VILNIUS GEDIMINAS TECHNICAL UNIVERSITY

FACULTY OF fundamental sciences

DEPARTMENT OF information systems

Islam Salamzade

**MINING METHOD APPLICATION FOR SOFTWARE SYSTEM EVENT LOG ANALYSIS**

Master‘s degree Thesis

Information Sytems Software Engineering study programme, state code 6211BX017

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_specialisation

Informatics Engineering study field

**Supervisor** Dr. Titas Savickas\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_5 (Title, Name, Surname)

Vilnius, 2025

**Table of Content**

[1. Introduction 4](#_Toc201709012)

[1.1. Investigation Object 4](#_Toc201709013)

[1.2. The Aim and Tasks of the Thesis 4](#_Toc201709014)

[1.3. Novelty of the Topic 5](#_Toc201709015)

[1.4. Relevance of the Topic 5](#_Toc201709016)

[1.5. Research Methodology 6](#_Toc201709017)

[1.6. Scientific Value of the Thesis 6](#_Toc201709018)

[1.7. Main Results of the Thesis 7](#_Toc201709019)

[1.8. Structure of the Work 8](#_Toc201709020)

[2. Related Works Analysis 8](#_Toc201709021)

[2.1. Main Concepts Analysis 8](#_Toc201709022)

[2.1.1. Process Mining 8](#_Toc201709023)

[2.1.2. Event logs 9](#_Toc201709024)

[2.1.3. System logs 15](#_Toc201709025)

[2.1.4. Extracting Software System Logs as Event Logs 17](#_Toc201709026)

[2.1.5. Applying process mining on software systems 20](#_Toc201709027)

[2.2. Review of Main Related Works 22](#_Toc201709028)

[2.3. Systematization of Related Works 27](#_Toc201709029)

[2.4. Main Results of the 2nd Section 32](#_Toc201709030)

[3. Proposed Approach 33](#_Toc201709031)

[3.1. General Overview of the Proposed Approach 33](#_Toc201709032)

[3.2. Methodological Workflow (BPMN Representation) 34](#_Toc201709033)

[3.3. Advantages of proposed Method 38](#_Toc201709034)

[3.4. Main Results of 3rd Section 39](#_Toc201709035)

[4. Empirical Study and Results 40](#_Toc201709036)

[4.1. Initial Experimental Plan 40](#_Toc201709037)

[4.2. Datasets will be used in Experiments 42](#_Toc201709038)

[5. Conclusion 44](#_Toc201709039)

[6. References 45](#_Toc201709040)

**List of Images**

[Figure 2.1 Process Mining Manifesto (van der Aalst & Process Mining Group, 2011) 9](#_Toc201709041)

[Figure 2.2. Example of Event logs (Process Mining, n.d.) 11](#_Toc201709042)

[Figure 2.3 Example of event log(2) (Process Mining, n.d.). 13](#_Toc201709043)

[Figure 2.4. Example of system logs 15](#_Toc201709044)

[Figure 2.5. Raw system log 18](#_Toc201709045)

[Figure 2.6. XML Snippet of XES file 19](#_Toc201709046)

[Figure 2.7. Overview of Process Mining 20](#_Toc201709047)

[Figure 3.1. BPMN representation of transforming heterogeneous software system logs into structured event logs for process mining. 35](#_Toc201709048)

[Figure 3.2. BPMN sub-process representing Smart Event Grouping 35](#_Toc201709049)

[Figure 4.1. Example: Apache Raw Log (Before Preprocessing) 43](#_Toc201709050)

**List of Tables**

[Table 2.1. Summary and extraction of research papers based on mining methods and event log analysis 28](#_Toc201709051)

[Table 3.1. Evaluation Metrics For Software System Event Log Analysis 37](#_Toc201709052)

[Table 4.1. Example characteristics of selected log sources 41](#_Toc201709053)

[Table 4.2. Summary of datasets will be used in experiments 42](#_Toc201709054)

