggested by Adriansyah et al. [2010], García-Bañuelos et al. [2018], Weber et al. [2008], and how they relate to each other. Most relations are straightforward, except for difference in the ways that Adriansyah et al. [2010] and Weber

[1]The 18 change patterns are *insert process fragment*, *delete process fragment*, *replace process fragment*, *swap process fragment*, *copy process fragment*, *move process fragment*, *extract sub-process*, *in line sub-process*, *embed an existing fragment in a loop*, *parallelize a process fragment*, *embed an existing process fragment in a conditional branch*, *add control dependency*, *remove control dependencies*, *update transition conditions*, *late selection of fragments*, *late modeling of fragments*, *late composition of fragments*, and *multi instance activity*

Mismatch patterns	Patterns	Deviation Patterns by Adriansyah et al. [2010]	Mismatches by García-Bañuelos et al. [2018]	Change Patterns by Weber et al. [2008]
Existence	skip an activity	skip	task skipping	delete process fragment
	insertion an extra activity	insert	task absence/ insertion	insert process fragment
	replacement two activities	replacement	task substitution	replace process fragment
Sequence	swapping two activities	swapping	task relocation	swap process fragment
	repetition an activity	insert	unmatched repetition	copy process fragment
	loop on an activity	repetition	–	embed process fragment in loop

Table 4.1: Comparison of the deviation categories proposed in the literature

et al. [2008] use 'insert' which requires some attention. The latter makes a distinction between inserting and copying a process fragment, where the former categorizes both these two patterns as 'insert'. García-Bañuelos et al. [2018] is more in line with Weber et al. [2008], also making a distinction. Therefore, we follow this more detailed split-up in inserting and repeating an activity. It is important to keep this design choice in mind when analyzing our data later on. The first column in Table 4.1, Categories, is the overarching description that we will use throughout this chapter when referring to these patterns.

4.1.1 Process deviation patterns in literature

The previous subsections provided the categories from both conformance checking and business process management literature. Table 4.1 summarizes the relevant literature on the process deviation that was described above. The table contains the patterns suggested by Adriansyah et al. [2010], García-Bañuelos et al. [2018], Weber et al. [2008] each with their counterparts. For each deviation pattern, an example of a deviating case is provided by considering $< ABCEFGH >$ which is one of the model paths illustrated in Figure 3.4. The generic deviation patterns are described in the first column. The second column shows the terms used in this thesis referring to the different deviation patterns. As illustrated in the table, the possible process deviations from literature for existence mismatch are *skip of an activity*, *insertion of an activity* and *replacement* of two activities which can be seen as the combination of skipping of an activity and inserting of another activity. For the sequence or order mismatch, the possible process deviations are *swapping*, *repetition* and *loop* of activities. In this chapter, these six deviation types are considered as the hypothesis, based on the existing theory.

4.2 Methodology and research design

The literature study provides us with different *deviation patterns*[2] that can be identified when comparing a set of transactions with a model. These patterns are stemming from computer science and business process management theory and were never tested on their alignment with the auditing profession. For auditors to be able to use conformance checking techniques in their profession, the alignment of the deviation categories with their objective is crucial. This section explains the methodology which is used to investigate to what extent the *deviation patterns* identified in Section 4.1

[2]To the end of this chapter this font is used for deviations patterns stemming from literature to help the reader to distinguish them easily from the categories discovered from field study

can be applied by auditors. Therefore, one first needs to gain insight into how auditors interpret deviations when being confronted with deviations. Starting from these interpretations potential alignments with existing *deviation patterns* can be identified. For this purpose, a field research is performed. Fourteen auditors were interviewed to obtain their interpretations of deviations from a control-flow perspective. The following subsections explain the methodology of this research and the instrument design.

4.2.1 Field research methodology

Starting from the literature review, a set of process *deviation patterns* is proposed as it is identified in the literature and illustrated in Table 4.1. These are the general process deviation categories and are considered as the assumption for this research.

The proposition is that this set of *deviation patterns* is applicable in the auditing domain. To test this proposition, a field research is executed. An inductive (bottom-up) construction of a theory [Wilson, 2014] is conducted in this research, the same as almost all the other field research performed in the auditing domain [Malsch and Salterio, 2015]. Gathering the deviation types interpreted by auditors is an objective, independent, and value-free investigation. These are the characteristics of a positivism research paradigm [Johannesson and Perjons, 2014]. Note that, although in this research the interpretation of auditors of different deviations is considered, the nature of these interpretations per se is objective (not subjective), and it is independent of the human meaning. The auditors' interpretation of the control-flow deviations are based on their knowledge and experience. We postulate that there is a reasoning behind each of their interpretations, although not yet formalized in the auditing literature.

For the field research 'interviews' are applied as the data collection method. By asking questions, we tried to discover how auditors investigate and assess deviations, and to what extent their interpretation is similar or in line with the process *deviation patterns* discovered from the literature.

We applied the Gioia method [Gioia et al., 2013] to analyze qualitative data and develop deviating categories. Gioia's method combines two approaches: a systematic inductive approach to develop the concepts from data and an abduction approach with the input from researchers and existing literature [Gioia et al., 2013]. In the first stage or first-order analysis, the analysis of the gathered data and codes is done. In this stage we used induction and extract information terms from the participants' terms and wording. In the second stage, or the second-order analysis, the concepts are aggregated to develop the themes. The second-order analysis is where the abduction is used and researchers' knowledge and existing literature is taken into account. This method is explained later in this chapter.

4.2.1.1 Theoretical sampling

Theoretical sampling in qualitative research involves collecting data in the field while checking the emerging theory against reality [Willig, 2013]. In this method firstly some data is collected from the field and it is analyzed and coded by the researcher. According to the result of coding, more data will be selected from the field and analyzed. These back and forth steps between collecting and analyzing data continues until data saturation is achieved.

In the field research that so far performed in the auditing domain, a number between 15 and 30 interviewees had been selected by researchers [Malsch and Salterio, 2015]. For this research study, we targeted 20 auditors in our sampling plan. To get to this number, the interview request has been sent to over 40 auditors in Belgium and The Netherlands. However, only 14 could be reached for the interview. All the 14 auditors were interviewed. Nine were employed at Big four auditing firms and five were employed as internal auditors at multinational companies. This resulted in 12 individual interviews. We considered the data saturation criterion, as suggested by Malsch and Salterio [2015].

Saturation in data collection occurs when the result of a sequence of interviews is similar, which means further data collection does not provide new results. Reaching the saturation, we stopped as soon as we observed the same list of interpretations for deviations. In many field research studies in the auditing domain, the saturation is reached before the end of the sampling plan [Malsch and Salterio, 2015]. If this would not be the case in our study, we planned to apply the snowball approach.

In the snowball approach, each interviewee introduces another person who can provide the researcher with more information on the topic [Miles and Huberman, 1994]. However, it turned out this was not necessary for our study.

Table 4.2 summarizes the sample composition and interview statistics in our research. The focus for this field research was on managers and senior managers with at least five years of experience. The reason is that those below this level, may not have enough experience, and those who are at a higher level, may have not executed the deviation detection task for the last years, and they might have forgotten the complete picture [Malsch and Salterio, 2015].
Each interview was designed to last approximately one hour, however, the interviews varied in length from 0:51' to 1:44'.
The interviews were conducted over a 12-week period from April 2017 to June 2017. The period is chosen to be after the busy period of yearly auditing.

Most of the interviews were conducted by myself in the offices of the participants,

four interviews were conducted at an office in the university and one interview was conducted in a hotel lobby. All interviews are recorded and transcribed. The transcriptions have been done by the interviewing researcher.

		Number of auditors	Median years of experience	Mean interview time
External auditors	Manager	4	10	1:05'
	Senior manager	3		
	Director	1		
	Partner	1		
Internal auditors	-	5	9	1:30'

Table 4.2: Interviews statistics

We have chosen to perform a semi-structured interview, since additional questions were asked by the interviewer to probe further the line of responses, where appropriate.

Prior to each interview, a part of the instrument, together with the objectives of the interview, was distributed to the auditors via email. At the outset of the interview, the researcher introduced herself and her research, the study and briefly reiterated its objectives, including the exclusive focus on the control-flow perspective. The interviewees were informed that the study was based on their professional experience and there are no predefined correct or incorrect answers. They were further assured that all information provided would be used solely for academic purposes and their personal information and data will be handled confidentially. The interviewees were invited to indicate whether any part of the pre-distributed material was unclear or confusing; additional clarification and explanation was provided where necessary. Subsequently, the structure of the interview was explained, and consent for audio recording was obtained.

General questions such as the level of expertise of the auditors and their experience with data analytic techniques (e.g., process mining) were asked at the beginning of the interview.

4.2.1.2 Theoretical coding

Theoretical coding is the process of identifying categories from the collected data. The coding process starts with low-level coding which is based on the frequency of observations in the research. Subsequently, the high-level categories will be systematically created by integrating the low-level categories.

In this research, the interviews are firstly transcribed using the *Otranscribe* tool[3]. The coding and analysis of interview data is performed using the latest version of *Nvivo*[4] (Nvivo 11) a data analysis software.

The interview data is analyzed in two phases. In the first-order analysis, all data from the interviews are coded. This results in a high number of information terms. In the second-order analysis, similarities and differences between the first-order codes are considered, which leads to compressing the codes into 'themes' [Gioia et al., 2013]. In the first cycle of data analysis *In Vivo coding*[5] is applied. In the In Vivo coding, codes refer to the actual phrases which are used by the interviewees themselves [Strauss, 1987]. This helped to find the terms and concepts used by auditors, to represent the meanings inherent in their daily practice [Stringer, 2014]. The In vivo coding also helps to avoid inserting existing theory into the analysis [Willig, 2013]. To reorganize the first-order codes into a selected list of categories, code mapping is applied as a transition between the first and the second coding phase [Saldaña, 2016]. In the second-order, conceptual coding is used to categorize the interviewees' interpretation terms into a higher dimension, called themes.

4.2.1.3 Constant comparative method

Constant comparative method together with theoretical sampling constitute the core of qualitative analysis [Boeije, 2002]. The constant comparative method is used for analyzing the data gathered. The goal of constant comparative method is to compare new data with emerging theory and to identify its similarity or difference with the other categories. Each new observation can challenge, refine, or validate the identified categories. The goal of constant comparative analysis is to ensure that all possible categories are captured. For this purpose, the data should be gathered and the comparison should continue until no new categories can be identified.

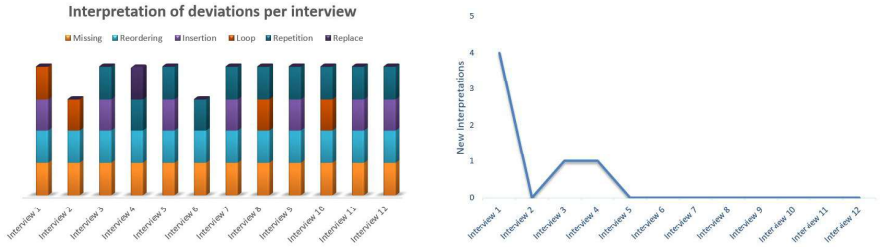
4.2.1.4 Theoretical saturation

In the qualitative research method, the process of data collection and data analysis continues until the theoretical data saturation occurs. The data saturation occurs when more data from the sample will not lead to more information regards to the research question [Seale, 1999]. In this research for generating the results, we made

[3]http://otranscribe.com/

[4]http://www.qsrinternational.com/nvivo/nvivo-products

[5]In Vivo coding is also known as literal coding, natural coding, inductive coding, verbatim coding, indigenous coding [Saldaña, 2016]. In the In vivo coding the label, which is word or phrase is taken from the data itself. In other word, the participant's own language is chosen as the label.



(a) Deviation categories mentioned in each interview

(b) New interpretations discovered in each interview

Figure 4.1: Data saturation is reached after the fourth interview since afterwards no more new concept has been discovered.

sure that our sample is large enough to reach theoretical data saturation. Analyzing the interview data shows that the data saturation is reached from the fifth interview since no new category (theme) is discovered afterwards. However, we have collected and investigated more data to make sure that no more category can emerge from the field. Figure 4.1 illustrates whether and how many new interpretations were discussed in the interviews. It is shown that from the fifth interview on, no new deviation category is mentioned by the interviewees.

4.2.2 Instrument Design

In the designing phase of the research, the purpose of the interview, and the research question was transformed into interview questions, to reach suitable responses. To facilitate the understanding of the interview questions, a purchase-to-pay process model and a set of procurement transaction instances are designed. The interview structure comprises two parts. In part 1, a general and open-ended question is designed to discuss possible deviations that auditors can predict such a situation might happen in a process execution, or have experienced it before. In part 2, the interviewee is confronted with the model and examples. To this end, 18 process execution examples have been designed based on the process *deviation patterns* discovered in the literature summarized in Table 4.1. This section firstly explains how the model and the set of deviating process executions are designed. Subsequently, the two parts of the interview instrument are discussed. Both parts of the instrument were used in all the interviews.

4.2.2.1 Reference model design

We have chosen the purchase-to-pay (also known as P2P, procurement, and procure-to-pay) process as the process under investigation. The purchase-to-pay (P2P) process is the well-known process within organizations' processes and auditors are familiar with it. As notational language, BPMN (rather than Petri net notation) is chosen because of its simplicity and clarity for understanding.

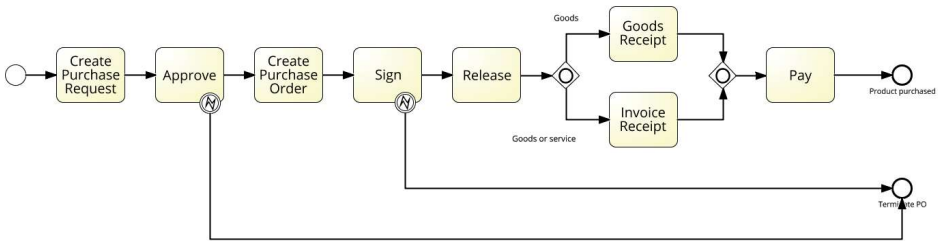
Although the P2P process design is highly dependent on the companies processes and objectives, we have designed three simple models based on two real P2P event logs. All the three models were also in line with P2P processes proposed in Jans et al. [2014] and Sadiq et al. [2007].

The models are created in a way to be able to cover all the deviations proposed in Table 4.1, but also to maintain an acceptable level of simplicity to be understandable.

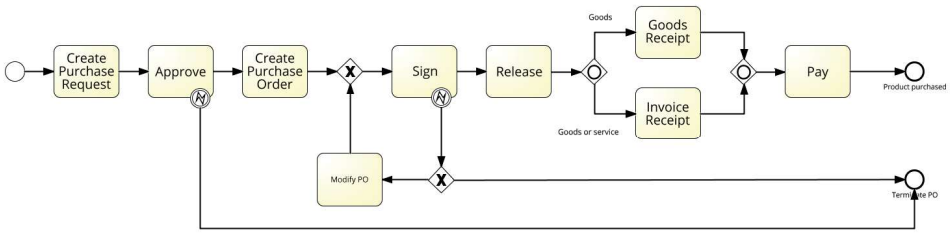
The first model is a simple P2P model with eight tasks, i.e., *Create Purchase Request, Approve Purchase Request, Create Purchase Order, Sign Purchase Order, Release, Goods Receipt, Invoice Receipt* and *Pay*. Although the tasks *Approve Purchase* and *Sign Purchase Order*, both refer to the approval by a senior, for the clarity of the model, we preferred to use different labeling. In case the *purchase request (PR)* is not approved or *purchase order (PO)* is not signed, the procurement case is terminated. The termination is shown for both *Approve Purchase Request* and *Sign Purchase Order*, with a special kind of event in BPMN language, i.e., the intermediate error event (See *Sign* and *Approve* events in the models). *Goods Receipt* and *Invoice Receipt* are designed in an OR branch, which means that either a *Goods Receipt* or an *Invoice Receipt* can take place, but their combination is also possible. Often, if the purchase request concerns goods, both *Goods Receipt* and *Invoice Receipt* should occur. However, if the purchase concerns service, an *Invoice Receipt* is enough.

The second model is the same as the first one, but an extra task *Modify Purchase Order* is added. The *Modify Purchase Order* is designed in a loop format after *Sign Purchase Order*, which means that if the PO is not acceptable to be signed, the purchase order can be modified and resent to be signed.

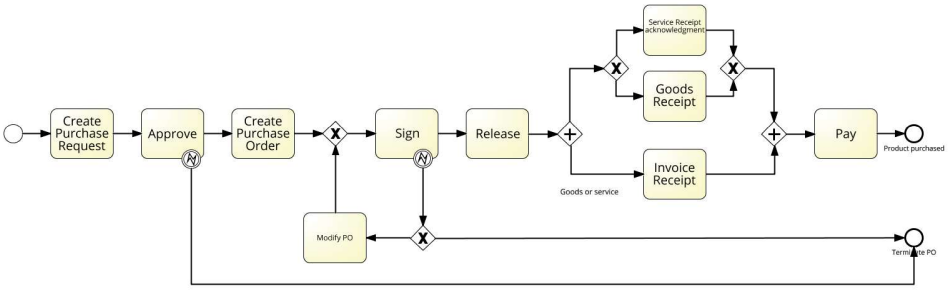
The third model is the same as the second model, but another task for *Service Receipt* is added next to *Goods Receipt* within an *XOR* branch. Having an XOR gateway, means that only one of the branches can be executed, never both together. For instance, in this model, either *Service Receipt* or *Goods Receipt* can occur in one process execution, never both of them. The OR gateway for *Goods Receipt* and *Invoice Receipt* from models (a) and (b) is changed into an *AND* gateway. An AND gateway allows for parallelism. This implies that both branches need to be executed. For instance, in this model, two tasks, are expected to be executed, *Invoice Receipt*



(a)



(b)



(c)

Figure 4.2: (a) A simple P2P model with eight tasks, (b) a simple model the same as *a* but with an extra Modify PO task, (c) a simple model the same as *a* and *b*, but with an XOR gateway to differentiate between 'Service Receipt' and 'Goods Receipt' tasks.

and one of the *Goods Receipt* or *Service Receipt*. This model design, would allow us to investigate the deviations that arise from parallelism (i.e., concurrency) and choice in the model.

The goal was to select the best model among these three models in a way that it will be complete and realistic, but also basic enough to keep the field discussion as close as possible to the core of the research interest. The models are sent to three auditors along with the following questions on the completeness and complexity of the models.

(i) Do these models resemble the purchase-to-pay processes that you as an auditor see on your day-to-day profession? If yes, which one of the three is the closest one to the frequently used P2P process?

(ii) Is everything with regard to the labeling of the tasks or in their order clear in the models?

(iii) Is there any other frequent task in the P2P process that is not mentioned in these models?

(iv) Is any of these models too complex or too simple from an auditor's perspective?

(v) Are the labels of the tasks '*Sign*' and '*Approve*' confusing for auditors? If yes, is there another label for such tasks that is usually used by process analysts/ auditors?

(vi) Which of the models do you prefer, considering its completeness and simplicity?

The answers received from two auditors (one internal and one external auditor) were as follows: The three models are realistic and understandable for the auditors and there is no confusion or mislabeling. They suggested to remove the 'Modify Purchase Order' from the second and third models, because most of the time the P2P process begins with an informal prior agreement. Therefore, modification of a PO occurs very infrequently.

It is also mentioned that for service, there should always be a receipt, the same as for goods, which makes the first model incomplete, although they said that the *Goods Receipt* can be used as the receipt for services as well. The internal auditor confirms that it is possible to combine *Goods Receipt* and *Service Receipt* together as one task. However, since the XOR branch for *Goods Receipt* and *Service Receipt* allows us to check more deviations, as explained before, we did not combine both receipts together. It was also mentioned that auditors see the *Goods Receipt* mostly before the *Invoice Receipt*. Although this can be a correct remark from an event log perspective, from a process modeling perspective the parallelism of these two tasks is

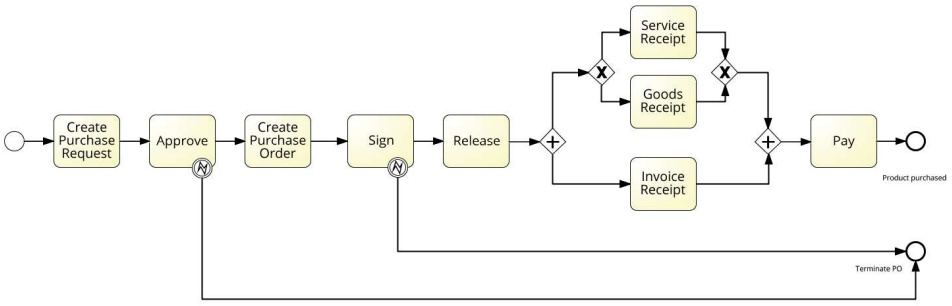


Figure 4.3: A simple model created by modifying the models in Figure 4.2 for instrument, considering auditors remarks

correct. This remark from auditors confirms our assumption that auditors perceive audit data as a set of sequences.
The difference between *Approve* and *Sign* was clear for the auditors. However, the internal auditor said the signature after *Create Purchase Request*, i.e., *Approve*, is the main approval for both purchasing orders and contracts. This remark shows that this auditor gives a higher score to *Approve* compared to *Sign*. This subject is beyond the purpose of this part of the instrument, which is the selection of the best representing model, but it is considered to discuss this during the interviews.

There were also some comments about the consideration of data perspective (such as matching *Create PO* and *Goods Receipt* both price- and quantity-wise) which is beyond the scope of this research, as we consider only the control-flow perspective. Such a remark admits the fact that in the beginning of each interview, the interviewer has to emphasize on the control-flow focus of this research.

Based on the feedback that we got from the auditors, we have modified the models, and created the model in Figure 4.3 and used it as the reference model in the instrument. We decided to discuss some cases which consist of *Modify Purchase Order* task in the interviews. In the *deviation patterns* in the literature, shown in Table 4.1, such cases can be categorized in the *insertion* (existence of an extra task) type.

4.2.3 Process Instances Design

For our instrument, we have created 18 procurement process traces[6]. The series of deviating process instances are based on the *deviation patterns* of Table 4.1.

We have created the process instances in such a way that they cover all possible *deviation patterns* as identified in existing theory. Beware that not all possible deviations (for all the eight tasks, and all six process *deviation patterns* from the literature)

[6] 'Traces' or 'sequences' are used interchangeably throughout this chapter.

are considered in this research. The focus is on the deviation categories and not on the combination of the activity-deviation category. On the one hand, deviating cases should be chosen based on their uniqueness and informativeness, which is needed for later generalization. On the other hand, the examples should be kept realistic enough for auditors, rather than presenting deviating instances that they never have encountered, or find it difficult to imagine them in a real world situation. To reach this goal, we have used theoretical sampling [Eisenhardt, 1989] for creating the instances. In our instrument examples, theoretical sampling means to select the process instances which are likely to occur in reality or process instances that we assume they can facilitate better our purpose and can be used to extend the theory.

Trace	Deviating Purchase-to-pay trace
1	< CreatePR → ApprovePR → CreatePO → SignPO → Release → IR → SR → ModifyPO → ModifyPO → Pay >
2	< CreatePR → ApprovePR → CreatePO → SignPO → ModifyPO → Release → GR → IR → Pay >
3	< CreatePR → ApprovePR → Pay → CreatePO → SignPO → Release → GR → IR → Pay >
4	< CreatePR → ApprovePR → CreatePO → SignPO → Release → GR →SR→ IR → Pay → Pay >
5	< CreatePR → CreatePO → SignPO → Release → GR → IR >
6	< CreatePR → ApprovePR → GR → IR → Pay >
7	< CreatePR → ApprovePR →CreatePO → SignPO →GR → IR → Pay >
8	<CreatePR → ApprovePR → CreatePO → SignPO → CreatePO → SignPO → Release →SR→SR → IR → Pay → Pay >
9	<CreatePR → ApprovePR → CreatePO → SignPO → CreatePO → SignPO → Release → GR → IR → SR → IR → Pay → Pay >
10	< CreatePR → ApprovePR→ CreatePO → SignPO → Release → GR → IR → Pay → CreatePO → SignPO → Release →SR → IR → Pay >
11	< CreatePR → ApprovePR → CreatePO → SignPO → Release → GR → SignPO → IR → SignPO → Pay >
12	< CreatePR → ModifyPO → ModifyPO →ModifyPO → SignPO → Release → SignPO → Release → IR → SR → Pay >
13	< CreatePR → ApprovePR → CreatePO → SignPO → Pay → GR → IR → Pay >
14	< CreatePR → ApprovePR → CreatePO → ApprovePR→ Release → SR → IR → Pay >
15	< CreatePR → Pay → ApprovePR → CreatePO → SignPO → Release → GR → IR >
16	< CreatePR → GoodsReceipt → IR → Pay → ApprovePR → CreatePO → SignPO → Release >
17	< CreatePR→ ApprovePR → PO → Goods Receipt → SignPO → Release → IR → Pay >
18	< CreatePR → ApprovePR → IR → Pay → SignPO >

Table 4.4: Deviating process instances used during the interviews (a selection of this list was taken for each interview)

The list of 18 deviating traces is shown in Table 4.4. An arrow symbol ($\rightarrow$) is chosen to illustrate the order and sequence of the tasks, instead of using a column to separate tasks from each other. For the sake of space in this chapter, the abbreviations of process activities are used in this table and also in Table 4.5. *PR* and *PO* stand for *Purchase Request* and *Purchase Order* respectively. and *IR, GR* and *SR* are accordingly the abbreviations for *Invoice Receipt*, *Goods Receipt* and *Service Receipt.* However, in the instrument used in the interviews, the full name of activities had been illustrated.

As it can be seen in most of the process executions selected here, the deviations can be interpreted in different ways. For instance, the deviation in Trace 13, can be interpreted in three ways: i) *Release* is skipped and *Pay* is inserted, or ii) *Release* is skipped and, $< GoodsReceipt \rightarrow InvoiceReceipt >$ are swapped, *Pay* is repeated, or iii) *Release* is replaced by *Pay*. Among all these interpretations that are possible from a theoretical point of view, we have considered the one that has the least combinations and therefore the least ambiguity. Our classification is only relevant for the theoretical sampling and the analysis afterward. This classification was not shared with the interviewees, not to bias their interpretations. So, for Trace 13, we consider the third deviation pattern. Table 4.5 shows which patterns of process deviations found from the literature are present in each process execution.

4.2.4 Part 1: Discussion on possible deviations based on the process model

In the first part of the interview, the interviewees are asked an open-ended question. An open-ended question, as stated by Saunders [2011], allows the interviewees to define a situation by themselves and let them provide an extensive and developmental answer. The question is as follows:

"Considering this model, what kind of deviations can you come up with in the P2P process executions, from a control-flow perspective?"

Before each interview, this question has also been sent along with the P2P model to the interviewees. An explanation of 'the control-flow perspective' was added as clarification. Receiving the model and question prior to the interview, allows auditors to think of the possible deviations in advance, which generated some additional time for further discussion during the interview.

4.2.5 Part 2: Comparison of process traces with the reference model

The second part of the interview consists of the designed model and a subset of

Trace	Deviation pattern	Description of deviation in example
1	*Insertion*	<ModifyPO → ModifyPO> is inserted after $< SR >$
2	*Insertion*	$< ModifyPO >$ is inserted after $SignPO$
3	*Insertion*	$< Pay >$ is inserted after $ApprovePR$
4	*Insertion*, *Loop*	$< SR >$ or $< GR >$ is inserted, $< Pay >$ is on loop
5	*Skip*, *Skip*	$< ApprovePR >$ is skipped, $< Pay >$ is skipped
6	*Skip*	<CreatePO → SignPO → Release> is skipped
7	*Skip*	$< Release >$ is skipped
8	*Loop*, *Loop*, *Loop*	<CreatePO → SignPO> is on loop, $< SR >$ is on loop, $< Pay >$ is on loop
9	*Loop*, *Insertion*,*Repetition*, Loop	<CreatePO → SignPO> is on loop, SR is inserted, IR is repeated, Pay is on loop
10	*Repetition*, *Insertion*, *Repetition*	<CreatePO→ SignPO → Release> is repeated, SR is inserted, <IR →Pay> is repeated.
11	*Repetition*	$< SignPO >$ is repeated two times
12	*Replace*, *Loop*	< ApprovePR → CreatePO> is replaced by <ModifyPO → ModifyPO → ModifyPO> and <SignPO → Release> is on loop
13	*Replace*	$< Release >$ is replaced by $< Pay >$
14	*Replace*	$< SignPO >$ is replaced by $< ApprovePR >$
15	*Swap*	$< Pay >$ and <ApprovePR → CreatePO → SignPO → Release → GR → IR > are swapped
16	*Swap*	<ApprovePR → CreatePO → SignPO → Release> and <GR → IR → Pay> are swapped
17	*Swap*	$< SignPO \rightarrow Release >$ and GR are swapped
18	*Skip*, *Skip*, *Swap*	$< CreatePO >$ is skipped. <SignPO → Release → GR> is skipped, $SignPO$ and <IR → Pay> are swapped

Table 4.5: The deviating examples used for the interview along with their process *deviation patterns* from literature

the list of 18 process traces with deviating behavior. Per interview eight examples are selected to show to the interviewees. The interviewees compared them with the model and interpreted the deviations.

In the second part of the interview, the model and a couple of traces selected from the list are shown to the interviewees. The examples have been presented to the auditors one by one, and in a random order to avoid the ordering effect. A set of specific and closed questions are asked while presenting each deviating trace:

(i) Do you see any deviation in this example? If yes, what kind of deviation is that?

(ii) Would this trace cause further investigation by your team?

(iii) What is the actual risk level you relate to this deviation (a score between 1 and 5)?

(iv) How do you assess the severity level of this deviation?

4.2.5.1 Pilot run

As a pilot run, a preliminary draft of the instrument is used with a group of 10 academic researchers, before the actual interviews. The result of the pilot test is used to evaluate and refine the instrument. There were six researchers with a business and process mining background and four researchers with a computer science background. Unlike the interviews that are conducted individually, in the pilot run, participants were gathered at the same time in a place but did not communicate with each other during the test. They have reviewed the process instances to assure their clarity and completeness. The pilot test also helped to test whether the collected data from interviews will enable us to answer the research questions [Saunders, 2011].

4.3 Analyses and results

This section contains the analysis of the field research data and its results. Firstly, we explain how the coding of the interview data is done and we discuss the results of analysis. Next, we compare our results with the process *deviation patterns* from Section 4.1.

4.3.1 Interview coding

During the coding phase, in the first-order analysis, we coded in total 1,519 information terms, among which 388 codes were explicitly on the interpretation of the

Exemplary quotation including the information terms	First-order code	Concept
(Interviewee 11) When there are payments without an invoice for a service, that's typically a risk.	Pay without an IR	Without
(Interviewee 4) If you have for example an invoice but no purchase order, then first of all, okay is that they really order it?	No purchase order	No X

Table 4.6: Deviation concepts in first-order analysis and code mapping quotations

deviations from a control-flow perspective. In the appendix some of these first-order codes are presented in Table 9.1 with their associated quotations. Table 4.6 below illustrates only two examples of Table 9.1. The exemplary interview quotations are shown in the first column and the second column shows the example of In Vivo codes we assigned to those quotations. In the transition phase between two cycles of coding, the goal is to categorize the codes and generate the underlying concepts. We categorized the first-order codes, based on the common term that was used in them. For instance, we categorized all the information terms like "task X is missed", "missing task X is risky for the company, not me as an auditor", "I miss task X here in this sequence" or "they just miss a few of the steps", etc., in a concept called *'Miss'*. The last column of Table 4.6 shows the concepts used in the code mapping phase to reorganize and assemble the codes developed from the first-order process. The last column of Table 9.1 in the appendix, besides the concepts, contains their frequency to show how many times these concepts were mentioned in total. Please note that the table only shows the examples for concepts which were mentioned by auditors in the first part of the interview, and for sake of space, we selected only the examples for the concepts which had a frequency higher than 10 for this table.

In the second-order coding, we have used conceptual coding [7]. Conceptual coding, as Saldaña [2016] says, "functions like an umbrella that covers... all other codes and categories". In this type of coding, all terms and concepts which are found in the

[7] Conceptual coding is known also as theoretical coding and selective coding.

first-order analysis and the transition phase, are systematically integrated into a set of core categories called themes. These conceptual categories should be able to describe all its subsets. In our second-order analysis, we have found five themes: Missing, Reordering, Duplication, Insertion, and Substitution. Among these five themes, three core categories were composed of several subsets: Missing, Reordering, and Duplication. The other two themes, Insertion and Substitution were created only based on one or two subsets. We explain these themes further in this section.

Tables 4.7, 4.8, and 4.9 show how the concepts are categorized in themes. The tables contain the concepts, their frequency and the themes, each set of concepts categorized to for missing, reordering, and duplication, along with their total frequency. As illustrated in the tables, the themes missing, reordering, and duplication all cover several related concepts used by auditors.

Concept	Frequency	Theme (Total frequency)
Without	41	Missing (150)
No	31	
Not/ never + verb	21	
Miss	15	
Not + verbs: be, have, happen, exist, execute	14	
Straight / Immediately / Automatic	5	
Skip	5	
Forgotten	4	
Lack of	4	
Bypass	4	
A and C (B is missed)	3	
Go (Start) with X	1	
Gap in the flow	1	
Loose invoice	1	

Table 4.7: Deviation concepts which aggregated into 'Missing' theme in the second-order coding

Table 4.7 shows for missing the concepts with the highest frequency are *Without*

activity X, and *No activity X* with a frequency of 41 and 31 respectively. Table 4.8 illustrates for reordering, the most used concepts are *After* with frequency of 29, and *Before* with frequency of 25. Table 4.9 shows for duplication, *Twice activity X*, *N-times activity X*, or *Two-three-four activity X* are the most mentioned concepts by our interviewees. The frequency of using these concepts by auditors were 17, 13, and 13 respectively.

The other two conceptual categories are insertion, illustrated in Table 4.10 and substitution, illustrated in Table 4.11. However, as it can be seen in Table 4.10, insertion category has only two concepts. This theme has only *Verb X* and *expect X or Y, but here there are both* as subsets.

The substitution theme has only one concept as a subset as illustrated in Table 4.11 which is *instead of activity X you have activity Y*. This only occurred once, hence it was not needed to be aggregated as a theme. However, since the same category existed in the literature *deviation patterns* set, we assign a theme for it during our coding phase.
In the next section, we compare the five themes with the process deviation types from the literature. We also discuss further on the last two themes.

4.3.2 Audit deviation categories

Based on the analysis of the terms that are used by auditors to describe deviations we identified five themes. However, only the first three categories, cover many concepts: missing, reordering, and duplication. Tables 4.7 to 4.11 show how these categories are used in the auditors' daily practice. These three themes were mentioned widely by auditors in the first part of the interviews when they were answering the general question on what kind of deviations they come up with in the P2P process executions. They mentioned these deviation concepts based on their experience with real world processes. The category Inserting was expressed fewer times. The last category, Substituting, was only mentioned once, so we did not get many explicit information terms and concepts in the interview coding.

To decide which deviation categories are applicable for auditors, we compare these deviation themes with their potential counterparts in process deviations in the literature. To do so, we focus and analyze the data that are gathered during the second part of the interview. In that part, auditors are confronted with the deviating traces,

Concept	Frequency	Theme (Total frequency)
After	29	
Before	25	
Afterwards	8	
Advanced	6	
Pre-	4	
Up front/In front	4	
Too late/ Later	4	
Prior	3	
Postpone	2	
Timing issue	2	
Too soon	1	Reordering (98)
Skipping the queue	1	
Not timely X	1	
Not yet X	1	
Somebody can X whenever he want	1	
X in the middle of process	1	
X happens without any order	1	
X is not in the right place	1	
Y in 2016 and X in 2017	1	
X is a step behind	1	
X with a back date	1	

Table 4.8: Deviation concepts which aggregated into 'Reordering' theme in the second-order coding

Concept	Frequency	Theme (Total frequency)
Twice	17	Duplication (88)
N-times	13	
Two-three-four	13	
Duplication	11	
Double	8	
Re-	7	
Multiple/several	6	
Second	4	
Again	4	
Over-	2	
Repeat	1	
Split in different lines	1	
A lot of X and you get only one Y	1	

Table 4.9: Deviation concepts which aggregated into 'Duplication' theme in the second-order coding

Concept	Frequency	Theme (Total frequency)
Verb X	11	Insertion (13)
Expect X or Y, here you have both	2	

Table 4.10: Deviating concepts from the code mapping phase which related to insertion of a new task are categorized into 'Insertion' theme in the second-order coding

Concept	Frequency	Theme (Total frequency)
Instead of X, you have Y	1	Substitution (1)

Table 4.11: The only deviation concept which can be seen as 'Substitution' theme in the second-order coding

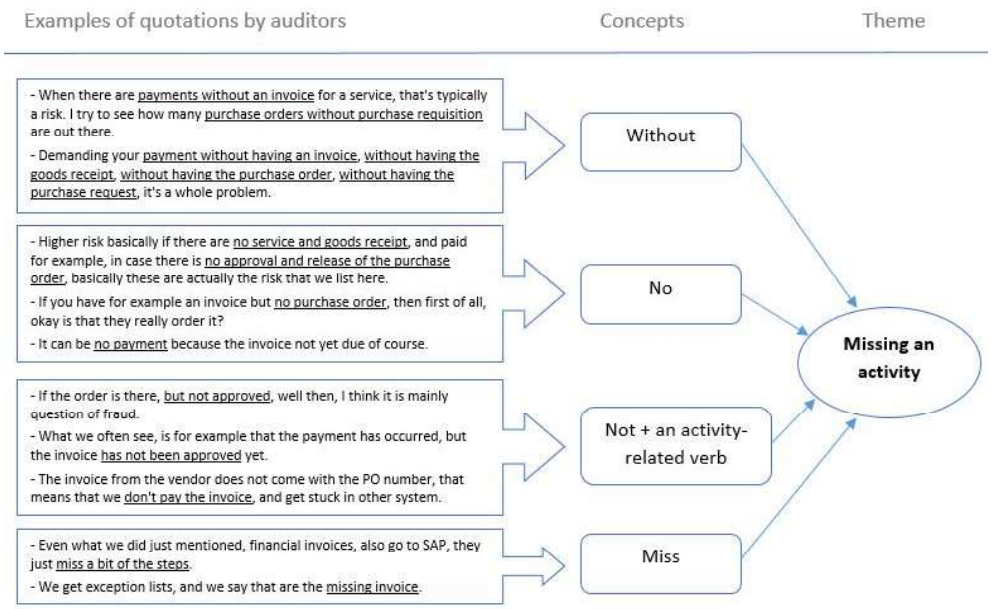


Figure 4.4: Coding structure for the Missing theme

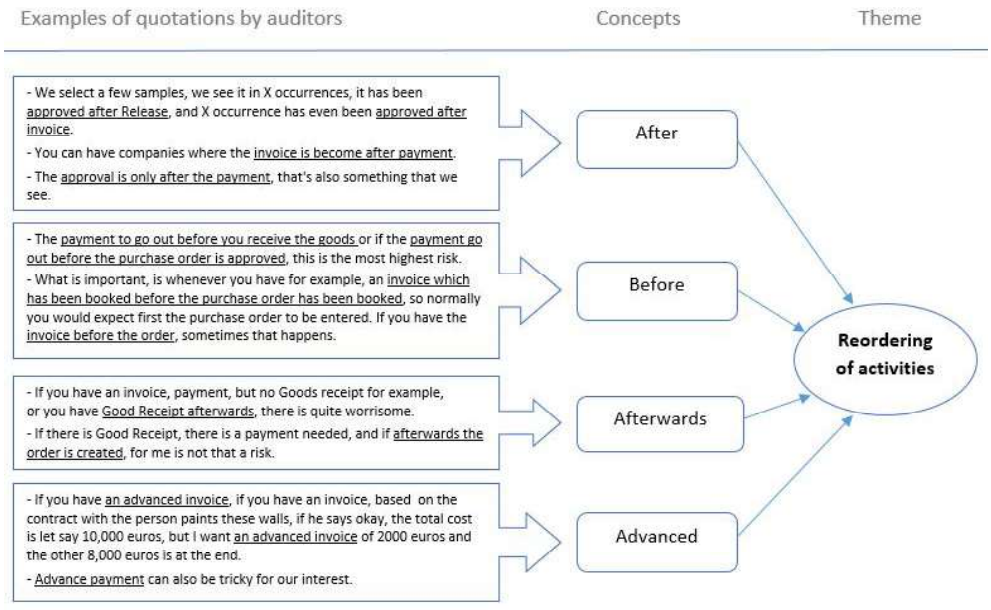


Figure 4.5: Coding structure for the Reordering theme

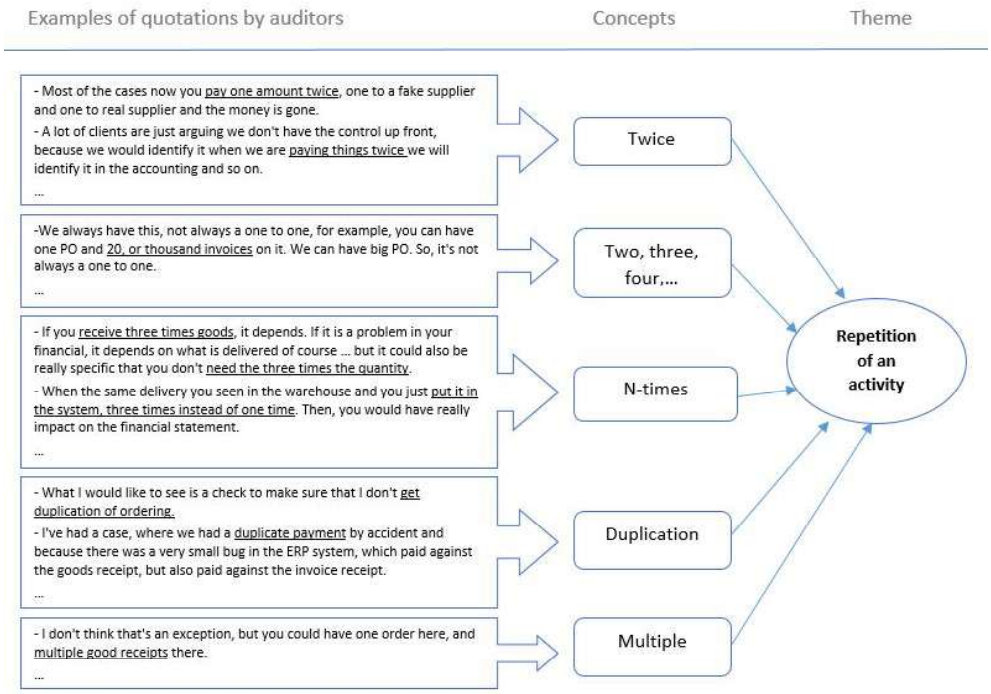


Figure 4.6: Coding structure for the Duplication theme

		Deviation category in literature					
		insertion	*skip*	*replace*	*swap*	*repeat*	*loop*
Deviation category used by auditors	Missing	-	24	14	9	-	-
	Reordering	5	-	7	19	-	-
	Duplicating	5	-	5	-	22	33
	Inserting	13	-	-	-	-	-
	Substituting	-	-	1	-	-	-

Table 4.12: Matrix with frequencies of deviation categories used by auditors when interpreting deviations that are categorized according to previous literature

and they are asked to describe those deviations (in case they find any deviation in the trace). We generated a frequency table to provide an overview of connection between the *deviation patterns* of the examples from a literature point of view in Table 4.12.

The process *deviation patterns* from literature are shown in the columns and the themes that are used by the auditors are shown in the rows.

The frequency table shows that missing, reordering, duplication, and insertion are considered in both sets of *deviation patterns* and categories from the field study. For missing, reordering, and duplication, it also exhibits high correlations between the *deviation patterns* and the deviation themes used by auditors.