**Abbreviations**

**PM** – Process Mining

**EL** – Event Log

**SE** – Software Engineering

**IS** – Information Systems

**DA** – Data Analytics

**CS** – Computer Science

**PL/SQL** – Procedural Language / Structured Query Language

**CI/CD** – Continuous Integration / Continuous Deployment

**API** – Application Programming Interface

**CRM** – Customer Relationship Management

**IT** – Information Technology

**RPA** – Robotic Process Automation

**ERP** – Enterprise Resource Planning

# Introduction

Running software systems generates a great volume of event logs from the fast expansion of digital systems. These event logs provide great possibilities for perceptive studies on system performance, user behavior, and operational efficiency. Still, the sheer number and intricacy of these logs provide a great difficulty. Manual analysis is not only time-consuming and prone to mistakes but also useless, hence new mining methods must be used for automatic and efficient study.

This thesis's research problem is the challenge of deriving significant understanding from event logs. Because of the diversity of logs, different logging techniques, and lack of standardized procedures, current methods can fail in spotting actionable trends. Using these logs to enhance systems and processes presents difficult tasks for organizations. In advanced software systems, where small inefficiencies can cause major operational interruptions, this problem is very important.

This work addresses the challenge of analyzing heterogeneous software system logs by applying advanced process mining techniques. Grounded in the broader field of process mining—which has been shown to enhance operational visibility and performance analysis—this study builds on established research. For example, Milani and Maggi (2018) evaluate how log-based process performance techniques can support decision-making. Similarly, Ter Hofstede et al. (2023) emphasize the importance of high-quality data for ensuring reliable process insights, which directly impacts system effectiveness.

## Investigation Object

The Investigation Object – mining methods for analyzing software system event logs.

## The Aim and Tasks of the Thesis

**The Aim of the Research** is to enhance the analysis of software system event logs by developing a method that improves the extraction of actionable insights through advanced process mining techniques.

To achieve this aim, the following tasks were formulated:

1. **To analyze and compare existing methods for event log analysis** in terms of their capabilities, limitations, and applicability, focusing on the challenges of handling heterogeneous and large-scale data.
2. **To propose a method for event log analysis** tailored to address the identified challenges, incorporating advanced process mining techniques to improve the accuracy and efficiency of extracting actionable insights.
3. **To implement and evaluate the proposed method** by developing a software prototype and testing its performance on real-world datasets, assessing its effectiveness and usability in practical scenarios.

## Novelty of the Topic

Software systems increasingly rely on event logs, which contain valuable insights about system performance, user behavior, and operational efficiency. However, current methods for analyzing these logs often struggle with mixed data types, noise, and incomplete or poorly structured entries (Shahzadi et al., 2024).

This thesis addresses these challenges by proposing an enhanced mining approach tailored for software system event logs. The method incorporates artificial intelligence and flexible analysis techniques to improve upon traditional log mining approaches (Effendi & Kim, 2024). While existing research emphasizes the value of process mining, it often overlooks the specific complexities found in technical system logs (Marin-Castro & Tello-Leal, 2021).

The proposed method aims to bridge this gap and contribute new insights for event log analysis, with potential applications in domains such as healthcare, IT infrastructure, and business operations.

## Relevance of the Topic

Software systems are growing quickly in many industries, leading to the generation of vast amounts of event logs. These logs contain valuable information about system performance, user interactions, and operational flow. However, they are often underutilized due to their complexity, unpredictability, and the absence of standard analytical approaches. This study emphasizes the importance of better understanding event logs to enhance system performance and support improved decision-making.

Analyzing event logs plays a crucial role in detecting anomalies, ensuring compliance, and identifying inefficiencies across domains such as finance, manufacturing, and healthcare. Ter Hofstede et al. (2023) illustrate how process mining reveals hidden patterns in event data, while Shahzadi et al. (2024) discuss challenges related to noisy and incomplete logs. Ter Hofstede et al. (2023) also highlight the growing need for improved methods to manage complex event data in high-stakes contexts.

This research seeks to bridge recent advances in process mining with practical applications in software system logs. It aims to enhance scalability, improve log preprocessing, and support data protection, thereby improving system reliability, operational efficiency, and strategic decision-making.