Skip, Repeat, and Loop

The frequency table shows some clear patterns. We start with discussing the categories from literature that were grouped under exactly one category of the auditors: '*skip*', '*repeat*, and '*loop*. All the '*skip*' deviation examples that are shown to auditors, are described only with concepts (such as*without, no, miss, skip,*, etc.) that are coded as the 'missing' category. None of the 24 interpretations of auditors is classified under another category. We can state that the '*skip*' deviation category from literature is used by auditors in the same way as it is interpreted in process literature.

Almost the same situation holds for the examples of '*repetition* (22 examples, such as *twice, n-times, duplication*, etc.) and '*loop*' (33) *deviation patterns*. All the concepts that auditors used to describe these examples are grouped under one particular category ('duplication'). None of the examples is categorized in another theme. However,

Deviation categories (Themes)			Deviation patterns					
			Insertion	Skip	Replace	Swapping	Repetition	Loop
	Missing	Without		3	1	7		
		No		3	3			
		Not / never + verb		3	4	1		
		Miss		6	4			
		Not +verbs: be, have, happen, execute			1	1		
		Skip		2				
		Forgotten		1				
		Lack of		3	1			
		Bypass		2				
		Gap in the flow		1				
	Reordering	After				4		
		Before				6		
		Afterwards				3		
		Advanced			3			
		Pre-						
		Up front/In front			2			
		Too late/ Later				2		
		Prior			2	1		
		Timing issue	2					
		Too soon				1		
		Skipping the queue				1		
		Somebody can X whenever he want	1					
		X in the middle of process	1					
		X happens without any order	1					
		X is not in the right place				1		
	Duplication	Duplication	2		3			
		Twice					2	8
		N-times	1		1		2	6
		Re-					5	2
		Multiple/several					2	1
		Two-three-four	1				3	8
		Second			1		3	
		Again (after)					4	
		Double						8
		Both	1				1	

Figure 4.7: Frequency table of themes used by auditors describing the examples

reading the table in the opposite direction reveals that 'duplication' is an umbrella category from auditors for more than one deviation category from literature. It seems that auditors bundle the deviation categories '*repetition*' and '*loop*' and to a certain extent also '*insertion*' and '*replace*', into this category. Auditors apparently do not make a distinction between repeating an activity "somewhere in the process execution" and repeating an activity "right after it has been executed," which would be called a (self-)loop in process literature.

Missing, Reordering, and Duplicating

Continuing the analysis from the deviation categories as used by the auditors, following observations can be made. The "Missing" category hosts several deviation categories from literature: '*skip*' (24), '*replace* (14), and *swap* (9). As already pointed out, the '*skip*' examples all find their way to this category. On top of that, some examples of activities that are '*replaced*' by other activities are interpreted as 'missing' by auditors. Further, some of the *swap* examples are somewhat surprisingly categorized as "missing" by auditors. For example, the *swap* examples in Traces 16 and 17 (where a part of approval is swapped with the whole sequence of $< GoodsReceipt \rightarrow InvoiceReceipt \rightarrow Pay >$ or with $< GoodsReceipt >$) are interpreted by auditors as missing an approval. They describe these deviations as the sequence of $< GoodsReceipt, InvoiceReceipt, Pay >$ or the activity $< GoodsReceipt >$ was done *"without approval"* or when *"no approval"* was in place.

A comparable pattern is found for the *"Reordering"* category. This category also hosts examples of several categories from literature: '*insertion* (5), '*replace*' (7), and '*swap*' (19). From a theoretical, content-wise point of view, swapping activities matches most with reordering them. This is partly confirmed by our data: 19 *swap* examples are indeed classified as *"Reordering"*. However, nine other *swap* examples are classified as "Missing". Hence there is no one-on-one translation between the auditor's *Reordering* and the literature's *swap* category. The auditors' *"Reordering"* classifications also include *insertion* and '*replace*' examples and not all '*swap*' examples are perceived as "Reordering."

Also, for "*Duplicating*," a link with deviation categories from literature is present, but it is not a sharp-cut translation. Aside from all '*repetition* and '*loop* examples that are classified as "*Duplicating*," also some *insertion* and '*replace*' examples are classified under the same deviation category. As mentioned before, '*insertion*' examples can refer to the execution of an activity earlier than expected. If that same activity is also executed at the expected moment, the process execution exhibits twice the same activity, hence the link to duplication. For instance, in example 3, based on

the deviation patterns from literature, *Pay* is inserted after *ApprovePurchaseOrder*, since it is executed in its proper time as well. However, in the auditors' interpretation, it is seen as a combination of *"Reordering"* and *"Duplicating"*. The same line of thought exists with the '*replace*' examples. A '*replace*' can result in having the same activity twice.

In those situations, an interpretation as "*Duplicating*" is logical. These observations could indicate that auditors are more focused on the consequences (one activity is executed twice, hence duplicated), and not on the underlying mechanism (an activity is inserted in the process execution, or is replaced by another).

Inserting

In contrast to the first three categories, the category of "*Inserting*" category was backed up by only a few underlying concepts. This might indicate that this category is either not frequently used, or that it represents a well-understood concept to auditors. When inspecting the usage of the *"Inserting"* category by the auditors, we observe that when this category was used, this was indeed during a discussion of *insertion* examples. However, the used concepts are only weakly connected to the word "*Inserting*," as shown in Table 4.10. On top of that, only some '*insertion*' from the literature point of view examples reside under this category, since the majority was classified as *"Reordering"* (five examples) and/or *"Duplicating"* (another five examples). Given this ambiguous situation, not only here but also in literature, we explore this category more in depth.

Six out of 18 example traces contained an *insertion* deviation deviation pattern (examples 1, 2, 3, 4, 9 and 10). In two examples *Modify Purchase Order* is inserted somewhere in the process, while it was not in the set of activities allowed by the model. In one example, *Pay* is inserted somewhere in the middle of the process, and in other examples, *Service Receipt* or *Goods Receipt* is inserted, while only one of both was expected according to the model.

In examples 1 and 2, the insertion of $ModifyPurchaseOrder$ in the traces did not raise any concern or curiosity to the auditors, as long as they can see an approval after it. For instance, Interviewee 7 describes the second example as follows: "The purchase order is modified, it should be re-approved as well, before being released. So, at this point, I would say that this is a non-authorized modification." So, although the $ModifyPurchaseOrder$ was not in the set of activities in the model, it was acceptable for the interviewees. Their main concern was whether there is a control on this inserted task. These two examples are in all interviews interpreted as "Inserting"

by the auditors. So, it seems that when the *insertion* refers to a new activity, auditors also categorize this as an "*Inserting*" deviation.

When the inserted activity is part of the set of allowed activities in the model, we notice that auditors do not use a specific term for the *insertion* examples. The auditors describe this sort of deviations as "the modification of the Purchase Order was performed after approval" and "this Purchase Order is overwritten" in Interview 7, or "you can still change the PO, and this is dangerous of course" in Interview 9. To include these observations in our data, we used the concept "Verb + activity X" (mentioned 11 times in the interviews) and "Verb X" (mentioned two times in the interviews) for these information terms, which we grouped into "*Inserting*."

For the examples in which either *Service Receipt* or *Goods Receipt* is allowed by the model and both are present in the traces (traces 4 and 10), auditors do consider them deviations. For instance, a participant in the first interview said, "this is a bit strange, because normally you have Goods Receipt or you have Service Receipt." In the second interview it was mentioned that "in most cases you'd expect a GR or SR, here we have both." So, although they do not use a specific information term, it is obvious that they consider such traces as deviations and they look for justifications. We conclude that although there are only a few terms used when interpreting the '*insertion*' deviations, auditors consider it as a type of deviation that needs to be explored further and can be interpreted in different ways, depending on the context of the '*insertion*'. As a result, this deviation category from literature cannot be transposed uniquely to the *"Inserting"* deviation category that is sometimes used by auditors.

4.3.3 Process deviation categories that are not adopted by auditors

Some of the process deviation categories that are described in literature, are not used by auditors or are merged into bigger deviation categories. They are '*replace*' and '*loop*'.

Replace

From all process deviation categories from literature, '*replace*' does not seem to be adopted by auditors. The majority of '*replace*' examples are described by auditors in a less abstract level: they are categorized under *"Missing"* in combination with *"Missing"* with *"Reordering"* or *"Duplicating"*. For instance, Interviewee 7 describes the deviating Trace 13 as "there is [...] an approval missing." Or, Interviewee 8 describes the deviation in Trace 14 as "the approval of the PO, that's missing," indicating the importance of absence of activities.

The other terms that auditors use when describing the other deviations are ei-

ther linked to *"Reordering"* (7), or *"Duplicating"* (5) (see Table 4.12). For instance, Interviewee 1 considers the deviation in Trace 13 as an "advanced payment," and Interviewee 7 describes it as "paying prior to the goods and invoice receipt." Both of these information terms are concepts of the reordering theme. Interviewee 2 describes the deviation in the same example as a "second payment" and Interview 7 uses the term "duplicate payment." These terms belong to the duplication theme.

Over the course of all the interviews, we heard the term "instead" only once. This occurred when the third interviewee was describing example 14. He said "Signature is missing, so instead of the signature, you have again the approval." We considered this term as an interpretation of *"Substituting"*, which would have been the most straightforward transformation of the '*replace*' examples. However, even in this quotation, he described his interpretation by mentioning *"Missing"* and *"Duplicating"* categories ("Signature is missing" and "you have again the approval"). No other term similar to or related to *"Substituting"* was used by any other interviewee.

As a result, we consider the '*replace*' pattern from literature as too abstract to use in an auditing context. The analysis suggests it is better to map this deviation pattern into a combination of deviation categories like *"Missing"* with *"Reordering"* or *"Missing"* with *"Duplicating."*

Loop

Table 4.12 shows the categories that are used to describe '*loop*' examples. As can be seen, all the 33 concepts used by auditors to describe these examples are categorized as *"Duplicating"*. However, as mentioned before, this category also hosts all '*repetition*' examples. In other words: auditors do not use a dedicated category for deviations that relate to looping an activity.

4.4 Discussion and Limitations

As mentioned in the results section, the three deviation categories that we have identified in interviewing auditors are partly in line with the process *deviation patterns* that exist in the literature. The concepts used by auditors to describe examples from the literature *skip* and *repetition patterns* were 100% classified as the same category by auditors. The literature *swapping pattern* was classified mostly as reordering by auditors but sometimes also as missing where the approval activity was involved. For *insertion*, which is one of the basic *deviation patterns*, either the concepts from other categories are used or the used concepts were not explicitly related to the insertion category.

This might be due to the limited possible paths in our proposed model, but also due to the limited number of activities allowed by the model. Hence, as a future research study, an application of a more complex model with different branches and paths can be used as the model example.
The literature *replace pattern*, according to our results seems to be a bit too abstract for auditors. Our interviewees interpreted this sort of deviations at a lower level, as a combination of a missing activity with a reordering or duplication. The *loop pattern*, as mentioned before, is seen as a special kind of duplication, in which the activity is repeated right after itself.

Moreover, beyond the categories discovered from analyzing the interviews, we have noticed that the auditors interpret certain types of deviations as "uncontrolled", "unauthorized", "control failure", and so on. This deviation interpretation occurs when an approval is missing or not timely executed. In our example, this could be the *Approve Purchase Request*, *Sign Purchase Order* or *Release*. An example of this type of deviation is what Interviewee 10 gives as examples of deviation in the process: "Unauthorized payments to third parties, unauthorized acceptance of prices". This example can be categorized into the missing theme. It also can be categorized into the reordering theme, as in this quote of interviewee 6, "If you postpone this approval, or the purchase requisition creation, the risk is that you have unauthorized purchases". However, it can also be related to the duplication theme, as interviewee 6 mentioned that: "Every time there is an invoice receipt and a payment, then it automatically decrements the purchase order. And the automation of that would prevent duplicate payments, and unauthorized payments". Or even it can be related to insertion, like when interviewee 10 describes Trace 3: "Lack of control over the payment process. Uncontrolled payments". These examples show that these kinds of controls in the process are very important for auditors and the absence or reordering of them is considered as a deviation.

It is worth to mention that, we consider the procedural normative model as our reference. The control aspect of transactions refers to the sequential order in which process activities are executed. Considering the procedural model for this purpose, have two advantages compared with the set of predefined rules (which is used as a reference in Kuhn Jr and Sutton [2010], Alles et al. [2006b], Issa [2013], Li et al. [2015], Alles et al. [2006c, 2008b]): Firstly, it skips the time-consuming rules definition procedure prior to analyzing the transaction data. Secondly, although the rules are defined by a process expert panel, it is still possible that some potential risky behavior is not considered due to human error (false-negatives). In comparing event logs with the corresponding procedural normative model, the number of false-positive cases might be higher than in the case of comparing event log with a predefined set of

rules. However, we use the procedural normative models to avoid the false-negative cases, which are critical in auditing. In misstatement detection, it is a well-known fact that a false-negative error is usually more costly than a false-positive error [Phua et al., 2010].

The other limitation of this research is the investigation of only the control-flow perspective. Some important aspects of audit practice, such as the data aspect that is needed for checking the quantities and prices or the segregation of duties, does not exist in the control-flow perspective. However, in order to investigate business processes completely in an automatized fashion, the contribution of this research with the control-flow analysis can be seen as the first phase of this automation. Data and resource perspectives are needed to be checked in the next phases, as it will be discuss in the next chapter.

4.5 Concluding note

This chapter addressed one of the limitations of applying data analysis techniques in the auditing domain: the generation of too many false alarms that must be investigated in a follow-up. This issue is also pressing when analyzing the full population of business process executions, a possibility that is enabled by process mining techniques. These techniques, and especially conformance checking techniques, allow for the identification of all process executions that deviate from the expected process. However, those algorithms produce a set of outcomes that is too large for humans to process. One way to overcome this challenge is to group deviating process executions into categories of similar deviations. Building on the cognitive fit theory, we recognize the importance that these deviation categories are aligned with the way auditors process information on deviating process executions. That is, if the presentation of the problem does not fit the representation of the task or the way domain knowledge is structured internally in the expert's mind, the task performance decreases [Vessey and Galletta, 1991]. So, to move forward with data analytics in the context of auditing, we need a better understanding of how information on identified deviations is interpreted by auditors. Only then can this information be presented in a way that supports the auditor. In this research, we focus on process deviations from the perspective of an activity sequence, called control-flow perspective in process mining literature. A literature study and a field research were performed. First, we identified six patterns in the literature of process mining and business process management to describe mismatches between a logged process execution and the modeled process execution: skip, insert, replace, swap, repeat, and having a loop on an activity.

Next, we investigated the extent to which these categories are applied by auditors when they are presented with information on process executions that deviate from the expected process model. Although some of these patterns from process mining are indeed used by auditors, they are used with a different interpretation. Some patterns that are suggested in the literature are used together in one category by the auditors. Other patterns are scattered over different categories, depending on the example the auditors are shown.

As a result of our research, six *deviation patterns* are found in the literature of process mining and business process management to describe a mismatch between a real process execution and the modeled process execution: *skip*, *replacing*, *swapping*, *repeating* an activity, insertion of an activity, and having a loop on an activity. In a field study, we investigated to what extent these categories are applied by auditors in their daily practice. We found three out of these six patterns are commonly used by auditors: missing, reordering and duplication. For these three categories, the concept used by auditors was in line with the *deviation patterns*. In other words, the auditors used the same themes to describe deviations as those from the *deviation patterns*. Only for the literature *swapping pattern*, it was sometimes classified as a missing category by auditors, only where the approval activities were involved.
The other three *deviation patterns*, i.e., *insertion*, *replace*, and *loop*, although used in some situations by auditors, are not consistently applied. Our results further indicate that the *replace pattern* can be mapped into a combination of two other categories (missing and reordering, or missing and duplication). The results show that the *loop pattern* is also seen as a subcategory of duplication category by auditors, hence it can be mapped into duplication category. The *insertion* pattern was not consistently interpreted as a separate category by auditors; depending on the context, it was interpreted as inserting, reordering, or duplicating.

Our study also shows that merely starting from the output of current process mining algorithms without adapting to the auditing context would hamper the exploitation of the full potential of these techniques. Therefore, the aim of this research is to facilitate future research in process deviation analysis by providing a set of deviation types that are aligned with how auditors perceive deviating process executions.

Chapter 5

Process Deviation Classification

As discussed in Chapter 3, internal auditors are increasingly required to analyze large volumes of process execution data in order to assess whether business processes are executed as intended and whether internal rules and controls are respected. Modern information systems record detailed traces of process executions, making it possible to analyze the full population of process instances rather than relying on sampling, i.e., partial or manual inspection. However, due to certain limitations such as increasing complexity of business processes, the volume of recorded data, and the time constraints, auditors face significant challenges in identifying and interpreting deviations in an efficient and systematic manner.

The problem addressed in this chapter concerns the classification of deviations in process executions by comparing observed behavior recorded in an event log with a normative process model. As introduced in Chapter 3, the normative model represents the intended or expected process behavior, while the event log contains recorded events together with case attributes and event attributes. As explained in Chapter 1, it is assumed that some information relevant for auditing decisions is implicit and not explicitly captured in the normative model. The goal is to investigate the event log and identify deviations from the normative model. Since the normative model is typically incomplete and primarily captures the intended ("happy path") behavior, the approach also supports auditors in distinguishing between exceptions, which can be justified based on contextual or organizational rules, and anomalies, which indicate undesired or unjustified violations and may indicate risks or errors.

Conformance checking techniques provide a systematic way to compare observed process executions with normative models and are among the most promising data analytics approaches for analyzing process behavior in internal auditing [Berghout et al., 2023]. By comparing the full population of process executions recorded in

event logs with a normative process model or predefined rules, these techniques enable the automated detection of deviations. However, as discussed in Chapter 3, the normative process model is typically incomplete and primarily captures the intended behavior, while implicit, exceptional, or context-dependent rules remain unmodeled. Consequently, existing conformance checking techniques often detect a large number of deviations that correspond to acceptable but unmodeled behavior, resulting in numerous false positive cases.

In addition, these techniques typically generate outputs at a highly technical and atomic level [Russell et al., 2006], which does not align with how auditors reason about process behavior and makes timely and meaningful investigation difficult. While some conformance checking approaches, such as the state-of-the-art technique proposed by García-Bañuelos et al. [2018], improve interpretability, they do not fully address the issue of false positives arising from the incompleteness of the normative model.

As a result, auditors are required to manually interpret and classify detected deviations using expert judgment, often considering different perspectives and levels of information. This leads to information overload and extensive manual effort.

Given these limitations, the problem addressed in this chapter is how to assist auditors in classifying deviating process instances more efficiently than current manual approaches. While conformance checking techniques are effective in detecting deviations, they implicitly assume the availability of a complete and exhaustive normative process model. This assumption rarely holds in internal auditing practice. In practice, normative models often capture only the intended or "happy path" behavior and do not fully reflect implicit, exceptional, or context-dependent rules. As a result, existing conformance checking techniques do not sufficiently support auditors in distinguishing true anomalies from acceptable exceptional cases across the full population of process executions. In particular, auditors are confronted with large volumes of deviations, technical representations, and heterogeneous information needs, all of which hinder efficient classification. Insights from the field study presented in Chapter 4 show that auditors prefer to reason about deviations using high-level, interpretable deviation categories rather than technical or atomic descriptions of mismatches between models and logs. This finding directly motivates the design of the framework proposed in this chapter, which follows the algorithmic design principles advocated by Mendling et al. [Mendling et al., 2025]. To address the identified problem, the proposed framework builds upon conformance checking results and incrementally provides the required level of information to auditors, guiding them to classify large volumes of deviations through a human-in-the-loop mechanism that reduces information load while preserving audit effectiveness. By structuring deviations into high-level deviation categories and incrementally extending them with additional information only when necessary,

the framework aims to support auditors in classifying deviating process instances in a more efficient and interpretable manner.

The deviation classification framework is designed as an algorithmic solution to support auditors in systematically classifying control-flow deviations by comparing process executions recorded in an event log with a normative process model. The goal of the framework is not only to detect deviations but to transform conformance checking outputs into interpretable, auditor-oriented representations that facilitate efficient decision-making. From an algorithmic engineering perspective, the framework adapts a conformance checking technique by mapping low-level differences into high-level deviation categories aligned with auditors' mental models, thereby reducing cognitive load and improving usability.

The framework operates under several key assumptions and requirements. It assumes the existence of a normative process model, capturing the intended "happy path" of process behavior, as well as the availability of a structured event log containing traces, activities, timestamps, and contextual attributes. It further assumes that non-process-related internal controls have already been validated prior to applying the framework. The design of the framework is guided by four design principles: minimizing information overload through incremental information disclosure, reducing manual classification effort via automated grouping and pattern extraction, ensuring full population coverage by analyzing all process executions, and aligning deviation representations with auditors' reasoning processes. In addition, the framework must satisfy a set of requirements, namely support for reduction of manual evaluation effort and scalability to real-life event logs. Together, these assumptions and requirements enable a scalable, interpretable, and human-centered approach to deviation classification.

The remainder of this chapter is structured as follows. Section 5.1 introduces the core concepts and definitions required for deviation classification. Section 5.2 presents the deviation classification framework and its algorithmic design. Sections 5.2.4 to 5.2.6 then describe the three stages of the framework in detail, covering deviation identification, iterative deviation analysis, and the labeling of remaining deviations.

5.1 Preliminaries

In this section, besides the preliminaries provided in Chapter 3, we explain some more definitions which are needed to introduce the deviation classification framework in this chapter.

As explained in Chapter 3, in García-Bañuelos et al. [2018] conformance check-

ing technique, both the process model and the event log are converted into an event structure. Subsequently, the two event structures are aligned via an error-correcting synchronized product. Each difference detected in the synchronized product is reported as a mismatch between the normative model and the event log.

The approach distinguishes nine distinct differences:

i. Causality/Concurrency, ii. Conflict, iii. Task skip, iv. Task substitution, v. Unmatched repetition, vi. Task relocation, vii. Task absence, viii. Unobserved acyclic interval, and ix. Unobserved cyclic interval.

For the auditing purpose, we are only interested in the deviations at the trace-level; we can ignore differences i and iii, as they are only applicable when analyzing several traces together.

Below, we formally define the concepts which are essential for understanding the proposed process classification framework. These terms play a crucial role in distinguishing variations in process executions throughout this chapter. The definitions build upon the concepts of event logs and traces, as established by Van der Aalst [2016], and are presented in Definition 1 in Chapter 3.

Definition 3 (Event)**.** *Let $\mathcal{E}$ denote the universe of events. An event $e \in \mathcal{E}$ is defined as a tuple*

$$e = \langle e.cid,\, e.ts,\, e.act,\, e.ea \rangle,$$

where:

- *$e.cid \in CID$ is the* case identifier *from the set of all case identifiers, indicating the process instance to which the event belongs,*
- *$e.ts \in TS$ is the* timestamp *of the event from a totally ordered set of timestamps,*
- *$e.act \in \mathcal{A}$ is the* activity label *associated with the event from the finite set of activities,*
- *$e.ea \subseteq EA$ is a set of* event attributes *from the universe of event attributes.*

We assume a single timestamp per activity execution, representing either the start or the completion of the activity.

Definition 4 (Case)**.** *Let $\mathcal{E}$ denote the universe of events. A* case *c is defined as a tuple*

$$c = \langle c.id,\, c.trace,\, c.ca \rangle,$$

where:

- *c.id is the unique* case identifier,
- $c.trace = \langle e_1, e_2, \ldots, e_n \rangle$ *is the* trace *of the case, defined as an ordered sequence of events such that*

$$\langle e_i | e_i.cid = c.id \rangle,$$

- *c.ca is a set of* case attributes.

We denote $c.trace = \sigma$ *where* σ_i *denotes the i-th event in the trace. All events in a trace share the same case identifier:*

$$\forall i : \sigma_i.cid = c.id$$

.

Definition 5 (Event log)**.** *Let* $\mathcal{C}$ *denote the universe of cases. An* event log L *is defined as a (finite) set of cases:*

$$L \subseteq \mathcal{C}.$$

Equivalently, a simplified event log can be represented as a multiset of traces:

$$L = \{\sigma_1^{n_1}, \sigma_2^{n_2}, \ldots, \sigma_k^{n_k}\},$$

where

$$L = \{\sigma_i^n \mid n = |\{\, c \mid c.trace = \sigma_i \,\}| \ \wedge \ n > 0\}$$

meaning that each $\sigma_i = c_i.trace$ *is the trace associated with a case* $c_i \in L$*,* $n_i > 0$ *denotes the multiplicity of trace* σ_i*, and* k *is the number of distinct traces in the log.*

Definition 6 (Normative process model)**.** *A* normative process model M *defines, either explicitly or implicitly, the set of process executions that are considered allowed or intended.*

Formally, let Σ *denote the universe of all possible traces. The normative model* M *induces a set of allowed traces:*

$$\mathcal{L}(M) \subseteq \Sigma,$$

where each trace $\sigma \in \mathcal{L}(M)$ *represents a process execution that conforms to the intended (normative) behavior.*

Traces that do not belong to $\mathcal{L}(M)$ *are considered non-conforming and may correspond to deviations.*

Definition 7 (Deviation type)**.** *A deviation type is an element*

$$t \in \mathcal{T} = \{Ins, Mis, Reo, Rep\},$$

where Ins*,* Mis*, Reo, and Rep correspond to Insertion, Missing, Reordering, and Repetition deviations, respectively.*

Definition 8 (Deviation)**.** *Let L be an event log, M a normative process model, and let $c \in L$ be a case with trace*

$$c.trace = \sigma = \langle \sigma_1, \sigma_2, \ldots, \sigma_n \rangle,$$

and $\sigma = c.trace$ be the trace of case $c \in \mathcal{C}$, and let M be the normative process model. A deviation is a 5-tuple:

$$\delta = (c, t, a, p, q),$$

$c \in \mathcal{C}$, the case in which the deviation occurs
$t \in \mathcal{T}$, the deviation type i.e., $t \in \{\textsc{Ins}, \textsc{Mis}, \textsc{Reo}, \textsc{Rep}\}$.
$a \in \mathcal{A}$, the activity label associated with the deviation
$p \in \mathbb{N}$, the primary position index
$q \in \mathbb{N} \cup \{\perp\}$, the secondary position index ($\{\perp\}$ when not applicable) We access components via dot notation: $\delta.c, \delta.t, \delta.a, \delta.p, \delta.q$.

The four deviation types are instantiated as follows:

Insertion deviations *An* insertion deviation *occurs when an event is observed at a position where the corresponding activity is not allowed by the normative model. Formally, an insertion deviation is a tuple*

$$\delta = (c, \textsc{Ins}, \sigma_p.act, p, \perp)$$

such that activity $a \in \mathcal{A}$ is inserted in $c \in \mathcal{C}$ while it was not allowed to occur at position p according to M. Let

$$\Delta^{Ins} = \{\delta^{Ins} \mid \delta^{Ins} \text{ is an insertion deviation}\}.$$

Missing deviations *A* missing deviation *occurs when an activity that is required by the normative model does not occur in the observed trace. Formally, a missing deviation is a tuple*

$$\delta = (c, \textsc{Mis}, a, p, \perp)$$

such that activity a is required to occur at position p in case c according to M, but such event is not recorded. Let

$$\Delta^{Mis} = \{\delta^{Mis} \mid \delta^{Mis} \text{ is a missing deviation}\}.$$

Reordering deviations *A* reordering deviation *occurs when an activity is executed, but at a position different from that prescribed by the normative model. Formally, a reordering deviation is a tuple*

$$\delta = (c, \textsc{Reo}, \sigma_p.act, p, q)$$

such that activity a in $(c, p) \in \Delta^{Ins}$, $(c, q) \in \Delta^{Mis}$, *and the activity a appears at position p, where a is expected at position q according to* M. *Let*

$$\Delta^{Reo} = \{\delta^{Reo} \mid \delta^{Reo} \textit{ is a reordering deviation}\}.$$

When a reordering deviation $\delta_{\textsc{Reo}} = (c, \textsc{Reo}, \sigma_p.\text{act}, p, q)$ *is identified, the corresponding insertion and missing deviations are removed from* $\Delta^{\textsc{Ins}}$ *and* $\Delta^{\textsc{Mis}}$, *respectively, thereby ensuring that the four deviation subsets remain pairwise disjoint.*

Repetition deviations *A* repetition deviation *occurs when an activity is executed more times than allowed by the normative model. Formally, a repetition deviation is a tuple*

$$\delta = (c, \textsc{Rep}, \sigma_q.act, p, q)$$

where p is the first occurrence position, q is the repetition position, such that $p < q$, $(c, p) \notin \Delta^{Ins}$, $(c, q) \in \Delta^{Ins}$,
and
$\nexists k : p < k < q$ *such that* $(c, k) \notin \Delta^{Ins}$, $\sigma_k.act = \sigma_q.act$

meaning that σ_p *is the closest preceding non-inserted occurrence of the same activity to* σ_q.

Let

$$\Delta^{Rep} = \{\delta^{Rep} \mid \delta^{Rep} \textit{ is a repetition deviation}\}.$$

When a repetition deviation $\delta_{\textsc{Rep}} = (c, \textsc{Rep}, \sigma_p.\text{act}, p, q)$ *is identified, the corresponding insertion deviation is removed from* $\Delta^{\textsc{Ins}}$, *thereby ensuring that the four deviation subsets remain pairwise disjoint.*

The deviation universe is defined as the disjoint union of category-specific deviation sets: Δ *denotes the set of all deviations across all cases.*

$$\Delta \;=\; \Delta^{Ins} \;\cup\; \Delta^{Mis} \;\cup\; \Delta^{Reo} \;\cup\; \Delta^{Rep},$$

where:

- Δ^{Ins} *denotes the set of* Insertion *deviations,*
- Δ^{Mis} *denotes the set of* Missing *deviations,*

- Δ^{Reo} *denotes the set of* Reordering *deviations,*
- Δ^{Rep} *denotes the set of* Repetition *deviations.*

Each deviation $\delta \in \Delta$ is represented as a tuple containing a case identifier, a position index, and an activity label. The activity associated with a deviation, denoted by $\delta^{dev}.a$, is defined depending on the deviation category as follows:

$$\delta^{dev}.act = \begin{cases} \sigma_i^{Ins}.act, & \text{if } \delta \in \Delta^{Ins} \text{ and } \delta = (c, \textsc{Ins}, \sigma_p.act, p, \perp) \\ \sigma_i^{Mis}.a, & \text{if } \delta \in \Delta^{Mis} \text{ and } \delta = (c, \textsc{Mis}, a, p, \perp), \\ \sigma_i^{Reo}.act, & \text{if } \delta \in \Delta^{Reo} \text{ and } \delta = (c, \textsc{Reo}, \sigma_p.act, p, q), \\ \sigma_i^{Rep}.act, & \text{if } \delta \in \Delta^{Rep} \text{ and } \delta = (c, \textsc{Rep}, \sigma_q.act, p, q). \end{cases}$$

Here, σ_i denotes the i-th event of the trace associated with case c, and $\sigma_i.act$ denotes the activity label of that event.

Definition 9 (Deviation classes)**.** *A* deviation class *is any subset $D \subseteq \Delta$ whose elements share a common defining condition (e.g., deviation type, activity, ordering pattern, or contextual attributes).*

(9a) ***Deviation sequence of a case.***

For a case $c \in C$ with deviations $\Delta(c) = \{\delta \in \Delta \mid \delta.c = c\}$, the deviation sequence is the sequence of (type, activity) pairs ordered by primary position:

$$\mathrm{Seq}(c) = \langle (\delta_1.t, \delta_1.a), (\delta_2.t, \delta_2.a), \ldots, (\delta_m.t, \delta_m.a) \rangle$$

where $\delta_1, \delta_2, \ldots, \delta_m$ are the deviations in $\Delta(c)$ ordered by $\delta.p$.

(9b) ***Level 1 – Deviation classes***

For a deviation type $t \in T$ and an activity label $a \in A$, the Level 1 deviation class for (t, a) is defined as

$$D_1(t, a) = \{\delta \in \Delta \mid \delta.t = t \wedge \delta.a = a\}.$$

Level 1 groups deviations solely by their type and activity, without considering execution context or ordering. The set of all Level 1 classes is

$$\{D_1(t, a) \mid t \in T,\, a \in A,\, D_1(t, a) \neq \emptyset\}.$$

(9c) ***Level 2 – Deviation classes based on ordered deviation types***

Let $p = \langle (t_1, a_1), (t_2, a_2), \ldots, (t_n, a_n) \rangle$ be a deviation type pattern, where each $t_i \in T$ and $a_i \in A$. Let $\mathcal{C}_D \subseteq C$ denote the set of deviating cases, i.e., cases that contain at least one deviation. The Level 2 deviation class is defined as :

$$D_2(p) = \{c \in \mathcal{C}_D \mid p \sqsubseteq \mathrm{Seq}(c)\}$$

where $\sqsubseteq$ denotes the subsequence relation and $\mathrm{Seq}(c)$ is the ordered sequence of (type, activity) pairs for case c. Level 2 groups deviating cases by shared sequential deviation patterns.

This level groups deviations by recurring sequences of deviation types and activities within the same case.

Example:

$$D(\langle (Mis, a)(Ins, b) \rangle)$$

(9d) ***Level 3 – Deviation classes with case attributes***

Let $\mathcal{T}^*$ be the set of extended deviation type as

$$\mathcal{T}^* = \mathcal{T} \cup \{N\} = \{Ins, Mis, Reo, Rep, N\}$$

where Ins, Mis, Reo, Rep are deviation categories and N represents a normal (non-deviating) event.

For a case $c \in C$, let $ca(c) \subseteq \mathcal{CA}$ denote the set of case attribute. The Level 3 extended deviation sequence is:

$$S_3(c) = (ca(c), \langle (t_1, a_1), (t_2, a_2), \ldots, (t_n, a_n) \rangle)$$

where $t_i \in \mathcal{T}^*$, $a_i \in \mathcal{A}$ is an activity and $ca(c) \subseteq \mathcal{CA}$ is a subset of case-attributes for the i-th case. The pair structure separates case attributes from the deviation pattern. Normal (matched) events are represented by $t_i = N$. The Level 3 deviation class for a pattern p with case attributes ca is defined as

$$D_3(ca, p) = \{c \in \mathcal{C}_D \mid ca \subseteq ca(c) \wedge p \sqsubseteq S_3(c).\mathrm{seq}\}$$

where $S_3(c).\mathrm{seq}$ is the sequence component of $S_3(c)$.

This level enriches deviation patterns with case-level contextual information in order to support case-level context classification.

(9f) ***Level 4 – Deviation classes with event attributes***

For a case $c \in \mathcal{C}_D$, the Level 4 extended deviation sequence augments Level 3 with event-level attributes:

$$S_4(c) = (ca(c), \langle (t_1, a_1, ea_1), (t_2, a_2, ea_2), \ldots, (t_n, a_n, ea_n) \rangle)$$

where each $t_i \in \mathcal{T}^*$, $a_i \in A$, and $ea_i \subseteq \mathcal{EA}$ is the set of event attributes associated with the i-th position. A Level 4 pattern is a pair

$$p = (ca_p, \langle (t'_1, a'_1, ea'_1), \ldots, (t'_m, a'_m, ea'_m) \rangle)$$

where $ca_p \subseteq \mathcal{CA}$ specifies required case attributes and each $ea'_j \subseteq \mathcal{EA}$ specifies required event attribute values at the matched position.

The Level 4 deviation class for a pattern p is:

$$D_4(p) = \{c \in \mathcal{C}_D \mid ca_p \subseteq ca(c) \wedge p.\text{seq} \sqsubseteq_{ea} S_4(c).\text{seq}\}$$

where $\sqsubseteq_{ea}$ denotes the event-attribute-aware subsequence relation: $p.\text{seq} \sqsubseteq_{ea} s$ if there exists an order-preserving injection mapping each pattern element (t'_j, a'_j, ea'_j) to a sequence element (t_i, a_i, ea_i) such that $t'_j = t_i$, $a'_j = a_i$, and $ea'_j \subseteq ea_i$.

Note change from Level 3: Level 4 patterns contain event attribute constraints at specific positions. The subsequence match $\sqsubseteq_{ea}$ extends $\sqsubseteq$ by additionally requiring that the pattern's event attribute values are present in the matched position's event attributes.

Level 4 refines deviation classes by incorporating event attributes, such as timestamps, resources attributes. This level is used only when deviation classification cannot be resolved using control-flow structure and case-level information alone.

Definition 10 (Exception)**.** *An exception is a deviation that is considered acceptable due to predefined or implicit rules not directly encoded in the process model. Exceptions are process variants that occur under special conditions, such as urgent cases. In conformance checking terms, these deviations may align with alternative rules not captured in M. For example, let $\sigma_e \subseteq L$ denote traces that deviate but are validated against a secondary exception rule set Exc. For a trace $\sigma \in L$, if σ deviates from M and $Exc(\sigma) = true$, then σ is classified as an exception.*

Definition 11 (Anomaly)**.** *An anomaly is a deviation that cannot be justified by any additional rules or exception policies. These represent irregularities that signal potential issues such as errors, fraud, or process inefficiencies.*

A trace $\sigma \in L$ is classified as an anomaly if it contains at least one deviation and does not satisfy the exception rule:

$$\text{Anom}(\sigma) = \textit{true} \iff \Delta(\sigma) \neq \emptyset \wedge \text{Exc}(\sigma) = \textit{false}$$

where

$$\Delta(\sigma) = \{\delta \in \Delta \mid \delta.c = c \wedge c.\text{trace} = \sigma\}$$

is the set of deviations in that trace.

Anomalies are critical for auditors as they highlight risks or violations in process execution.

As proposed in Chapter 4, deviations in the auditing context are defined using a finite set of high-level deviation categories that reflect how auditors conceptualize deviations in practice. In this chapter, these categories are formalized as a set of *deviation categories*.

To facilitate readability and provide a structured reference, the notation used throughout this chapter is summarized in Tables 5.1 and 5.2.

The notation is divided into two tables to distinguish between the underlying data structures and the concepts proposed in the framework. Table 5.1 covers standard process mining concepts related to the event log and normative process model, such as events, cases, and traces. Table 5.2 introduces the notation specific to the deviation classification framework developed in this thesis, including deviation types, deviation sets, and the different levels of deviation classes. These concepts extend the basic event log representation by structuring deviations in a way that supports iterative and human-in-the-loop classification.

5.2 Deviation Classification Framework

This section explains how the deviation classification framework works and how it is positioned with respect to the algorithmic engineering perspective of the thesis. The framework is designed as an algorithm for auditors that supports the classification of control-flow deviations by analyzing differences between process executions and a normative process model. In line with the algorithmic design principles introduced earlier in this chapter, the focus is not only on detecting deviations but also on engineering the output and interaction in a way that supports auditor decision-making efficiently.

Notation	Description
$\mathcal{E}$	Universe of events
$e \in \mathcal{E}$	Event, $e = \langle e.cid, e.ts, e.act, e.ea \rangle$
CID	Set of case identifiers
TS	Set of timestamps
$\mathcal{A}$	Set of activity labels
EA	Universe of event attributes
$c \in \mathcal{C}$	Case, $c = \langle c.id, c.trace, c.ca \rangle$
σ	Trace (ordered sequence of events)
$\mathcal{C}$	Universe of cases
$L \subseteq \mathcal{C}$	Event log (set of cases)
Σ	Universe of all possible traces
M	Normative process model
$\mathcal{L}(M)$	Set of allowed traces defined by M

Table 5.1: Event log and process model notation

The framework takes as input (i) a normative model, assumed to exist and represented as a Petri net, and (ii) an event log containing events, case identifiers, timestamps, case attributes, and event attributes. The normative model serves as the baseline against which observed executions are compared. As discussed in the introductory sections of this chapter, the normative model may not contain all information required for auditing decisions, because some relevant context or exceptional rules can be implicit and therefore not explicitly captured in the normative model. The framework further assumes that internal controls such as segregation of duties and three-way match have already been validated using existing auditing software, and that the deviation classification procedure is applied after this validation step.

From an algorithmic engineering perspective, the framework builds on an existing conformance checking technique and adapts it to meet the requirements identified in the earlier sections of this chapter. Concretely, it uses the first class of the conformance checking technique introduced by García-Bañuelos et al. [2018] as the deviation detection mechanism, and it modifies the output so that differences between the process model and event log are mapped into the high-level deviation categories introduced in Chapter 4. This design choice ensures that deviations are represented

Notation	**Description**
$\mathcal{T}$	Set of deviation types, $\{\textsc{Ins}, \textsc{Mis}, \textsc{Reo}, \textsc{Rep}\}$
$\delta = (c, t, a, p, q)$	Deviation tuple
Δ	Set of all deviations
$\Delta^{Ins}, \Delta^{Mis}, \Delta^{Reo}, \Delta^{Rep}$	Sets of deviations by type
$\Delta(c)$	Set of deviations in case c
t	Deviation type
a	Activity label
p, q	Position indices in a trace
$\perp$	Undefined / not applicable position
$\text{Seq}(c)$	Deviation sequence of case c
$DT(c)$	Set of distinct (t, a) pairs in case c
$p \sqsubseteq \text{Seq}(c)$	Pattern p is a subsequence of $\text{Seq}(c)$
$\sqsubseteq_{ea}$	Subsequence relation considering event attributes
$\mathcal{C}_D$	Set of deviating cases
$D_1(t, a)$	Level 1 deviation class (by type and activity)
$D_2(p)$	Level 2 deviation class (by deviation sequence pattern)
$D_3(ca, p)$	Level 3 deviation class (with case attributes)
$D_4(p)$	Level 4 deviation class (with case and event attributes)
$\ell_1, \ell_2, \ell_3, \ell_4$	Labels assigned by the auditor (e.g., OK, NOK)

Table 5.2: Deviation and classification notation

in an interpretable form that matches auditors' mental models, rather than being presented as technical or atomic mismatches. The framework is motivated by the practical limitations of current conformance checking techniques in internal auditing. As the normative model is not complete, conformance checking tools typically produce a large deviation set that mixes potentially problematic behavior with exceptional but acceptable behavior that is not explicitly modeled in the normative models.

Deviations are grouped and presented to the auditor through successive levels of information, starting with high-level deviation categories and incrementally incorporating additional contextual information only when needed. This incremental design operationally addresses the requirements formulated earlier in this chapter, in particular minimizing information overload, reducing manual classification effort, ensuring full population coverage, and aligning the analysis with auditors' reasoning processes.

5.2.1 Algorithmic Design of the Deviation Classification Framework

The deviation classification framework proposed in this chapter is intended to support auditors by comparing process executions recorded in an event log with a normative process model, with a specific focus on the classification of control-flow deviations as anomalies or exceptional cases. The design follows algorithmic design principles that emphasize interpretability, efficiency, and alignment with user needs. In this chapter, the proposed deviation classification framework is developed under the following assumptions, consistent with those introduced in Chapter 1:

Assumption 1: Existence and representation of a normative process model
A normative process model is assumed to exist. The model represents the intended and standard execution of the process and captures only the happy path of process behavior. Exceptional and infrequent execution paths are not explicitly modeled.

Assumption 2: Availability and structure of the event log
An event log is assumed to be available. The event log contains, at a minimum, recorded events with case identifiers, timestamps, and activity labels, as well as case attributes and event attributes. We also assume that all information that is relevant to the auditor to make an informed assessment is present in the event log.

Assumption 3: Prior validation of internal controls
The framework operates under the assumption that non-process related internal controls, such as segregation of duties and two-way or

three-way match, have already been tested and validated. These controls can be evaluated using existing auditing software and are assumed to be reliable before applying the deviation classification framework.

The algorithmic design emphasizes clarity, efficiency, and alignment with user needs. In line with these principles, the proposed framework is guided by four key design principles:

Design Principle 1: Minimization of information overload
The framework is designed to minimize information overload by providing deviation-related information incrementally. Deviations are not presented with all available information at once; instead, auditors initially receive only the minimal information required for classification. Additional information is introduced progressively as needed, particularly when the available information is insufficient in subsequent steps of the analysis. This design reduces cognitive load by construction.

Design Principle 2: Reduction of manual classification effort
The framework aims to reduce manual classification effort by automating the initial categorization of deviations. Auditors are expected to manually review only those deviations that remain ambiguous or complex after automated processing. The effectiveness of this principle is evaluated in the subsequent chapter through a case study.

Design Principle 3: Full population coverage
The framework is designed to ensure full population coverage by evaluating all process executions recorded in the event log rather than relying on sampling. This supports comprehensive auditing and is inherently satisfied by the design. Its implications are discussed in detail in the following sections.

Design Principle 4: Alignment with auditors' mental models
The framework aims to align with auditors' mental models by classifying deviations using high-level, interpretable deviation categories that reflect how auditors conceptualize anomalies and exceptional cases, as identified in the field study presented in Chapter 4. Instead of presenting deviations in an atomic or

purely technical form (e.g., event-level insertions or deletions), the framework emphasizes auditor-oriented representations.

In addition to the design principles, the framework must satisfy a set of functional and performance requirements that ensure its applicability in continuous auditing settings.

Requirement 1: Reduction of manual evaluation effort
The framework must demonstrably reduce the number of manual evaluation actions compared to a fully manual assessment. This requirement is operationalized and measured through the effort reduction metric in Chapter 6.

Requirement 2: Scalability to real-life event logs
The framework must be applicable to real-life event logs of substantial size and complexity, such as the purchase-to-pay dataset used in the case study, without requiring sampling.

The algorithmic design of the framework proceeds in three conceptual stages.

Stage 1) Deviation Identification and Type Assignment

The algorithm starts by identifying deviations between observed process executions and the normative model. Process executions are analyzed with respect to the expected order and occurrence of activities, that is, their observed control-flow behavior as recorded in the event log. Deviations are detected by contrasting the observed control-flow patterns of individual process executions with the expected control-flow patterns defined by the normative model. Based on this comparison, deviations are assigned to one of four deviation categories: missing activities, reordering of activities, insertion of unexpected activities, and repetition of activities. The primary deviations categories are defined as follows:

Missing – an expected activity in the normative model is absent in a case's trace.

Reordering – activities occur in a different sequence than prescribed by the normative model.

Insertion – an unexpected activity appears in a case's trace.

Repetition – an activity is executed more times than expected in a case's trace.

These deviation categories correspond directly to the deviation categories identified in Chapter 4 and reflect the way auditors conceptualize deviations in practice. This stage ensures that deviations are initially categorized in a way that matches the auditors' mental model, as discussed in the previous chapter, facilitating comprehension and reducing cognitive load.

Stage 2) Iterative Deviation Analysis

After identifying and categorizing deviations, the framework applies a multi-level analysis and iterative classification procedure in which deviations are incrementally presented to the auditor for labeling. In each iteration, groups of deviations are formed based on shared deviation characteristics and presented together with the available explanatory information, allowing the auditor to assess whether the provided information is sufficient to label the group as anomalies or exceptions. If the information is considered insufficient, the group is further split into lower-level groups which allow the presentation of more detailed and contextual information in subsequent iterations. As the iterations progress, the size of the deviation groups decreases while the richness of the information provided increases, until the process converges at the level of individual deviations if necessary. Based on the auditor's input, similar process executions are automatically assigned to the same deviation categories and candidate classification. The loop and iterations are designed to incrementally increase the level of information provided to the auditor. Initial iterations focus only on high-level deviation types, while subsequent iterations incorporate additional contextual information, such as common variation patterns and, when necessary, information derived from case attributes and event attributes. This incremental disclosure of information ensures that auditors are not overwhelmed and are only presented with additional context when it is necessary for decision-making.

The iterative loop has multiple levels designed to minimize information overload. To illustrate the distinction between these levels, consider a single process execution where the normative process model specifies the precedence constraint $A \rightarrow B$, meaning that activity A must be executed before the approval activity B.

> Level 1: This process execution is presented to the auditor solely in terms of its deviation type. For example, if the recorded execution in the event log shows activity A occurring after activity B, this level presents the deviation as *reordering(A)*. No additional information about the surrounding process context or execution details is provided. This level supports rapid, high-level screening based on deviation type alone.
>
> Level 2: Deviations of the same type are grouped into frequent deviation pat-

terns. For instance, if a trace (B, A, B) occurs in multiple process executions, this level presents the deviation sequence $\langle reordering(A), repetition(B)\rangle$ to the auditor. This enables auditors to reason about recurring deviation structures rather than isolated instances.

Level 3: These deviation patterns are augmented with case-level contextual information, such as attributes associated with the process execution (e.g., transaction value or case category).

Level 4: The information from the previous levels is augmented with event attributes including information such as timestamps, responsible resources, or activity-specific data. This level provides the most detailed view and supports the classification of complex deviations that cannot be resolved using control-flow and case-level information alone.

Table 5.3 summarizes the inner loop design of the deviation classification framework, detailing the four level iteration, the information each level provides and the purpose of each iteration for categorizing deviations.

	Unit of analysis	**Information Provided**	**Purpose**
1	High-level deviation types only	Only the existing deviation types	Minimizes information overload and matches the auditors' mental model
2	Frequent deviation types patterns	Summaries of common deviation patterns across process executions	Facilitates pattern recognition and reduces manual classification effort
3	Frequent deviation sets with case attributes	Deviation sets enriched with case-level contextual information	Supports auditors in disambiguating deviations using contextual information
4	Frequent deviation sets with case and event attributes	Deviation sets enriched with both case-level and event-level attributes	Enables accurate classification of complex deviations

Table 5.3: Inner Loop Design of the Deviation Classification Framework

In each iteration, the framework goes deeper in the level of information, only revealing additional details when necessary for classification. This design balances efficiency and comprehensiveness while adhering to auditors' cognitive constraints.

This iterative procedure continues until no further deviations can be confidently classified using the available information.

Stage 3) Labeling Remaining Deviations

In the final stage, after the iterative loop converges, any remaining unclassified deviations are labeled using the accumulated knowledge from previous iterations. This final step ensures that all process instances in the event log are classified, addressing the requirement of full population classification by design. Overall, the deviation classification framework supports internal auditors in analyzing deviations in a structured and efficient manner. By combining high-level deviation categories, incremental information provision, and interaction, the framework reduces manual intervention and effort, avoids unnecessary complexity, and focuses auditor attention on the most informative deviations and systematically aligns technical conformance results with auditors' reasoning processes across the entire population of process executions.

In summary, the proposed framework puts insights from prior chapters into a coherent algorithmic design that enables scalable, interpretable, and auditor-centered deviation classification. It bridges the gap between technical conformance checking outputs and practical auditing needs by structuring deviation analysis as an incremental, interactive process grounded in auditors' mental models.

Figure 5.1 provides an overview of the three stage deviation classification framework introduced in this section.

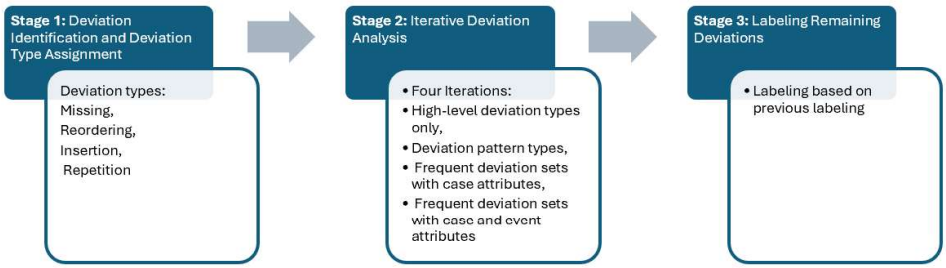


Figure 5.1: Three-stage deviation classification framework

Figure 5.2 summarizes how the proposed deviation analysis approach works.

5.2.2 Academic Example

To illustrate the framework and explain its steps in a transparent and reproducible manner, the section uses an edited version of an academic running example described

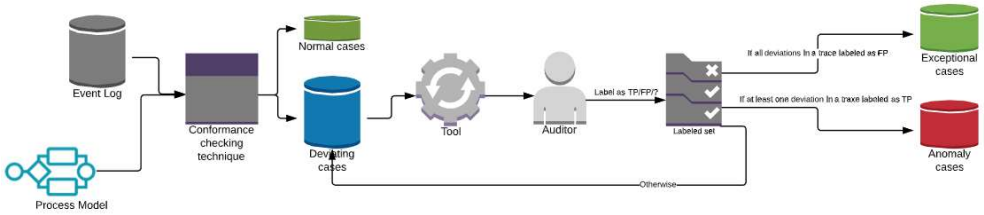


Figure 5.2: The proposed deviation analysis approach

by de Leoni and van der Aalst [2013a] and used in other process mining studies such as Mannhardt et al. [2016]. The example is a process to handle credits request by clients to buy small home appliances. The client request is registered, verified, assessed, and communicated. It means the client's financial data should be checked and verified when the credit is requested. If the verification is positive, the request is assessed, and a notification is sent to the client. Otherwise, the client is informed about the negative assessment and rejection. The BPMN diagram of this process model is depicted in Figure 5.3. The original model (from de Leoni and van der Aalst [2013a]) is edited in a way to include both AND and OR gateways. The process model is enriched with information perspectives (control-flow, data, resource, and time) that allow demonstrating how different levels of information can support deviation classification. Activities' names and their simplified version of them are shown in the boxes. Each activity is performed by a person with a particular role (Assistant, Expert, or Manager). These performers are also shown in the model.

Example traces with case and event attributes are used throughout the following sections to show how deviations are detected, grouped, and incrementally presented to auditors for labeling, and how these labels are used to classify process executions as anomalies or exceptional cases. Table 5.4 shows traces as examples of the execution of this process which we use to explain our approach. In each case, each bracket refers to one of its events. According to the model, the first item in the event refers to the abbreviation of the activity. Letter A refers to the requested amount. Letter R is the name of the requester of the loan, $E_{activity}$ is the resource or performer of the activity, and $T_{activity}$ is the timestamp or the time the activity is performed. V is a binary event attribute that refers to the verification result, which can be OK or NOK, I is the interest, and finally, D is a binary attribute whether the assessment decision is OK or NOK.

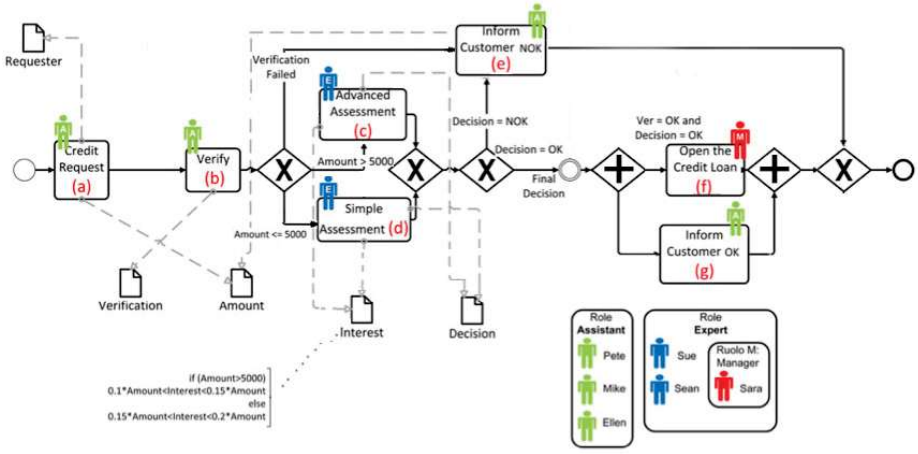


Figure 5.3: A simplified version of BPMN Model of credit request handling process from de Leoni and van der Aalst [2013a]

Case no.	Process Execution Activities and Attributes
Case 1	$< (a, \{A = 7000, R = Ana, E_a = Pete, T_a = 13Feb\}), (c, \{I = 150, D = OK, E_c = Sue, T_c = 16Feb\}),$ $(g, \{E_g = Pete, T_g = 17Feb\}), (f, \{E_f = Sara, T_f = 18Feb\}) >$
Case 2	$< (a, \{A = 3000, R = Alice, E_a = Pete, T_a = 03Jan\}), (d, \{I = 450, D = NOK, E_d = Ellen, T_d = 3Jan\}),$ $(b, \{V = OK, E_b = Sue, T_b = 05Jan\}), (d, \{I = 450, D = OK, E_d = Ellen, T_d = 7Jan\}), (g, \{E_g = Pete, T_g = 17Jan\}),$ $(f, \{E_f = Sara, T_f = 18Jan\}) >$
Case 3	$< (a, \{A = 5000, R = Jack, E_a = Pete, T_a = 04Nov\}), (b, \{V = OK, E_b = Ellen\}),$ $(c, \{I = 750, D = OK, E_c = Sue, T_c = 06Nov\}), (e, \{E_e = Pete, A = 5000, T_e = 11Nov\}) >$
Case 4	$< (a, \{A = 7000, R = Sam, E_a = Pete, T_a = 07Dec\}), (b, \{V = OK, E_b = Sue\}),$ $(c, \{I = 1050, D = OK, E_c = Pete, T_c = 9Dec\}), (d, \{I = 1050, D = NOK, E_d = Sue, T_d = 16Dec\}),$ $(e, \{E_e = Pete, A = 7000, T_e = 13Dec\}) >$
Case 5	$< (a, \{A = 3000, R = John, E_a = Pete, T_a = 04Apr\}), (b, \{V = OK, E_b = Ellen, T_b = 24Apr\}),$ $(d, \{I = 150, D = NOK, E_d = Ellen, T_d = 29Apr\}), (e, \{E_e = Pete, A = 6000, T_e = 1May\}) >$
Case 6	$< (a, \{A = 2000, R = John, E_a = Pete, T_a = 04Oct\}), (d, \{I = 300, D = OK, E_d = Pete, T_d = 9Oct\}),$ $(b, \{V = OK, E_b = Ellen, T_b = 11Oct\}), (g, \{E_g = Pete, T_g = 18Oct\}), (f, \{E_f = Sara, T_f = 23Oct\}) >$

Table 5.4: Example cases with attributes for credit request process

Case no	Trace Activities	Normal/Deviating case
Case 1	$\langle a, c, g, f \rangle$	Deviating case
Case 2	$\langle a, d, b, d, g, f \rangle$	Deviating case
Case 3	$\langle a, b, c, e \rangle$	Normal case
Case 4	$\langle a, b, c, d, e \rangle$	Deviating case
Case 5	$\langle a, b, d, e \rangle$	Normal case
Case 6	$\langle a, d, b, g, f \rangle$	Deviating case

Table 5.5: Example traces for credit request process with the output of conformance checking tool

As described before, the output of conformance checking techniques is two sets of cases: normal cases and deviating cases. The cases in the normal set follow the business rules and process models, so they are in line with the organizations' expected behavior. The deviation set contains cases that are not conforming the normative process but consists of both true-positives and false-positives. The true-positives are *anomalies* and the false-positives are the *exceptional cases* from the auditor's point of view. Table 5.5 shows the example traces of the credit request process from the control-flow perspective with the output of any of the current conformance checking tools.

In the following sections the approach is explained in detail. First, the deviation classification framework is described in more detail. Stage 1 is the mapping from model–log mismatches to the four deviation categories (missing, insertion, reordering, and repetition) is described based on the selected conformance checking mechanism. Next, the incremental classification procedure is presented, showing how auditor labels are collected and propagated across similar cases through successive levels of information, and how remaining unclassified executions are handled to ensure classification of the full population.

5.2.3 Process Deviation Classification Framework in Detail

This section describes the operational steps of the deviation classification framework. The framework starts with model–log comparison to identify deviations and structures them to support auditor-oriented deviation analysis within audit procedures. As discussed above, model–log comparison typically yields a large set of deviating cases. This is partly due to the fact that the normative model does not capture all business rules, while additional implicit rules may exist that are not formally documented. As

a result, the classification of many deviations requires expert judgment from auditors who are aware of, or can obtain, such implicit rules in order to distinguish true anomalies from acceptable exceptions. Analyzing these cases individually is often impractical. To address this challenge, the proposed framework groups deviations and presents them to auditors using iterative refined levels of information, enabling classification with the minimum information necessary.

The following subsections first introduce Stage 1, which maps model–log mismatches to high-level deviation types forming the basis for subsequent analysis. This is followed by Stage 2, which describes the iterative deviation analysis, to minimize the information overload for auditor at each step and reduce the manual classification effort and Stage 3, which focuses on labeling the remaining deviations to ensure full population classification.

5.2.4 Stage 1: Mapping Model–Log Mismatches to Deviation Types

The first stage of the deviation classification framework focuses on identifying control-flow mismatches between observed process executions and the normative process model, and mapping these mismatches to a finite set of high-level deviation types. This stage constitutes the foundation of the framework, as it translates technical conformance checking outputs into interpretable deviation representations that align with auditors' mental models. To detect mismatches, each process execution recorded in the event log is compared against the normative process model, which serves as the reference for expected behavior. The comparison is performed at the control-flow level, examining both the occurrence and the ordering of activities in the observed execution relative to the model. The outcome of this comparison is a set of differences that indicate where and how the observed execution deviates from the normative behavior. Rather than exposing these differences in a technical or atomic form, the framework maps them to four predefined deviation categories: missing, insertion, reordering, and repetition. A missing deviation occurs when an activity that is expected according to the normative model does not appear in the observed execution. An insertion deviation corresponds to the occurrence of an activity in the execution that is not expected at that point according to the model. A reordering deviation captures situations in which activities occur in a different sequence than prescribed by the model, while a repetition deviation indicates that an activity is executed more times than expected.

To reduce the complexity at the earliest stage of analysis, and to build on an interpretable and auditor-oriented deviation representation, this mapping step is designed to abstract from technical model–log mismatches and to provide a uniform, high-level representation of deviations that is meaningful for auditors. The selected

deviation types correspond directly to the deviation categories identified in Chapter 4 and reflect how auditors conceptualize deviations in practice, as evidenced by the field study results. The goal is to preserve the detection capability of conformance checking and, at the same time, produce a representation that is interpretable and suitable for subsequent deviation classification.

The output of Stage 1 is, for each process execution, a structured set of deviation types describing how its control-flow behavior deviates from the normative model. These deviation types serve as the primary input for the subsequent iterative deviation analysis stage, where they are grouped and enriched with additional information to support efficient classification as anomalies or exceptional cases.

We use part of the conformance checking technique proposed by García-Bañuelos et al. [2018] as the starting point for detecting differences between the event log and the process model. The technique creates a Prime Event Structure (PES) from the event log and a PES from the process model, and then compares these structures using a Partially Synchronized Product (PSP). Figure 5.4 provides an overview of this method.

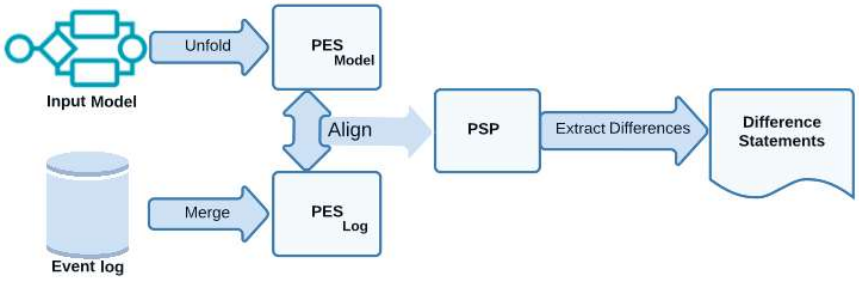


Figure 5.4: The alignment method used in García-Bañuelos et al. [2018]

In this step, we create Petri-net model of the normative model. This is required because the proposed algorithm builds upon the conformance checking technique introduced by García-Bañuelos et al. [2018] is defined for Petri-net–based process models. Figure 5.5 shows the Petri net model of the BPMN model in Figure 5.3. The Petri-net graph consists of places depicted in circles and transactions depicted in rectangles. Transactions represent the activities, and each of them has a set of input places and a set of output places.

the PES of the process model is constructed by unfolding the model behavior into an event structure that captures causal dependencies, conflicts, and concurrency relations. In this thesis, the PES derived from the model is denoted as PES_{model}. As shown in Figure 5.6, nodes are labeled by an event identifier and an activity label. An

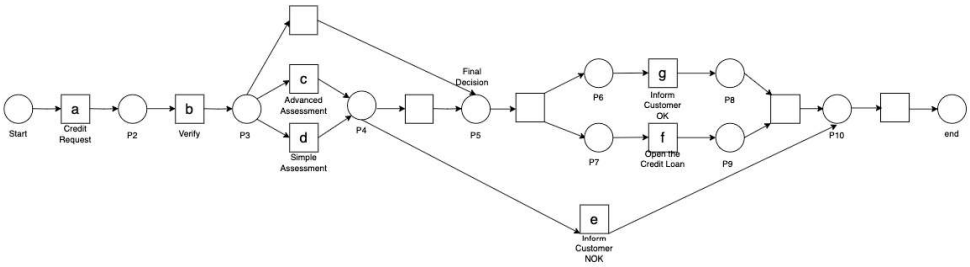


Figure 5.5: Petri Net model of the Credit Request model of Figure 5.3

example is the first node e0:A, which indicates that the first event, e0, represents an occurrence of activity A. Solid edges from each node to the following node represent causal dependencies, dotted edges represent conflict relations, and the absence of both indicates concurrency.

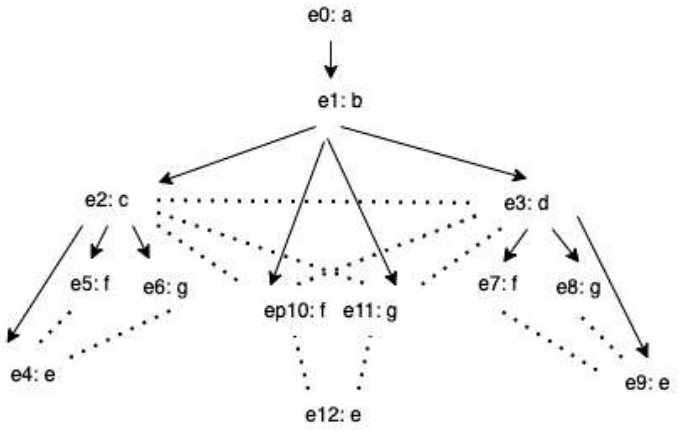


Figure 5.6: Prime Event Structure created from the credit request model (PES Model)

To construct the PES from the event log, PES_{log}, the event log is processed in two steps. First, a set of partially ordered runs is derived from the observed traces in the event log using the $\alpha+$ concurrency oracle, proposed in de Medeiros et al. [2003]. Figure 5.7 illustrates the set of runs associated with the traces in our credit request example. For each of the six traces in Table 5.4, its corresponding run is created. in Table 5.4. The notation *e:A* indicates the *e*-th event in the trace that represents

an occurrence of activity A in the log. Second, these set of runs are merged into a single event structure by combining events that share the same activity label and the same history (i.e., the same prefix). The resulting structure is denoted as PES_{log}. Figure 5.8 shows the PES constructed from the set of runs of the example shown in Figure 5.7. The notation $e1, e2...ei : A$ indicates that events $e1, e2...ei$ represent occurrences of activity A in individual runs.

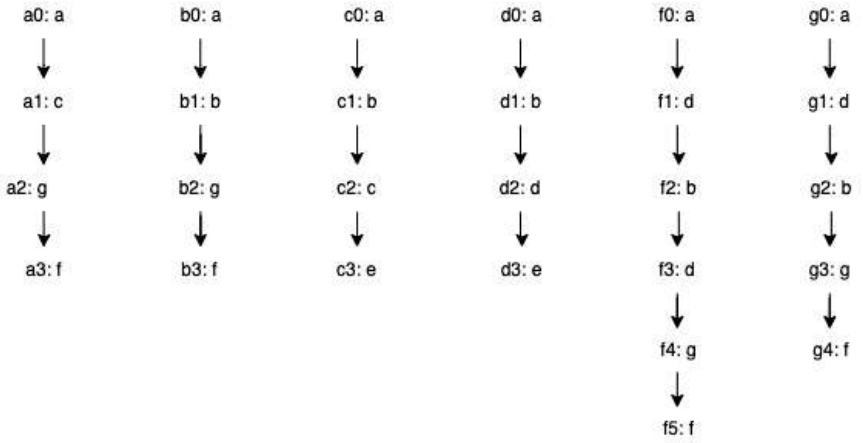


Figure 5.7: Runs from the credit request event log example in this thesis

After constructing PES_{model} and PES_{log}, the technique compares them by building a Partially Synchronized Product (PSP). The PSP aligns states from PES_{log} and PES_{model} to identify correspondences and mismatches. When two compared states carry the same activity label, an event match is asserted in the PSP, the PSP records a "match" operation. For example, consider PES_{Model} and PES_{Log} from 5.6 and 5.8, the initial state $e0$ in PES_{model} and the corresponding initial event in PES_{log} both carry the label "a". Therefore, the PSP asserts a match when comparing these two initial states. When synchronization is not possible because an event in one structure does not have a corresponding event in the other, the PSP proceeds by marking one side as "hidden". Following García-Bañuelos et al. [2018], we place PES_{log} on the left-hand side and PES_{model} on the right-hand side. Consequently, when there is a state in the PES_{Log} and there is no matched event in the PES_{model} (i.e., a log-only event) it is called *"lhide* α*"*. If there is event $\alpha\prime$ in PES_{Model} and no matched event in PES_{Log} (i.e., a model-only event), it is called *"rhide* $\alpha\prime$*"*.

In the alignment procedure of García-Bañuelos et al. [2018]'s conformance checking technique, the notion of position is derived from the progression of the alignment. At

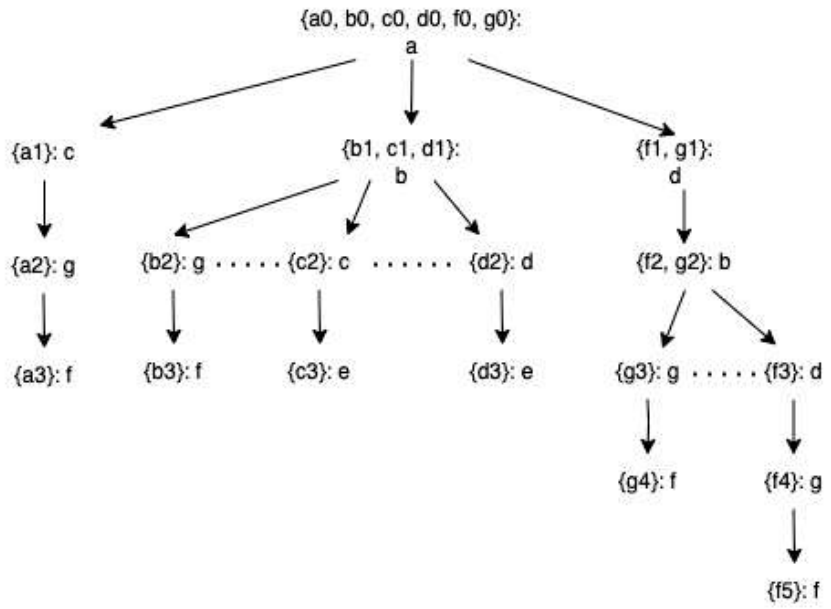


Figure 5.8: Prime Event Structure constructed from the set of runs created in Figure 5.7 (PES log)

each step, the algorithm considers a current candidate pair (e_1, e_2), where $e_1, e_2 \in \mathbb{N}$, i.e., integers serving as positional identifiers associated with events in the log and in the model, respectively.

The alignment maintains a set of previously matched events (f_1, f_2), corresponding to log events and model events. This formulation is consistent with the definition of event matching in the PSP, where "two requirements for two events to be matched in the PSP" are imposed, namely that "an event matching must be label preserving (i.e. both events must have the same label) and order-preserving (i.e. event matchings in the PSP are consistent with the causal relation of the input PESs)" [García-Bañuelos et al., 2018]. The order-preserving condition is evaluated by comparing the current candidate pair (e_1, e_2) with all previously matched pairs (f_1, f_2). That means the relative behavioral relation between f_1 and e_1 in the log is identical to the relation between f_2 and e_2 in the model, which can be expressed as:

$$\forall (f_1, f_2), \quad \mathrm{rel}_{\log}(f_1, e_1) = \mathrm{rel}_{\mathrm{model}}(f_2, e_2)$$

Based on this alignment, three types of operations are distinguished. A *match* corresponds to the synchronization of a log event and a model event with identical

labels that satisfy these conditions. A *lhide* operation represents the consumption of a log event without a corresponding model event, indicating an insertion in the log. Conversely, a *rhide* operation represents the consumption of a model event without a corresponding log event, indicating a missing event in the log.

Consequently, positions are interpreted relationally rather than absolutely, i.e., their validity depends on consistency with previously established correspondences,amd not only the fixed sequential indices.

The PSP output is informative but still too technical for the purposes of this thesis. Therefore, we abstract the PSP operations into deviation types that correspond to the deviation categories introduced in Chapter 4. The mapping is performed as follows. A log-only event ("lhide") indicates that an activity occurs in the observed execution that is not expected at that point according to the model; this mismatch is mapped to an *Insertion*. A model-only event ("rhide") indicates that an activity is expected by the model but does not occur in the observed execution; this mismatch is mapped to a *Missing*. At this stage, the algorithm yields a preliminary list of insertions and missing activities for each trace.

In the next step, the algorithm refines these preliminary mismatches into higher-level deviation types on the whole dataset. If an insertion of a specific activity is observed and a missing of the same activity is also observed later in the mismatch sequence, this indicates that the activity is present but occurs in the wrong position. The algorithm therefore maps this pair to a *Reordering* deviation rather than reporting one insertion and one missing. In addition, if an activity is reported as inserted while the same activity has already occurred as a matched event earlier in the trace, the insertion reflects an additional execution of an already expected activity; this situation is mapped to a *Repetition*. Algorithm 1 summarizes the mapping rules.

By applying these mapping rules, model–log mismatches are translated into the four deviation categories identified in Chapter 4, namely Missing, Insertion, Reordering, and Repetition. These deviation types serve as the output of Stage 1 and provide the input to Stage 2, where deviations are incrementally presented to auditors for classification as anomalies or exceptional cases. Considering the deviating traces in the credit request example, Trace 1 contains a deviation of type Missing(b). Trace 2 contains Reordering(b) and Repetition(d). Trace 4 contains an Insertion(d). Trace 6 contains Reordering(b). These results illustrate how the PSP mismatch operations are transformed into auditor-oriented deviation types that are interpretable and aligned with the deviation categories defined in the thesis.

Algorithm 1 Abstraction of PSP operations into deviation categories and types

Given:
a normative model M (Petri net) and an event log L (set of traces).
$PosSeq(\sigma) \leftarrow \langle\rangle$ ▷ ordered sequence of (e_1, e_2, a, op) with $\perp$ for missing components
$e_1, e_2 \in \mathbb{N}$ are positional identifiers for current event in the log and current event in the model
For each trace $\sigma \in L$, let $Diff(\sigma)$ denote the PSP operations returned by the conformance checker
Output: a set of deviation types $\mathcal{D}(\sigma)$ for each trace σ.
for all $\sigma \in L$ **do**
  $PreDev(\sigma) \leftarrow \emptyset$
  $\mathcal{D}(\sigma) \leftarrow \emptyset$
  for all $op(e_1, e_2, a) \in Diff(\sigma)$ **do**
    if $op = match(a)$ **then**
      append $(e_1, e_2, a, match)$ to $PosSeq(\sigma)$
      $PreDev(\sigma) \leftarrow PreDev(\sigma) \cup \{a\}$
    else if $op = lhide(a)$ **then**
      append $(e_1, \perp, a, lhide)$ to $PosSeq(\sigma)$
      $PreDev(\sigma) \leftarrow PreDev(\sigma) \cup \{insertion(a)\}$
    else if $op = rhide(a)$ **then**
      append $(\perp, e_2, a, rhide)$ to $PosSeq(\sigma)$
      $PreDev(\sigma) \leftarrow PreDev(\sigma) \cup \{missing(a)\}$
    end if
  end for
  for all a involved in $PreDev(\sigma)$ **do**
    if $insertion(a) \in PreDev(\sigma)$ **and** $missing(a) \in PreDev(\sigma)$ **then**
      $\mathcal{D}(\sigma) \leftarrow \mathcal{D}(\sigma) \cup \{reordering(a)\}$
    else if $insertion(a) \in PreDev(\sigma)$ **and** a occurs earlier in a $match(a)$ **then**
      $\mathcal{D}(\sigma) \leftarrow \mathcal{D}(\sigma) \cup \{repetition(a)\}$
    else if $insertion(a) \in PreDev(\sigma)$ **then**
      $\mathcal{D}(\sigma) \leftarrow \mathcal{D}(\sigma) \cup \{insertion(a)\}$
    else if $missing(a) \in PreDev(\sigma)$ **then**
      $\mathcal{D}(\sigma) \leftarrow \mathcal{D}(\sigma) \cup \{missing(a)\}$
    end if
  end for
end for

5.2.5 Stage 2: Iterative Deviation Analysis

After mapping model–log mismatches to high-level deviation types in Stage 1, the second stage of the framework focuses on the iterative, multi-level analysis and classification of deviations. The objective of this stage is to support auditors in distinguishing anomalies from exceptional cases efficiently while minimizing information overload and manual classification effort. Instead of presenting all deviations together with all available information at once, the framework adopts an incremental and interactive analysis strategy. Deviations are systematically grouped and presented to the auditor across multiple levels of abstraction. It supports auditors in their classification decision by providing only the information that is necessary at each step.

Following Stage 1, each deviating process execution is associated with one or more deviations. Each deviation category may be further instantiated with respect to specific activities; these instantiations are referred to in this chapter as deviation types. For example, missing(Verification) denotes the absence of the verification activity in a given process execution. Table 5.6 illustrates the deviation types identified in the running example.

Category	Deviation Type
Missing	missing (c)
Insertion	insertion (c)
	insertion (d)
Reordering	reordering (b)
Repetition	repetition (c)

Table 5.6: Deviation types in credit request example

The overall aim of this stage is to allow auditors to classify deviations using the minimal amount of information required, thereby reducing information load and limiting unnecessary manual effort. Across all levels of analysis, auditors are asked to label deviations as OK (false-positive), NOK (true-positive), or leave them unlabeled (?) when the available information is insufficient to support a confident decision.

After each iteration, the framework propagates the auditors' labels to all process executions that exhibit the same deviation characteristics, thereby reducing the need for repeated manual classification. A process execution is classified as an anomaly if at least one of its deviation types is labeled as NOK, since a single true-positive deviation is sufficient to warrant further investigation of the entire case. Conversely,

if all deviation types associated with a process execution are labeled as OK, the execution is classified as an exceptional case, indicating that the observed deviations can be justified by valid but implicit rules that are not explicitly captured in the normative model.

Process executions that contain one or more unlabeled deviation types remain in the deviation set and are forwarded to the next iteration, where the deviation groups are enriched with additional information. Each iteration increases the level of detail in a systematic manner, for example by incorporating common deviation patterns and, when necessary, contextual information derived from case attributes and event attributes. This iterative refinement continues until deviations can be confidently classified or until no further informative refinement is possible. Throughout this process, classification decisions provided by the auditor are reused to automatically label similar deviations across the remaining population. As a result, the number of unclassified cases decreases with each iteration, and auditor effort is gradually focused on the most ambiguous and informative deviations. This design ensures full population coverage as well as maintaining a manageable information load.

The iterative deviation analysis is organized into four levels that incrementally enrich the information provided to the auditor. Level 1 presents deviations solely in terms of their high-level deviation types based on control-flow deviations. Level 2 augments this view by grouping deviations into frequent deviation patterns, to enable auditors to recognize recurring combinations of deviations across process executions. Level 3 further incorporates case-level contextual information, such as relevant case attributes. Finally, Level 4 introduces event-level information, including event attributes and derived features, to facilitate the classification. The event attributes level is the lowest level of information in the event log. Level 3 and 4 aim to support the classification of deviations that depend on business context not explicitly captured in the normative model. The following subsections describe these levels in detail.

5.2.5.1 Level 1– Deviation Identification and Type Assignment: Control-flow perspective

The first level of analysis focuses on individual deviation types from the control-flow perspective. At this level, deviation types are presented in isolation, without contextual or data-related information. For certain deviations, this level is sufficient for the auditor to decide whether the deviation is an anomaly or not. For instance, in the running example, *repetition(Verification)*, although it might indicate inefficiency in the system, can be immediately classified as acceptable when repeated verification yields the same outcome. In contrast, *missing(Verification)* is considered critical

regardless of any other information and it is therefore classified as an anomaly (i.e., true-positive). As a result, some process executions can be classified and removed from the deviation set at this early stage.

Algorithm 2 provides the pseudo-code of how Level 1 is designed to perform an initial classification of deviating process executions based solely on individual deviation types. For each deviating trace $\sigma \in D$, the framework considers the set of deviation types $\delta \in \Delta$, where each deviation type δ represents the occurrence of deviation for activity $a \in \mathcal{A}$. All distinct deviation types observed in D are presented to the auditor for labeling as *OK*, *NOK*, or undecidable, and the decisions are captured by the labeling function ℓ_1. These labels are then propagated to the trace level: any trace containing at least one deviation type labeled *NOK* is classified as an anomaly and moved to the anomaly set *Anom*, whereas traces for which all deviation types are labeled *OK* are classified as exceptions and moved to the exception set *Excp*. Traces that cannot be conclusively classified at this level remain in D and are forwarded to Level 2 for further analysis.

Considering the deviations mentioned in our four traces of the credit request process, the deviation in Case 1 has a *missing(b)*, the verification. Case 2 contains a *reordering(b)*, the verification and *repetition(d)*, the assessment. Case 3 is a normal case from the control-flow perspective. Case 4 has an insertion(d) or insertion(c), the simple assessment. Case 5, again, is a normal case from the control-flow perspective. Case 6 contains a *reordering(b)* deviation. Table 5.7 shows the deviation types in Cases 1, 2, 4, and 6 with the auditor's labels for each. The auditor labels missing (b) as NOK, because the verification activity is a critical step and should never be missed in the process. The other deviations are labeled as ?. Consequently, the algorithm classifies Case 1 as an anomaly at this step and removes it from the set of deviating cases.

Category	Deviation Type	Label (OK, NOK, ?)
Missing	missing (b)	NOK
Insertion	insertion (c)	?
	insertion (d)	?
Reordering	reordering (b)	?
Repetition	repetition (c)	?

Table 5.7: Deviation types in credit request example, labeled by an auditor

At the end of this level, Case 1 will be classified as an anomaly and removed from

Algorithm 2 Level 1 classification using deviation types

Given:

D: set of deviating traces (process executions), $|D| = N$ where $mathcalT = \{\text{MIS}, \text{INS}, \text{REO}, \text{REP}\}$-deviation types

$Anom \leftarrow \emptyset$: anomaly set

$Excp \leftarrow \emptyset$: exception set

for each $\sigma \in D$ **do**

$\Delta(\sigma) = \{\delta \in \Delta \mid \delta.c.\text{trace} = \sigma\}$ set of deviations for trace σ

end for

For a trace σ, let

$$DT(\sigma) = \{(\delta.t, \delta.a) \mid \delta \in \Delta(\sigma)\}$$

distinct (type, activity) pairs denote the set of deviation types in σ.

$\ell_1(t, a) \mapsto \{\text{OK}, \text{NOK}, ?\}$ auditor labeling function over deviation types over deviation types

(1) Collect labels for deviation types

for all distinct (t, a) such that $\exists \sigma \in D$ **do**

Present (t, a) to the auditor to label as *NOK*, *OK*, or ?

Record the label as $\ell_1((t, a))$

end for

(2) Propagate labels and classify traces

for all $\sigma \in D$ **do**

if $\exists\,(t, a) \in DT(\sigma)\ :\ \ell_1((t, a)) = NOK$ **then**

$Anom \leftarrow Anom \cup \{\sigma\}$

Remove σ from D

else if $\forall\,(t, a) \in DT(\sigma)\ :\ \ell_1((t, a)) = OK$ **then**

$Excp \leftarrow Excp \cup \{\sigma\}$

Remove σ from D

else

/ σ remains in D for the next level */*

end if

end for

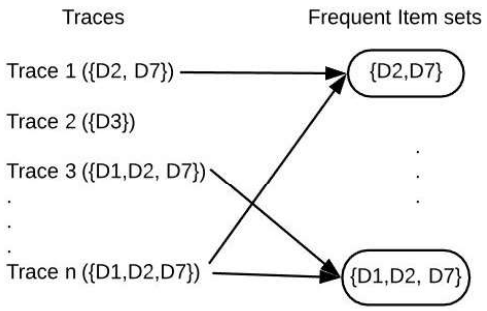


Figure 5.9: Mining frequent sequence pattern in Level 2

the set of deviating cases. The rest of deviating cases remain.

5.2.5.2 Level 2– Combination of deviation types: Control-flow perspective

The second level extends the analysis by considering combinations of deviation types based on the sequence in which they occurred after each other within process executions. Deviation sequences are extracted using a sequence mining algorithm and are presented to auditors in descending order of support. This level enables auditors to reason about recurring deviation patterns rather than isolated deviations. Process executions containing deviation sequences labeled as NOK are classified as anomalies, while those whose deviation sequences and individual deviation types are all labeled as OK are classified as exceptional. Deviations that cannot yet be assessed remain for further analysis.

The frequent closed sequential patterns mining algorithm proposed by Gomariz et al. [2013] is applied in this level. Some definitions of the frequent sequence pattern mining problem are modified to adapt them to suit the proposed approach in this level. The definitions are as follows:

Definition 12 (Set of items)**.** *The set of items is defined as* $D = \delta_1, \delta_2, \delta_3, \ldots, \delta_m$*, where each* δ_i *refers to each deviation in the event log* L *and* m *refers to the number of deviations in* L*.*

Definition 13 (Sequence and sequential pattern)**.** *A sequence is an ordered list of itemsets:* $s = \langle D_1, D_2, \ldots, D_k \rangle$*, where each* $D_j \subseteq D$*.*

A sequential pattern is a sequence $p = \langle P_1, P_2, \ldots, P_m \rangle$ *a set of deviation types which is repeated frequently in the event log as a pattern. (e.g., see Figure 5.9).*

Definition 14 (Sequence database). *A sequence database DB is a set of sequences $\{s_1, s_2, \ldots, s_n\}$ over D.*

Given these definitions, the support of a sequence pattern is defined as follows:

Definition 15 (Support of a sequential pattern). *The support of a sequential pattern p in DB is*

$$sup(p) = \frac{|\{s \in DB \mid p \sqsubseteq s\}|}{|DB|}.$$

A pattern is frequent *if $sup(p) \geq \theta$ for a minimum support threshold $\theta \in (0, 1]$.*

For example, if $x_1 = \{D1, D4, D7\}$ is a sequence pattern, and the event log L has 100 deviating instances and x_1 has occurred in 30 instances, then the support of sequence pattern x_1 will be 30/100.

Definition 16 (Frequent sequence pattern). *A sequence pattern is called frequent if its support is equal or greater than a given minimal support threshold θ, where $0 \leq \theta \leq m$.*

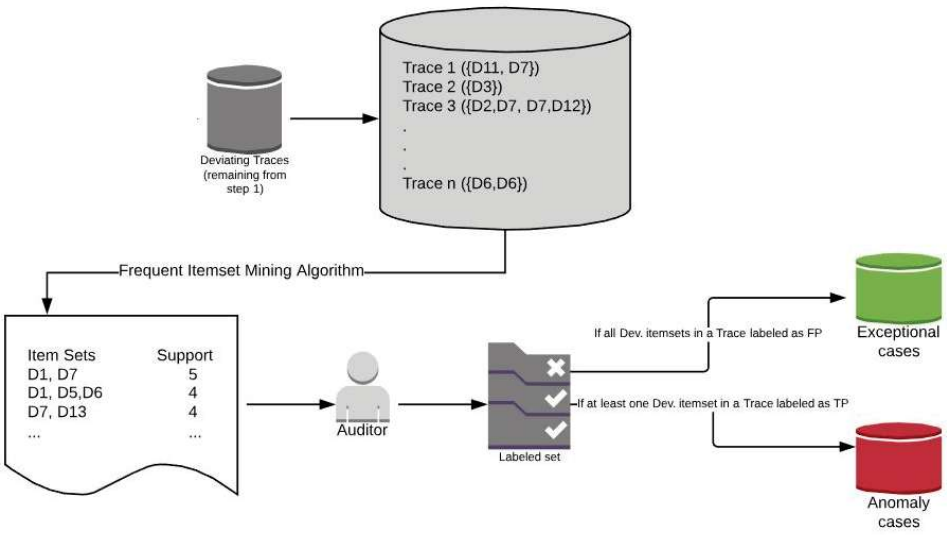


Figure 5.10: Labeling the frequent sequence patterns in Level 2

Figure 5.10 summarizes how we apply *frequent closed sequence pattern mining* in this step. When the sequence patterns are discovered, the framework sorts them based on their supports (from the largest support to the smallest support) and presents them to the auditor. The auditor labels the deviation patterns as OK, NOK, and leaves others unlabeled. Afterward, the framework applies these labels to the deviation

Frequent deviation sequences	Support	Label(OK, NOK, ?)
⟨reordering(b), repetition(d)⟩	1/6	?