## Research Methodology

This thesis uses a mix of analysis, conceptual development, and experimentation to advance the study's objectives.

In this part, library research and comparative analysis are employed to examine different approaches to analyzing event logs in software systems. This section offers an overview of process mining methods, their application areas, and the challenges posed by diverse types of event log data. By comparing these methods, their limitations are identified to inform the design of a tailored solution (Ter Hofstede et al., 2023; Marin-Castro & Tello-Leal, 2021).

The proposed mining method will be developed using logical reasoning, generalization, and conceptual modeling. These strategies synthesize insights from existing literature to construct a framework addressing key issues in event log analysis, with a focus on accuracy, scalability, and real-world applicability.

During the validation phase, a prototype implementation of the proposed method will be developed and tested on real event log datasets. Case studies will evaluate the method's usability and effectiveness, and performance will be assessed using metrics such as fitness, precision, efficiency, and scalability. While a direct comparison with existing techniques may be difficult due to dataset heterogeneity, a baseline evaluation will be conducted using known benchmarks (Milani & Maggi, 2018; Effendi & Kim, 2024).

## Scientific Value of the Thesis

This thesis contributes to the advancement of process mining and software log analysis by addressing a critical real-world challenge: the transformation of heterogeneous, unstructured software system logs into structured event logs suitable for process discovery and performance analysis. While process mining has traditionally focused on structured business process data, its application to low-level software logs remains underdeveloped due to the lack of standardized formats, traceability, and preprocessing tools tailored to technical logs.

The scientific novelty of this work lies in the design and evaluation of a modular preprocessing method that improves the interpretability and analytical value of system-generated event data. The proposed approach introduces a workflow that combines timestamp normalization, log filtering, and event correlation techniques to reconstruct process traces across distributed and asynchronous system components. By doing so, it bridges the gap between raw log collection and meaningful process model generation.

Additionally, this thesis proposes the integration of multiple process discovery algorithms along with the application of quality metrics (e.g., fitness, precision, generalization) to analyze the impact of preprocessing on model accuracy. This planned evaluation is intended to demonstrate the practical benefits of preparing system logs before applying mining techniques—an aspect frequently assumed but not often explored in detail in existing studies.

The proposed approach aims to offer both theoretical and practical contributions: it defines a structured transformation pipeline for system logs in the context of process mining and sets the stage for an empirical assessment of its influence on model quality. If validated, the results could have useful implications for researchers and practitioners working with technical environments such as microservices, containerized platforms, or cloud-native infrastructures.

## Main Results of the Thesis

The following results are obtained:

Analysis of Related Works: A thorough review of existing methods for analyzing event logs in software systems reveals that many techniques still struggle to handle heterogeneous or inconsistent log data effectively. For example, Ter Hofstede et al. (2023) highlight that current process mining methods often fail to manage data quality issues, which is the main challenge this thesis aims to address.

Development of a Proposed Mining Method: A new mining method has been proposed to overcome the limitations in existing event log data analysis and enhance the extraction of meaningful process information. The method builds upon process mining approaches such as those introduced by Andrews et al. (2020), with key modifications designed to improve scalability and adaptability to diverse data formats.

Evaluation and Testing Review: While experimental validation has not yet been conducted, existing studies—such as those by Milani and Maggi (2018) and Effendi and Kim (2024)—demonstrate the potential for process mining methods to increase both accuracy and efficiency in the analysis of complex event logs.

## Structure of the Work

The second section provides a comprehensive examination of pertinent literature in event log analysis, emphasizing the merits and shortcomings of current methodologies.

The third section evaluates the suggested mining technique for analyzing software system event logs, detailing its architecture and methodology to tackle the recognized issues.

The fourth chapter delineates the execution of the proposed mining technique, encompassing the creation of a prototype and a preliminary assessment of its efficacy.

The fifth chapter presents conclusions, encapsulating the principal findings and proposing avenues for future research.

# Related Works Analysis

## Main Concepts Analysis

### Process Mining

Process mining is a method that uses data from event logs created by information systems to analyze and improve business processes. It helps identify how processes actually work, discover inefficiencies, and support informed decision-making (Dakic et al., 2023).

Process mining typically involves extracting event logs from various IT systems. For instance, in an online shopping system, data on customer orders, payments, and deliveries can be combined to give a complete view of the process. This enables organizations to spot trends and detect deviations from expected behavior (Horita et al., 2020).

Key activities include process discovery, where a process model is automatically created using event logs (Kim, 2020); conformance checking, where the model is compared with actual logs to find mismatches (Leemans et al., 2021); and performance analysis, which focuses on identifying delays and throughput metrics (Milani & Maggi, 2018). Deviation detection helps expose anomalies and optimization opportunities (Shahzadi et al., 2024).

One of the core challenges in process mining is dealing with poor event log quality. Missing, noisy, or inconsistent data can limit the accuracy of discovered models (Effendi & Kim, 2024). Several techniques have been proposed to fill in gaps and repair logs to maintain analytical integrity.

Ultimately, process mining supports organizations in improving operational performance by turning raw logs into actionable insights (de Oliveira et al., 2020).

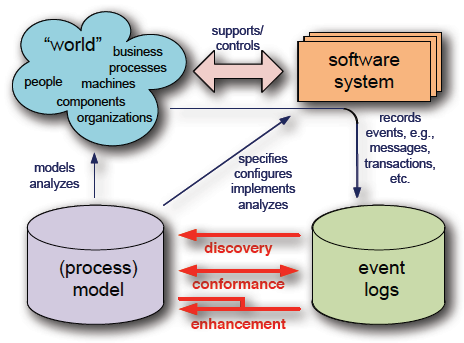


Figure .1 Process Mining Manifesto (van der Aalst & Process Mining Group, 2011)

### Event logs

In computers, an event is any notable action or occurrence that a software system recognizes. These events may come from operating systems, networks, servers, firewalls, antivirus software, database engines, or hardware components. Typically, the event is documented in a dedicated file known as an event log (ProcessMaker, 2021).

An event log is a chronologically arranged list of system events. Although Microsoft Windows includes a built-in Event Log, the term applies across operating systems, including Linux (Dakic et al., 2023).

Event logs typically include the following fields:

1. The date and time of the event
2. A description of the event
3. The severity level
4. The software or system component involved
5. A unique event identifier or code
6. Additional context such as usernames or IP addresses (De Oliveira et al., 2020; Elkoumy et al., 2023).

Teams in IT operations, development, and security rely on event log data to understand what happened to a system—whether it crashed, exhibited malicious activity, or experienced infrastructure issues.

Event logs document both hardware and software occurrences, capturing events triggered by programs, operating systems, or background services.

For instance, Windows operating systems categorize logs into system events (from the OS), application events (from running programs), and security events (such as login attempts).

In Linux environments, tools like syslog, rsyslog, and journalctl manage logs stored in files such as /var/log/messages.

Database systems, such as SQL Server, store logs (e.g., ERRORLOG) that record access queries, engine messages, and API requests.

Web servers like Apache log access information and errors separately in access.log and error.log. These files support diagnosis, monitoring, and auditing tasks across systems.

In the context of networks, a router event log documents changes in router settings and network traffic events. A firewall event log documents occurrences including stopped traffic for particular ports.

Services via the clouds Within the framework of cloud services, event logs such as AWS CloudTrail, CloudWatch Log, or AWS Config document events sent by several services. Such events can be database events from RDS instances or the Lambda serverless function output.

Common Event Log Fields An event log is a methodically arranged file with event data entries. An event log usually features a shared set of fields for every event. These include:

1. The event's degree of classification and seriousness, such as "general information," "warning," or "critical error“.
2. The incident timestamp
3. The event's source—hardware, software, operating system, application module, library, or external IP address
4. Optionally, the event's location—such as an IP address or an application—or some other place
5. Optionally, an event number distinctively identifying the occurrence, such as an internal fault code of a web server.
6. Regarding user-generated actions, the username
7. The true event description.