Table 5.8: The frequent deviation sequences created in Level 2 together with their support and label as OK, NOK, and ?

sets and classifies the cases. Similar to Level 1, the cases with NOK (true-positive) sequence patterns are moved to the anomaly set. Furthermore, if all the sequence patterns and deviating types in a case are labeled as OK (false-positive), the tool moves it to the exception set.

Algorithm 3 below provides the pseudo-code of how Level 2 designed to extend the initial classification of individual deviation types by considering their sequential occurrence within process executions. For each deviating trace $\sigma \in D$, the framework constructs a deviation-type sequence $Seq(\sigma)$ that preserves the order in which deviation types occur in the trace. Frequent deviation type sequences are then mined from the set $\{Seq(\sigma) \mid \sigma \in D\}$ using a minimum support threshold θ_2 and are sorted in descending order of support. These frequent sequences are presented to the auditor for labeling as *OK*, *NOK*, or *?*. The labels are recorded by the labeling function ℓ_2. The assigned labels are subsequently propagated to all deviating traces: any trace containing at least one frequent sequence labeled *NOK* is classified as an anomaly and moved to *Anom*, whereas traces for which all matched frequent sequences and all individual deviation types (as labeled in Level 1) are labeled *OK* are classified as exceptions and moved to *Excp*. Traces that cannot be conclusively classified at this level remain in D and are forwarded to Level 3 for further analysis.

Consider Case 2 in our credit request example; it has a combination of two deviations. First, the verification is reordered (reordering(b)) and then the assessment is repeated (repetition(d)). When the combination of deviations is presented to the auditor it turns out that the sequence of two deviations is not a meaningful combination for the auditor, and there is no rule to clear it. Therefore, this combination of deviations is still labeled as ? at the end of Level 2. Table 5.8 shows this deviation sequence together with its support and label.

5.2.5.3 Level 3– Case level- adding data perspective

In the third level, additional contextual information is introduced at the case-level. Alongside control-flow deviations and their sequences, relevant information from the data perspective at the case-level is incorporated. Consider the credit request ex-

Algorithm 3 Level 2 classification using frequent deviation-type sequences

Given:

D: set of remaining deviating traces from Level 1
$Seq(c)$: deviation sequence of case c (Definition 9a)
θ_2: minimum support threshold for Level 2
F_2: set of frequent sequential patterns with $\sup(p) \geq \theta_2$
$Anom$: anomaly set
$Excp$: exception set
$\mathcal{C}$: deviation categories; $\mathcal{A}$: activities
ℓ_1: labeling function from Level 1
ℓ_2: auditor labeling function for patterns, $\ell_2(p) \in \{\text{OK}, \text{NOK}, ?\}$

(1) Mine frequent deviation-type sequences
Compute the set F_2 of frequent sequences from $\{Seq(c) \mid c \in D\}$
For each $p \in F_2$, compute $supp(p)$ and keep only those with $supp(p) \geq \theta_2$
Sort F_2 in descending order of $supp(\cdot)$

(2) Collect labels for frequent sequences
for all $p \in F_2$ (in sorted order) **do**
  Ask the auditor to label p as *NOK*, *OK*, or ?
  Record the label as $\ell_2(p)$
end for

(3) Propagate labels and classify traces
for all $\sigma \in D$ **do**

  if $\exists p \in F_2 \ : \ p \sqsubseteq Seq(c) \wedge \ \ell_2(p) = NOK$ **then**
    $Anom \leftarrow Anom \cup \{\sigma\}$
    Remove σ from D
  else if $\Big(\forall\, p \in F_2 : (p \sqsubseteq Seq(c) \Rightarrow \ell_2(p) = OK)\Big) \ \wedge \ \Big(\forall\, (t,a) \in DT(\sigma) : \ \ell_1((t,a)) = OK\Big)$ **then**
    $Excp \leftarrow Excp \cup \{\sigma\}$
    Remove σ from D
  else
    / σ remains in D for Level 3 */*
  end if
end for

ample: if the requested loan amount is below a certain threshold (e.g., 200 euros), sending a notification to the client may not be required. Such rules are often not explicitly modeled because these cases are rare, yet they are known to process owners. In this example, when the deviation is *missing(g)* for specific requests, the deviation type alone is insufficient for classification; the auditor also needs to know the requested loan amount. To assess such cases, auditors must examine a case attribute, namely *Amount*. Therefore, for analyzing process instances that contain *missing(g)*, case-level data attributes must be provided by the framework.

At this level, case attributes such as the requested amount are integrated into the analysis. This enrichment enables auditors to assess deviations that depend on business context not explicitly encoded in the normative model. Frequent combinations of deviation types and case attribute values are identified and presented to the auditor for labeling. Based on the auditors' decisions, additional process executions are classified and removed from the deviation set.

In Level 3, the position of deviation types relative to other normal events in the process instance is also considered, as the timing of a deviation is often relevant for audit judgment. In addition to control-flow information, case attribute data are added to the instances that remained unclassified in previous levels. Similar to Level 2, frequent sequence pattern mining is applied to identify frequent deviation sequences enriched with case attribute values. A minimum support threshold is used to select the most frequent itemsets, which are then presented to the auditor for classification. Using the combined control-flow and case-level information, the auditor labels the itemsets as OK, NOK, or leaves them unlabeled when the information is still insufficient. Process executions containing itemsets labeled as NOK are moved to the anomaly set, while those for which all deviations and itemsets are labeled as OK are moved to the exception set, consistent with the previous levels. Figure 5.11 illustrates the creation of frequent itemsets from deviation types and case attributes.

Algorithm 4 below provides the pseudo-code about how Level 3 extends the classification by incorporating case-level contextual information in addition to control-flow deviations. For each remaining case $c \in D$, a *case-level deviation sequence* $S_3(c)$ is constructed by combining case attributes $c.ca \subseteq \mathcal{CA}$ with ordered deviation events (t_i, a_i), where $t_i \in \mathcal{T}^*$ denotes the event in the case and $a_i \in \mathcal{A}$ its corresponding activity. Frequent sequential patterns $\mathcal{F}_3$ are mined from these sequences using a minimum support threshold θ_3 and are presented to the auditor for labeling as *OK*, *NOK*, or *?*. The obtained labels are then propagated to all cases: cases containing at least one pattern labeled *NOK* are classified as anomalies and moved to *Anom*, whereas cases for which all matched patterns and all previously identified deviation

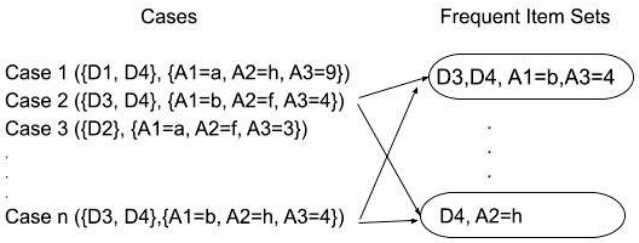


Figure 5.11: Cases with case attributes and frequent itemsets in Level 3

Frequent deviation sequences and cases attributes	Support	Label
⟨ insertion(d), Amount>5000 ⟩	1/6	OK

Table 5.9: The frequent sequences in Level 3 and their supports. The itemsets are labeled by the auditor as OK, NOK, or ?

types are labeled *OK* are classified as exceptions and moved to *Excp*. Cases that cannot be conclusively classified remain in $\mathcal{D}$ for further analysis at Level 4.

Case 4 in our running example, has a deviation: either advanced assessment or simple assessment is inserted. To decide about this case, we need to know the requested amount, which is a case attribute. According to the case attributes, the amount is 7000 for this case. As mentioned in the model, if the amount is 5000 or more, an advanced assessment is needed. Therefore, activity c is expected, therefore the deviation is *insertion(d)*. As the advanced assessment is more comprehensive than the simple assessment, the auditor labels this deviation as OK. Table 5.9 shows this deviation together with its related case attribute. Note that we do not mention all other deviations with their case attributes in the table.

At the end of this level, Case 4 is classified as a normal case because its only deviation is labeled OK. As a result, the algorithm moves it from the set of deviating cases to the set of normal cases.

5.2.5.4 Level 4– Event level- adding data perspective

The fourth and final level incorporates event-level data, which represents the most granular information available in the event log. At this level, deviation types are

Algorithm 4 Level 3 deviation classification (adding case-level attributes)

Given:

$\mathcal{D}$: set of remaining deviating traces from Level 2
Anom: anomaly set
$\mathcal{T}^* = \mathcal{T} \cup \{N\}$, where N denotes a normal (non-deviating) event
For each case c, its Level 3 sequence is defined as

$$S_3(c) = \Big(ca(c), \langle (t_1, a_1), \ldots, (t_n, a_n) \rangle\Big),$$

where $ca(c) \subseteq \mathcal{CA}$ is the set of case-level attributes,
$t_i \in \mathcal{T}^*$, $a_i \in \mathcal{A}$(activities), and $t_i = N$ for non-deviating positions
θ_3: minimum support threshold for Level 3
$\mathcal{F}_\ni$: set of frequent sequential patterns from $\{S_3(c) \mid c \in \mathcal{D}\}$ with $\text{supp}(p) \geq \theta_3$
ℓ_1, ℓ_2: labeling functions from Levels 1 and 2
ℓ_3: auditor labeling function for Level 3 patterns, $\ell_3(p) \in \{OK, NOK, ?\}$

(1) Construct enriched sequences
for all $c \in D$ **do**
Construct

$$S_3(c) = \Big(ca(c), \langle (t_1, a_1), \ldots, (t_n, a_n) \rangle\Big)$$

where $t_i = \delta_i.t$ if position i contains a deviation δ_i, and $t_i = N$ otherwise
end for
(2) Mine frequent patterns
Construct sequence database $DB_3 = \{S_3(c) \mid c \in \mathcal{C}_D\}$
$F_3 \leftarrow \text{FreqSeqMine}(DB_3, \theta_3)$

(3) Collect labels for frequent sequences
for all $p \in F_3$ (in sorted order) **do**
Present pattern p (with case attributes) to auditor to label as *NOK*, *OK*, or ?
Record $\ell_3(p)$
end for

(4) Propagate labels and classify traces
for all $c \in D$ **do**
if $\exists p \in F_3 \ : \ p \sqsubseteq S_3(c) \ \wedge \ \ell_3(p) = NOK$ **then**
$Anom \leftarrow Anom \cup c$
Remove c from D
else if $\Big(\forall p \in F_3 : \ p \sqsubseteq S_3(c) \Rightarrow \ell_3(p) = OK\Big) \wedge \Big(\forall p \in F_2 : \ p \sqsubseteq Seq(c) \Rightarrow \ell_2(p) = OK\Big) \wedge \Big(\forall (t,a) \in DT(\sigma) : \ \ell_1((t,a)) = OK\Big)$ **then**
$Excp \leftarrow Excp \cup \{c\}$
Remove c from $\mathcal{D}$
else
/ c remains in D for Level 4 */*
end if
end for

enriched with event attributes such as resource (originator) information, timestamps, and activity-specific data. When necessary, derived event attributes may be constructed to capture domain-specific rules, such as the relationship between verification and assessment outcomes in the running example. This level supports the classification of complex deviations that cannot be resolved using control-flow and case-level information alone.

As in the previous levels, sequence pattern mining is applied to identify the most frequent itemsets. These itemsets consist of deviation types combined with frequent event attribute values and are presented to auditors for labeling as OK, NOK, or left unlabeled when the available information is still insufficient. Based on the auditors' labels, the framework classifies the remaining deviating process executions and determines whether they can be moved to the exception or anomaly set, consistent with the procedure applied in the earlier levels.

Figure 5.12 depicts how frequent itemsets are created from case attributes and event attributes.

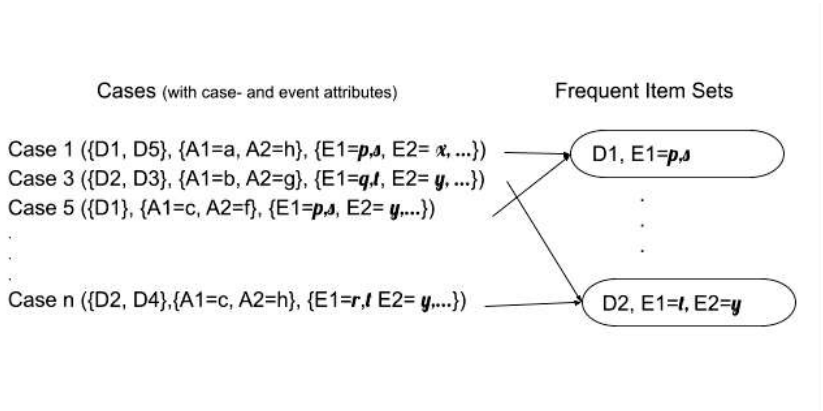


Figure 5.12: Cases with case attributes and event attributes and frequent itemsets in Level 4.

Algorithm 5 summarizes the pseudocode on how Level 4 further refines the classification by incorporating event-level information in addition to control-flow deviations and case attributes. For each remaining case $c \in D$, an *event-level deviation sequence* $S_4(c)$ is constructed by extending the case-level representation with event attributes, yielding tuples (t_i, a_i, ea_i), where $t_i \in \mathcal{T}^*$ denotes the event, $a_i \in \mathcal{A}$ its corresponding activity, and ea_i the associated event attributes. Frequent sequential patterns $\mathcal{F}_4$ are mined from these sequences using a minimum support threshold θ and are presented to the auditor for labeling as *OK*, *NOK*, or *?*. The obtained labels are then propagated to all remaining cases: cases containing at least one pattern labeled *NOK*

are classified as anomalies and moved to *Anom*, whereas cases for which all matched patterns and all previously identified deviation types are labeled *OK* are classified as exceptions and moved to *Excp*. This final level enables auditors to assess deviations whose classification depends on fine-grained, event-specific data that is not captured at higher abstraction levels.

Each level increases the expressiveness of deviation representation while progressively reducing the size of deviation groups.
Cases 2 and 6 in the credit request example exhibit the deviation type reordering(b), corresponding to the verification activity. Event **b** includes an event attribute **V**, which is a binary attribute indicating the outcome of the verification; it takes the value OK when the verification result is positive. A domain rule applies in this context: if the verification outcome (**V**) is OK and the assessment decision (Dec), an event attribute of the assessment activities (**c** or **d**), is also OK, the deviation can be exceptionally accepted.

To incorporate this rule within the framework, a derived event attribute, denoted as *Ver-Dec*, is introduced. This attribute captures the relationship between the verification and assessment outcomes by comparing **V** from event **b** with *Dec* from event **c** or **d**. The attribute Ver-Dec is defined as binary: it takes the value Yes when both V and *Dec* are OK, and No otherwise. Table 5.10 summarizes the definition, conditions, and possible values of this derived attribute.

New event attribute Ver-Dec	Value
If V from event b = OK and *Dec* from event c = OK	Yes
If V from event b = OK and *Dec* from event d = OK	Yes
Otherwise	NO

Table 5.10: A new event attribute *Ver-Dec* needs to be created in Level 4

Using this new event attribute, an auditor can label the frequent deviation sequences with the event attributes. Table 5.11 shows these sequences together with the support and labels.

At the end of Level 4, the reordering(b) deviation in Case 6 and Case 2 are labeled as OK. Consequently, we can remove Case 6 from deviating cases set and add it to the set of normal cases. However, although Case 2 is partially cleared at this level because we have no rule yet about repetition(d), we cannot decide about Case 2, and it remains in the deviating cases set.

Algorithm 5 Level 4 deviation classification (adding event-level attributes)

Given:

$\mathcal{D}$: set of remaining deviating traces from Level 3
Anom: anomaly set
Excp: exception set
$\mathcal{T}^* = \mathcal{T} \cup \{N\}$, where N denotes a normal (non-deviating) event
For each case c, its Level 3 sequence is defined as

$$S_4(c) = \Big(ca(c), \langle (t_1, a_1, ea_1), \ldots, (t_n, a_n, ea_n)\rangle\Big),$$

where $ca(c) \subseteq \mathcal{CA}$ is the set of case-level attributes,
$t_i \in \mathcal{T}^*$, $a_i \in \mathcal{A}$(activities), and $t_i = N$ for non-deviating positions
θ_4: minimum support threshold for Level 4
F_4: set of frequent sequential patterns from $\{S_4(c) \mid c \in \mathcal{D}\}$ with $\text{supp}(p) \geq \theta_4$
ℓ_1, ℓ_2, ℓ_3: labeling functions from Levels 1, 2 and 3
ℓ_4: auditor labeling function for Level 4 patterns, $\ell_4(p) \in \{OK, NOK, ?\}$

(1) Construct enriched sequences with event attributes
for all $c \in D$ **do**
Construct

$$S_4(c) = \Big(ca(c), \langle (t_1, a_1, ea_1), \ldots, (t_n, a_n, ea_n)\rangle\Big)$$

where $t_i = \delta_i.t$ if position i contains a deviation δ_i, and $t_i = N$ otherwise
and $ea_i = \sigma_i.ea$ (event attributes of the i-th event)
end for

(2) Mine frequent patterns
Construct sequence database $DB_4 = \{S_4(c) \mid c \in D\}$
$F_4 \leftarrow \text{FreqSeqMine}(DB_4, \theta_4)$

(3) Collect labels for frequent sequences
for all $p \in F_4$ (in sorted order) **do**
Present pattern p (with case and event attributes) to auditor to label as *NOK*, *OK*, or ?
Record the label as $\ell_4(p)$
end for

(4) Propagate labels and classify traces
for all $c \in D$ **do**
if $\exists p \in F_4 \ : \ p \sqsubseteq S_4(c) \ \wedge \ \ell_4(p) = NOK$ **then**
$Anom \leftarrow Anom \cup c$
Remove c from D
else if $\Big(\forall p \in F_4 \ : \ p \sqsubseteq S_4(c) \Rightarrow \ell_4(p) = OK\Big) \wedge \Big(\forall p \in F_3 \ : \ p \sqsubseteq S_3(c) \Rightarrow \ell_3(p) = OK\Big) \wedge \Big(\forall p \in F_2 \ : \ p \sqsubseteq Seq(c) \Rightarrow \ell_2(p) = OK\Big) \wedge \Big(\forall\, (t,a) \in DT(\sigma) : \ \ell_1((t,a)) = OK\Big)$ **then**
$Excp \leftarrow Excp \cup \{c\}$
Remove c from $\mathcal{D}$
end if
end for

Frequent deviation sequences and event attributes	Support	Label
⟨ reordering(b), Ver-Dec= Yes ⟩	1/3	OK
⟨ reordering(b), V= OK ⟩	1/3	?

Table 5.11: The frequent sequences in Level 4 and their supports. The itemsets are labeled by the auditor as OK, NOK, or ?

5.2.6 Stage 3: Labeling Remaining Deviations

After completing the iterative deviation analysis in Stage 2, the labeling decisions provided by the auditors have already been propagated to all similar deviations identified across the event log. Through this process, the majority of deviating process executions are classified as anomalies or exceptions based on control-flow patterns and incrementally enriched contextual information.

Nevertheless, a small number of deviations may remain unclassified. These remaining cases typically correspond to rare, complex, or highly context-specific process executions for which none of the previously presented deviation types, sequences, or contextual combinations were sufficient to enable an automatic classification. While the framework allows additional deviation combinations to be presented to the auditors at Levels 2, 3, and 4 by adjusting the minimum support threshold, such refinements may still fail to capture all exceptional behaviors.

In this final stage, the remaining deviations require full observation and manual classification by the auditors. These cases are examined individually, using complete trace-level information and domain knowledge that cannot be systematically encoded within the framework. The auditors' judgments at this stage ensure that unusual but legitimate behaviors are not incorrectly flagged, and that genuine anomalies are properly identified.

Stage 3 thus acts as a controlled fallback mechanism that guarantees complete population classification while preserving the auditors' authority over complex and non-repetitive cases. By limiting manual analysis to a small residual set, the framework maintains scalability and efficiency without compromising audit quality or professional judgment.

For the running example, the application of the proposed approach allows three out of four deviating cases to be classified. Case 1 is classified as an anomaly, while Cases 4 and 6 are classified as exceptions. Case 2 requires further investigation, although part of its deviations have been labeled.

5.3 Concluding note

This chapter introduced a structured deviation classification framework to assist auditors in classifying the extensive output of conformance checking tools into two sets of exceptions and anomalies. By mapping model–log mismatches to a small set of interpretable deviation types and incrementally enriching them with control-flow, case-level, and event-level information, the framework embeds auditor knowledge within an algorithmic process, while explicitly managing the information load. The staged design ensures full population coverage and enables systematic propagation of auditor judgments, thereby reducing manual effort and focusing attention on complex cases. At the same time, the framework acknowledges the inherent incompleteness of the normative models and incorporates a controlled fallback to manual analysis for residual deviations that cannot be resolved automatically. As such, the framework enables internal auditors to systematically analyze process executions and to structure detailed investigations of deviations within assurance activities.

The next chapter demonstrates how auditors can apply this proposed framework to an actual data set using a case study.

Chapter 6

Empirical Evaluation of the Process Deviation Classification Framework

The process deviation classification framework proposed in Chapter 5 was developed to support auditors in classifying control-flow deviations in a structured and incremental manner. While the previous chapter presented the conceptual foundations, formal definitions, and algorithmic logic of the framework, its practical relevance and effectiveness must be demonstrated on real process data. For this reason, the present chapter evaluates the framework through a real-life purchase-to-pay event log, gathered in the context of a previous case study.

In line with the algorithm engineering perspective outlined in Chapter 1, this chapter represents the evaluation phase of the design cycle, where theoretical constructs are assessed in an empirical setting. The framework was developed following an iterative process that integrates problem understanding, algorithm design, and validation on real data. Consequently, the purpose of this chapter is not merely to illustrate how the framework operates, but to assess whether the proposed incremental classification design is useful to effectively support auditing tasks in practice. Therefore, this chapter has two objectives. First, it aims to demonstrate how the framework can be applied step by step to a real-world event log, thereby illustrating its operational workflow. Second, it evaluates to what extent and how the framework supports the reduction of manual auditor effort while maintaining the ability to distinguish between anomalous and acceptable process behavior. The chapter examines how the deviation set is incrementally reduced across the classification levels and to what extent cases

can be resolved without detailed inspection. This enables a structured assessment of how classification performance evolves across the different levels and how effectively the framework reduces manual effort while preserving interpretability.

The chapter proceeds as follows. First, the experimental setup, data, and supporting techniques are introduced. Next, the evaluation data set and normative model are described. The chapter then applies the framework step by step, beginning with the mapping of process differences into deviation types, followed by the successive classification levels based on deviation types, frequent deviation patterns, case-level attributes, and event-level attributes where each level introduces additional information. The chapter concludes with insights into the strengths and limitations of the framework, and reflects on its role within the broader context of continuous auditing.

6.1 Goal of the Experimental Evaluation

The objective of this chapter is twofold.

(1) Demonstration goal: to illustrate how the proposed framework can be applied step by step on a real-world event log.

(2) Evaluation goal: to assess whether and to what extent the framework reduces manual auditor effort while maintaining the ability to distinguish anomalous from acceptable process behavior.

6.2 Evaluation Metrics

The performance of the framework was assessed based on:

Input Metric:

- **Auditor Effort:** The number of evaluation actions performed by the auditor at each level. An evaluation action corresponds to labeling an aggregated representation (e.g., deviation type, pattern, or itemset).

This metric provides an approximation of auditor effort based on the number of evaluation actions; its limitations are discussed in Section 6.6.

Output Metrics:

- **Reduction Rate:** the proportion of deviation instances (or cases) that are classified at a given level, defined as the ratio between the number of resolved elements at that level and the total number of elements entering that level.

- **Remaining Set Size**: The number of deviation instances (or cases) that remain unresolved after each level.

Performance Metric:

- **Effort Reduction**: The ratio between the number of manual evaluation actions required using the framework and the number of inspections required in a fully manual setting.

The input metric captures the effort performed by the auditor. The output metrics capture how much of the deviation set is resolved, and the performance metric quantifies the overall reduction in effort compared to a fully manual baseline.

The choice of an empirical evaluation setting is appropriate for two reasons. First, the framework is intended for use in realistic auditing environments in which deviations occur in large numbers and in heterogeneous forms. Second, the framework explicitly relies on auditor judgment for labeling deviation representations at different levels of abstraction. Its usefulness therefore cannot be assessed solely through technical correctness, it must also be examined in terms of how effectively it structures the interaction between computational outputs and human judgment.

Accordingly, the evaluation does not only consider whether deviations can be detected, but also whether they can be organized in a way that reduces the amount of detailed manual inspection required. The successive levels of the framework are therefore evaluated with respect to their contribution to grouping, and resolving deviating cases before a full event log review becomes necessary.

6.3 Experimental Design and Setup

The evaluation of the process deviation classification framework was conducted using a real-world event log from a purchase-to-pay process. The primary objective was to systematically assess the framework's ability to classify control-flow deviations incrementally to reduce the amount of manual auditor effort required to distinguish anomalies from acceptable exceptions. Below, we describe the dataset and setup used for this evaluation.

6.3.1 Data set

The event log used in this chapter is the case study presented in Chapter 2. The data set represents a purchase-to-pay process in a real organizational setting. Section 6.4 elaborates on this dataset in more detail.

6.3.2 Implementation and Techniques

The evaluation leverages the following implementation and techniques:

Conformance Checking: The framework utilized a conformance checking algorithm presented by García-Bañuelos et al. [2018] to identify discrepancies between the normative model and the event log. The resulting deviations were categorized into predefined types, such as Insertion, Missing, Reordering, and Repetition. For this categorization, we adapted the algorithm from García-Bañuelos et al. [2018] as the foundation of our approach, with valuable insights and assistance from one of its authors Nick R. T. P. van Beest.

Frequent Pattern Mining: Sequence mining techniques were applied to discover frequent combinations of deviation types, aiding in the classification process. The sequential pattern mining algorithm from SPMF, an open-source data mining library written in Java and developed by Philippe Fournier-Viger, was used for this purpose. SPMF specializes in pattern mining and is available at https://www.philippe-fournier-viger.com/spmf.

6.3.3 Procedure

The proposed framework is applied by following the sequence of steps defined in Chapter 5. Subsequently, the evaluation is performed based on the outcomes of this application.

Stage 1: Mapping process differences into deviation types

Differences between observed traces and the normative model are identified through conformance checking and mapped to deviation types such as insertion, missing, reordering, and repetition. This step is further explained in Section 6.5.1.

Stage 2: Incremental deviation classification (Levels 1–4)

After deviation mapping, the remaining deviating cases are classified incrementally by presenting the auditor with increasingly enriched representations of the deviations. The classification starts from single deviation types, then considers frequent deviation patterns, then adds case-level attributes, and finally event-level attributes. These

steps are explained in Section 6.5.2.

Stage 3: Manual classification of remaining deviations

Finally, the cases that remain unresolved after Level 4 are inspected manually by the auditor. This stage ensures that all remaining deviations are classified, thereby achieving full population coverage.

6.4 Data Set Description

The case under study for exploratory analysis is an event log of a real-life purchase-to-pay process, which we previously introduced in Chapter 2. The data set used in this evaluation consists of 26,185 purchase-to-pay process instances generated and recorded within an organization between 02.01.2007 and 25.01.2008. As stated in Chapter 2 a normative model is also provided by the organization.

This data set is particularly suitable for evaluation because it combines several characteristics that make deviation analysis difficult in practice. First, the number of cases reflects a realistic process scale in which deviations occur in a significant number of cases. This increases the complexity of systematic analysis without structured support. Second, the purchase-to-pay process contains both regular process paths and numerous deviations, including deviations that may reflect actual anomalies as well as deviations that may correspond to acceptable exceptions. Third, the event log contains both case-level and event-level attributes, which makes it possible to evaluate the full incremental enrichment logic of the framework across all four classification levels.

To support both interpretability and formal analysis, the normative model is presented using two complementary representations. Figure 6.1 illustrates the process in BPMN notation from Chapter 2, providing an intuitive and business-oriented view of the process flow. Figure 6.2 presents the corresponding Petri net representation, which is used as the formal basis for conformance checking tool. The normative model serves as a benchmark for identifying deviations between observed and expected process behavior in this evaluation.

Table 6.1 presents four process instances from this event log. For the sake of space, we only presented process instances that deviate from the normative process model. In this table, for each process instance, the process variants, their deviation type, case attributes, and event attributes of the deviating events are provided. For the sake of space, only the event attributes of the deviating events are shown, while

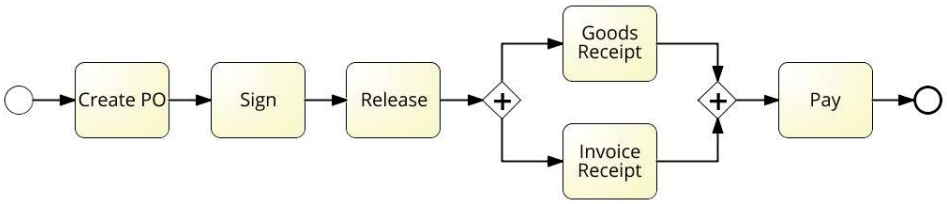


Figure 6.1: The case study normative model for the purchase-to-pay process in BPMN

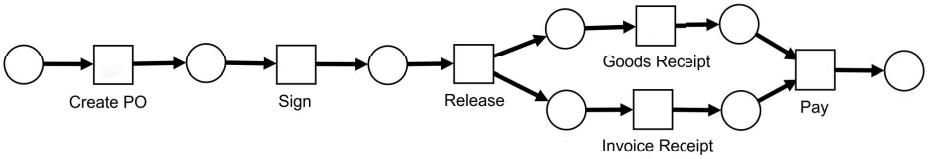


Figure 6.2: Normative model for the purchase-to-pay process in Petri net notations

attributes of normal events are excluded from these process instances. As shown, an overwhelming amount of information exists for each deviating instance, making the analysis complicated for auditors.

In the following sections, we apply our process deviation analysis approach to the case study data set and show how this approach reduces the labeling effort by providing the information step by step when needed.

No.	Deviating instance	Attributes
1: 450040304610	**Variant**	Create PO, Sign, Release, IR, Pay
	Case Attributes	Value-PO:128250, Quantity-PO:30, Conceptname:450040304610, Doc-Type:FO, PG:H01, Value-Pay:128200, Supplier:82706, Quantity-IR:30, Value-IR:128250, Unit:BL
	Event Attributes	Conceptname:Create PO, resource:U70162, lifecycle:complete, timestamp:2007-01-30T00:00:00 Conceptname:Sign, resource:G22364, lifecycle:complete, timestamp:2007-01-31T00:00:00 Conceptname:Release, resource:G40152, lifecycle:complete, timestamp:2007-02-06T00:00:00 Conceptname:IR, IR-Value:128250, resource:G10849, IR-Quantity:30, IR-LFBNR:No value defined, IR-Objectkey:5105739598200 7, lifecycle:complete, timestamp:2007-03-05T00:00:00 No info for GR Conceptname:Pay, resource:G10849, Pay-Value:128250, Pay-Objectkey:5105739598200 7, lifecycle:complete, timestamp:2007-03-23T00:00:00
2: 450040096380	**Variant**	Create PO, Sign, Release, Change Line, Sign, Release, GR, Pay
	Case Attributes	Value-PO:1316276, Quantity-PO:1, Quantity-GR:1, Value-Pay:1592600, Quantity-IR:1, Unit:EA, Conceptname:450040096380, Value-GR:1592694, Doc-Type:FO, PG:I17, Supplier:2343, Value-IR:1592694, GR-Ind:X
	Event Attributes	Conceptname:Create PO, resource:G24237, lifecycle:complete, timestamp:2007-01-23T00:00:00 Conceptname:Sign, resource:U66052, lifecycle:complete, timestamp:2007-01-23T00:00:00 Conceptname:Release, resource:U77555, lifecycle:complete, timestamp:2007-01-26T00:00:00 Conceptname:Change Line, Modification:0, resource:G30708, Rel-Modif:0, lifecycle:complete, timestamp:2007-07-13T00:00:00 Conceptname:Sign, resource:G38262, lifecycle:complete, timestamp:2007-07-16T00:00:00 Conceptname:Release, resource:U66052, lifecycle:complete, timestamp:2007-07-16T00:00:00 Conceptname:GR, resource:G24237, GR-Quantity:1, GR-Value:1592694, GR-LFBNR:No value defined, lifecycle:complete, timestamp:2007-07-26T00:00:00 Conceptname:Pay, resource:U71980, Pay-Value:1592694, Pay-Objectkey:5105787307 2007, lifecycle:complete, timestamp:2007-07-31T00:00:00

Variant	Create PO, Change Line, Sign, Release, GR, IR, Pay
3: 450039661580	
Case Attributes	Value-PO:1074, Quantity-PO:1, Quantity-GR:1, Value-Pay:1000, Quantity-IR:1, Unit:EA, Conceptname:450039661580, Value-GR:1074, Doc-Type:FO, PG:L07, Supplier:1816, Value-IR:1074, GR-Ind:X
Event Attributes	Conceptname:Create PO, resource:U45859, lifecycle:complete, timestamp:2007-01-08T00:00:00 Conceptname:Change Line, Modification:0, resource:U45859, Rel-Modif:0, lifecycle:complete, timestamp:2007-01-08T00:00:00 Conceptname:Sign, resource:G16977, lifecycle:complete, timestamp:2007-01-09T00:00:00 Conceptname:Release, resource:U29598, lifecycle:complete, timestamp:2007-01-09T00:00:00 Conceptname:GR, resource:U45859, GR-Quantity:1, GR-Value:1074, GR-LFBNR:No value defined, lifecycle:complete, timestamp:2007-01-12T00:00:00 Conceptname:Pay, resource:G15330, Pay-Value:1074, Pay-Objectke:51057242552007, lifecycle:complete, timestamp:2007-01-24T00:00:00
Variant	Create PO, Sign, Change Line, Sign, Sign, Release, GR, IR, Pay
4: 450039860620	
Case Attributes	Value-PO:1328, Quantity-PO:1, Quantity-GR:1, Value-Pay:1600, Quantity-IR:1, Unit:EA, Conceptname:450039860620, Value-GR:1607, Doc-Type:DI, PG:L14, Supplier:6508, Value-IR:1607, GR-Ind:X
Event Attributes	Conceptname:Create PO, resource:U21356, lifecycle:complete, timestamp:2007-01-15T00:00:00 Conceptname:Sign, resource:G20186, lifecycle:complete, timestamp:2007-01-15T00:00:00 Conceptname:Change Line, Modification:0, resource:U21356, Rel-Modif:0, lifecycle:complete, timestamp:2007-01-16T00:00:00 Conceptname:Sign, resource:G20186, lifecycle:complete, timestamp:2007-01-16T00:00:00 Conceptname:Sign, resource:G20186, lifecycle:complete, timestamp:2007-01-16T00:00:00 Conceptname:Release, resource:G56639, lifecycle:complete, timestamp:2007-01-17T00:00:00 Conceptname:GR, resource:U50978, GR-Quantity:1, GR-Value:1607, GR-LFBNR:No value defined, lifecycle:complete, timestamp:2007-01-23T00:00:00 Conceptname:Pay, resource:G55584, Pay-Value:1607, Pay-Objectkey:51057300802007, lifecycle:complete, timestamp:2007-02-14T00:00:00

Table 6.1: Illustrative deviating process instances with their corresponding case- and event-level attributes (4 out of 15,056 deviating instances).

The examples in Table 6.1 illustrate the central challenge addressed by the framework. Even a small number of deviating cases already contains a large amount of information, including control-flow deviations, contextual case attributes, and event-level details. In practice, auditors are not able to inspect such information exhaustively for all deviating cases. This motivates the need for a procedure that first presents deviations in a highly abstract form and introduces additional contextual information only when earlier levels are insufficient for classification.

6.4.1 Data preparation and control validation

This subsection describes the preliminary control validation performed on the case study data prior to applying the process deviation classification framework. In line with the assumptions defined in Chapter 5, these checks are treated as input preparation steps rather than components of the framework itself. In particular, we consider two commonly used control mechanisms: segregation of duties and the three-way match. These controls are generally essential for ensuring process integrity and preventing financial and operational risks. Here the segregation of duties and the three-way match controls are applied to identify and remove cases that already violate established business rules, ensuring that the subsequent deviation classification focuses on control-flow deviations that require further analysis.

6.4.1.1 Segregation of duties

This test ensures that critical activities within the process are performed by separate individuals, reducing the risk of fraud or error. As explained in the work of Jans [2009], who worked with the same data set and had access to the domain experts, there were three tests on the segregation of duties:

i) Whether activities 'Sign' and 'Release' are always executed by two distinct performers?
ii) Whether activities 'Goods Receipt' and 'Invoice Receipt' are always executed by two distinct performers?
iii) Whether 'Release' and 'Goods Receipt' are always executed by two distinct performers?

Jans [2009] has performed these three tests and reported that the first check, the segregation of duties for 'Sign' and 'Release,' is valid for all the cases. The second proposition is then checked using the LTL checker, and they reported that in all cases, 'Goods Receipt' and 'Invoice Receipt' were performed by different performers. The

third proposition is also checked using the LTL checker, and they found out there are 175 cases that do not respect this rule, and the performers of 'Release' and 'Goods Receipt' are the same person. For this thesis, we did not use the LTL checker, but we got the same number (175 cases) in which 'Release' and 'Goods Receipt' have been performed by the same performers. There are three performers who are involved in these cases (users *U15816* (42 cases), *U61532* (4 cases), and *U66602* (129 cases)).

While investigating these 175 process instances, it turned out that these cases are spread in 12 process variants. There are 89 cases which have a Change Line inserted between Create PO and Sign (variant 2). There are 41 cases in which the signature is missing (variant 24). Fifteen cases (variant 8), 12 cases (Variant 20), 11 cases (variant 50), and the seven other process instances are unique cases with many combinations of GR, IR, and Pay. These 175 cases are extracted from the event log as they violate the segregation of duties to be discussed further with the process owners. By removing these cases from the event log, 1,820 events are removed from the log. So, the remaining are 26,018 cases containing 180,025 events.

Figure 6.3 illustrates the distribution of violations against the segregation of duties.

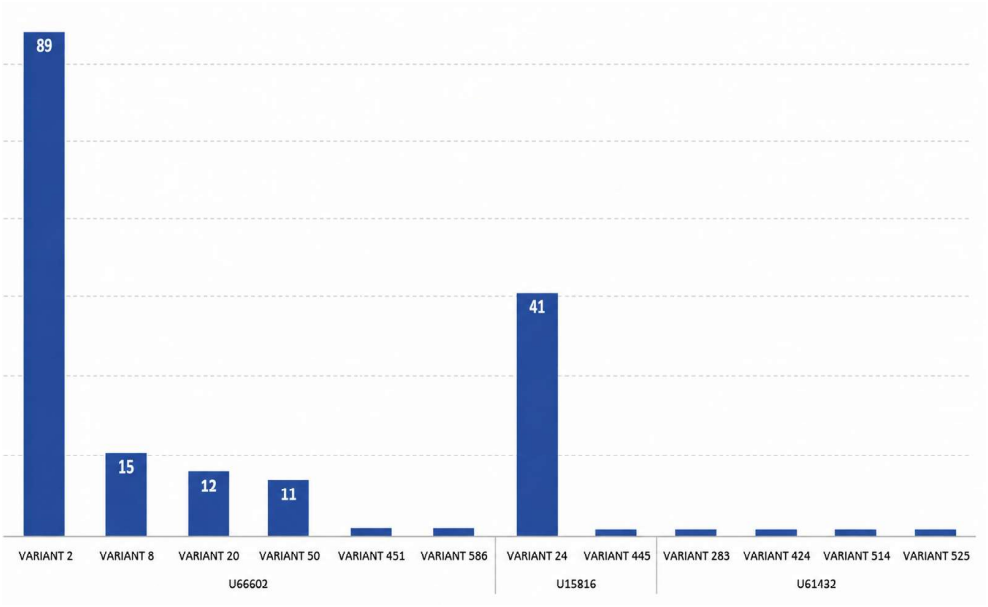


Figure 6.3: Segregation of duties violated by three resources

New feature	Description
Diff-Quantity-GR-PO	Difference between the quantity of GR and PO
Diff-Price-IR-PO	Difference between amount of IR and PO
Rel-diff-unit price-IR-GR	Relative difference between unit price IR and unit price of GR. Unit prices for both IR and GR are calculated by Total-price divided by Total Quantity.

Table 6.2: New features created for testing three-way match of the case study data set

6.4.1.2 Three-way match

The three-way match is a procedure of verifying invoice and testing whether the payment is complete and accurate. Three-way matching ensures that there are no discrepancies in the three most important documents of the purchase-to-pay process, i.e., purchase order, goods receipt, and invoice receipt documents. Auditors check the existence of these documents and test whether the price and quantity of the invoice match the amount and quantity of the purchase order and the goods received. This test is to control that the organization does not overspend or pay for what they did not ever receive.

While creating the event log for purchase-to-pay processes, one should also consider extracting this information.

For performing the three-way match test on our data set, the following values should be checked against each other:
i) Goods Receipt quantity and purchase order quantity
ii) Invoice Receipt price and purchase order price
iii) Invoice Receipt unit price and Goods Receipt unit price

The event log contains the total value and the total quantity of purchase order, goods receipt, and invoice receipt. We create three numerical features to compare this information together. Table 6.2 shows these three features.

Goods Receipt quantity- purchase order quantity match

When the difference between quantity of GR and quantity of PO is calculated, there were 830 cases in which the difference was not 0. The range of diff-Quantity-GR-PO varied from -200 to 3,749,900. There are 403 cases in which Rel-diff-Quantity-GR-PO was -200. Figure 6.4 shows the distribution of Diff-Quantity-GR-PO of these

830 cases and their frequency. The X-axis shows the difference (the value of Rel-diff-Quantity-GR-PO), and the two Y-axes (on the left and right) illustrate the frequency of each difference. Most of the values occur once, and some occur several times, between 1 and 20 times. Only value -200 has shown up 403 times, so to be able to illustrate the values nicely on the plot, a dual Y-axis is used. The left Y-axis is used for all differences except -200, and the right Y-axis is used for -200. These 830 cases are removed from the event log as they fail the three-way match test.