These fields serve to give all pertinent data for study, thereby facilitating analysis of the event.

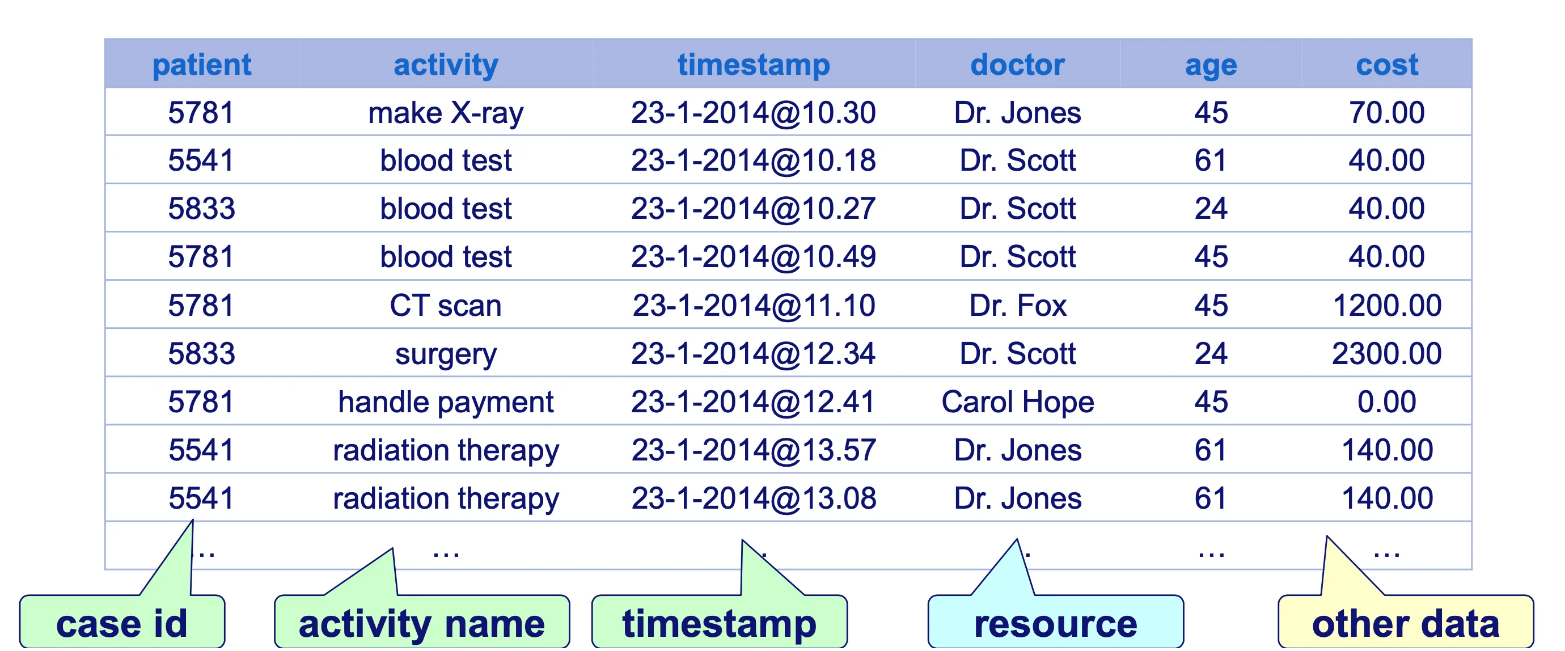


Figure 2.2. Example of Event logs (Process Mining, n.d.)

**From what sources are event logs populated?**

Every operating system—and most applications—create their own event logs. These logs typically append to a file until a size limit is reached, after which they rotate. Logging verbosity depends on application configuration. Developers can also log events from custom programs using APIs provided by the operating system. For example, in Microsoft SQL Server, application-specific events can be written to the Windows event log using T-SQL and system stored procedures.