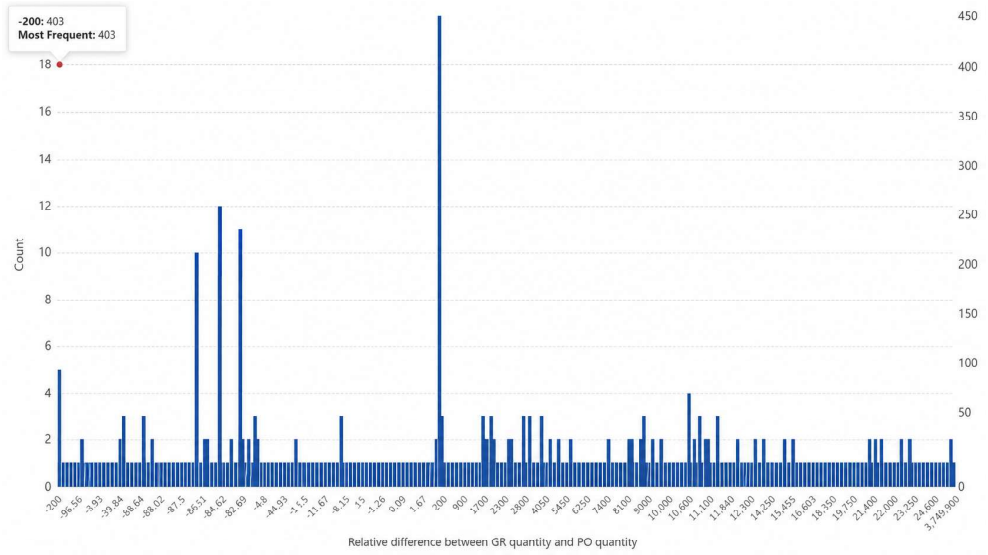


Figure 6.4: The relative difference between quantity of goods receipt and quantity of purchase order

Invoice receipt price and purchase order price

The difference between invoice receipt price and purchase order price is calculated for all cases in the data set. There are 21,062 cases from 28,185 cases in which the price on the purchase order is not the same as the price on the invoice receipt. Figure 6.5 shows the differences in the price of invoice receipt and purchase order. As the X-axis shows the values of Diff-Price-IR-PO. The range of values is so wide (from -64,230,114 to 25,779,600). The Y-axis shows the frequency of each difference value.

Normally, a tolerance limit is set on the differences between invoice receipts and purchase orders. If the difference exceeds this tolerance, there should be a control that prevents the invoice from being paid immediately to the vendor. In other words, the invoice should get blocked due to the tolerance control, and the blocked invoice should be returned to the buyer for a suitable organizational decision. The tolerance

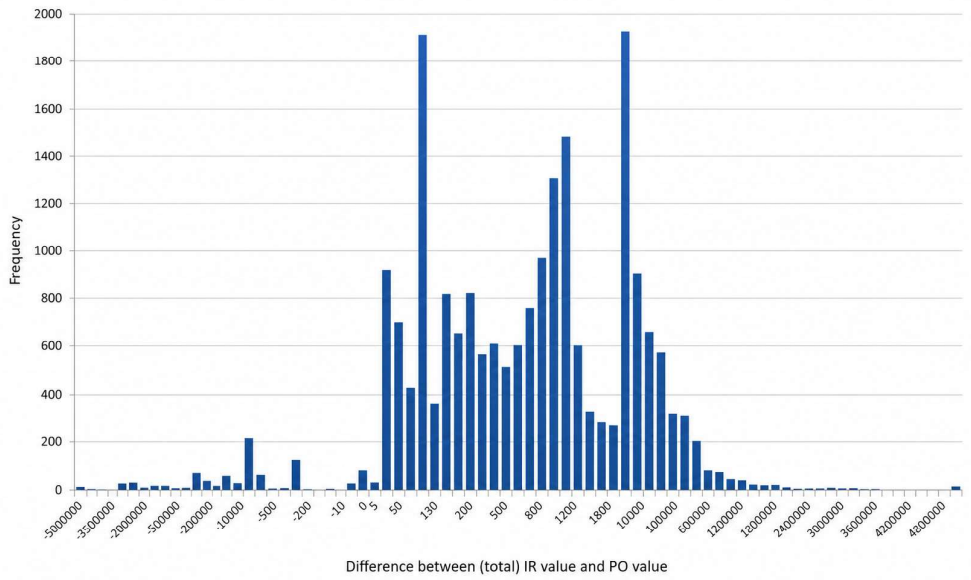


Figure 6.5: The difference between price of invoice receipt and price of purchase order

on this event log is set to 10,000. So all the cases in which the difference between the purchase order price and invoice price is higher than 10,000 are considered anomalies. This raises 2,436 cases which are higher than the tolerance. These cases are extracted from the event log to be discussed with the process owner.

Invoice receipt unit price and goods receipt unit price

The last test in the three-way match is comparing the unit prices of invoice receipt and goods receipt. We calculate the relative difference between the unit price of the invoice receipt and goods receipt (Rel-Diff-Unit price IR-GR). The analysis shows that there are 1,133 cases out of 28,185 cases in which the difference between the unit prices of invoice and the goods receipt is not zero. Figure 6.6 illustrates this analysis. The same as in the previous plots, the X-axis shows the value of Rel-Diff-unit price-IR-GR, and the Y-axis shows the frequency of each value.

The tolerance limit for the unit price differences is set to 6%, considering the possible tax added to the goods price. Moreover, the negative differences (where the unit price on the invoice is less than the unit price on the goods receipt) are not considered risky by the auditor. Considering this, there are 89 cases whose Rel-Diff-Unit price IR-GR exceeds the tolerance. The auditor considers these cases anomalies, so they can be removed from the event log for further investigation.

As a result of tests of controls phase, 3,167 cases are removed from the event log as they are segregation of duties and three-way match violations. Please note that

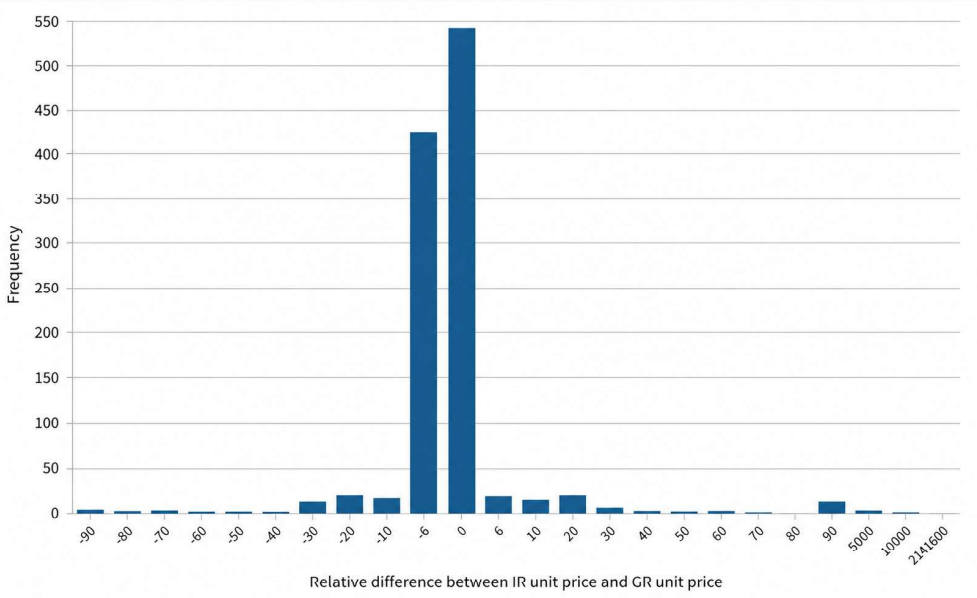


Figure 6.6: The relative difference between the unit price on invoice and unit price on Goods Receipt

there were 363 cases in which more than one violation against controls occurred. For instance, the analysis shows 74 cases in which violation against segregation of duties has occurred. In addition, the difference between the invoice value and purchase order was larger than the tolerance (10,000). Moreover, 12 cases fail all tests in the three-way match, meaning that none of GR, IR, or PO matches the other one. Figure 6.7 shows the result of the tests of the controls phase.

Although these control checks are not part of the deviation classification framework itself, they play an important preparatory role in the empirical evaluation. Such control analyses are well supported by a wide range of existing auditing tools and are typically performed as part of standard audit procedures. This makes the input to the framework more focused and the framework can concentrate on those deviating executions whose status are less feasible to be resolved by the existing tools. In this way, the input to the framework is restricted to deviations that require further classification.

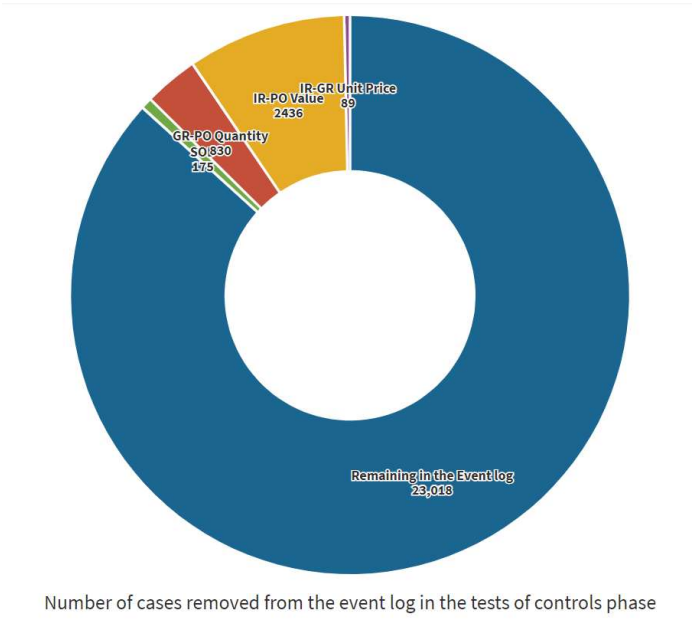


Figure 6.7: Tests of controls phase result

6.5 Empirical Results

This section presents the empirical results obtained by applying the proposed framework to the case study data set. The results are reported in terms of the evaluation metrics defined in Section 6.2, including reduction rate, remaining set size, and effort reduction. The analysis follows the successive stages of the framework and shows how the deviation set is incrementally reduced across the classification levels. In line with the framework design, Stage 2 consists of four incremental classification levels (Levels 1–4), each introducing additional information to support deviation analysis.

6.5.1 Stage 1: Mapping process differences into process deviations

The first step involves mapping differences between process traces and the expected model into process deviations. This process is grounded in the methodology detailed in Chapter 5, which emphasizes the systematic identification of deviations. The process deviation classification framework constructs the prime event structure (PES) from the normative model, referred to as PES_{Model}. This event structure is derived directly from the Petri-net normative model. The PES_{Model} acts as a benchmark against which the event log is evaluated.

For this comparison, the Prime Event Structure (PES) from the event log should

be generated. To create this PES which is called PES_{Log}, the conformance checker creates a set of distinct runs.

The runs are subsequently merged to construct the consolidated event structure, PES_{Log}.

The comparison of PES_{Model} and PES_{Log} produces a Partially Synchronized Product (PSP), which highlights discrepancies between the expected and actual process flows. As for the next step, the Insertion and Missing mismatches are discovered in accordance with the pseudocode of the algorithm 1 presented in Chapter 5. Afterward, we map these mismatches into the four deviation categories: *insertion*, *missing*, *reordering*, and *repetition*.

This mapping stage is essential because the raw output of conformance checking is not yet directly suitable for auditor interaction. The PSP identifies mismatches between the normative and observed behavior, but these mismatches must be translated into a smaller and more interpretable terminology before they can support efficient classification.The four deviation categories used in this chapter serve as such a representation and are derived from the field study presented in Chapter 4. They reduce the complexity of the conformance-checking output while preserving the distinction between different forms of control-flow deviations for auditors. The mapping performed in this step therefore creates the bridge between the technical output of conformance checking and the higher-level representations needed for the subsequent classification levels.

Each category is further subdivided into specific sub-categories based on the events involved in the process. These correspond to the deviation types defined in Chapter 5. An example of these deviation types is missing(Sign). Although any combination of these four categories for each activity in the event log is possible, in reality, only some happen. Table 6.3 provides a comprehensive listing of deviation types identified in the case study data set.

The identified deviation types provide an initial indication of whether cases can be classified based on the control-flow level information alone. Deviation types that are clearly interpretable can be resolved at this stage, whereas those whose meaning depends on additional information remain unresolved and are carried forward to subsequent levels for further analysis. Some deviation types, such as missing(Release), are intuitively more critical because they suggest the absence of an expected approval step. Others, such as repetition(Sign), may be less problematic and may instead reflect operational rework or benign process variation. This difference in interpretability motivates the incremental labeling approach adopted in the next section.

After applying this step and comparing the event log with the normative pro-

Category	Deviation Type
Insertion	insertion(Change Line)
Missing	missing(Sign)
	missing(GR)
	missing(Release)
Reordering	reordering(IR)
	reordering(Pay)
	reordering(Release)
Repetition	repetition(Sign)
	repetition(Release)
	repetition(GR)
	repetition(IR)
	repetition(Pay)
	repetition(Change Line)

Table 6.3: Deviation types identified in the case study derived from the mapping of conformance checking results.

No.		Process Instance
1: 450040304610	**Variant**	Create PO, Sign, Release, IR, Pay
	Deviation Type	missing(GR)
2: 450040096380	**Variant**	Create PO, Sign, Release, Change Line, Sign, Release, GR, Pay
	Deviation Type	insertion(Change Line), repetition(Sign), repetition(Release)
3: 450039661580	**Variant**	Create PO, Change Line, Sign, Release, GR, IR, Pay
	Deviation Type	insertion(Change Line)
4: 450039860620	**Variant**	Create PO, Sign, Change Line, Sign, Sign, Release, GR, IR, Pay
	Deviation Type	insertion(Change Line), repetition(Sign), repetition(Sign)

Table 6.4: Deviation types associated with the process instances presented in Table 6.1

cess model, process instances presented in Table 6.1 exhibit the deviation illustrated in Table 6.4. As shown, the first and third process instances each contain a single deviation. The first process instance has deviation missing(GR) while the third process instance has insertion(Change Line). In contrast, the second and fourth process instances exhibit three deviation types each, showing that multiple deviations might occur in one process instance. The second process instance in Table 6.4 has deviations insertion(Change Line), repetition(Sign), repetition(Release), and the fourth process instance has insertion(Change Line), repetition(Sign), repetition(Sign).

This stage defines the initial deviation set that serves as the input for the subsequent classification levels.

6.5.2 Stage 2: Classification of deviating cases through incremental information enrichment

In this section, the deviating cases are analyzed and classified through an incremental information enrichment procedure. As introduced in Chapter 5, the framework

progressively augments the representation of deviations, starting from control-flow information and gradually incorporating additional contextual data. At each level, auditors use the provided information to classify cases from the deviation set into anomaly and exception sets, while deferring those that require further detail to subsequent levels. The auditor is not required to inspect individual cases, but only to label aggregated representations, which are then propagated automatically to all corresponding cases.

This section illustrates the application of the incremental deviation classification procedure on the case study data set, following the steps defined in Chapter 5.

Figure 6.8 presents the corresponding process map after the preliminary control validation phase. The figure provides a visual overview of the process complexity, highlighting the presence of multiple variants and alternative execution paths. This complexity motivates the need for a structured classification approach, as manual inspection of such a process landscape would be difficult to perform systematically. In total, the log contains 147,330 events and 385 variants of process executions.

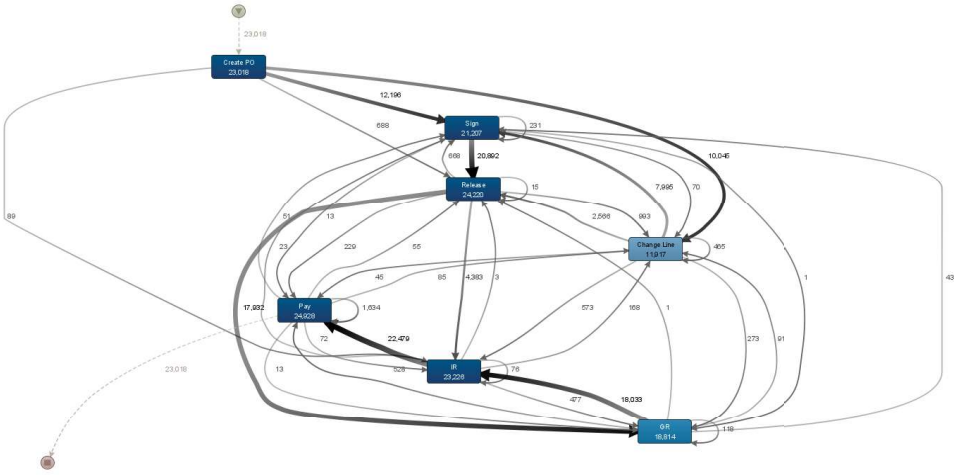


Figure 6.8: Process map after tests of controls

Applying the conformance checking tool on the event log divides the event log into two sets of normal and deviating cases. Of the 385 variants, two are the normal cases (45,76 % of the event log, which consists of 10,532 cases and, in total, 63,192 events). We put them in the normal set. The remaining 12,486 cases (with 84,138 events) deviate from the process model, and we classify them as deviating cases. Figure 6.9 shows the distributions of these two sets.

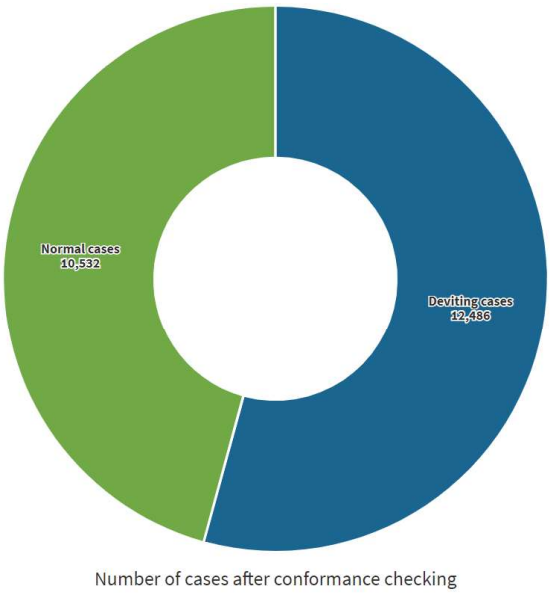


Figure 6.9: After Stage 1 the event log is divided into two sets of normal cases and deviating cases

Level 1: Single deviation type classification

At Level 1, deviating cases are represented only by their individual deviation types. This corresponds to the highest abstraction level of the framework. The list of deviation types is provided to the auditor to label as OK, NOK, or to leave unlabeled if this level of information is not sufficient). These labels are then propagated to the case-level according to the classification logic defined in Chapter 5.

The role of Level 1 is to determine whether some deviations can already be classified based on their type, without requiring any additional context. This is the most abstract level of the framework, as it presents only the minimal amount of information to the auditor. If a deviation type can be clearly assessed as acceptable or problematic independently of its context, then resolving it at this stage reduces both cognitive load and the need for further detailed analysis.

As an example, consider *repetition(Sign)* as a deviation type. When auditors confront this deviation for analysis, only seeing the deviation type may be enough to clear it and classify it as acceptable. The reason is that the repetition of *Sign*, while it may indicate inefficiency in the system, is not considered a significant deviation from an auditing perspective. As such, it can be considered an acceptable exception and cleared in this stage with this level of information. On the other hand, a deviation

Category	Deviation Type	Label (OK, NOK, ?)
Insertion	insertion(Change Line)	?
Missing	missing(Sign)	?
	missing(GR)	?
	missing(Release)	NOK
Reordering	reordering(IR)	OK
	reordering(Pay)	?
	reordering(Release)	?
Repetition	repetition(Sign)	OK
	repetition(Release)	OK
	repetition(GR)	?
	repetition(IR)	?
	repetition(Pay)	?
	repetition(Change Line)	?

Table 6.5: Deviation types (subcategories) in the data set for tests of details

type such as *missing(Release)* may indicate the absence of a required approval step and can be considered problematic irrespective of any further information. Therefore, such a deviation type would be classified as NOK.

Classification decisions are propagated at the case-level. If a case contains at least one NOK-labeled deviation instance, the entire case is classified as an anomaly and removed from the remaining deviation set. Consequently, all other deviation instances within that case are also removed from further analysis, even if they are not individually labeled.

At the same time, Level 1 highlights the limitations of classification based on control-flow information alone. Certain deviation types cannot be assessed reliably without knowing more about the sequence or the associated data. The presence of undecided labels at this level is therefore expected and motivates the need for the following enrichment levels.

In the case study data set, the deviation set has 12,486 cases, and there are 24,417 deviation instances in these cases (one case can contain multiple deviation instances). We categorized these deviation instances into 13 deviation types. In Level

1, the deviation types are presented to the auditor to label. The labeling process was instrumental in filtering deviations and isolating critical anomalies for further investigation. Among 13, only one deviation type has been labeled as NOK. Three deviation types are labeled as OK (i.e., acceptable exceptions), and nine deviation types remain unlabeled. Table 6.5 shows these 13 deviation types with the auditor's labels for each of them.

At level 1, only one deviation type, *missing(Release)*, was labeled as NOK. This deviation occurred in three cases, which were therefore classified as anomalies. However, as a result of case-level propagation, additional deviation instances contained within these cases are also removed from the remaining deviation set. In total, 13 such deviation instances are excluded, including *repetition(IR)*, *repetition(GR)*, *repetition(Pay)*, *insertion(Change Line)*, and *repetition(Change Line)*.

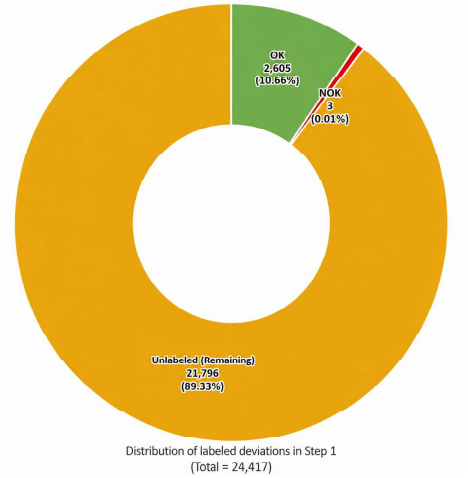
In addition, three deviation types were labeled as OK: *repetition(Sign)*, *repetition(Release)*, and *reordering(IR)*. These labels covered 2,605 deviation instances in total, consisting of 1,394 instances of *repetition(Sign)*, 1,203 instances of *repetition(Release)*, and eight instances of *reordering(IR)*. At the case-level, 280 cases were moved to the exception set. These were the cases in which all deviation instances belonged to deviation types labeled as OK at this level (only contained repetition (Sign) and repetition (Release)). Cases that contain a combination of OK-labeled and unlabeled deviation instances remain unclassified and will be forwarded to the next level. This initial classification step helped auditors to systematically narrow down the scope of analysis, ensuring that their efforts were focused on deviations with the highest risk potential.

Figure 6.10 summarizes the result of Level 1. Figure 6.10a shows the distributions of deviations after the auditor labels the 13 deviation types, and Figure 6.10b shows the distribution of cases in three sets of deviation, exception, and anomaly.

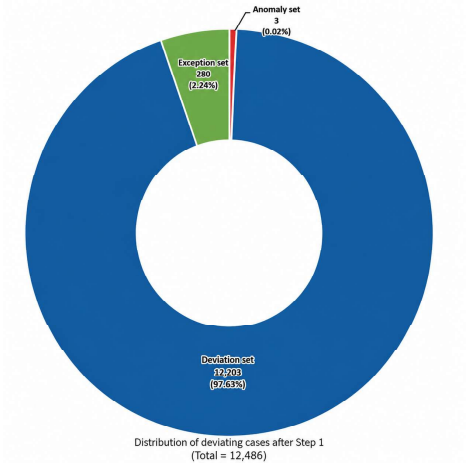
In this level, 283 out of 12,486 cases (approximately 2.25%) were resolved, including 280 cases (approximately 2.24%) classified as exceptions and three cases (approximately 0.02%) classified as an anomaly. In addition, 2,606 out of 24,417 deviation instances (approximately 10.67%) were labeled at this level.

The outcome of Level 1 shows that only a limited subset of the overall deviation set can be resolved at the highest abstraction level. Nevertheless, this level contributes to reducing the problem size by resolving clearly classifiable cases at an early stage and by identifying a subset of cases that require further analysis in the subsequent levels. This is consistent with the observation that control-flow categories alone are often insufficient for reliable auditing decisions.

In terms of the evaluation metrics, these results show that a subset of the deviation



(a) The distribution of deviation instances: 2,605 labeled as OK, 3 as NOK, and 21,811 remain unlabeled.



(b) The distribution of cases: 280 classified as exceptions, 3 as anomaly, and 12,203 remain unclassified.

Figure 6.10: Distribution of cases and deviating instances after Level 1

set can already be resolved at the highest level of abstraction. While the reduction rate remains limited, the identification of clearly interpretable deviation types enables early classification and supports a more focused analysis in subsequent levels.

Level 2: Frequent deviation-pattern classification

The transition from Level 1 to Level 2 reflects the need to move beyond isolated deviation types when they are insufficient for classification. Before immediately inspecting all available data which can increase the information load, Level 2 is designed to investigate whether the combination of deviations itself provides enough information for classification. The combinations of deviation types are provided in the form of frequent deviation patterns. These patterns are identified by applying sequence mining techniques to the remaining deviating cases from level 1. The frequent closed sequential patterns mining algorithm proposed by [Gomariz et al., 2013] is applied in this step. The resulting set of patterns is presented to the auditor, who assigns a label to each pattern as OK, NOK, or leave unlabeled (when this level of information is not sufficient). These labels are then propagated to all corresponding cases according to the classification logic defined in Chapter 5. The same as in level 1, the cases with NOK sequence patterns are moved to the anomaly set. Note that, the presence of a single NOK-labeled deviation instance within a case is sufficient for the entire case

to be classified as an anomaly. If all the sequence patterns and deviating types in a case are labeled as OK, the framework assigns it to the exception set.

The role of Level 2 is to determine whether deviations that could not be labeled at Level 1 can be classified by considering their combination with other deviations. While individual deviation types may be insufficient for classification, their occurrence with other deviations within a trace may provide additional context that supports the auditor's decision. This level therefore introduces a structured sequence-based representation of deviations while remaining within the control-flow perspective.

A minimum support threshold of 0.02 (2%) was applied to identify significant patterns, ensuring that only frequent and recurrent sequences were considered. At this level, support is computed with respect to the number of unique deviating traces, as pattern mining is performed on trace-level representations. It illustrates that certain combinations of deviation types may consistently occur together across multiple cases. The analysis revealed 11 deviation sequence patterns to present to the auditor. Table 6.6 provides a detailed breakdown of these sequence patterns, including their frequency and auditor classifications. The most frequent itemset is $\langle$*insertion(Change Line), missing(GR)*$\rangle$ with support 3,612 (15%). Among all the sequences, the auditor only labeled four deviation patterns, and all of them are labeled as OK (acceptable exceptions). For instance, sequences involving 'insertion(Change Line)' followed by either 'Sign' or 'Release' were classified as acceptable due to their alignment with expected process variations. The auditor reasons that if after any *Change Line* activity, a *Sign* or *Release* activity has occurred, it means that there is a control on the Modification happening in such cases. Hence, this combination of deviation types is exceptional and can be cleared.

The support values are computed with respect to the 4,324 unique traces representing the deviating cases. At this level, the resulting patterns reflect structurally distinct traces rather than repeated occurrences of identical cases.

The results show that only a limited number of deviation patterns exceed the minimum support threshold of 2%. Even for the most frequent patterns, the support remains moderate relative to the total number of traces. This indicates that the deviation space is not dominated by a single or a few recurring patterns, but is instead distributed across a variety of distinct combinations of deviations. At the same time, the identified frequent patterns still provide meaningful structure for classification. By focusing on patterns that occur across multiple traces, the framework enables the auditor to group similar deviation behaviors and assess them collectively, thereby reducing the need for case-by-case inspection. This confirms that, despite the diversity of deviations, pattern-based analysis contributes to reducing the classification effort while maintaining interpretability.

Frequent deviation sequences	Count	Support (%)	Label (OK,NOK,?)
⟨ins(Change Line), mis(GR)⟩	1485	34.3	?
⟨mis(Sign), mis(GR)⟩	1167	27.0	?
⟨ins(Change Line), rep(Change Line)⟩	528	12.2	?
⟨ins(Change Line), mis(Sign)⟩	519	12.0	?
⟨ins(Change Line), mis(Sign), mis(GR)⟩	519	12.0	?
⟨ins(Change Line), rep(Change line), mis(GR)⟩	436	10.1	?
⟨rep(Sign), rep(Release)⟩	226	5.2	OK
⟨ins(Change Line), rep(Sign)⟩	219	5.1	OK
⟨ins(Change Line), rep(Release)⟩	196	4.5	OK
⟨ins(Change Line), rep(Sign), rep(Release)⟩	196	4.5	OK
⟨ins(Change Line), rep(Pay)⟩	86	0.2	?

Table 6.6: List of frequent deviation sequences in Level 2 and their supports and labels where min-supp = 0.02(2%). 'ins', 'mis' and 'rep' are used as abbreviations for deviation types insertion, missing, and repetition respectively

The results of Level 2 indicate that deviation pattern information provides additional support for classification beyond isolated deviation types. Only a subset of the mined patterns could be confidently labeled, and in this case, all labeled patterns were assessed as acceptable exceptions.

To be more specific, at this level 824 itemset instances were identified, corresponding to 1,373 deviation instances. Of these, 614 deviation instances had already been labeled in Level 1, as they belong to deviation types previously classified (e.g., *repetition(Sign)* and *repetition(Release)*). The remaining 759 deviation instances were newly labeled at this level based on their occurrence within deviation patterns (e.g., *insertion(Change Line)* when appearing in specific combinations).

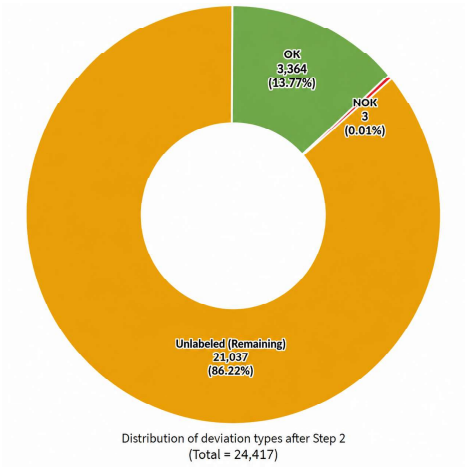
At the case-level, 328 cases associated with these OK-labeled patterns were removed from the deviation set and added to the exception set. A case is removed from the deviation set only if all deviation instances within that case are labeled as OK. This means that no NOK or undecided deviation instances remain for that case, allowing it to be fully classified as an acceptable exception.

As no deviation pattern was labeled as NOK, no case was added to the anomaly set. In this step, 328 out of the 12,203 cases entering Level 2 (approximately 2.69%) were resolved, all of which were classified as exceptions. In addition, 759 out of the 21,811 deviation instances entering Level 2 (approximately 3.48%) were newly labeled at this level.

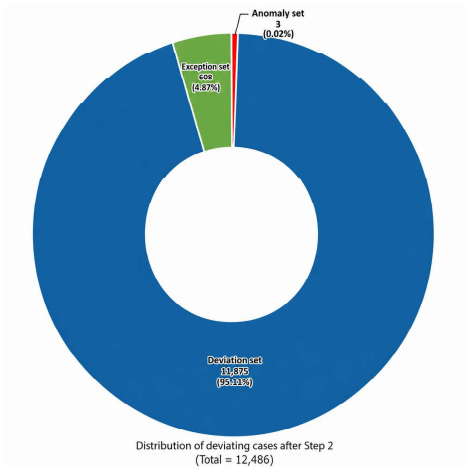
As shown in Table 6.6, some of the identified patterns are related through subsequence relationships, in the sense that longer patterns may contain shorter ones as subsequences. For example, the pattern ⟨ins(Change Line), rep(Sign), rep(Release)⟩ includes both ⟨ins(Change Line), rep(Release)⟩ and ⟨ins(Change Line), rep(Sign)⟩. This is consistent with the nature of sequence-based pattern mining, where related patterns can appear at different levels of granularity and may capture similar behavioral structures across cases.

Figure 6.11 demonstrates the cases and deviations statistics after this step. Figure 6.11a shows the distributions of deviations after the labeling in Level 2, and Figure 6.11b shows the distribution of cases in three sets of deviation, exception, and anomaly when Level 2 is finished. The remaining 21,037 deviations (11,875 cases) for subsequent analysis is sent to Level 3.

Level 2 demonstrates that deviation patterns can support the classification process beyond isolated deviation types. By grouping deviations into recurring combinations, this level enables additional cases to be classified without requiring detailed contextual information. However, the results also highlight the limitations of classification based solely on control-flow patterns. Although the combination of deviation types provides more information than Level 1, only a subset of patterns can be assessed re-



(a) The distribution of deviation instances: 3,364 labeled as OK, 3 as NOK, and 21,037 remain unlabeled.



(b) The distribution of cases: 608 classified as exceptions, 3 as anomaly, and 11,875 remain unclassified.

Figure 6.11: Distribution of cases and deviating instances after Level 2

liably, and a considerable number remain undecided. This indicates that control-flow information alone is insufficient for consistent classification decisions. The presence of these undecided patterns motivates the need for further enrichment. In the next step, additional case-level information is incorporated to support a more detailed classification of the remaining deviations.

According to the defined metrics, Level 2 provides an additional reduction of the deviation set by leveraging recurring combinations of deviation types. While the reduction remains moderate, the use of frequent deviation patterns enables the grouping of structurally similar cases and supports further classification without requiring detailed contextual information.

Level 3: Classification with case-level attributes

At Level 3, the framework enriches deviation patterns with case-level attributes for further refinement of deviation classifications. This additional context supports cases in which control-flow information alone is insufficient for classification. The auditor labels the resulting enriched patterns as OK, NOK, or leave unlabeled. The labels are then propagated to the remaining cases.

Level 3 is particularly important because it marks the point at which the framework moves beyond pure control-flow analysis and begins to integrate data perspective

information. This is crucial for auditing, since many deviations cannot be interpreted correctly without considering the business context of the case. A control-flow deviation may be suspicious in one setting and entirely acceptable in another, depending on attributes such as document type, payment amount, or whether the process concerns goods or services.

The inclusion of case-level attributes preserves the principle of incremental enrichment. Instead of exposing the auditor immediately to all event-level details, the framework first introduces a more aggregated layer of contextual information that may already be sufficient for classification.

For example, in the case study data set, a significant deviation type is 'missing(Goods Receipt)'. When such a deviation occurs in a case, besides the deviation type, the auditor needs to perform an investigation into whether the purchase order was for goods or services. This distinction is vital because goods delivery always should be documented with a goods receipt, while when service is delivered typically there is nothing to document and it is normal that the goods receipt activity is missing. In order to investigate this, auditors have to check one of the case attributes, called '*Goods Receipt Indicator*' which is a binary attribute. If it is equal to "X" value, it indicates goods should have been delivered; otherwise, if it is equal to "NA," it signifies the purchase order was a service; making the absence of goods receipt acceptable. Hence, for analyzing such a process instance that contains missing(Goods Receipt), the data perspective is crucial for auditors to make decisions.

By analyzing case attributes, auditors can label itemsets as OK, NOK, or undecided if the available information is insufficient on the case-level. The event log in this case study includes 12 case attributes, detailed in Table 6.7 which outlines their types, descriptions, and the attributes' range.

In the feature selection part of this step, some attributes like Doc-Type, PG, Supplier, and Unit are filtered out from the list of case attributes as they were deemed irrelevant. According to the auditor, these attributes do not contain any information to help them decide upon. Some attributes such as *Quantity-PO*, *Quantity-GR*, and *Quantity-IR* and Value-GR, Value-PO, and Value-IR do not have meaningful added value per se for auditing purposes, but their differences are relevant. The differences in value and quantity of purchase order, goods receipt, and invoice receipt are already considered and analyzed in the preliminary control validation phase, so they are not included as primary features at this level. Therefore, only GR-Ind (Goods receipt indicator) and Value-Pay will be considered in this step as they provide essential insights for evaluating deviations. This enables the auditors to identify meaningful patterns and anomalies that may not have been apparent in previous steps. By integrating case-level data attributes, the framework enhances the granularity and

Case attributes	Attr. type	Case attribute description	Attribute range
Quantity-GR	Numerical	The total quantity of Goods Receipts	-31178.99 to 65250000
GR-Ind	Binary	Goods Receipt Indicator shows whether Goods receipt is recorded	NA / X
Quantity-PO	Numerical	The total quantity of purchase order	0.75 to 65800000
Value-GR	Numerical	The total value of Goods Receipts	-35166714 to 148539600
Unit	Categorical	unit to express the quantity, e.g, hour, km,...	12 unique values
Supplier	Categorical	Supplier	>500 unique values
Value-PO	Numerical	The total value of purchase order	0 to 122760000
Value-Pay	Numerical	The total value of pay	35166700 to 63458800
Doc-Type	Categorical	Document type	6 unique values
Value-IR	Numerical	The total value of Invoice Receipt	-35166714 to 92915900
PG	Categorical	Purchasing group	8 unique values
Quantity-IR	Numerical	The total quantity of Invoice Receipts	-31178.99 to 65250000

Table 6.7: Case attributes of the case study data set

precision of deviation analysis, further reducing the complexity of auditing large-scale processes.

The feature selection at this level is guided by audit relevance rather than by statistical availability alone. Although the event log contains numerous case attributes, only those attributes that can plausibly influence the interpretation of a deviation are retained. This reflects the broader design logic of the framework: the goal is not to maximize the volume of information presented to the auditor, but to maximize the usefulness of that information for classification.

As mentioned in third level of the framework in Chapter 5, we also consider normal events in this step. As all the cases started with Create PO and finished with a Pay, we removed these two from the sequences. GR-Ind and Value-Pay are also added to all deviation types. We defined Value-Pay as a binary attribute to check whether the price paid is higher or less than 1000. In this step, we set the minimum support as 0.1 (10%), which resulted in 31 itemsets. We presented the itemsets to the auditor to label. At this level, support is computed with respect to the number of deviating cases, since classification decisions are made at the case-level.

The most frequent itemset identified is $\langle$*ins(Change Line), Sign*$\rangle$ with count 9,706. Although a signature follows a *Change Line*, the auditor was uncertain whether a single signature is sufficient after such a modification. So to play safe and avoid the risk of a false-positive assessment, they refrained from labeling it as "OK". However, when confronted with itemset $\langle$*ins(Change Line), Sign, Release*$\rangle$, which includes two

signatures following the modification, they labeled it as OK. The transaction amount (whether *Pay-value* >1000 or *Pay-value* <1000) was deemed irrelevant to most of the deviations except for cases of missing (Sign). If the transaction amount exceeds 1000, there must always be two signatures (in this event log: Sign and Release). This requirement aligns with the principle that transactions exceeding 1000$ are always considered material as professional judgment under ISA 320 dictates that the materiality threshold depends on factors such as organizational size, revenue, and expenses. Conversely, transactions below this threshold are regarded as less risky.

The GR-Ind value (whether NA or X) was also found irrelevant to most deviation types, except in cases involving missing(Goods Receipt). The combination ⟨*missing(GR), GR-Ind=NA*⟩ was labeled as false-positive because *GR-Ind=NA* indicates the purchase order pertains to a service, for which a Goods Receipt is not required.

Following the labeling process for the itemset, these labels were applied to the deviation set. The framework subsequently classified 9,303 cases as anomalies and moved them to the anomalies set. Furthermore, 36 process instances were moved to the exceptions set. In these instances, the deviations were limited to ⟨*missing(GR), GR-Ind=0*⟩ or ⟨*missing(GR), GR-Ind=0*⟩ combined with other deviation types that had previously been labeled as false-positives on the previous steps.

Following the propagation of the auditor's labels to the remaining deviation set, 2,765 cases were moved to the anomaly set due to the presence of at least one NOK-labeled deviation instance. It is important to note that classification decisions are propagated at the case-level. Consequently, in addition to the 2,765 deviation instances corresponding to the NOK-labeled itemset ⟨mis(Sign), Pay-value > 1000⟩, these cases also contain 2,804 additional deviation instances (consisting of *insertion(Change Line)*, *repetition(Pay)*, *repetition(IR)*). These deviation instances are therefore also removed from the remaining unlabeled deviation set as part of the case-level propagation mechanism.

On the exception side, at the deviation instance level, 4,181 instances were labeled as OK due to the presence of ⟨missing(GR), GR-Ind=NA⟩, reflecting cases where the absence of a goods receipt is consistent with transactions involving services. As a result, 631 cases were moved because all deviation instances within those cases were labeled as OK. In addition, 6,227 cases were propagated to the exception set based on the pattern ⟨ins(Change Line), Sign, Release⟩, which was consistently labeled as acceptable.

Overall, Level 3 leads to a substantial reduction of the remaining deviation set. From an input of 11,875 cases and 21,037 deviation instances, 9,623 cases (81.0%)

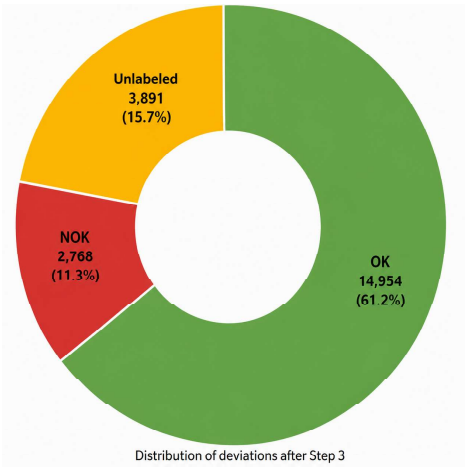
Frequent deviation sequences and cases attributes	Count	Label
⟨ins(Change Line), Sign ⟩	9706	?
⟨ins(Change Line), Sign, $Pay - value > 1000$⟩	7266	?
⟨ins(Change Line), Sign, Release ⟩	6946	OK
⟨ins(Change Line), Sign, GR-Ind=X ⟩	6835	?
⟨ins(Change Line), Sign, Release, GR-Ind=X ⟩	6759	OK
⟨ins(Change Line), Sign, Release, GR ⟩	5680	OK
⟨ins(Change Line), Sign, Release, GR, IR ⟩	5680	OK
⟨ins(Change Line), Sign, GR-Ind=X, $Pay - value > 1000$⟩	4460	?
⟨ins(Change Line), Sign, Release, GR-Ind=X, $Pay - value > 1000$⟩	4405	OK
⟨mis(GR), GR-Ind = NA ⟩	4381	OK
⟨mis(GR), GR-Ind = NA, $Pay - value > 1000$⟩	3908	OK
⟨ins(Change Line), Sign, Release, $Pay - value > 1000$⟩	4252	OK
⟨mis(Sign), GR-Ind= NA ⟩	3204	?
⟨ins(Change Line), Sign, Release, GR, $Pay - value > 1000$⟩	3336	OK
⟨ins(Change Line), Sign, Release, GR, IR, $Pay - value > 1000$⟩	3336	OK
⟨mis(Sign), Release ⟩	3169	?
⟨mis(Sign), Release, IR ⟩	3033	?
⟨ mis(Sign), Pay-value >1000 ⟩	2765	NOK
⟨mis(Sign), GR-Ind= NA, $Pay - value > 1000$⟩	2744	?
⟨ins(Change Line), Sign, $Pay - value < 1000$⟩	2440	?
⟨ins(Change Line), Sign, Release, $Pay - value < 1000$⟩	2419	OK
⟨ins(Change Line), Sign, GR-Ind=X, $Pay - value < 1000$⟩	2375	?
⟨ins(Change Line), Sign, Release, GR-Ind=X, $Pay - value < 1000$⟩	2354	OK
⟨ins(Change Line), Release ⟩	2345	?
⟨ins(Change Line), mis(Sign), Release ⟩	2345	?
⟨ins(Change Line), mis(Sign), Release, IR ⟩	2345	?
⟨ins(Change Line), mis(Sign), Release, IR, mis(GR) ⟩	2345	?
⟨ins(Change Line), mis(Sign), Release, IR, mis(GR), GR-Ind=NA ⟩	2345	?
⟨mis(Sign), Release, IR, mis(GR) ⟩	2345	?
⟨ins(Change Line), Sign, Release, GR, $Pay - value < 1000$ ⟩	2344	OK
⟨ins(Change Line), Sign, Release, GR, IR, $Pay - value < 1000$ ⟩	2344	OK

Table 6.8: The most frequent sequences in Level 3 and their supports where minsupp = 10%. The itemsets are labeled by the auditor as OK, NOK, or left unlabeled(?).

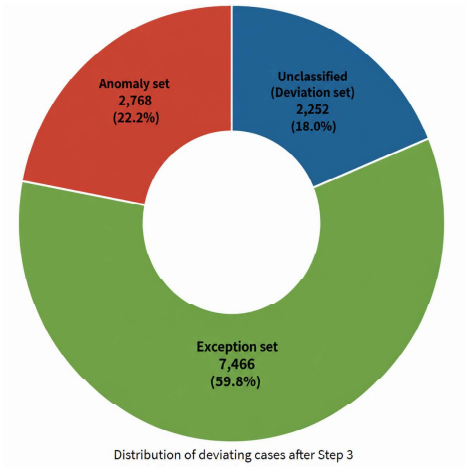
and 9,799 deviation instances (40.1%) are resolved through label propagation. Of the resolved cases, 2,765 cases (23.0% of the input cases) are classified as anomalies due to the presence of at least one NOK-labeled deviation instance, while 6,858 cases (57.7%) are classified as exceptions because all deviation instances within those cases are labeled as OK.

It is important to note that a portion of these cases could only be classified at this level despite having a combination of labeled and unlabeled deviation instances from previous levels. The incorporation of case-level attributes enables the resolution of such cases by providing sufficient contextual information for a definitive classification. Consequently, 2,254 cases (19.0%) and 11,238 deviation instances (53.4%) remain unresolved and are passed to Level 4.

Figure 6.12 presents the statistics of cases and deviations after this step. Figure 6.12a illustrates the distributions of deviations following the labeling process in Level 3, and Figure 6.12b depicts the distribution of cases across three sets of deviation, exception, and anomaly upon the completion of Level 3.



(a) The distribution of labeled deviation instances: 14,954 labeled as OK, 2768 as NOK, and 3,891 remain unlabeled.



(b) The distribution of cases: 7,466 classified as exceptions, 2,768 as anomalies, and 2,252 remain unclassified.

Figure 6.12: Distribution of cases and deviating instances after Level 3

The results of Level 3 demonstrate the practical importance of case-level context. Compared with the previous levels, this stage leads to a much larger reduction of the remaining deviation set, showing that many unclassified cases can indeed be explained once limited case-level information is added. In particular, the labels attached to

combinations involving GR-Ind and Value-Pay illustrate how the same control-flow deviation may take on a different meaning depending on the surrounding process context.

In terms of the evaluation metrics, Level 3 achieves the highest reduction rate, confirming the importance of incorporating case-level attributes. The addition of contextual information enables a substantial portion of previously unresolved cases to be classified, significantly decreasing the remaining set size and demonstrating that control-flow information alone is insufficient for full-population classification.

Level 4: Classification with event-level attributes

At Level 4, the framework incorporates event-level attributes, providing the most detailed level of information used in the classification procedure. This level is applied only to the cases that remain unresolved after Levels 1–3. Event attributes, the most granular contextual information in the event log, offer insights into individual events within a process trace. This helps the auditor classify deviations that could not be resolved at higher abstraction levels.

This level is also important from an algorithmic-engineering perspective, because it demonstrates that the framework does not simply accumulate information indiscriminately. Rather, it controls the escalation of informational detail. Only after higher-level representations fail to resolve the remaining cases does the framework expose the auditor to event-level attributes such as modification values or reference matches between related documents.

Examples of event attributes in our case study include the two event attributes of Change Line: '*Modification*,' and '*Rel-Modification*.' The *Modification* attribute is a numerical attribute where a value "0", indicates that no modification has occurred in the purchase order's quantity or price. However, when the Modification is a non-zero value, it represents the magnitude of the difference between the original and the modified amount or price. Similarly, *Rel-Modification* attribute contains the relative difference between the amounts or the prices pre- and post- Modification. These attributes are critical for auditors seeking to understand what transpired during modifications in the process.

The *Modification* attribute illustrates the role of event-level data in supporting auditor investigations. Each event in a trace is associated with multiple attributes that provide detailed contextual information. At this stage, these event-level attributes are incorporated to support the classification of the remaining deviations. Frequent itemsets are identified and presented to the auditor for labeling, and the resulting

labels are propagated to classify the corresponding cases as anomalies or exceptions. For example, itemsets containing *insertion(Change Line)* or *repetition(Change Line)* were not labeled unless they were followed by both *Sign* and *Release*, indicating that the modification had been properly controlled. This reflects the need to consider the nature and magnitude of changes when assessing such deviations. The event attribute 'Modification' contains this information. According to the auditor, if the attribute is "0", the deviation can be cleared and safely classified as an exception. Conversely, if the value is non-zero, the auditor flags the case for further investigation with the process owner. Notably, in this particular event log, only modifications affecting the amount or price considered material and raised the auditor's concern. Other types of modifications, such as minor changes in the documents, are deemed insignificant and not suspicious.

Table 6.9 outlines the event attributes associated with each activity. These attributes are further discussed in the feature selection section. Note that all events in this data set also include the following three general attributes: *org:resource*, which identified the resource performing the event; *time:timestamp*, indicating when the event was completed; and *lifecycle:transition* specifying whether the event is completed. These three general attributes are not detailed in Table 6.9 or used in this approach due to space limitations and their lower relevance for auditing purposes.

To apply event-level analysis in a structured manner, this level consists of two main steps: (i) feature selection, where relevant event attributes are identified, and (ii) labeling, where the resulting representations are evaluated by the auditor.

Feature Selection

As already explained, the attributes such as GR-Quantity, IR-Quantity, GR-Value, IR-Value, and Pay-Value are excluded from the event log analysis, as their aggregate values are already available at case-level, and they are already considered. The event-level analysis in this step focuses on attributes that are directly informative for the remaining unresolved deviations.
For deviating instances involving *insertion(Change Line)* or *repetition(Change Line)* deviation types, we focus on the Modification attribute as it is more pertinent to auditing purposes.

To streamline the analysis, we define a new binary attribute *Modification* which equals "0" if Modification is 0, and equals "1" if Modification is not zero.

For each Goods Receipt event, there is a reference code, GR-LFBNR, which should correspond and match to an invoice document, IR-LFBNR, the reference code to the goods document. For every case, for all Goods Receipt events, we verify whether

Event	Event attribute	Event attribute description	Type
Change Line	Rel-Modification	Relative difference between old value and new value	Numerical
	Modification	Absolute difference between old value and new value	Numerical
GR	GR-Quantity	Quantity of Goods Receipt	Numerical
	GR-LFBNR	SAP field with the reference to the invoice doc.	Numerical
	GR-Value	Value of Goods Receipt	Numerical
IR	IR-Objectkey	SAP field with the reference to the payment doc.	Numerical
	IR-Value	Value of Invoice Receipt	Numerical
	IR-Quantity	Quantity of Invoice Receipt	Numerical
	IR-LFBNR	SAP field with the reference to the goods doc.	Numerical
Pay	Pay-Value	Value of Pay	Numerical
	Pay-Objectkey	SAP field with the reference to the invoice doc.	Numerical

Table 6.9: Event-level attributes associated with activities in the case study

each GR-LFBNR matches an IR-LFBNR in at least one Invoice Receipt event in that case. Note that a case may include multiple goods receipts and invoice receipt events, and the sequence of related goods receipts and invoice receipts are not guaranteed. To capture this, we define a new binary attribute, *GR-IR-LFBNR* which equals "Matched," if there is a match between GR-LFBNR in a goods receipt and a IR-LFBNR in an Invoice Receipt. It equals "Not-Matched", if there is no match between the GR-LFBNR and an IR-LFBNR, meaning that there is an extra goods (invoice) receipt that has no reference to an invoice (goods) receipt. A similar check is performed for IR-Objectkey and Pay-Objectkey to ensure the references match each other. We introduce another binary attribute, *IR-Pay-Objectkey*, to compare these two event attributes per case. It equals "Matched" if, for this Pay (Invoice) event, a matched Invoice (Pay) is found in the same case. Otherwise, if there is no match, it equals "Not-Matched". As a result, in this step, we consider these three features to help the auditor investigate the data at the event-level. Table 6.10 summarizes these features, linking deviation types to relevant event attributes.

Deviation type	Feature	Feature description	Feature Type
ins(Change Line), rep(Change Line)	Modification	Difference between old value and new value	Binary
rep(GR), rep(IR)	GR-IR-LFBNR	Compare the goods and invoice references	Binary
rep(IR), rep(Pay)	IR-Pay-Objectkey	Compare the IR-Objectkey and IR-Pay-Objectkey references	Binary

Table 6.10: Three new event attributes calculated and added to database in Level 4

Labeling

In this step, the approach provides the deviation types, alongside frequent event attributes. With this additional context, the auditor classifies deviations into the predefined classes. The auditor used the three event attributes to label the deviations in the deviation set as OK, NOK, or leave them unlabeled. Table 6.11 summarizes how the auditor labeled deviations using these attributes in this step. The minimum support threshold was initially set to 10% but since this produced only five itemsets to label, we lowered the minimum support threshold to 1% resulting in 19 itemset to present to the auditor.

Four itemsets are labeled as NOK:
Itemsets ⟨*rep(Pay), IR-Pay-objectkeys= Not-matched*⟩ and ⟨*rep(GR), IR-GR-LFBNR= Not-matched*⟩ were labeled NOK because at least one of the references does not match each other.
Itemset ⟨*ins(Change Line), mis(Sign), mis(GR), GR-Ind = NA, Pay-value <1000, Modification <>0*⟩ was labeled as NOK, due to lack of control (signature) following the change line and a non-zero modification.
The itemset ⟨*ins(Change Line), GR, IR, Pay, Modification <>0*⟩ was also labeled NOK for the same reason because the goods receipt, invoice receipt, and payment after a change line are performed without any approval on a Modification.

Some itemsets, such as ⟨*mis(Sign), Pay-value <1000)*⟩ and ⟨*rep(Change Line), Sign, Release*⟩, appear as frequent patterns at this level due to the lower support threshold, even though they do not explicitly include event-level attributes. These patterns were nevertheless presented to the auditor and labeled accordingly, as they contribute to the classification of the remaining cases. They were not identified in Level 3 because of the higher minimum support threshold applied at that stage.

At the end of Level 4, 1,116 itemset instances were labeled as NOK, covering 1,204 deviation instances, resulting in 338 cases (15.0% of the input cases to this level) being moved to the anomaly set.
Note that some NOK-labeled itemsets are related through subsumption relationships; for instance, in Table 6.11, the itemset ⟨ins(Change Line), mis(Sign), Modification $\neq 0$⟩ covers more specific itemsets listed below (with identical support), and therefore these are not counted separately.

Additionally, 2,522 deviation instances were labeled as OK, moving 1,778 cases (78.9%) to the exception set as all corresponding deviation instances were labeled as OK.
Similarly, some OK-labeled itemsets exhibit subsumption relationships; for example, the itemset ⟨rep(Change Line), Sign, Release⟩ covers its more specific variants that include additional conditions (e.g., Modification), and should therefore be interpreted as representing the same set of cases rather than distinct contributions.

Overall, 3,726 additional deviation instances were labeled in this level, consisting of 2,522 labeled as OK and 1,204 labeled as NOK (including those covered by overlapping itemsets). In total, 2,116 out of 2,252 cases (94.0%) were resolved at this level, leaving only 136 cases (6.0%) for final manual inspection in Stage 3.

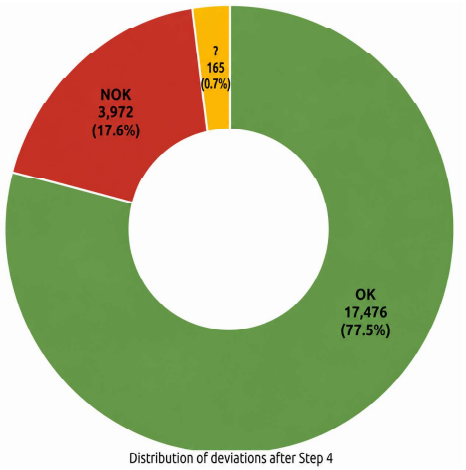
Figure 6.13 shows the cases and deviation statistics after this step. Figure 6.13a displays the distributions of deviations after the labeling in Level 4, and Figure 6.13b shows the distribution of cases in three sets of deviation, exception, and anomaly after the completion of Level 4.

The results of Level 4 show that event-level enrichment is able to resolve a substantial share of the cases that remained ambiguous after the earlier levels. This confirms that some deviations cannot be judged reliably without examining fine-grained transactional details. At the same time, the fact that only a relatively small residual set reaches this stage illustrates one of the main strengths of the framework: detailed inspection is reserved for the minority of cases that genuinely require it.

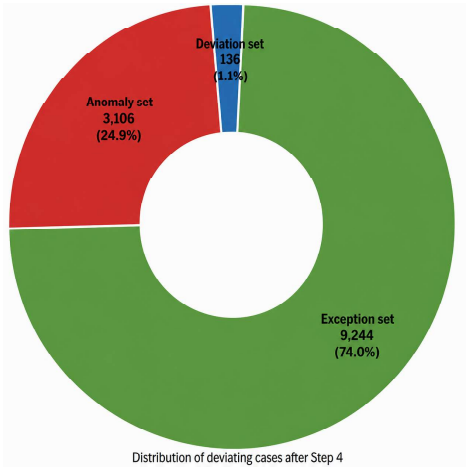
In terms of the evaluation metrics, Level 4 further reduces the remaining deviation set by leveraging event-level attributes for analysis. The results show that only a small subset of cases requires this level of detail, demonstrating the effectiveness of the framework in deferring detailed inspection until it is necessary.
Taken together, the four classification levels illustrate the main contribution of the framework: the incremental reduction of the unclassified deviation set through increasingly informative representations. The results show that different levels contribute in different ways. Level 1 resolves only a limited subset of clearly interpretable deviations. Level 2 adds information by identifying recurrent combinations. Level 3

Frequent deviation sequences and cases attributes	Count	Label
⟨rep(Pay), IR-Pay-objectkeys= Matched⟩	1076	OK
⟨**rep(Pay), IR-Pay-objectkeys= Not-matched**⟩	786	NOK
⟨rep(Change Line), Modification= 0⟩	464	OK
⟨mis(Sign), Pay-value <1000⟩	439	OK
⟨ins(Change Line), Modification= 0⟩	414	OK
⟨ins(Change Line), Modification <> 0 ⟩	229	?
⟨**rep(GR), IR-GR-LFBNR= Not-matched** ⟩	166	NOK
⟨rep(Change Line), Sign, Release⟩	156	OK
⟨rep(IR), IR-GR-LFBNR= Matched⟩	129	OK
⟨rep(IR), IR-GR-LFBNR= Matched, IR-PAY-Objectkeys= Matched⟩	129	OK
⟨ins(Change Line), GR, Modification <> 0 ⟩	114	?
⟨ins(Change Line), mis(Sign), Pay-value <1000 ⟩	88	?
⟨ins(Change Line), mis(Sign), mis(GR), Pay-value <1000⟩	88	?
⟨ins(Change Line), mis(Sign), mis(GR), GR-Ind = NA, Pay-value <1000⟩	88	?
⟨**ins(Change Line), mis(Sign), Modification** <> 0⟩	88	NOK
⟨ins(Change Line), mis(Sign), Pay-value <1000, Modification <> 0 ⟩	88	NOK
⟨ins(Change Line), mis(Sign), mis(GR), Pay-value <1000, Modification <> 0 ⟩	88	NOK
⟨ins(Change Line), mis(Sign), mis(GR), GR-Ind = NA, Pay-value <1000, Modification <> 0 ⟩	88	NOK
⟨**ins(Change Line), GR, IR, Pay, Modification** <> 0⟩	76	NOK

Table 6.11: The most frequent sequence patterns in Level 4 and their supports and labels- minsupp=1%. The itemsets are labeled by the auditor as OK, NOK, or left unlabeled(?).



(a) The distribution of labeled deviation instances: 17,476 labeled as OK, 3,972 labeled as NOK, and 165 remain unlabeled.



(b) The distribution of cases: 9,244 classified as exceptions, 3,106 as anomalies, and 136 remain unclassified.

Figure 6.13: Distribution of cases and deviating instances after applying Level 4

contributes the largest reduction by incorporating case-level contextual information that is highly relevant for auditing interpretation. Level 4 then resolves a further part of the residual ambiguity by using event-level attributes. This process confirms the central premise of Chapter 5: the classification of process deviations should not begin with the most detailed information available, but with the least detailed information that is sufficient for classification decision-making. Doing so reduces the auditor's effort, structures the interaction between algorithm and expert judgment, and preserves interpretability throughout the process.

6.5.3 Stage 3: Classification of Remaining Deviations

In accordance with Stage 3 of the framework defined in Chapter 5, the remaining deviations after Level 4 are classified through manual case by case inspection. At this stage, all deviations that could not be labeled through the preceding levels of incremental enrichment are addressed individually by the auditor.

After Level 4, a total of 165 deviation instances across 136 cases remain unlabeled. These deviations correspond to cases for which neither deviation types, deviation patterns, nor case- and event-level attributes provided sufficient information for classification. As such, manual inspection is required to determine whether they should be labeled as anomalies (NOK) or acceptable exceptions (OK).

It is important to note that these additional deviations could potentially have

been classified in Levels 3 and 4 by adjusting the minimum support threshold. Lowering this threshold would result in a larger number of patterns, which could increase coverage. However, this would also increase the volume of information presented to the auditor and may lead to information overload. In this evaluation, a balance was maintained between classification coverage and interpretability. Nevertheless, the framework allows the auditor to adjust such parameters, providing flexibility to trade off between completeness and information load depending on the specific auditing context.

A closer examination of the remaining deviations reveals that several of them exhibit characteristics that can be interpreted using detailed transactional information. In particular, six instances of repetition(IR) are identified where the object key (LFBNR) does not match the corresponding reference within the same case. These deviations indicate inconsistencies in invoice referencing and are therefore classified as NOK.

Furthermore, three instances of missing(GR) with GR-Ind = X are observed. Given that the GR indicator confirms the expected presence of a goods receipt, the absence of the corresponding event constitutes a confirmed deviation and is therefore labeled as NOK.

In addition, 100 instances of repetition(Change Line) and 56 instances of insertion(Change Line) are identified with a non-zero modification value (Modification $\neq 0$), and without a subsequent control activity such as Sign or Release. The absence of these control activities following a modification suggests that the changes were not properly validated. These deviations are therefore also classified as NOK.

This final stage ensures that all remaining deviations are systematically reviewed, thereby completing the classification of the full deviation set and ensuring full population coverage in line with the objectives of the framework.

6.6 Evaluation of Framework Performance

To ensure a structured evaluation in line with the algorithm engineering perspective, the performance of the framework is assessed along clearly defined metrics. These include (i) the reduction rate across classification levels, (ii) the remaining set size after each level, and (iii) the effort reduction achieved through the use of the framework. The evaluation is conducted at both the case-level and the deviation instance level, providing a transparent view of how the framework affects different units of analysis.

The primary objective of the proposed framework is to reduce the amount of manual effort required for deviation classification. To assess this objective, the required

effort under the framework is compared with a fully manual baseline, and analyzed in relation to the reduction achieved across the different classification levels.

Auditor effort (input metric)

After Step 1, the deviation set consists of 12,486 cases and 24,417 deviation instances. In a fully manual setting, the auditor would need to inspect either all deviating cases or all deviation instances with full contextual information. These values therefore represent the baseline effort required without any support mechanism.

In the proposed framework, auditor effort is distributed across two components. First, the auditor performs evaluation actions on aggregated representations during Stage 2. Across all classification levels, a total of 74 evaluation actions were performed (13 in Level 1, 11 in Level 2, 31 in Level 3, and 19 in Level 4).

Second, a residual set of 136 cases remains unresolved after Level 4 and requires manual case-by-case inspection in Stage 3. These cases represent the portion of the deviation set that cannot be classified through aggregated representations and therefore must be inspected individually.

The total auditor effort within the framework can therefore be expressed as the combination of these two components, resulting in 74 structured evaluation actions and 136 manual case inspections. This corresponds to a total of 210 evaluation decisions, compared to 12,486 case-level inspections in a fully manual setting.

It should be noted that the auditor effort metric is based on counting evaluation actions and therefore assumes that these actions are comparable in terms of effort. In practice, this assumption does not fully hold. Different types of evaluations involve different levels of cognitive complexity. For example, labeling a high-level deviation type typically requires less effort than analyzing an enriched itemset or performing a detailed case-by-case inspection.

This implies that the metric should be interpreted as a structural measure of effort, reflecting the number of decisions required, rather than a precise measure of cognitive load or time required. Nevertheless, it remains appropriate for comparative analysis across the different levels of the framework, as it consistently reflects the extent to which the classification task is shifted from detailed case-by-case inspection toward higher-level decision-making on aggregated representations.

The distribution of effort across levels is not uniform. While the highest number of evaluation actions occurs at Level 3, this is not solely a consequence of increased complexity introduced by case-level attributes, but also reflects the choice of the minimum support threshold.

In particular, the number of itemsets presented to the auditor at Levels 3 and 4

is directly influenced by the selected support threshold, which determines how many patterns are considered for labeling. Lower thresholds increase coverage but also increase the number of evaluation actions required, whereas higher thresholds reduce the labeling effort at the cost of leaving more cases unresolved.

Therefore, the observed effort distribution across levels is partially controlled by the framework configuration. This highlights an important characteristic of the approach: auditor effort can be explicitly managed by adjusting the level of granularity at which patterns are presented, allowing a trade-off between labeling effort and classification coverage.

Compared to the baseline of 12,486 case-level inspections, the framework reduces the number of explicit evaluation decisions by several orders of magnitude. This demonstrates that auditor effort is not only reduced, but also restructured toward higher-level decision-making over aggregated representations.

Effort reduction (performance metric)

In the proposed approach, the auditor evaluates aggregated representations (deviation types, sequential patterns, and enriched itemsets), and the corresponding labels are propagated automatically to all associated cases and deviation instances. Across all levels in Stage 2, the auditor performed 74 evaluation actions in total (13 in Level 1, 11 in Level 2, 31 in Level 3, and 19 in Level 4).

It is important to note that evaluation actions correspond to labeling aggregated representations rather than individual cases, and therefore each action may affect a large number of cases simultaneously through label propagation.

This result is particularly significant in auditing contexts, where manual inspection is the primary bottleneck. The ability to classify large groups of deviations through a small number of evaluation actions demonstrates that the framework effectively shifts the auditor's role from case-by-case inspection to higher-level decision-making.

The effort reduction achieved by the framework is defined as the ratio between the total number of evaluation actions required when using the framework, including both aggregated labeling decisions and the remaining manual inspections, and the total number of cases that would require inspection in a fully manual setting. In this evaluation, this corresponds to 74 evaluation actions and 136 remaining cases, relative to a baseline of 12,486 cases.

The effort reduction at case-level can be formally expressed as:

$$ER_{cases} = \frac{74 + 136}{12,486}$$

This corresponds to an effort reduction of approximately 98.32%, meaning that the framework eliminates the need for manual inspection in the vast majority of cases.

The effort reduction can also be expressed at the deviation instance level:

$$ER_{instances} = \frac{74 + 165}{24,417}$$

This corresponds to an effort reduction of approximately 99.02%, indicating that nearly all deviation-instance inspections are avoided through the use of the framework. Compared to the baseline of 12,486 case-level inspections or 24,417 deviation-instance inspections required in a fully manual setting, the framework reduces the auditor's effort to 74 evaluation actions, supplemented by the manual inspection of the remaining 136 cases (corresponding to 165 deviation instances) that could not be classified through the incremental levels.
This corresponds to a major improvement in evaluation efficiency, where each individual evaluation action results in the classification of a large number of deviation instances. This amplification of auditor effort is a strength of the framework, as it enables efficient large-scale deviation analysis with minimal manual input.

Reduction rate (output metric)

To provide a consolidated view of the framework's performance, the reduction rate achieved at each classification level is summarized below.
At Level 1, 2,606 deviation instances (10.7%) were labeled, resulting in 283 cases (2.2%) being classified and removed from the deviation set. At Level 2, a further 759 deviation instances (3.1%) were newly resolved through deviation-pattern analysis, leading to 328 additional cases (2.6%) being classified.
The largest reduction occurs at Level 3, where 9,799 deviation instances (40.1%) are resolved, corresponding to 9,623 cases (77.0% of the Stage 2 input) being classified through the incorporation of case-level attributes. Finally, Level 4 resolves 2,116 of the remaining cases, corresponding to 16.9% of the initial deviation set (and 94.0% of the Level 4 input), leaving only 136 cases (1.1%) for final manual inspection. Table 6.12 summarizes these results.

These results show that the majority of deviations can be resolved before reaching the most detailed level, with the most significant contribution stemming from the integration of case-level context in Level 3. This confirms that the incremental enrichment strategy effectively concentrates detailed inspection on a small residual subset of cases, thereby substantially reducing the overall manual effort required for deviation classification.

Reduction rates are computed with respect to the number of elements entering each level.

Level	Labeled Deviation Instances	(% of total)	Classified Cases	(% of total)
Level 1	2,606	10.7%	283	2.2%
Level 2	759	3.1%	328	2.6%
Level 3	9,799	40.1%	9,623	77.0%
Level 4	3,726	15.3%	2,116	16.9%
Remaining	165	0.68%	136	1.1%

Table 6.12: Summary of reduction achieved at each level relative to the input of Stage 2 (12,486 cases and 24,417 deviation instances).

Due to case-level propagation, some deviating cases are classified without all their individual deviation instances being explicitly labeled.
These results reveal an important characteristic of the framework. The majority of deviations are not resolved at the highest abstraction levels, but at intermediate levels where contextual information becomes available. In particular, the dominant contribution of Level 3 indicates that control-flow information alone is insufficient for reliable classification, and that incorporating business context is essential for aligning automated analysis with auditor reasoning.

In addition to the reduction in manual effort, the framework also supports the identification of anomalous behavior through the labeling of NOK deviations. This aspect is assessed qualitatively based on auditor judgment rather than against a predefined reference classification. A formal evaluation of classification accuracy is not conducted, as it would require a fully labeled dataset, which is not available in this setting.

Remaining set size (output metric)

The remaining set size reflects the number of unresolved elements after each classification level, both at the case-level and at the deviation instance level. At the case-level, the number of unresolved cases decreases from 12,486 at the beginning of Stage 2 to 136 after Level 4. Similarly, at the deviation instance level, the number of unresolved instances decreases from 24,417 to 165. This illustrates how the framework systematically reduces the analysis space across levels. In particular, the most

significant reduction occurs at Level 3, after which only a small residual set remains. This confirms that the majority of deviations can be resolved before requiring the most detailed level of inspection.

To provide a consolidated overview of the evaluation results, Table 6.13 summarizes the four metrics defined in Section 6.2 together with their observed values.

Metric	**Definition**	**Observed Value**
Auditor Effort (Input)	Number of evaluation actions performed by the auditor across all levels, including aggregated labeling and residual manual inspection	74 aggregated evaluation actions + 136 manual case inspections (total: 210 decisions)
Reduction Rate (Output)	Proportion of deviation instances or cases resolved at each classification level	Level 1: 10.7% (instances), 2.2% (cases) Level 2: 3.1%, 2.6% Level 3: 40.1%, 77.0% Level 4: 15.3%, 16.9%
Remaining Set Size (Output)	Number of unresolved deviation instances or cases after each level	Reduced from 12,486 to 136 cases Reduced from 24,417 to 165 instances
Effort Reduction (Performance)	Ratio between effort required using the framework and effort required in a fully manual setting	Case level: 98.32% reduction Deviation instance level: 99.02% reduction

Table 6.13: Summary of evaluation metrics and observed results for the proposed framework.

Overall, the results confirm that the framework achieves its primary objective of substantially reducing manual auditor effort. The evaluation across the four defined metrics provides a comprehensive view of its performance.

The auditor effort metric shows that only 74 evaluation actions are required to classify the majority of deviations. The reduction rate demonstrates that a large proportion of cases and deviation instances are resolved progressively across the classification levels, with the most significant contribution at Level 3. The remaining set size analysis confirms that only a small subset of cases (136 out of 12,486) requires detailed manual inspection. Finally, the effort reduction metric quantifies this

improvement, showing that more than 98% of manual inspection effort is eliminated.

Together, these results demonstrate that the framework enables efficient, structured, and scalable deviation classification while preserving interpretability in an auditing context.

6.7 Discussion, limitations and conclusions

In this chapter, we evaluated the process deviation classification framework of Chapter 5 on a real-life purchase-to-pay event log containing 26,185 cases and 181,845 events. In line with the framework assumptions, preliminary control validation was treated separately from the classification procedure itself. The proposed framework was then applied to the prepared event log through Step 1 and four incremental classification levels.

The evaluation demonstrated that the framework can substantially reduce the amount of manual case-by-case review required from the auditor. Starting from the set of deviating cases identified after conformance checking, the framework progressively reduced the unresolved deviation set by first using deviation types, then frequent deviation patterns, then case-level attributes, and finally event-level attributes. At the end of the procedure, only 165 deviations across 136 cases remained unresolved and required direct manual classification into the exception or anomaly set based on their deviation types

These results provide quantitative evidence of the practical applicability of the framework, particularly in reducing manual auditor effort while maintaining structured and interpretable classification decisions. More specifically, they show that the proposed incremental information-enrichment strategy supports auditor decision-making by postponing detailed data inspection until it is actually needed. This makes the classification process more structured, more transparent, and more efficient than a fully manual review of all deviating cases.

The results also indicate that the framework could, in principle, support the analysis of additional control mechanisms beyond control-flow deviations. For instance, checks related to three-way matching and segregation of duties could be incorporated through the inclusion of appropriate case- and event-level attributes. However, integrating such controls directly into the classification phase would shift the focus of the framework and may limit its ability to cover both normal and deviating cases consistently.

These findings can be further understood by positioning the framework relative to existing conformance checking and process mining approaches. Unlike traditional

conformance checking approaches, which often produce large volumes of low-level deviations requiring manual inspection e.g., [van der Aalst, 2016], the proposed framework structures these deviations into progressively enriched representations. This enables auditors to classify large groups of deviations with limited effort.

Compared to existing approaches in process mining that focus primarily on deviation detection, the results show that the proposed framework contributes not only to identifying deviations but also to supporting their interpretation and classification in an auditing context. This distinction is particularly important in continuous auditing settings, where the main challenge is not only detecting deviations but managing and interpreting them efficiently.

While this chapter, focuses on complete process instances, it is equally important for auditing purposes to address incomplete processes, particularly in the case of "hard close" or "fast close." It is possible that some processes are not completed in a defined audit cycle (year). Incomplete processes that extend beyond an audit cycle (e.g., pending payments) should be flagged for follow-up or considered in the subsequent cycle to ensure comprehensive coverage.

By methodically applying the framework's steps to real event logs, this chapter has shown how the methodology reduces the burden of manual analysis for auditors in detecting deviations. Furthermore, the integration of theoretical concepts, such as deviation categorization and auditing principles, into actionable strategies demonstrates the framework's robustness and relevance. These findings underscore the framework's potential as a catalyst for improving auditing practices, bridging the gap between theoretical constructs and practical applications, and achieving the continuous auditing objectives discussed in Chapter 5.

Limitations and generalizability The evaluation is conducted on a single purchase-to-pay process. While this enables an in-depth analysis of the framework's behavior, the results may depend on process characteristics such as variability and control structure. Consequently, the generalizability of the findings remains limited, and further validation on different process types is required to assess the broader applicability of the framework.

Nevertheless, the selected case provides a realistic and representative setting for assessing the framework in a complex auditing context, where large volumes of heterogeneous deviations must be analyzed in a structured manner. Future research should evaluate the framework on processes with different levels of structure, variability, and domain characteristics, such as order-to-cash or service management processes, to assess its robustness across contexts.

Chapter 7

Conformance Checking and Active Learning in Auditing*

As discussed in Chapter 3, continuous auditing involves information systems to automate the audit process, striving for (near) real-time assurance [Li et al., 2016]. With the rapid advances in technology and techniques, research on the topic of audit analytics – a term that is often used interchangeably with continuous auditing– has emerged and expanded [Moffitt et al., 2016, Lombardi et al., 2025, Kokina et al., 2025, Torroba et al., 2025, Sewpersadh, 2025]. Most recently published works on this topic describe both the opportunities and the challenges of combining data analytics with auditing and accounting. Examples of articles on possible roles data analytics can play in auditing can be found in Kokina et al. [2025], Torroba et al. [2025], Brown-Liburd et al. [2015], Warren et al. [2015], Krahel and Titera [2015], Titera [2013]. Recent research in audit analytics has increasingly focused on the application of data-driven techniques to support auditing tasks, including anomaly detection, risk assessment, and decision support. While early contributions primarily addressed architectural aspects of continuous auditing systems, more recent work emphasizes the integration of advanced analytics and intelligent systems into audit procedures.

Two key elements that were, right from the start, part of the principles of continuous auditing, are the 'audit by exception' principle and a business process view. Both elements have been subjects of separate research investigations. With regard to the exceptions, the CA pilot implementation of Alles et al. [2006a], among others, confirmed the idea of 'discrepancy analysis', coined by Vasarhelyi et al. [2004]. The

*This chapter is based on a study previously published as the journal article "How Active Learning and Process Mining Can Act as a Continuous Auditing Catalyst" in the International Journal of Accounting Information Systems (IJAIS).

focus on exceptions is a result of alarm floods (too many false positives). These can, for example, incapacitate an internal audit department and present a critical problem in the adoption of Continuous Auditing [Li et al., 2016]. The challenge of managing the alarm floods was the subject of study in the work of Perols and Murthy [2012a] and Li et al. [2016], who suggested information fusion and the use of belief functions respectively to deal with the exceptions.

The second element, the business process view, has been part of the continuous auditing principles since the early beginning [Vasarhelyi et al., 2004]. A first experiment on the influence of a process-focused system on auditing was presented by O'Donnell and Schultz Jr [2003]. Although this research was not explicitly related to the CA context, the process aspect stayed under the radar for a long time and was only picked up more recently [Perdana et al., 2023]. The expansion of data analysis opportunities within the broader field of auditing presumably triggered this revival. One type of data analytics that might influence the audit is process mining [Jans et al., 2011b]. The purpose of process mining is to discover, monitor, and improve real processes (as opposed to assumed processes) by extracting knowledge from event logs in information systems [Van der Aalst, 2016]. Process mining could be described as data analytics from a process point of view. To date, some initiatives on applying process mining in an auditing setting have been reported (for example Jans et al. [2014], Werner [2017]).

Where previous research investigated opportunities and challenges of continuous auditing, exception management, data mining, or process mining in isolation from each other, we connect these related issues with each other in a concrete manner. In this chapter, we aim to link the opportunities that both data mining and process mining present for continuous auditing, including the key challenge of dealing with alarm floods. In order to do so, we propose incorporating an active learning mechanism that combines the power of artificial intelligence techniques with the expertise of the human auditor [Sewpersadh, 2025, Torroba et al., 2025]. The auditor functions as an *oracle* to feed a machine learning algorithm that classifies the exceptions. These exceptions are first formulated in a process context. Only later, pure data analysis is added to enrich patterns. To present this holistic, but concrete approach, a framework is proposed.

To ensure an embedding in the continuous auditing principles, the framework relates specifically to one of the proposed levels of Monitoring and Control by Vasarhelyi et al. [2004]: the transaction evaluation level. This level is primarily linked with the tests of controls procedure. In this chapter, we will first explain the underlying principles of the internal control testing procedure and formalize the procedure in an unambiguous way. After formalizing the internal control testing procedure, we

present our framework that explains the use of analytics for this procedure, displaying the following three characteristics: 1) a process view is taken, 2) managing the alarm flood is incorporated in the approach, and 3) human interpretation (the professional judgment of the auditor) is used as input for the data mining algorithm to address this alarm flood. The approach to start from an existing procedure and to identify how analytics can support this, builds on the expectation that CA will be adopted by automating tasks in the first place. Only in a later phase, it is expected that the procedure itself is reconsidered [Vasarhelyi et al., 2004].

By starting from one specific layer, addressing one audit procedure in depth, and integrating previously identified challenges, the proposed framework contributes to the literature by creating synergies between research initiatives that are related to each other, but were previously conducted isolated from each other. This chapter presents a structured proposal on an actionable continuous auditing procedure that incorporates both data mining and process mining techniques and employs the ideas of an active learning approach as leverage to deal with the alarm flood. This last aspect is highly important since exception sampling undermines the strength of full-population testing. The main contributions of our framework are:

- the classification procedure is not required to assign a label 'OK' or 'Not OK' to every transaction. Less certain cases may remain unlabeled and be deferred for further analysis

- professional judgment of the auditor is incorporated into the learning process, to feed the classifier with additional expert knowledge

Incorporating human intervention is also important in light of broader regulatory and compliance with a new European law that forces algorithms to explain their decisions. Additionally, the research stream of process mining for auditing purposes framed in a larger context, along with all its current research challenges, is a relevant new assessment.

Although not the main contribution, the formalization of the internal control procedure in a continuous auditing setting can also be valued as an increase of the knowledge base. The task of internal control testing itself has been represented as a generic process in Business Process Model and Notation (BPMN) specifications. Having a general representation of the process of internal control testing facilitates research on this topic. Namely, having an overview of business processes, and their 'as is' enactment, enables organizations to identify potential improvements. We include the proposition that this rationale on business processes can be extended to the processes and procedures of auditing as well.

7.1 Related Literature

In Chapter 3 a background and literature review on continuous auditing and process mining is provided. This section provides a background on active learning and the related literature on managing alarm floods.

7.1.1 Active learning

In many real-world applications, labeling data manually is a tedious, time-consuming and expensive task, particularly when expert knowledge is required. It may also suffer from significant errors due to human annotation. Moreover, not all instances are equally informative or equally easy to label. For example, a process instance similar to what the learner has already seen may provide limited additional value, while other instances may require greater labeling effort from the oracle. Active learning is a machine learning paradigm that aims to reduce labeling effort by selectively choosing the most informative instances for annotation by an oracle (e.g., a human expert) Settles [2009].

In contrast to traditional supervised learning, where the model is trained on a fixed labeled dataset, active learning operates in a setting where a large pool of unlabeled instances is available, along with a limited set of labeled examples. The learning algorithm iteratively selects informative instances from the unlabeled pool and queries the oracle for their labels. The newly labeled instances are then incorporated into the training set, and the model is updated accordingly. The informativeness of an instance is assessed based on its potential to improve the model when added to the training set [Huang et al., 2010]. Adding the most informative instances to the training set improves the performance of the classifier which consequently, leads to minimizing the amount of required labeled data, reducing classification error and variance over the distribution of instances [Cohn et al., 1996]. As a result, the model is able to learn from fewer labeled instances.

Formally, an active learning system consists of the following components:

- a labeled dataset, initially small,
- an unlabeled dataset, typically large,
- a learning algorithm that builds a classifier based on the labeled data set,
- a query strategy that selects instances from the unlabeled data set to be labeled by the oracle.

Active learning can be categorized into different scenarios depending on how instances are selected for labeling. Two commonly distinguished settings are selective sampling and pool-based sampling [Settles, 2009, Tharwat and Schenck, 2023].

In the *selective sampling* setting, instances are observed sequentially, and the learning algorithm decides, for each incoming instance, whether to query its label or discard it. This setting is particularly relevant in streaming environments where data arrives continuously, and immediate decisions must be made [Freund et al., 1997].

In contrast, *pool-based sampling* assumes the availability of a large pool of unlabeled instances. The learning algorithm can inspect this pool and iteratively select the most informative instances according to a query strategy, such as uncertainty sampling. These selected instances are then presented to the oracle for labeling.

Given that, in the context of transactional verification, a complete set of deviations is available after the conformance checking phase, the proposed framework follows a pool-based active learning setting. This allows the system to prioritize the most informative deviations for expert labeling, thereby reducing the overall labeling effort while maximizing the impact of each interaction.

The key element of active learning lies in the query strategy, which determines which unlabeled instances should be presented to the oracle. Different query strategies have been proposed in the active learning literature (see, e.g., [Hanneke, 2014, Settles, 2009]), including:

- *uncertainty sampling*, where the model selects instances for which it is least confident about its prediction
- *query by committee*, where multiple models are trained and instances with the highest disagreement are selected,
- *expected model change* or *expected error reduction*, where instances are selected based on their potential impact on the model.

These strategies aim to maximize the informativeness of each labeled instance, thereby improving the performance of the model while minimizing the number of required annotations.

Active learning is particularly suitable for transactional verification. As mentioned above, first, labeling deviations often requires domain expertise and professional judgment, making it expensive and difficult. Second, not all deviations are equally informative; some are trivial, while others provide valuable insight into underlying patterns or risks. By focusing on the most informative and uncertain cases, active learning allows the system to learn efficiently while leveraging the expertise of the auditor in a targeted manner.

In the proposed framework, the auditor acts as the oracle, and active learning is introduced in later iterations, where the system selectively queries the auditor for the most informative cases, as will be discussed in Section 7.3.

7.1.2 Managing Alarm Floods

Research has reported already that a continuous auditing system can generate a large volume of 'exceptions' [Debreceny et al., 2003, Alles et al., 2006a, 2008a, Perols and Murthy, 2012a, Li et al., 2016]. Human judgment is required to process this information (over)load, leading to sub-optimal decision making. Recent research emphasizes the role of intelligent decision support systems to assist auditors in handling large volumes of exceptions [Sewpersadh, 2025, Kokina et al., 2025]. On this topic, Chan and Vasarhelyi [2011a] state: "*[...] the automation of all traditional audit procedures may not be immediately feasible. Audit procedures requiring complex judgment and professional skepticism will still require manual performance by the auditor [...]*". The work of Perols and Murthy [2012a], Swinnen et al. [2011] and Li et al. [2016] address the issue of excessive alarm floods and propose a solution.

Perols and Murthy [2012a] addressed the issue of exceptions by suggesting an information fusion approach. Although the authors do not go into details, their first layer, the monitoring layer, relates to processes. This is also visually expressed in their framework figure (p. 39). The monitoring layer starts from detected exceptions, gathers information on these exceptions, and suggests the use of machine learning algorithms to classify the exceptions. The presented architecture aims to draw meaningful conclusions by processing the exceptions (and their different sources of information).

Li et al. [2016] propose a different framework; one to prioritize exceptions, making use of Dempster-Shafer belief functions. In this framework, the exception detection is limited to applying defined business rules. Based on these rules, the theory of belief functions is used to assign suspicion scores for each transaction that was identified as an exception. The authors followed the arguments of Srivastava and Shafer [1992] that these functions could represent the auditor's intuitive understanding of audit risk and included an iterative component to bring the exceptions to a manageable number. Next, based on the suspicion scores, the exceptions are ranked and auditors manually investigate the cases with the highest likelihood of being errors or fraud. The auditors classify the exceptions into 'normal', 'erroneous', or 'fraudulent'. The manual classifications are subsequently used as new information to fine-tune the initial confidence levels of the rules. An iterative approach and the intervention of the human auditor is inherent to this framework. This can be seen as an example of active learning.

7.2 The procedure of internal control testing in continuous auditing- a process representation

The implementation of a continuous auditing system is primarily the responsibility of the internal auditor [Chan and Vasarhelyi, 2011a, Li et al., 2016] and internal control testing is assumed to serve as a good starting point for this. Both the focus on high-risk processes and the reporting requirements on the effectiveness of the internal controls contribute to the suitability of this procedure as the backbone of a continuous auditing system. Internal control testing is an auditing procedure that corresponds to the first level of continuous assurance and analytical monitoring: transactional verification.

In order to propose a way how analytics can support transactional verification, we first present the generic process of internal control testing. The process is illustrated in Figure 7.1 and makes abstraction of the level of automation that is applied[1]. This way, the core process of internal control testing is presented, regardless of the maturity level of continuous auditing. The process does, however, start from a contemporary financial reporting setting, where information systems like Enterprise Resource Planning systems are deployed. In addition, the exception-based principle of continuous auditing is taken as a foundation. Having this generic process overview facilitates the identification of opportunities to employ analytics for transaction verification. Again, the underlying assumption is to move to continuous auditing by automating existing manual procedures in a first stage.

Although not incorporated in the process itself, testing internal controls starts with the selection of a business process. As suggested by Chan and Vasarhelyi [2011a] in Stage 1, *"the auditor identifies a business process area where continuous auditing can be applied"*, we also start from a selected business process with available data and control settings to test. After the selection, the phases of the depicted process in Figure 7.1 take place.

- **Understand the normative process**
  Once a business process is selected to test the related internal controls, the first step is to get a general understanding of the process. The purpose is to understand the expected process, which is called the normative process. This understanding can be gained through document review, interviews, previous experience with the company, etc. The output of this activity is either implicit

[1]Business Process Modeling and Notation specifications are used as modeling language. This language is chosen for its high level of understandability by non-technical users.

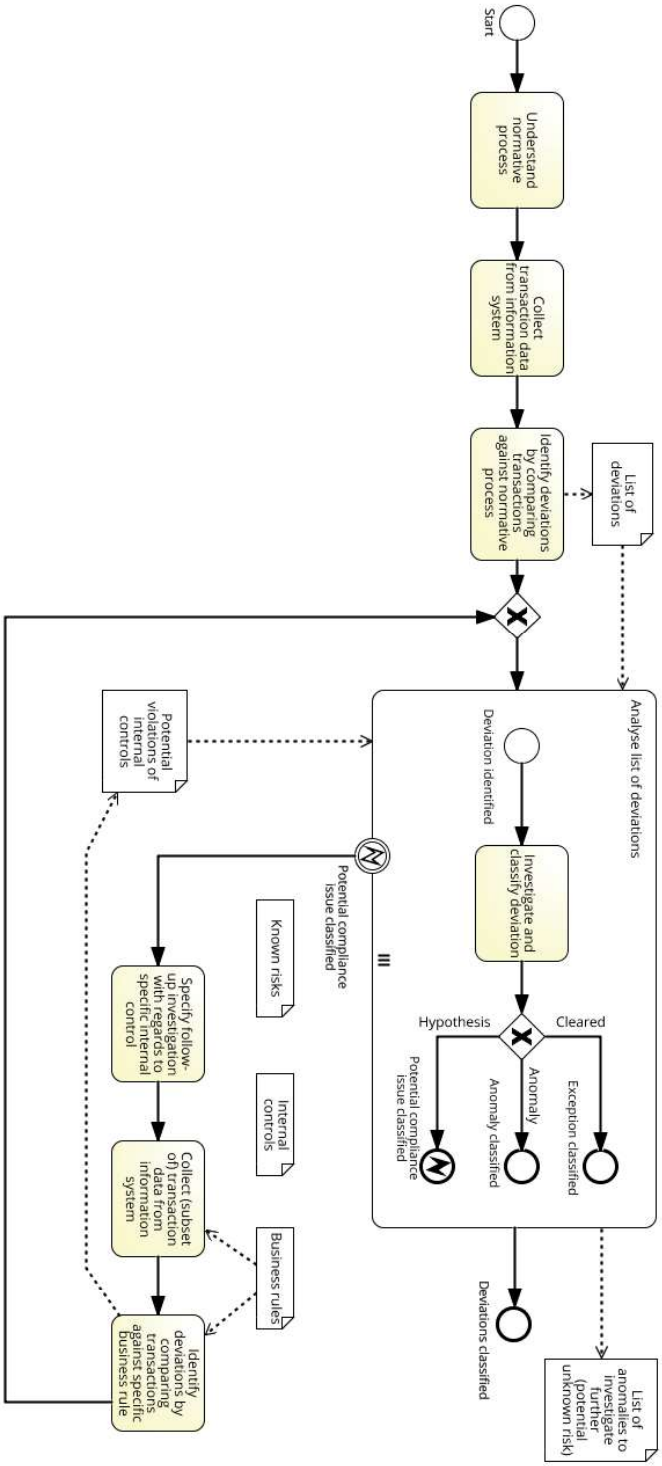


Figure 7.1: Process of Internal Control testing in BPMN specifications

-in the auditor's head-, or explicit in the form of a narrative or a process diagram that describes the normative process.

Aside from understanding the normative process execution, a general understanding of the related process risks must be obtained. The process risks, on their turn, trigger the internal control settings. Although not every single risk and configured control might be disclosed in this first step, understanding the most important risks is an elementary component of understanding the normative process and its controls.

Consider a procurement (P2P) process in our running example and its normative model in Figure 2.2. An example of a process risk can be an execution of a purchase order with no approval. This general level of understanding process risks and their related controls (having an approval strategy in place) are necessary to conduct the procedure. It is important to note that the normative model, like the one in Figure 2.2b, only captures the perspective of activity presence and activity order. Perspectives on who does what, the implied value, or other aspects that do not relate to the presence or order of activities, are not included in this notation. Of course, these perspectives also need to be taken into account, but when analyzing processes, these are often only involved in a second phase.

- **Collect transaction data from information system**
  In the second step, transactions of the process under investigation are extracted from the information system.

  A traditional selection of data of the P2P process is a list of some purchase orders and related invoices, whether or not this process is investigated through analytics or manually. Tables 7.1a and 7.1b show two short exemplary lists that could be the output of this step. In case an automated audit would take place, these lists would contain all relevant purchase orders and invoices of the period under investigation.

- **Identify deviations by comparing transactions against the normative process**
  In the third step, the normative process and the available transaction data are compared with each other. If the transaction, or the process execution that precedes the transaction, is in line with the expected normative model, there is evidence for assurance. If the transaction deviates from what is expected, the deviation is stored in a separate list. Comparing the available transaction data, illustrated in Tables 7.1a and 7.1b, with the normative model in Figure 2.2, it seems that the purchase orders 1, 2, 4, 8, 9, and 10 meet the expectations of

Table 7.1: Transactional data

Purchase Order						
Nr.	Date	Value	Signed	Released	Goods Received	Invoice ...
1	Sep. 1 2016	23 300	☑	☑	☑	101
2	Sep. 1 2016	7 329	☑	☑	☑	102
3	Sep. 1 2016	2 090	☐	☑	☐	103
4	Sep. 1 2016	4 675	☑	☑	☑	104
5	Sep. 1 2016	135	☐	☑	☐	105
6	Sep. 1 2016	4 500	☑	☑	☐	106
7	Sep. 1 2016	7 210	☑	☐	☑	107
8	Sep. 1 2016	69 023	☑	☑	☑	108
9	Sep. 1 2016	12 000	☑	☑	☑	109
10	Sep. 1 2016	40 329	☑	☑	☑	110

(a) An example list of a purchase orders for internal control testing

Invoice			
Nr.	Purchase Order	Date	Value ...
101	1	Sep. 12 2016	23,300
102	2	Sep. 14 2016	7,329
103	3	Sep. 2 2016	2,090
104	4	Sep. 20 2016	4,675
105	5	Sep. 4 2016	135
106	6	Sep. 25 2016	4,500
107	7	Sep. 18 2016	3,605
108	8	Sep. 29 2016	69,023
109	9	Sep. 19 2016	120,052
110	10	Sep. 11 2016	40,329
111		Sep. 15 2016	67

(b) An example list of invoices for internal control testing

the normative model. At least, based on the information that is available in our example, these orders have been signed, released, goods have been received and an accompanying invoice is present. With regard to following the normative model, these orders do not seem to deviate. The remaining purchase orders are listed as deviations. Also, invoice number 111 is listed as deviation since no related purchase order was presented. As mentioned in Chapter 3 we use the terminology of Depaire et al. [2013] and start from the neutral term 'deviation' when a transaction does not match the normative model.

Take into account that if one would not limit this first investigation to the activities that have been executed, and also take the other information into account, purchase order 9 would be listed as a deviation: the value on the accompanying invoice is slightly higher than on the purchase order, possibly including transportation costs. The scope of elements that are taken into account when executing this third step depends on the approach of the auditor: whether the auditor includes all elements at once or splits the investigation into several parts. For reasons of understandability, we limit our example to only checking the process-flow in this step, assuming business rules like 'The value of a purchase order should equal the value of the invoice' are tested separately.

- **Analyze the list of deviations**
  Starting from the listed deviations of the previous step, these deviations have to be classified. In our process, we see three possible outcomes, after a closer inspection of a deviation.

  - **Exception**
    A first possibility is that, although the transaction deviates from the available normative model, this deviation is in fact also a valid process execution. In our example, invoice 111 deviates from the normative model, as it is not linked to any purchase order. However, although it is not ideal from a risk point of view to procure items without an order, it is an acceptable deviation. This might be the case when one orders flowers to thank an employee or some other minor, quick expenses. This deviation is classified as an exception, following the terminology of Depaire et al. [2013]: "*Deviations which are accepted and are used to guarantee the necessary flexibility to react fast and operate effectively, are called exceptions.*" Alles et al. use the term 'tolerable exceptions' [Alles et al., 2006a, p. 157].
  - **Anomaly**
    A second possibility is that the deviation is classified as an anomaly, '*an*

undesirable deviation'. These are deviations for which the auditor, even at closer inspection, cannot formulate a possible explanation for. These deviations are classified as anomalies and will be added to the list of anomalies that warrant further investigation. An example of an anomaly can be a purchase order that has not been released, like purchase order number 7, but goods have been received, along with an invoice (number 107).

  - **Potential compliance issue**
    The third possibility is that, although the deviation cannot be cleared immediately, the auditor can formulate a hypothesis that would clear this deviation. Returning to our normative procurement model and data sample, purchase orders 3 and 5 will be listed as deviation for missing out a signature. The auditor might however recognize the underlying business rule that states that purchases below 1000 do not need to be signed. To reach a final classification on the deviation, the hypothesis 'order value is below 1000$' needs to be tested. Without proof for accepting the hypothesis, the deviation is classified as a potential compliance issue. In case of this third possibility, the process of internal control testing continues. If no deviations are classified in this category, the internal control testing is finished by reaching the state 'deviations classified'. The output is a list of anomalies that warrant further investigation.

- **Specify follow-up investigation with regards to specific internal control**
  Starting from the deviations that reveal a potential compliance issue, a follow-up investigation is specified. The specifics relate to the potential risk and the configured business rule that, if effective, would mitigate this risk. Turning back to our previous example, the follow-up investigation could relate to the risk of missing a signature for high-value purchases. The configured business rule might be 'If purchase value is less than 1000, no signature is requested'. In this case, the follow-up investigation might be specified to test whether all purchases that miss out a signature, indeed have a purchase value of less than 1000$.

- **Collect (subset of) transaction data from information system**
  Based on the specifications of the follow-up investigation, transaction data has to be collected from the information system again. Depending on the data analytics maturity, this might be a new data extraction phase (in case of a sampling-based approach) or a phase of subsetting previously extracted data (in case of full-population testing). In our example, transactions that relate

to a purchase without a signature would be collected, along with the relevant attributes required for the analysis, such as the purchase value.

- **Identify deviations by comparing transactions against specific business rules** Having the follow-up investigation specified and the relevant transaction data available, transactions are again compared with an 'expected pattern'. This time, however, the pattern the transactions are compared with, are in the form of a specific business rule. This is different from the first comparative step, where transactions were compared against a full process. Without turning to specific formats of a process versus a rule, we distinguish between these two artifacts in terms of scope. Where we see 'process' as a holistic concept, a 'rule' is narrowed down to a few specific characteristics, for example, the value of a purchase and the presence/absence of a signature. In our example, the transactions under investigation (the ones without signature, like purchase order 3 and 5) would be tested for having a value that is lower than 1000. If this is not the case (like for purchase order number 3), the transaction ends up on the list of deviations again and a new cycle of 'Analyze the list of deviations' will follow. The three possibilities of 'exception', 'anomaly', and 'potential compliance issues' are revisited. Continuing on our purchase of higher than 1000, but not showing a signature, a new potential compliance issue can be identified, along with a newly formulated hypothesis. For example, 'If the supplier is on the list of 'trusted suppliers', no signature is required'. Then, the procedure continues with testing this hypothesis, and so on.

7.3 Transactional Verification Framework

In Section 7.2, the generic procedure of internal control testing is depicted in an unambiguous way. This allows us to present a framework that employs available data and process mining techniques for transactional verification. Based on the challenges that are reported on previous use cases, and on the principles of continuous auditing in general, we postulate three requirements of the framework:

- transaction verification starts from the business process
- managing the stream of alarms is crucial to make the approach actionable
- a human expert is used as an oracle in order to combine the human and computer intelligence to classify transactions

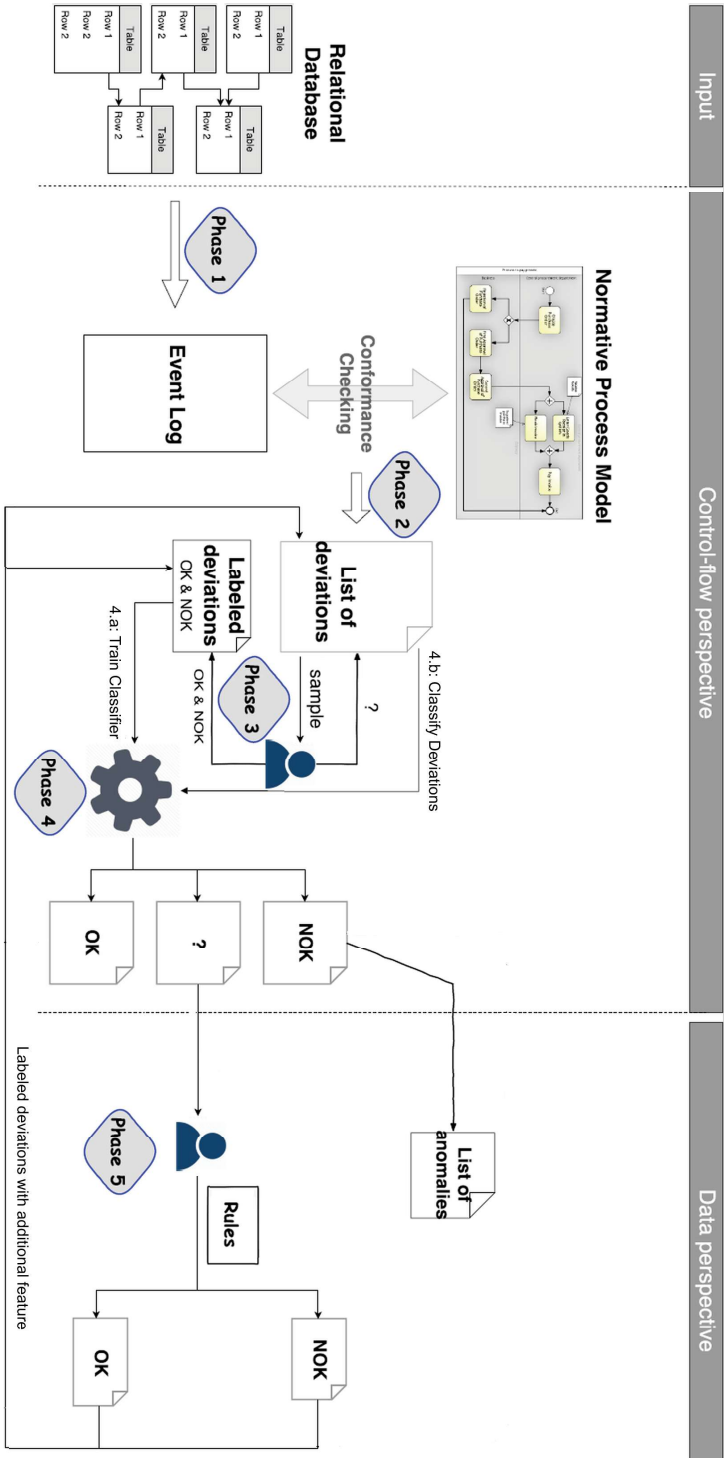


Figure 7.2: Transactional verification framework

Figure 7.2 presents the proposed framework. The framework follows a two-stage approach in terms of the applied perspective. In the first stage, only the control-flow perspective is employed. Transactions are viewed in the context of their process execution, and only the sequence of activities is taken into account. Only in the second stage, when additional data is necessary to classify transactions as an anomaly or not, the data perspective is added. This perspective brings information like the value, performer, etc. into the equation. The first four phases are part of the control-flow perspective. The following last two steps are situated in the data perspective.

Step 1 As a start, transactional data of the process under consideration is extracted from the relevant relational database. Data from this database is subsequently transformed into an event log. This event log will be used as input for a conformance checking phase.

Step 2 A conformance checking algorithm is applied to compare the real transactional data in the event log with the normative process model. The normative process model can be represented in a procedural way, or in a declarative way. The former is a graphical description of *how* process executions should take place (like for example the process in Figure 7.1), the latter is a description of *what* needs to be executed, along with constraints that are typically formulated in a set of business rules. As long as a process execution respects these constraints, the execution is compliant with the model. The framework presumes to start from a procedural process model that strictly postulates the sequence of activities that needs to be respected. This normative model, along with the event log, is used as input for a conformance checking algorithm. The conformance checking technique only takes into account the order of activities (i.e. the control-flow perspective), making abstraction from additional constraints. The output of this step is a list of deviations in terms of process flows.

The deviations which are identified in this step are related to process executions that are different from what is expected in the normative process model. An example can be an invoice that is not preceded by a purchase order. Since such a normative model is designed to be general and illustrating the *ideal* situation, it is not surprising that also deviations with a logical explanation are captured in this list of deviations. Therefore, this list needs to be processed to distinguish anomalies from exceptional deviations.

Step 3 The list of deviations will contain mainly two types of process executions. The first type includes deviations that, despite deviating from the normative model, remain valid and acceptable (OK). For example, a process execution that involves

more approvals than what is initially modeled. The second type consists of executions that under no circumstances can be justified (NOK), such as a process execution that involves a payment without an associated invoice. Of course, there are also executions where more information is needed before a proper classification can be made. Until further assessment is conducted, these executions cannot be labeled. An example of such a case is a booked invoice without a purchase order. This kind of deviation, may sometimes be OK, but sometimes it may turn out unacceptable.

Since the list of deviations is based only on the control-flow perspective, the classification of some cases is inconsistent. In this context, the term 'inconsistent' refers to situations where the same sequence of process executions may be classified as OK in one instance and NOK in another, depending on additional factors and characteristics.

In this step, an initial sample of deviations is selected (e.g., randomly) and presented to an expert for evaluation. The Auditor labels the deviations as either OK or NOK based on their assessment of the control-flow perspective. It is considered an uncertain instance if a deviation cannot be conclusively classified due to insufficient or ambiguous information at that given moment. Such cases are returned to the unlabeled deviation pool. At the end of the third step, a sample of deviations is labeled as OK or NOK by an expert auditor. This initial labeled dataset serves as a starting point for training the classifier in the next phase.

Step 4 This step is divided into two integral parts.

In part 4.a, the classifier undergoes training using the examples within the labeled deviation pool, ensuring that the model learns from a set of deviations already validated by expert judgment. At this stage, only the deviations labeled as OK or NOK are used to train the classifier. This training phase enables the classifier to generalize patterns from the labeled dataset and develop its ability to distinguish between normal and anomalous deviations.

In part 4.b, the trained classifier is then applied to the remaining unlabeled deviations to assign labels and classify deviations autonomously. The classifier generates an output consisting of three 'lists'. The first labeled as the 'OK' list, contains deviations that have been deemed non-problematic and, therefore, removed from further testing procedures. The second, labeled as the 'NOK' list contains deviations identified as anomalies and go directly to the list of anomalies for further investigation in a follow-up stage.

Finally, deviations for which the classifier cannot make a confident decision remain unlabeled (illustrated by '?' in this chapter). These deviations are deferred for further analysis as at this stage, the classifier is unable to classify them.

It is important to emphasize that the role of the classifier is not only to assign

labels (OK, NOK) but also to identify cases where it lacks sufficient confidence. This can be achieved by introducing a confidence threshold: only predictions exceeding this threshold are assigned a label, while the remaining instances are left unlabeled for further analysis.
It is also important to note that, at step 4, classification is still based on the control-flow perspective. That is, only the sequence of activities is considered, without incorporating additional data attributes. The inclusion of further contextual information is deferred to later phases when necessary to resolve uncertainty.

Step 5 The deviations that the classifier could not classify as OK or NOK (i.e., the unlabeled instances) are presented to a human expert, but not all at once. Instead, a subset of the most informative deviations is selected for further analysis. This selection is guided by a query strategy, for example uncertainty sampling, where instances for which the classifier is least confident are prioritized. By focusing on the most uncertain instances, the framework reduces labeling effort while maximizing the learning impact of each expert interaction. This constitutes the active learning component of the framework and aligns with pool-based active learning, where the model selectively queries an oracle for labels on the most informative instances.

The selected deviations are presented to the auditor, who acts as the oracle and provides rules that could classify the deviation. For each deviation, the expert is provided with the relevant contextual information, including the control-flow perspective as well as associated data attributes (e.g., values, resources, or other case and event related information), and provides (i) a label (OK or NOK) and (ii) a rule that explains or justifies the classification.

Consider the example of the invoice without a purchase order. The oracle presents the related rule(s) that could justify this deviation. Think, for example, of the rule that if the supplier is listed as a trusted supplier, no purchase order would be needed.

To integrate this knowledge in the process, this information is stored as a new data feature in the form of a rule. In our example, the information 'Supplier is listed as trusted supplier' is encoded as a binary attribute (0/1). This new data feature will be added to both the labeled and unlabeled sets of deviations from the previous steps, and its value is computed for all instances. The deviations evaluated by the expert are assigned a label (OK or NOK) and incorporated into the labeled dataset.

The newly labeled deviations are moved from the unlabeled set to the labeled deviation pool. As a result, two updates occur: (i) the labeled dataset is expanded with newly labeled instances, and (ii) both labeled and unlabeled datasets are enriched with the additional feature in the form of rules that have been precisely defined by the oracle.

After Step 5 has been executed, the procedure retakes Step 4a where the classifier is retrained using the updated labeled dataset. By incorporating newly labeled examples, enriched with additional data perspective features, the classifier continuously refines its learning process and decision boundaries leading to reduction of the number of uncertain instances and enhancing its accuracy and performance.

These newly labeled instances are further used as additional input for the classifier, which now receives more human-based knowledge to learn from and in the end will produce fewer and fewer deviations in the unlabeled ('uncertain') category in Step 4. It is important to note that the restriction to a purely control-flow perspective applies only to the classifier up to this stage. While the expert may rely on additional contextual information when labeling deviations, the classifier incorporates such information only at this stage, thereby transitioning from a control-flow to a data perspective in the learning process. Additional contextual information is incorporated only when required to resolve uncertainty. It helps to ensure that the complexity of the analysis remains controlled.

One could ponder on the option to immediately capture all rules from the human expert and program these. We postulate that it is difficult, if even feasible, for the expert to provide an exhaustive set of rules with 100% confidence. By confronting the expert with deviations in the activity sequence, the tacit knowledge on which rule is activated in which activity sequence is extracted and made explicit. The more a rule is presented in different situations, the higher the confidence level of that rule. In this way, the expertise of the human expert can be captured.

It is assumed that previously assigned labels remain valid after the introduction of new features.

7.4 Continuous Transactional Verification

Revisiting the framework as described in Section 7.3, continuous auditing opportunities and the link to process and data mining research can be made in several steps. The argument we make is that by having a streamlined process of an audit procedure, and knowing the specifics of how to optimize and automate this process, continuous auditing can be reached. The process of internal control testing was outlined before. In this section, we give an overview of outstanding research questions that bring auditing closer to continuous auditing. Given the close ties to the field of Business Process Management, some related research questions in this area are also presented. We structure the research opportunities around the different phases that were discussed before.

7.4.1 Phase 1 – Building the event log

In the context of testing internal controls over a process that affects financial reporting, a continuous auditing environment collects transaction data of the full process. This is a wider collection than the financial transactions, the journal entries, alone. As show-cased in Jans et al. [2014], all timestamped process activities leading to a financial transaction can be taken into account in a process mining investigation.

The main challenge is situated in preparing the data for a process mining analysis: building the event log. As described by González López de Murillas et al. [2015] and González López de Murillas et al. [2016], turning data from a relational database (like an ERP system is using) into an event log, is not straightforward. During event log building, decisions need to be taken on the view on the process. Opting for one specific view might rule out another view. These decisions, therefore, have a crucial impact on the type of analyses that are possible afterward. Current research only investigates this topic from a technical viewpoint, like the work of Jans [2019], Calvanese et al. [2015] and Lu et al. [2016]. However, the impact of these decisions on the analytical procedures needs to be investigated more thoroughly.

7.4.2 Phase 2 – Identify deviations by comparing the event log with the normative process model

This phase focuses on the construction of the normative process model and its role in identifying deviations.

Creating the normative process model of the process under investigation

In this framework, the decision on which process to examine thoroughly is taken before the transaction verification starts. In a future continuous auditing environment, where controls monitoring and detailed testing occurs simultaneously [Chan and Vasarhelyi, 2011a], the auditor gets an overview of how the key business processes are running. Ideally, this overview might be in a process model for each process, easily and quickly to understand. Based on this overview, resources can be allocated to the process with the highest risk assessment. In order to get this overview, process discovery algorithms would be suited. To date, however, depending on which algorithm is used, inputting the same data set will result in different process models [Jans and Laghmouch, 2022]. As explained before, this variation is linked to the underlying emphases these algorithms place, such as higher fitness, higher precision, etc). Future research could examine which process discovery technique's underlying assumptions are -most- compatible with the risk assessment task. Or perhaps new algorithms, dedicated to this task, need to be developed (see for example Pika et al. [2016]).

Once a process is selected, a normative process model must be created to check against logged behavior. When creating such a normative process model, there must be an understanding of the process first. Understanding the normative process can have multiple types of input and output. Possible inputs are (tacit or explicit) experiences of the auditor, oral or written process narratives, graphical process models, or a list of business rules. This will depend on the maturity of the organization. In a continuous auditing setting, the output of this activity is a formal process representation. This can be either in the form of a procedural model or in the form of a declarative model. This framework presented to formalize this process in a procedural way.

Artificial intelligence can play an important role to reach a formal process representation at the end of this activity. Natural Language Processing techniques that are capable of interpreting unstructured data that describe a normative process and pour it into a process representation can be further investigated. Some work on this narrative-model translation and on the alignment is already been done by Friedrich et al. [2011] and van der Aa et al. [2015] for example. Related work has explored the automatic generation of business process models from natural language descriptions by extracting activities and their relationships using linguistic analysis techniques [Honkisz et al., 2018]. In a full continuous auditing environment, this formal process representation is present at the start of the auditing procedure. The task 'Understand the normative process' is then fully in line with the key concern of the auditor: assessing the risks and the installed controls of the process under investigation.

Identify deviations by comparing transactions against the normative process model

Depending on the format of the normative model (procedural versus rules), different process mining techniques are available, as described in the previous chapter. In order for these techniques to be applicable in a continuous auditing environment, the required input and output specifications of these techniques need to be adapted to the auditing environment. In other words: if research proves that the normative process -for the goal of continuous auditing- is best formalized in a BPMN process model (as the process model in Figure 2.2 is illustrated in BPM notation), the conformance checking technique should be able to compare an event log with BPMN models. Second, how is a 'deviation' defined? Is it, for example, defined at the level of a complete process execution ('this process execution deviates from the model') or at the level of an activity ('this activity, Goods Receipt, should not take place in this sequence, according to the model')?

In process mining field, 'deviance mining' is described in Nguyen et al. [2014].

Chapter 4 investigated what type of deviations are 'typical' for an auditing setting. This shows if the 'list of deviations' could be compressed to a 'list of deviation types', this might be the first step in making full-population testing feasible. For example, instead of having a list of 10,000 deviations, the output could be a list of certain types of deviations (e.g., 'activity B is missing').

7.4.3 Phase 3 – Labeling sample deviations as OK or NOK

A key component of the proposed framework is the involvement of a human expert in the labeling process. In this phase, a sample of deviations is presented to the auditor, who classifies each deviation as either OK (acceptable) or NOK (anomalous). This assessment is based on the auditor's domain knowledge, understanding of business processes, and awareness of internal control objectives. The resulting labeled instances constitute the initial knowledge base that will be used to train the classification model in subsequent phases.

This step is essential, as the quality and representativeness of the labeled data directly influence the performance of the classifier. Inaccurate or inconsistent labels may lead to biased or unreliable model behavior in later stages. At the same time, labeling deviations can be time-consuming and cognitively demanding, especially when large volumes of process instances are involved or when deviations are complex and require detailed inspection.

This phase corresponds to the initial labeling stage required in supervised learning, where a first set of labeled instances is constructed before any model-driven selection of instances can take place. While labeling itself may appear straightforward, it plays a critical role in capturing expert knowledge and establishing the foundation upon which the subsequent learning process is built.

7.4.4 Phase 4 – Classification under uncertainty

In this phase, the objective is to train a classification model that assigns deviations to the classes OK or NOK based on the labeled data obtained in the previous phase. The classifier learns patterns from the labeled dataset and generalizes them to previously unseen deviations.

However, not all instances can be classified with sufficient confidence. Therefore, the classification model must be capable of distinguishing between cases where it can make a reliable prediction and cases where the classifier cannot support a confident decision given the available information and learned patterns. Instead of enforcing a classification, deviations for which the model does not reach a predefined confidence threshold (i.e., their confidence score falls below a predefined threshold) are left unlabeled. These deviations are deferred to the next phase of the framework. This

behavior can be implemented using a wide range of classification techniques. In general, any classifier that produces class probabilities or decision scores can be adapted to support this mechanism. For example, by introducing a confidence threshold, only instances for which the predicted probability (or score) exceeds this threshold are assigned a label (OK or NOK). Instances falling below this threshold are not assigned a label and are considered for further analysis.

This approach is not tied to a specific modeling technique. It can be applied to various classifiers, including decision trees, naive Bayes models, and support vector machines. In this context, Rough Set Theory (RST) and related approaches, such as Rough Cognitive Networks [Nápoles et al., 2016, Nápoles et al., 2017, 2018], offer an alternative way to identify regions of uncertainty through their approximation mechanisms.

In this framework, instances that cannot be decisively assigned to a class fall into a boundary region, which naturally corresponds to situations requiring further analysis. These RST-based methods should be viewed as one possible implementation among several alternatives, rather than as a technique or requirement of the framework. A key limitation in this context is that many commonly used classification techniques (including Rough Set Theory, decision trees, and support vector machines) operate on a declarative, attribute-based representation of data. As a result, they do not inherently capture procedural or sequential information. This is particularly relevant for process-oriented tasks such as transactional verification in our framework where actual process executions are represented as sequences of activities in event logs. In such settings, a classifier must be able to understand the order and structure of events. This procedural dimension cannot be directly captured by Rough Set Theory alone or using purely tabular data. One approach to address this limitation is to transform sequential information into engineered features encoding sequential information (e.g., binary indicators such as "activity A precedes activity B"). However, this transformation may lead to a rapid increase in the dimensionality of the feature space.

An alternative is to employ classifiers that are inherently designed to handle sequential data. Architectures such as Long Short-Term Memory (LSTM) networks [Hochreiter and Schmidhuber, 1997] or Recurrence-Aware Long-Term Cognitive Networks [Nápoles et al., 2022] are well-suited to learning complex temporal dependencies and patterns directly from sequences of events. These models can also be configured to output class probabilities, allowing the same confidence-based mechanism to be applied in order to determine whether a prediction is sufficiently reliable.

Further investigation is required to determine the most appropriate modeling approach in this context. While traditional classifiers offer interpretability and simplic-

ity, sequence-based models provide greater expressive power for capturing temporal patterns. For example, Rough Set Theory offers an interpretable, rule-based mechanism for handling uncertainty, but it lacks the expressive power required for temporal abstraction. Conversely, models such as LSTMs excel at handling sequential data but may compromise explainability, which is an essential requirement in auditing. Therefore, a hybrid approach that balances interpretability, the ability to model sequential behavior, and robust handling of uncertainty remains a promising direction for future research.

7.4.5 Phase 5 – Oracle presents rules that deviating transactions can be checked against

In this phase, not all unlabeled deviations are presented to the auditor at once. Instead, a subset of the most informative deviations is selected for further analysis. This selection is guided by a query strategy, such as uncertainty sampling, where instances for which the classifier is least confident are prioritized. By focusing on the most uncertain and informative cases, the framework reduces labeling effort while maximizing the impact of each expert interaction. This constitutes the active learning component of the framework.

The auditor (the oracle) feeds the framework with their expertise by formulating hypotheses that might explain the observed deviating behavior. These hypotheses are typically expressed in the form of a set of rules. Comparing real transactions against these specific rules can be seen as a particular form of conformance checking. In contrast to the conformance check in Step 2, this step is limited to testing deviating transactions against rules, and not against a process model.

This drives this step to the area of algorithms such as the *LTL Checker* and related work [van der Aalst et al., 2005]. As mentioned before, techniques that compare real behavior with rules, are capable of taking more characteristics into account than only the order of activities (such as contextual and attribute-based information).

Further research could investigate which types of characteristics are most relevant for auditors to formulate and test rules and assess risk, and whether these characteristics can be captured by current techniques effectively. Additionally, the question of which languages for representing rules are most suitable for enabling auditors to express their knowledge in a formal way remains an open research direction.

7.5 Limitations

Like every study, also this study holds some limitations. Although an extensive literature review has been conducted, the selection of concepts and level of detail was mostly guided by the presented framework. We attempted to construct a generic process for all internal control testing procedures, regardless of the level of maturity on continuous auditing to build our framework. Although in Chapter 5 some parts of this framework are considered and applied on real data of the case study, future research could involve a pilot implementation of the presented framework as a validation. Also to estimate to what extent this framework can reduce the time and effort of the experts.

7.6 Concluding Notes

The application of data mining and process mining on continuous auditing has been investigated in different studies. The main drawback of these applications, as reported, is the existence of a large number of false positives. This keeps continuous auditing from full adoption in practice. The framework presented in this chapter aims to increase the degree of automation in the first level of continuous auditing, namely transaction verification, while preserving auditor involvement for unresolved cases. The basic continuous auditing principle to start from a process point of view is respected, hence a process mining approach is applied as a start. This is subsequently complemented by a data mining approach. The suggested data mining approach combines supervised classification with confidence-based filtering, allowing uncertain cases to remain unlabeled and be investigated further by expert analysis.

The transaction verification framework presented in this chapter has the following characteristics:

- incorporates the process view in a concrete fashion, relating to existing process mining techniques and how these could be employed
- combines process mining and data mining techniques to truly enable full-population testing, including managing the alarm flood
- incorporates classification techniques capable of operating under uncertainty, while allowing uncertain cases to be deferred for further expert assessment

- leverages the human expertise of the auditor to increase the efficiency of the classification algorithm

The possibility of deferring uncertain cases, along with the combination of the machine learning and the human expert feed, ultimately leads to a combined machine-human intelligence (aligning with recent developments in accounting information systems research [Sewpersadh, 2025]), capable of dealing with full-population transaction data.

Chapter 8

Conclusion

This thesis addressed the challenge of effectively integrating process mining techniques into continuous auditing in the presence of increasing data volume, process complexity, and the limitations of existing analytical approaches. While conformance checking has been widely recognized as a powerful technique for detecting deviations in process executions, its practical adoption in auditing remains limited due to key challenges, including the generation of large volumes of non-meaningful outputs, alarm floods, and high false-positive rates, and the resulting information overload for auditors.

To address these limitations, this research focused on the problem of deviation analysis and classification, positioning it as a critical step between deviation detection and audit decision-making. In this thesis, deviations are not viewed as mere technical outputs, but are systematically analyzed in accordance with auditors' reasoning processes.

The main contribution of this thesis is the design of a structured, human-in-the-loop framework for process deviation classification that combines process mining, data mining, and active learning. This framework transforms low-level conformance checking outputs into meaningful, interpretable, and actionable audit insights, thereby supporting scalable and effective continuous auditing.

The thesis began by introducing the challenges of continuous auditing in the context of increasing data availability and process complexity, highlighting the limitations of traditional and data-driven approaches. It then positioned process mining, and in particular conformance checking, as a promising technique for detecting deviations in business process executions. Building on this foundation, the research developed methods to overcome key limitations of existing conformance checking approaches, particularly regarding interpretability, false positives, and the effective use of deviations in auditing.

8.1 Key Contributions

This thesis makes several contributions to the field of continuous auditing and process mining. **Conceptualization of Deviation Analysis for Auditing.** This research reframes deviation analysis as a multi-stage classification problem, bridging the gap between technical conformance checking outputs and interpretations meaningful for auditing practice. By explicitly modeling deviation classification as an iterative and knowledge-driven process, the research aligns analytical techniques with auditors' cognitive processes.

Deviation Categorization:

A framework introduced in Chapter 4 compresses the outputs of conformance checking into a fixed set of meaningful and interpretable deviation categories. This abstraction reduces complexity and addresses the issue of non-meaningful outputs by aligning the categories with auditors' mental models. It enables auditors to gain a high-level overview of deviations, supporting more efficient decision-making.

Incremental Classification Approach for False-Positive Reduction.

The thesis proposes an iterative classification procedure in Chapter 5 to reduce false-positive deviations, a common challenge in auditing that increases manual effort and undermines efficiency. The framework analyzes deviations at increasing levels of granularity, starting from basic deviation types and incrementally incorporating sequence patterns and contextual attributes. This design minimizes information overload and enables the reuse of knowledge across iterations, significantly reducing the burden on auditors while maintaining the integrity of the auditing process. While the primary focus is on the control-flow perspective, the approach is extended to include the data perspective, enabling more comprehensive analysis of deviations. This integration ensures that auditors can classify deviations with greater accuracy, addressing cases where control-flow information alone is insufficient.

Active Learning Framework for Alarm Flood Management.

The thesis introduces a human-in-the-loop active learning framework in Chapter 7 that combines process mining, data mining, and active learning mechanisms to address the challenge of alarm floods in continuous auditing. By incorporating auditor judgment and allowing uncertain cases to be deferred and iteratively refined, the framework balances automation with expert judgment and addresses the problem of alarm floods in continuous auditing.

8.2 Reflections on Research Questions

This thesis addressed the research questions introduced in Chapter 1, focusing on the critical challenges and opportunities associated with integrating process mining techniques into auditing practices. These research questions were designed to bridge gaps between theoretical advancements and practical application. The proposed approaches provide structured and interpretable mechanisms that align analytical outputs with auditors' needs and decision-making processes.

The research questions are addressed below using the numbering introduced in Chapter 1. Although the thesis chapters develop these contributions in a different logical order, the reflection retains the original numbering to preserve consistency with the introduction.

RQ1: How can a framework integrating process mining, data mining, and human-in-the-loop enhancement be designed to assist auditors in a continuous auditing practice?

Addressal:

This question is addressed through the design of a structured human-in-the-loop framework presented in Chapter 7. The framework integrates conformance checking with data mining and active learning mechanisms to address the challenges associated with alarm floods in continuous auditing and support continuous auditing over full-population data. By incorporating professional judgment into the learning process and allowing uncertain cases to be deferred, the framework balances automation with expert judgment. This design enables more efficient handling of large volumes of deviations and supports adaptive, scalable auditing practices.

This approach not only enhances the accuracy of the auditing process but also improves its efficiency. Auditors can focus on the most critical issues without being overwhelmed by irrelevant or repetitive alarm notifications. Combining machine learning and auditor input creates a more intelligent system that continually improves with use, ultimately streamlining the auditing process and reducing the manual effort required.

By addressing alarm floods through this framework, continuous auditing systems become more reliable, efficient, and aligned with the practical needs of auditors, contributing to the broader goal of automating auditing tasks while preserving the essential human oversight.

RQ2: How can the outputs of conformance checking techniques be made more interpretable and meaningful for auditors?

Addressal:
This question is addressed through the deviation categorization method introduced in Chapter 4. The approach transforms low-level conformance checking outputs into a fixed set of meaningful and interpretable deviation categories aligned with auditors' mental models. The objective was to design and develop the process deviation categories based on both theoretical insights and practical requirements, bridging the gap between technical outputs produced by process mining techniques and the practical needs of auditing professionals. This abstraction reduces complexity, improves usability, and enables auditors to quickly understand and prioritize deviations without being overwhelmed by technical details.

Interviews were conducted with experienced auditors and domain experts to validate and refine the proposed deviation categories. These interviews provided critical insights into how auditors perceive and evaluate deviations in real-world auditing contexts. Auditors often require outputs that are both accurate and intuitive, allowing them to quickly identify and prioritize significant anomalies without being overwhelmed by technical jargon or excessive details. The key findings from the interviews revealed:

- Auditors prefer high-level summaries of deviations that allow for an immediate understanding of the overall process health.
- Detailed categorization should remain accessible but should not detract from the main focus on key anomalies.

Based on these findings, the proposed deviation categories were refined to ensure they are both comprehensive and intuitive, enabling auditors to drill down into specific anomalies when necessary. The categorization approach presented in Chapter 4 improves interpretability to serve as a practical tool for auditors to better manage their workloads and prioritize significant cases. By focusing on aligning outputs with auditors' professional judgment and expertise, this research provides a pathway for improving the usability of conformance checking techniques in real-world auditing scenarios.

RQ3: To what extent can incremental information provision assist auditors in classifying deviating instances more efficiently than traditional manual approaches?
Addressal:
This question is addressed through the incremental classification approach presented in Chapter 5. The proposed method structures the analysis of deviations across multiple levels of granularity, starting from simple deviation types and progressively

incorporating sequence patterns and contextual attributes. This design reduces the cognitive and operational burden on auditors and enables knowledge reuse across iterations, significantly improving classification efficiency. The integration with active learning further enhances this process by prioritizing relevant cases and reducing unnecessary manual effort.

Traditional manual approaches to auditing often require auditors to process large volumes of data simultaneously, which can be both time-consuming and error-prone. By contrast, the incremental framework structures the presentation of deviations in a hierarchical manner, starting with high-level categorizations, such as whether a deviation falls under the categories of "missing," "reordering," or "insertion." Additional information, such as specific data attributes or resource details, is made available as auditors explore deviations further.

Chapter 7 further demonstrated the application of active learning mechanisms to support this incremental approach. Active learning leverages machine learning models that are iteratively trained using feedback from auditors. This integration ensures that the system learns from auditor decisions and prioritizes deviations that are most likely significant in subsequent analyses. By combining active learning with incremental information provision, auditors can classify deviations with greater speed and accuracy while minimizing unnecessary manual effort.

The results of this approach showed that auditors can achieve improved classification efficiency without compromising the quality of their analysis. The system enables a more focused investigation of deviations by reducing false positives and providing contextual information only when necessary. This incremental method not only streamlines the auditing process but also supports auditors in making more informed decisions based on relevant, targeted data.

Overall, the results demonstrate that combining structured abstraction, incremental analysis, and human-in-the-loop learning provides an effective foundation for making process mining techniques applicable and scalable in continuous auditing.

8.3 Limitations and Future Research

While this thesis provides insights and contributions to the integration of process mining and continuous auditing, several limitations should be acknowledged, and opportunities for future research exist. These limitations primarily stem from the scope of the study, the specific techniques applied, the methodological choices, and practical implementation considerations. Addressing these limitations can open avenues for further exploration and refinement of the approaches proposed in this thesis.

1. Focus on Conformance Checking and Control-Flow Perspective

One limitation of this research is its primary focus on conformance checking techniques and the control-flow perspective of process mining. Although these techniques provide significant insights into process deviations, they may not fully capture all real-world business processes' complexities. Other perspectives, such as the resource and time perspectives, could offer additional layers of analysis that might help auditors make more comprehensive evaluations. Future research could expand the scope to include these perspectives from the beginning, providing a more holistic view of business processes and enhancing the applicability of conformance checking techniques in diverse auditing scenarios.

2. Scalability of Active Learning Framework

The active learning framework introduced in Chapter 7, which integrates process mining, data mining, and auditor input, offers a significant advancement in managing alarm floods and improving auditing efficiency. However, this framework relies on auditor input to refine machine learning models, which could present scalability challenges in large-scale auditing environments. Manual involvement may become impractical for organizations with vast amounts of data or complex processes. Future research could explore the automation of certain aspects of this framework, reducing the dependency on human input while maintaining the quality and relevance of exception classification.

3. Limited Real-World Validation

Although the methods and frameworks proposed in this thesis were validated through theoretical analysis and case study examples, they have not been tested in diverse, real-world auditing environments. The effectiveness and practicality of these methods could vary when applied across different industries or organizational contexts. Future research could focus on implementing and testing the proposed solutions in real-world settings, allowing for a deeper understanding of their applicability, limitations, and potential for adaptation in various auditing practices.

4. Focus on Deviation Classification

This research primarily focused on improving the classification of process devia-

tions, particularly through the refinement of conformance checking techniques and the categorization of deviations. While this focus addresses critical challenges in auditing, there are other important aspects of the auditing process that were not fully explored, such as risk assessment, fraud detection, and compliance verification. Future research could investigate how the proposed methods could be integrated into broader auditing workflows, supporting auditors in more comprehensive tasks beyond deviation classification.

5. Auditing Professional Development and Training

As noted throughout the thesis, the successful application of process mining and data analytics in auditing relies heavily on auditors' ability to interpret technical outputs and incorporate them into their decision-making processes. This emphasizes the need for professional development and training programs that equip auditors with the skills and knowledge necessary to use these advanced techniques effectively. Future research could explore the design of specialized training curricula that help auditors understand and apply process mining, data mining, and active learning tools in their daily work.

Addressing these limitations and exploring the proposed avenues for future research can lead to significant advancements in continuous auditing practices, ultimately enhancing auditing efficiency, accuracy, and adaptability.

Chapter 9

Thesis Appendix

Exemplary quotation including the information terms	First-order code	Concept (frequency)
(Interviewee 11) When there are payments without an invoice for a service, that's typically a risk.	Pay without an IR	Without (41)
(Interviewee 6) I try to see how many purchase orders without purchase requisition are out there.	Purchase orders without PR	
(Interviewee 10) Demanding your payment without having an invoice, without having the goods receipt, without having the purchase order, without having the purchase request, it's a whole problem.	Pay without having an IR Pay without having the GR Pay without having the PO Pay without having the PR	
(Interviewee 4) If the order is there, but not approved, well then, I think it is mainly question of fraud.	Not approved	Not/ never+ verb (20)
(Interviewee 7) What we often see, is for example that the payment has occurred, but the invoice has not been approved yet.	Not been approved	
(Interviewee 9) The invoice from the vendor does not come with the Purchase order number, that means that we don't pay the invoice, and get stuck in other system.	Not pay the invoice	
(Interviewee 9) Even what we did just mentioned, financial invoices, also go to SAP, they just miss a bit of the steps.	Miss a bit of the steps	Miss (14)
(Interviewee 12) We get exception lists, and we say that are the missing invoice.	Missing invoice	

Exemplary quotation including the information terms	First-order code	Concept (frequency)
(Interviewee 7) Higher risk basically if there are no service and goods receipt, and paid for example, in case there is no approval and release of the purchase order, basically these are actually the risk that we list here.	No SR and GR No approval and release of the PO	No X (13)
(Interviewee 4) If you have for example an invoice but no purchase order, then first of all, okay is that they really order it?	No purchase order	
(Interviewee 3) It can be no payment because the invoice [is] not yet due of course.	No payment	
(Interviewee 5) I think already the first step and the second step, the creation of purchase request and approval of purchase request is usually, at least in my experience, something that the clients don't do.	Not doing the CreatePR and ApprovePR	Not + verbs like happen and execute (13)
(Interviewee 5) The signature of the purchase order, if it is not included as an automatic check in the system, it does not happen for 100 percent of the cases.	SignPO does not happen	
(Interviewee 12) The suppliers had sent the invoice, for loose invoice or something for which there is never been a delivery, that has to be detected.	Never been a delivery	

Exemplary quotation including the information terms	First-order code	Concept (frequency)
(Interviewee 8) For example, if you are the boss and you order things directly.	Order things directly	Straight, Directly or Immediately (5)
(Interviewee 7) Go straight from purchase order to payment, for example which should not be the case.	Straight from purchase order to payment	
(Interviewee 7) They can directly create purchase order, it gives approve or not and then, these two (create PR and Approve PR) are not done.	Directly create purchase order	
(Interviewee 8) You have ideal authorization but you skip a piece of process.	Skip a piece of process	Skip (5)
(Interviewee 8) If you do this, you are skipping this step	Skipping this step	
(Interviewee 7) We select a few samples, we see it in X occurrences, it has been approved after Release, and X occurrence has even been approved after invoice.	Approved after Release Approved after invoice	After (29)
(Interviewee 11) You can have companies where the invoice is become after payment.	Invoice after payment	
(Interviewee 3) The approval is only after the payment, that's also something that we see	Approval after the payment	

Exemplary quotation including the information terms	First-order code	Concept (frequency)
(Interviewee 6) The payment to go out before you receive the goods or if the payment go out before the purchase order is approved, this is the most highest risk.	Payment before goods Payment before the ApprovePO	Before (25)
(Interviewee 4) What is important, is whenever you have for example, an invoice which has been booked before the purchase order has been booked, so normally you would expect first the purchase order to be entered. If you have the invoice before the order. Sometimes that happens.	Invoice before the order	
(Interviewee 2) We see all deviations like some purchase order is created before Purchase request is approved.	PO is created before PR	
(Interviewee 4) If you have an invoice, payment, but no Goods receipt for example, or you have Good Receipt afterwards, there is quite worrisome.	Good Receipt afterwards	Afterwards (8)
(Interviewee 11) If there is Good Receipt, there is a payment needed, and if afterwards the order is created, for me is not that a risk.	Afterwards the order is created	

Exemplary quotation including the information terms	First-order code	Concept (frequency)
(Interviewee 4) If you have an advanced invoice, if you have an invoice, based on the contract with the person paints these walls, if he says okay, the total cost is let say 10,000 euros, but I want an advanced invoice of 2000 euros and the other 8,000 euros is at the end.	Advanced invoice	Advanced (7)
(Interviewee 1) Advance payment can also be tricky for our interest.	Advance payment	
(Interviewee 3) Most of the cases now you pay one amount twice, one to a fake supplier and one to real supplier and the money is gone.	Pay one amount twice	Twice (17)
(Interviewee 3) A lot of clients are just arguing we don't have the control up front, because we would identify it when we are paying things twice we will identify it in the accounting and so on.	Paying things twice	
(Interviewee 9) We always have this, not always a one to one, for example, you can have one PO and 20, or thousand invoices on it. We can have big PO. So, it's not always a one to one.	20, or thousand invoices	Two-three-four (13)

Exemplary quotation including the information terms	First-order code	Concept (frequency)
(Interviewee 11) If you receive three times goods, it depends. If it is a problem in your financial, it depends on what is delivered of course … But it could also be really really specific that you don't need the three times the quantity.	Receive three times goods	N-times (12)
(Interviewee 3) When the same delivery you seen in the warehouse and you just put it in the system, Three times instead of one time. Then, you would have really impact on the financial statement.	Three times	
(Interviewee 10) What I would like to see is a check to make sure that I don't get duplication of ordering.	Duplication of ordering	Duplication (11)
(Interviewee 10) I've had a case, where we had a duplicate payment by accident and because there was a very small bug in the ERP system, which paid against the goods receipt, but also paid against the invoice receipt.	A duplicate payment	
(Interviewee 10) This type of things, we do: [...] duplicate invoices.	Duplicate invoices	
(Interviewee 8) I don't think that's an exception, but you could have one order here, and multiple good receipts there	Multiple GRs	Multiple/several (5)

Table 9.1: Deviation concepts in first-order analysis and code mapping

Bibliography

Mohammad Javad Abdolmohammadi and Ahmad Sharbatouglie. *Continuous auditing: An operational model for internal auditors.* Institute of Internal Auditors Research Foundation Institute, 2005.

Arya Adriansyah, Boudewijn F. van Dongen, and Nicola Zannone. Controlling break-the-glass through alignment. *Social Computing (SocialCom), IEEE International Conference on Privacy, Security, Risk and Trust*, 2010. doi: 10.1109/SocialCom.2013.91.

Arya Adriansyah, Boudewijn F. van Dongen, and Wil M.P. van der Aalst. Conformance checking using cost-based fitness analysis. In *Enterprise Distributed Object Computing Conference (EDOC), 2011 15th IEEE International*, pages 55–64. IEEE Computer Society, 2011.

Arya Adriansyah, Jorge Muñoz-Gama, Josep Carmona, Boudewijn F. van Dongen, and Wil M.P. van der Aalst. Alignment based precision checking. In *Business Process Management Workshops*, volume 132 of *Lecture Notes in Business Information Processing*, pages 137–149. Springer, 2012. ISBN 978-3-642-36284-2.

AICPA. Rutgers and aicpa unveil data analytics research initiative, December 16, 2015 2015.

Michael Alles, Gerard Brennan, Alexander Kogan, and Miklos A. Vasarhelyi. Continuous monitoring of business process controls: A pilot implementation of a continuous auditing system at siemens. *International Journal of Accounting Information Systems*, 7(2):137–161, 2006a.

Michael Alles, Gerard Brennan, Alexander Kogan, and Miklos A. Vasarhelyi. Continuous monitoring of business process controls: A pilot implementation of a continuous auditing system at siemens. *International Journal of Accounting Information Systems*, 7(2):137–161, 2006b.

Michael G Alles, Fernando Tostes, Miklos A. Vasarhelyi, and Edson Luiz Riccio. Continuous auditing: the usa experience and considerations for its implementation in brazil. *JISTEM-Journal of Information Systems and Technology Management*, 3(2):211–224, 2006c.

Michael G. Alles, Alexander Kogan, and Miklos A. Vasarhelyi. Putting continuous auditing theory into practice: Lessons from two pilot implementations. *Journal of Information Systems*, 22(2):195–214, 2008a.

Michael G Alles, Alexander Kogan, and Miklos A. Vasarhelyi. Putting continuous auditing theory into practice: Lessons from two pilot implementations. *Journal of Information Systems*, 22(2):195–214, 2008b.

Adriano Augusto, Raffaele Conforti, Marlon Dumas, Marcello La Rosa, and Artem Polyvyanyy. Split miner: automated discovery of accurate and simple business process models from event logs. *Knowledge and Information Systems*, 59(2):251–284, 2019.

Peyman Badakhshan, Bastian Wurm, Thomas Grisold, Jerome Geyer-Klingeberg, Jan Mendling, and Jan Vom Brocke. Creating business value with process mining. *The Journal of Strategic Information Systems*, 31(4):101745, 2022.

Cátia Barros and Rui Pedro Marques. Continuous assurance for the digital transformation of internal auditing. *Journal of Information Systems Engineering and Management*, 7(1), 2022.

Tim Bäßler and Marc Eulerich. Predictive process monitoring for internal audit: Forecasting payment punctuality from the perspective of the three lines model. *Available at SSRN 4080238*, 2022.

Egon Berghout, Rob Fijneman, Lennard Hendriks, Mona de Boer, and Bert-Jan Butijn. *Advanced Digital auditing: Theory and practice of auditing complex information systems and technologies*. Springer Nature, 2023.

Stuart Black and Gregory J. Gerard. Accelerating the future of audit technologies: Introducing the special issue and emphasizing future research directions. *International Journal of Accounting Information Systems*, 56:100740, 2025. doi: 10.1016/j.accinf.2025.100740.

Hennie Boeije. A purposeful approach to the constant comparative method in the analysis of qualitative interviews. *Quality and quantity*, 36(4):391–409, 2002.

Helen Brown-Liburd, Hussein Issa, and Danielle Lombardi. Behavioral implications of big data's impact on audit judgment and decision making and future research directions. *Accounting Horizons*, 29(2):451–468, 2015. ISSN 08887993. doi: 10.2308/acch-51023.

Alessandro Burigana, Alessandro Gianola, Marco Montali, and Sarah Winkler. Glocal conformance checking. In *International Conference on Business Process Management*, pages 75–92. Springer, 2024.

Diego Calvanese, Marco Montali, Alifah Syamsiyah, and Wil M. P. van der Aalst. Ontology-Driven Extraction of Event Logs from Relational Databases. In *Business Process Management Workshops*, Lecture Notes in Business Information Processing, pages 140–153. Springer, Cham, August 2015.

Filip Caron, Jan Vanthienen, and Bart Baesens. Comprehensive rule-based compliance checking and risk management with process mining. *Decision Support Systems*, 54(3):1357–1369, 2013.

David Y. Chan and Miklos A. Vasarhelyi. Innovation and practice of continuous auditing. *International Journal of Accounting Information Systems*, 12(2):152–160, 2011a.

David Y. Chan and Miklos A. Vasarhelyi. Innovation and practice of continuous auditing. *International Journal of Accounting Information Systems*, 12(2):152–160, 2011b.

Tiffany Chiu and Mieke Jans. Process mining of event logs: A case study evaluating internal control effectiveness. *Accounting Horizons*, 33(3):141–156, 2019.

Victoria Chiu, Qi Liu, and Miklos A Vasarhelyi. The development and intellectual structure of continuous auditing research. *Journal of Accounting Literature*, 33 (1-2):37–57, 2014.

David A Cohn, Zoubin Ghahramani, and Michael I Jordan. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145, 1996.

Raffaele Conforti, Marcello La Rosa, and Arthur HM ter Hofstede. Filtering out infrequent behavior from business process event logs. *IEEE Transactions on Knowledge and Data Engineering*, 29(2):300–314, 2016.

Bruce I Davidson, Naman K Desai, and Gregory J Gerard. The effect of continuous auditing on the relationship between internal audit sourcing and the external auditor's reliance on the internal audit function. *Journal of Information Systems*, 27 (1):41–59, 2013.

Massimiliano de Leoni and Wil M. P. van der Aalst. *Aligning Event Logs and Process Models for Multi-perspective Conformance Checking: An Approach Based on Integer Linear Programming*, pages 113–129. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013a. ISBN 978-3-642-40176-3. doi: 10.1007/978-3-642-40176-3_10.

Massimiliano de Leoni and Wil M.P. van der Aalst. Data-aware process mining: Discovering decisions in processes using alignments. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, pages 1454–1461, New York, NY, USA, 2013b. ACM. ISBN 978-1-4503-1656-9. doi: 10.1145/2480362.2480633. URL `http://doi.acm.org/10.1145/2480362.2480633`.

A. K. A. De Medeiros, A. J. Weijters, and Wil M. P. van der Aalst. Genetic process mining: an experimental evaluation. *Data Mining and Knowledge Discovery*, 14(2): 245–304, 2007.

Ana Karla A de Medeiros, Wil MP van der Aalst, and AJMM Weijters. Workflow mining: Current status and future directions. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 389–406. Springer, 2003.

Jochen De Weerdt, Manu De Backer, Jan Vanthienen, and Bart Baesens. A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs. *Inf. Syst.*, 37(7):654–676, November 2012a. ISSN 0306-4379. doi: 10.1016/j.is.2012.02.004. URL `http://dx.doi.org/10.1016/j.is.2012.02.004`.

Jochen De Weerdt, Manu De Backer, Jan Vanthienen, and Bart Baesens. A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs. *Information Systems*, 37(7):654–676, 2012b.

Roger Debreceny, Glen L. Gray, Wai-Lum Tham, Kay-Yiong Goh, and Puay-Ling Tang. The development of embedded audit modules to support continuous monitoring in the electronic commerce environment. *International Journal of Auditing*, 7(2):169–185, 2003.

Benoît Depaire, Jo Swinnen, Mieke Jans, and Koen Vanhoof. A process deviation analysis framework. In *Business Process Management Workshops: BPM 2012 International Workshops, Tallinn, Estonia, September 3, 2012. Revised Papers*, pages 701–706. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

Kanthana Ditkaew and Muttanachai Suttipun. The impact of audit data analytics on audit quality and audit review continuity in thailand. *Asian Journal of Accounting Research*, 8(3):269–278, 2023.

Mengming Michael Dong, Theophanis C Stratopoulos, and Victor Xiaoqi Wang. A scoping review of chatgpt research in accounting and finance. *International Journal of Accounting Information Systems*, 55:100715, 2024.

Huijue Kelly Duan, Miklos A Vasarhelyi, and Mauricio Codesso. Integrating process mining and machine learning for advanced internal control evaluation in auditing. *Journal of Information Systems*, 39(1):55–75, 2025.

Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A. Reijers. *Fundamentals of Business Process Management*. Springer Publishing Company, Incorporated, 2013. ISBN 3642331424, 9783642331428.

Kathleen M Eisenhardt. Building theories from case study research. *Academy of management review*, 14(4):532–550, 1989.

Robert K Elliott. Assurance services and the audit heritage. *The CPA Journal*, 68 (6):40, 1998.

Marc Eulerich, Benjamin Fligge, Vanessa I López Kasper, and David A Wood. Patience is key: The time it takes to see benefits from continuous auditing. *Accounting Horizons*, 39(1):69–86, 2025.

Dirk Fahland and Wil M.P. van der Aalst. Model repair - aligning process models to reality. *Inf. Syst.*, 47(C):220–243, January 2015. ISSN 0306-4379. doi: 10.1016/j.is.2013.12.007. URL `http://dx.doi.org/10.1016/j.is.2013.12.007`.

Maia Farkas and Uday S Murthy. Nonprofessional investors' perceptions of the incremental value of continuous auditing and continuous controls monitoring: An experimental investigation. *International Journal of Accounting Information Systems*, 15(2):102–121, 2014.

Paolo Felli, Alessandro Gianola, Marco Montali, Andrey Rivkin, and Sarah Winkler. Cocomot: conformance checking of multi-perspective processes via smt. In *Business Process Management: 19th International Conference, BPM 2021, Rome, Italy, September 06–10, 2021, Proceedings 19*, pages 217–234. Springer, 2021.

Paolo Felli, Alessandro Gianola, Marco Montali, Andrey Rivkin, and Sarah Winkler. Multi-perspective conformance checking of uncertain process traces: An smt-based approach. *Engineering Applications of Artificial Intelligence*, 126:106895, 2023.

Yoav Freund, H Sebastian Seung, Eli Shamir, and Naftali Tishby. Selective sampling using the query by committee algorithm. *Machine learning*, 28(2):133–168, 1997.

Fabian Friedrich, Jan Mendling, and Frank Puhlmann. Process model generation from natural language text. In *Advanced Information Systems Engineering: 23rd International Conference, CAiSE 2011, London, UK, June 20-24, 2011. Proceedings*, pages 482–496. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

Tassilo L. Föhr, Valentin Reichelt, Kai-Uwe Marten, and Marc Eulerich. A framework for the structured implementation of process mining for audit tasks. *International Journal of Accounting Information Systems*, 56:100727, 2025. ISSN 1467-0895. doi: https://doi.org/10.1016/j.accinf.2025.100727. URL `https://www.sciencedirect.com/science/article/pii/S146708952500003X`.

Luciano García-Bañuelos, Nick RTP Van Beest, Marlon Dumas, Marcello La Rosa, and Willem Mertens. Complete and interpretable conformance checking of business processes. *IEEE Transactions on Software Engineering*, 44(3):262–290, 2018.

Dennis A. Gioia, Kevin G. Corley, and Aimee L. Hamilton. Seeking qualitative rigor in inductive research: Notes on the gioia methodology. *Organizational Research Methods*, 16(1):15–31, 1 2013. ISSN 1094-4281. doi: 10.1177/1094428112452151.

Antonio Gomariz, Manuel Campos, Roque Marin, and Bart Goethals. Clasp: An efficient algorithm for mining frequent closed sequences. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 50–61. Springer, 2013.

George C Gonzalez, Pratyush N Sharma, and Dennis F Galletta. The antecedents of the use of continuous auditing in the internal auditing context. *International Journal of Accounting Information Systems*, 13(3):248–262, 2012.

E. González López de Murillas, Hajo A. Reijers, and Wil M. P. van der Aalst. Connecting databases with process mining: A meta model and toolset. In *Enterprise, Business-Process and Information Systems Modeling: 17th International Conference, BPMDS 2016, 21st International Conference, EMMSAD 2016, Held at CAiSE 2016, Ljubljana, Slovenia, June 13-14,2016 , Proceedings*, pages 231–249. Springer International Publishing, Cham, 2016.

Eduardo González López de Murillas, Wil M. P. Aalst, and Hajo A. Reijers. Process mining on databases: Unearthing historical data from redo logs. In *Business Process Management: 13th International Conference, BPM 2015, Innsbruck, Austria, August 31 – September 3, 2015, Proceedings*, pages 367–385. Springer International Publishing, Cham, 2015.

Eduardo Goulart Rocha, Sebastiaan J van Zelst, and Wil MP van der Aalst. Mining behavioral patterns for conformance diagnostics. In *International Conference on Business Process Management*, pages 291–308. Springer, 2024.

S Micheal Groomer and Uday S Murthy. Continuous auditing of database applications: An embedded audit module approach. *Journal of Information Systems*, 3 (2):53–69, 1989.

Christian W. Günther and Wil M. P. Van Der Aalst. Fuzzy mining–adaptive process simplification based on multi-perspective metrics. In *Business Process Management*, pages 328–343. Springer, 2007a.

Christian W. Günther and Wil M.P. Van Der Aalst. Fuzzy mining: Adaptive process simplification based on multi-perspective metrics. In *Proceedings of the 5th International Conference on Business Process Management*, BPM'07, pages 328–343, Berlin, Heidelberg, 2007b. Springer-Verlag. ISBN 3-540-75182-3, 978-3-540-75182-3.

Steve Hanneke. Theory of disagreement-based active learning. *Foundations and Trends in Machine Learning*, 7(2-3):131–309, 2014.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

Krzysztof Honkisz, Krzysztof Kluza, and Piotr Wiśniewski. A concept for generating business process models from natural language description. In *International Conference on Knowledge Science, Engineering and Management*, pages 91–103. Springer, 2018.

Feiqi Huang and Miklos A Vasarhelyi. Applying robotic process automation (rpa) in auditing: A framework. *International journal of accounting information systems*, 35:100433, 2019.

Feiqi Huang, Won Gyun No, Miklos A Vasarhelyi, and Zhaokai Yan. Audit data analytics, machine learning, and full population testing. *The Journal of Finance and Data Science*, 8:138–144, 2022.

Sheng-Jun Huang, Rong Jin, and Zhi-Hua Zhou. Active learning by querying informative and representative examples. In *Advances in neural information processing systems*, pages 892–900, 2010.

Shi-Ming Huang, David C. Yen, Yu-Chung Hung, Yen-Ju Zhou, and Jing-Shiuan Hua. A business process gap detecting mechanism between information system process flow and internal control flow. *Decision Support Systems*, 47:436–454, 2009.

Hussein Issa. *Exceptional exceptions.* PhD thesis, Rutgers University-Graduate School-Newark, 2013.

Mieke Jans. *Internal fraud risk reduction by data mining and process mining: framework and case study.* Universiteit Hasselt, 2009.

Mieke Jans. Auditor choices during event log building for process mining. *Journal of Emerging Technologies in Accounting*, 16(2):59–67, 2019.

Mieke Jans and Marzie Hosseinpour. How active learning and process mining can act as continuous auditing catalyst. *International Journal of Accounting Information Systems*, 32:44–58, 2019.

Mieke Jans and Manal Laghmouch. Process mining for detailed process analysis. In *Advanced Digital Auditing: Theory and Practice of Auditing Complex Information Systems and Technologies*, pages 237–256. Springer, 2022.

Mieke Jans, Jan Martijn Van Der Werf, Nadine Lybaert, and Koen Vanhoof. A business process mining application for internal transaction fraud mitigation. *Expert System Applications*, 38(10):13351–13359, 2011a. doi: 10.1016/j.eswa.2011.04.159. URL `http://dx.doi.org/10.1016/j.eswa.2011.04.159`.

Mieke Jans, Jan Martijn van der Werf, Nadine Lybaert, and Koen Vanhoof. A business process mining application for internal transaction fraud mitigation. *Expert Systems with Applications*, 38(10):13351–13359, 2011b. ISSN 0957-4174. doi: http://dx.doi.org/10.1016/j.eswa.2011.04.159.

Mieke Jans, Michael G. Alles, and Miklos A. Vasarhelyi. The case for process mining in auditing: Sources of value added and areas of application. *International Journal of Accounting Information Systems 14*, 14(1):1–20, March 2013. ISSN 14670895. doi: 10.1016/j.accinf.2012.06.015.

Mieke Jans, Michael G. Alles, and Miklos A. Vasarhelyi. A field study on the use of process mining of event logs as an analytical procedure in auditing. *The Accounting Review*, 89(5):1751–1773, September 2014. ISSN 00014826. doi: 10.2308/accr-50807.

Paul Johannesson and Erik Perjons. *An introduction to design science.* Springer, 2014.

Yongbum Kim and Miklos A. Vasarhelyi. A model to detect potentially fraudulent/abnormal wires of an insurance company: An unsupervised rule-based approach. *Journal of Emerging Technologies in Accounting*, 9(1):95–110, 2012.

Harvey S Koch. Online computer auditing through continuous and intermittent simulation. *MIS Quarterly*, pages 29–41, 1981.

Julia Kokina, Shay Blanchette, Thomas H Davenport, and Dessislava Pachamanova. Challenges and opportunities for artificial intelligence in auditing: Evidence from the field. *International Journal of Accounting Information Systems*, 56:100734, 2025.

John Peter Krahel and William R. Titera. Consequences of big data and formalization on accounting and auditing standards. *Accounting Horizons*, 29(2):409–422, 2015. ISSN 08887993. doi: 10.2308/acch-51065.

Siripan Kuenkaikaew and Miklos A. Vasarhelyi. The predictive audit framework. *The International Journal of Digital Accounting Research*, 13:37–71, 2013.

John R Kuhn Jr and Steve G Sutton. Continuous auditing in erp system environments: The current state and future directions. *Journal of Information Systems*, 24(1):91–112, 2010.

Manal Laghmouch, Benoît Depaire, and Mieke Jans. Towards full population testing in auditing: How many process deviations should be labeled? In *2024 6th International Conference on Process Mining (ICPM)*, pages 49–56. IEEE, 2024.

Heejae Lee, Lu Zhang, Qi Liu, and Miklos Vasarhelyi. Text visual analysis in auditing: Data analytics for journal entries testing. *International journal of accounting information systems*, 46:100571, 2022.

Sander J. J. Leemans, Dirk Fahland, and Wil M. P. van der Aalst. Discovering block-structured process models from event logs-a constructive approach. In *Application and Theory of Petri Nets and Concurrency*, pages 311–329. Springer, 2013.

Sander J. J. Leemans, Dirk Fahland, and Wil M.P. Van Der Aalst. Process and deviation exploration with inductive visual miner. *Auditing: A Journal of Practice and Theory*, 2014.

Rainer Lenz and Kim K Jeppesen. The future of internal auditing: Gardener of governance. *EDPACS*, 66(5):1–21, 2022.

Huaxia Li, Haoyun Gao, Chengzhang Wu, and Miklos A Vasarhelyi. Extracting financial data from unstructured sources: Leveraging large language models. *Journal of Information Systems*, 39(1):135–156, 2025.

Pei Li, David Y Chan, and Alexander Kogan. Exception prioritization in the continuous auditing environment: A framework and experimental evaluation. *Journal of Information Systems*, 30(2):135–157, 2015.

Pei Li, David Y. Chan, and Alexander Kogan. Exception prioritization in the continuous auditing environment: A framework and experimental evaluation. *Journal of Information Systems*, 30(2):135 – 157, 2016. ISSN 08887985.

Danielle R Lombardi, Meehyun Kim, Janice C Sipior, and Miklos A Vasarhelyi. The increased role of advanced technology and automation in audit: A delphi study. *International Journal of Accounting Information Systems*, 56:100733, 2025.

Tim Loughran and Bill McDonald. Textual analysis in accounting and finance: A survey. *Journal of accounting research*, 54(4):1187–1230, 2016.

Xixi Lu, Dirk Fahland, Frank JHM van den Biggelaar, and Wil MP van der Aalst. Detecting deviating behaviors without models. In *International Conference on Business Process Management*, pages 126–139. Springer, 2016.

Bertrand Malsch and Steven E Salterio. "doing good field research": Assessing the quality of audit field research. *Auditing: A Journal of Practice & Theory*, 35(1): 1–22, 2015.

Felix Mannhardt, Massimiliano de Leoni, Hajo A. Reijers, and Wil M.P. van der Aalst. Balanced multi-perspective checking of process conformance. *Computing*, 98 (4):407–437, 2016.

Laura Marcus, Sebastian Johannes Schmid, Franziska Friedrich, Maximilian Röglinger, and Philipp Grindemann. Navigating the landscape of organizational process mining setups: A taxonomy approach. *Business & information systems engineering*, pages 1–22, 2024.

A. K. Medeiros, A. J. Weijters, and Wil M.P. Aalst. Genetic process mining: An experimental evaluation. *Data Min. Knowl. Discov.*, 14(2):245–304, April 2007. ISSN 1384-5810.

Azadeh Sadat Mozafari Mehr, Renata M de Carvalho, and Boudewijn van Dongen. Explainable conformance checking: understanding patterns of anomalous behavior. *Engineering Applications of Artificial Intelligence*, 126:106827, 2023.

Jan Mendling, Gustaf Neumann, and Wil Van Der Aalst. Understanding the occurrence of errors in process models based on metrics. In *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS, OTM Confederated International Conferences CoopIS,DOA, ODBASE, GADA, and IS*, pages 113–130. Springer, 2007.

Jan Mendling, Henrik Leopold, Henning Meyerhenke, and Benoît Depaire. Methodology of algorithm engineering. *ACM Computing Surveys*, 2025.

Krishnagopal Menon and David D Williams. Long-term trends in audit fees. *Auditing: A Journal of Practice & Theory*, 20(1):115–136, 2001.

Matthew B Miles and A Michael Huberman. *Qualitative data analysis: An expanded sourcebook*. sage, 1994.

Kevin C. Moffitt, Vernon J. Richardson, Neal M. Snow, Martin M. Weisner, and David A. Wood. Perspectives on past and future ais research as the journal of information systems turns thirty. *Journal of Information Systems*, 30(3):157 – 171, 2016. ISSN 08887985.

Jorge Munoz-Gama and Josep Carmona. Enhancing precision in process conformance: Stability, confidence and severity. In *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2011, part of the IEEE Symposium Series on Computational Intelligence 2011, April 11-15, 2011, Paris, France*, pages 184–191. IEEE, 2011. doi: 10.1109/CIDM.2011.5949451. URL `http://dx.doi.org/10.1109/CIDM.2011.5949451`.

Jorge Muñoz-Gama and Josep Carmona. A fresh look at precision in process conformance. In *BPM*, volume 6336 of *Lecture Notes in Computer Science*, pages 211–226. Springer, 2010. ISBN 978-3-642-15617-5.

Gonzalo Nápoles, Isel Grau, Elpiniki Papageorgiou, Rafael Bello, and Koen Vanhoof. Rough cognitive networks. *Knowledge-Based Systems*, 91:46—61, 2016.

Gonzalo Nápoles, Rafael Falcon, Elpiniki Papageorgiou, Rafael Bello, and Koen Vanhoof. Rough cognitive ensembles. *International Journal of Approximate Reasoning*, 85:79—96, 2017.

Gonzalo Nápoles, Carlos Mosquera, Rafael Falcon, Isel Grau, Rafael Bello, and Koen Vanhoof. Fuzzy-rough cognitive networks. *Neural Networks*, 97:19–27, 2018.

Gonzalo Nápoles, Yamisleydi Salgueiro, Isel Grau, and Maikel Leon Espinosa. Recurrence-aware long-term cognitive network for explainable pattern classification. *IEEE transactions on cybernetics*, 53(10):6083–6094, 2022.

Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, 1970. doi: 10.1016/0022-2836(70)90057-4.

Hoang Nguyen, Marlon Dumas, Marcello Rosa, Fabrizio Maria Maggi, and Suriadi Suriadi. Mining business process deviance: A quest for accuracy. In *On the Move to Meaningful Internet Systems: OTM 2014 Conferences: Confederated International Conferences: CoopIS, and ODBASE 2014, Amantea, Italy, October 27-31, 2014, Proceedings*, pages 436–445. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

Mogens Nielsen, Gordon Plotkin, and Glynn Winskel. Petri nets, event structures and domains, part i. *Theoretical Computer Science*, 13(1):85–108, 1981.

Ed O'Donnell and Joseph J. Schultz Jr. The Influence of Business-Process-Focused Audit Support Software on Analytical Procedures Judgments. *Auditing: A Journal of Practice & Theory*, 22(2):265–279, 2003. ISSN 02780380.

Association of Certified Fraud Examiners. Report to the nations: 2024 global study on occupational fraud and abuse, 2024.

AICPA: American Institute of Certified Public Accountants. *Guide to Audit Data Analytics*. American Institute of Certified Public Accountants, New York, NY, 2017.

Fynn Oldenburg, Kai Hoberg, and Henrik Leopold. Process mining in supply chain management: state-of-the-art, use cases and research outlook. *International Journal of Production Research*, 63(8):2889–2904, 2025.

PCAOB. Spotlight: Data and technology research project update. Technical report, Public Company Accounting Oversight Board, May 2021. URL `https://pcaobus.org/documents/data-and-technology-project-may-2021-spotlight.pdf`. Staff document, not a rule or statement of the Board.

Jari Peeperkorn, Seppe vanden Broucke, and Jochen De Weerdt. Global conformance checking measures using shallow representation and deep learning. *Engineering Applications of Artificial Intelligence*, 123:106393, 2023.

Marco Pegoraro, Merih Seran Uysal, and Wil MP van der Aalst. Conformance checking over uncertain event data. *Information Systems*, 102:101810, 2021.

Arif Perdana, W Eric Lee, and Chu Mui Kim. Prototyping and implementing robotic process automation in accounting firms: Benefits, challenges and opportunities to audit automation. *International journal of accounting information systems*, 51: 100641, 2023.

Johan L. Perols and Uday S. Murthy. Information fusion in continuous assurance. *Journal of Information Systems*, 26(2):35 – 52, 2012a. ISSN 08887985.

Johan L Perols and Uday S Murthy. Information fusion in continuous assurance. *Journal of Information Systems*, 26(2):35–52, 2012b.

Clifton Phua, Vincent Lee, Kate Smith, and Ross Gayler. A comprehensive survey of data mining-based fraud detection research. *arXiv preprint arXiv:1009.6119*, 2010.

A. Pika, W. M. P. van der Aalst, M. T. Wynn, C. J. Fidge, and A. H. M. ter Hofstede. Evaluating and predicting overall process risk using event logs. *Information Sciences*, 352–353:98–120, 2016.

Artem Polyvyanyy and Anna Kalenkova. Conformance checking of partially matching processes: An entropy-based approach. *Information Systems*, 106:101720, 2022.

Elham Ramezani, Dirk Fahland, and Wil M.P. van der Aalst. Where did i misbehave? diagnostic information in compliance checking. In *International conference on business process management*, pages 262–278. Springer, 2012.

Daniel Reißner, Raffaele Conforti, Marlon Dumas, Marcello La Rosa, and Abel Armas-Cervantes. Scalable conformance checking of business processes. In *OTM Confederated International Conferences On the Move to Meaningful Internet Systems*, pages 607–627. Springer, 2017.

Pall Rikhardsson and Richard Dull. An exploratory study of the adoption, application and impacts of continuous auditing technologies in small businesses. *International Journal of Accounting Information Systems*, 20:26–37, 2016.

Anne Rozinat and Wil M.P. van der Aalst. Conformance checking of processes based on monitoring real behavior. *Information Systems*, 33(1):64–95, 2008. ISSN 0306-4379. doi: 10.1016/j.is.2007.07.001.

Nick Russell, Wil M.P. van der Aalst, and Arthur H.M. ter Hofstede. Exception handling patterns in process-aware information systems. *BPM Center Report BPM*, pages 288–302, 2006.

Shazia Sadiq, Guido Governatori, and Kioumars Namiri. Modeling control objectives for business process compliance. In *International conference on business process management*, pages 149–164. Springer, 2007.

Johnny. Saldaña. *The coding manual for qualitative researchers.* SAGE London, 3 edition, 2016. ISBN 9781473902497.

Mark N.K. Saunders. *Research methods for business students, 5/e.* Pearson Education India, 2011.

Martin Schultz. Enriching process models for business process compliance checking in erp environments. In *International Conference on Design Science Research in Information Systems*, pages 120–135. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-38827-9.

Clive Seale. Quality in qualitative research. *Qualitative inquiry*, 5(4):465–478, 1999.

Elseline Senave, Mieke J Jans, and Rajendra P Srivastava. The application of text mining in accounting. *International Journal of Accounting Information Systems*, 50:100624, 2023.

Burr Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

Navitha Singh Sewpersadh. Adaptive structural audit processes as shaped by emerging technologies. *International Journal of Accounting Information Systems*, 56: 100735, 2025.

Herbert A. Simon. *The Sciences of the Artificial*. MIT Press, 3 edition, 1996.

Kishore Singh, Peter J Best, Mario Bojilov, and Catherine Blunt. Continuous auditing and continuous monitoring in erp environments: case studies of application implementations. *Journal of Information Systems*, 28(1):287–310, 2013.

Christian Sonnenberg and Jan vom Brocke. The missing link between BPM and accounting: Using event data for accounting in process-oriented organizations. *Business Process Management Journal*, 20(2):213–246, 2014.

Rajendra P. Srivastava and Glenn R. Shafer. Belief-Function Formulas for Audit Risk. *Accounting Review*, 67(2):249–283, April 1992. ISSN 00014826.

Anselm L Strauss. *Qualitative analysis for social scientists*. Cambridge University Press, 1987. ISBN 0521338069.

Ernest T. Stringer. *Action research*. Thousand Oaks, California : SAGE, 4 edition, 2014. ISBN 9781452205083.

Ting Sun and Miklos A Vasarhelyi. Embracing textual data analytics in auditing with deep learning. *International Journal of Digital Accounting Research*, 18, 2018.

Jo Swinnen, Benoît Depaire, Mieke Jans, and Koen Vanhoof. A process deviation analysis, a case study. In *Business Process Management Workshops*, volume 99 of *Lecture Notes in Business Information Processing*, pages 87–98. Springer Berlin Heidelberg, 2011.

Alaa Tharwat and Wolfram Schenck. A survey on active learning: State-of-the-art, practical challenges and research directions. *Mathematics*, 11(4):820, 2023.

Sutapat Thiprungsri and Miklos A. Vasarhelyi. Cluster analysis for anomaly detection in accounting data: An audit approach. *International Journal of Digital Accounting Research*, 11, 2011.

William R. Titera. Updating audit standard–enabling audit data analysis. *Journal of Information Systems*, 27(1):325–331, 2013. ISSN 08887985. doi: 10.2308/isys-50427.

Macarena Torroba, José Ramón Sánchez, Lidia López, and Ángela Callejón. Investigating the impacting factors for the audit professionals to adopt data analysis and artificial intelligence: Empirical evidence for spain. *International Journal of Accounting Information Systems*, 56:100738, 2025.

Stacie Tronto and Brenda L Killingsworth. How internal audit can champion continuous monitoring in a business operation via visual reporting and overcome barriers to success. *The International Journal of Digital Accounting Research*, 21(27):23–59, 2021.

Han van der Aa, Henrik Leopold, and Hajo A. Reijers. Detecting inconsistencies between process models and textual descriptions. In *Business Process Management: 13th International Conference, BPM 2015, Innsbruck, Austria, August 31 – September 3, 2015, Proceedings*, pages 90–105. Springer International Publishing, Cham, 2015.

W. M. P. van der Aalst. Process mining: Data science in action. 2016.

W. M. P. van der Aalst, K. M. van Hee, J. M. van der Werf, and M. Verdonk. Auditing 2.0: Using process mining to support tomorrow's auditor. *Computer*, 43(3):90–93, March 2010. ISSN 0018-9162. doi: 10.1109/MC.2010.61.

Wil Van der Aalst. *Process Mining Data science in action*. Springer, 2016.

Wil Van Der Aalst, Arya Adriansyah, and Boudewijn Van Dongen. Causal nets: a modeling language tailored towards process discovery. In *International conference on concurrency theory*, pages 28–42. Springer, 2011.

Wil M. P. van der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004a.

Wil M. P. van der Aalst, H. T. de Beer, and B. F. van Dongen. Process mining and verification of properties: An approach based on temporal logic. In *Proceedings of the 2005 Confederated International Conference on On the Move to Meaningful Internet Systems*, OTM'05, pages 130–147. Springer-Verlag, 2005. ISBN 3-540-29736-7, 978-3-540-29736-9. doi: 10.1007/11575771_11.

Wil M.P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer Publishing Company, Incorporated, 1st edition, 2011. ISBN 3642193447, 9783642193446.

Wil M.P. van der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *IEEE Trans. on Knowl. and Data Eng.*, 16(9):1128–1142, 2004b. ISSN 1041-4347. doi: 10.1109/TKDE.2004.47.

Wil M.P. van der Aalst, Arya Adriansyah, and Boudewijn van Dongen. Replaying history on process models for conformance checking and performance analysis. *Wiley Int. Rev. Data Min. and Knowl. Disc.*, 2(2):182–192, 2012. ISSN 1942-4787. doi: 10.1002/widm.1045.

Sebastiaan J van Zelst, Boudewijn F van Dongen, and Wil MP van der Aalst. Ilp-based process discovery using hybrid regions. In *ATAED@ petri nets/ACSD*, pages 47–61, 2015.

Miklos Vasarhelyi, Michael Alles, and Alexander Kogan. Principles of Analytic Monitoring for Continuous Assurance. *Journal of Emerging Technologies in Accounting*, 1:1–21, 2004.

Miklos A. Vasarhelyi. *Continuous Assurance for the Now Economy*. PhD thesis, Rutgers Business School, 2010.

Miklos A. Vasarhelyi and Fern B. Halper. The continuous audit of online systems. In *Auditing: A Journal of Practice and Theory*. Citeseer, 1991.

Iris Vessey and Dennis Galletta. Cognitive fit: An empirical study of information acquisition. *Information systems research*, 2(1):63–84, 1991.

Sonia Vitali and Marco Giuliani. Emerging digital technologies and auditing firms: Opportunities and challenges. *International Journal of Accounting Information Systems*, 53:100676, 2024.

Jr J. Donald Warren, Kevin C. Moffitt, and Paul Byrnes. How big data will change accounting. *Accounting Horizons*, 29(2):397–407, 2015. ISSN 08887993. doi: 10.2308/acch-51069.

Barbara Weber, Manfred Reichert, and Stefanie Rinderle-Ma. Change patterns and change support features–enhancing flexibility in process-aware information systems. *Data & knowledge engineering*, 66(3):438–466, 2008.

A. J. M. M. Weijters, Wil M. P. van Der Aalst, and Ana Karla Alves De Medeiros. Process mining with the heuristics miner-algorithm. *BETA Working paper series TU/e*, 166:1–34, 2006a.

AJMM Weijters and Wil MP van der Aalst. Process mining: discovering workflow models from event-based data. In *Belgium-Netherlands Conf. on Artificial Intelligence*. Citeseer, 2001.

AJMM Weijters, Wil MP van Der Aalst, and AK Alves De Medeiros. Process mining with the heuristics miner-algorithm. *Technische Universiteit Eindhoven, Tech. Rep. WP*, 166:1–34, 2006b.

Michael Werner. Financial process mining - accounting data structure dependent control flow inference. *International Journal of Accounting Information Systems*, 25:57–80, 2017. ISSN 1467-0895. doi: https://doi.org/10.1016/j.accinf.2017.03.004.

Michael Werner and Nick Gehrke. Multilevel process mining for financial audits. *IEEE Transactions on Services Computing*, 8(6):820–832, 2015.

Carla Willig. *Introducing qualitative research in psychology*. McGraw-Hill Education (UK), 2013.

Jonathan Wilson. *Essentials of business research: A guide to doing your research project*. Sage, 2014.

Kyunghee Yoon, Lucas Hoogduin, and Li Zhang. Big data as complementary audit evidence. *Accounting Horizons*, 29(2):431–438, 2015.

Chanyuan Abigail Zhang, Soohyun Cho, and Miklos Vasarhelyi. Explainable artificial intelligence (xai) in auditing. *International Journal of Accounting Information Systems*, 46:100572, 2022a.

Sicui Zhang, Laura Genga, Lukas Dekker, Hongchao Nie, Xudong Lu, Huilong Duan, and Uzay Kaymak. Fuzzy multi-perspective conformance checking for business processes. *Applied Soft Computing*, 130:109710, 2022b.