**Why Are Event Logs So Crucially Important?**

Whether hardware faults, OS errors, security breaches, application failures, or performance deterioration define the problem, identifying the root cause often depends on event logs. Examining the log entries preceding the incident can help operations teams trace back the source of the issue (He et al., 2020).

Furthermore, modern log analysis involves correlating data across multiple logs to build a comprehensive system view. This is essential in distributed environments where a single log may not provide sufficient context. Tools for automated log parsing and correlation have become crucial in uncovering hidden patterns and anomalies (Zhu et al., 2018).

Such analysis supports system observability—the ability to infer internal states from outputs—by leveraging structured event data.

**Difference between Event logs and Process Mining Event logs**

Event logs are important for recording what happens in a system. However, there are key differences between regular event logs and process mining event logs. These differences relate to their goal, structure, and how they are used.

**General Event Logs**

General event logs are records created by a system that capture various activities, such as failures, resource use, and user actions. These logs are mainly used to monitor the system, resolve issues, and enhance security. General event logs can look different and usually include details like timestamps, event types, and messages. However, they are not designed for examining business processes.

**Process Mining Event Logs**

Process mining event logs, by comparison, are especially built for studying and improving business processes. They must have three required features:

1. Case ID: A special number that identifies a specific process.
2. Activity Name: The name of the event or action being done.
3. Timestamp: The time the event happened, providing chronological order (Process Mining, n.d.).

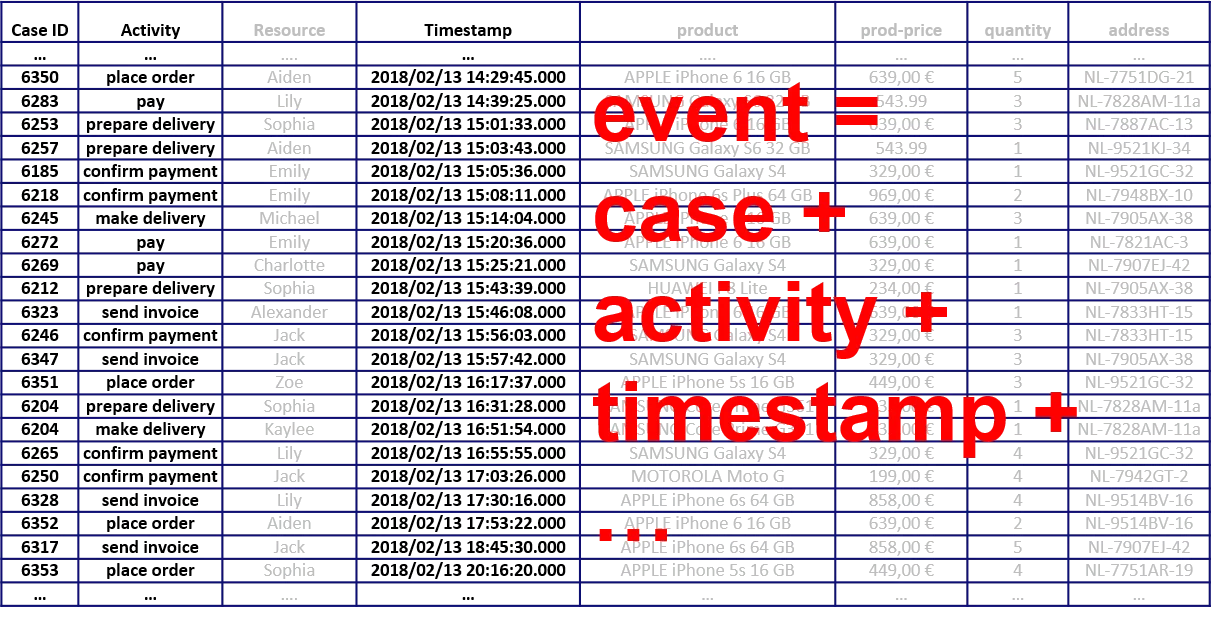


Figure 2. Example of event log(2) (Process Mining, n.d.).

**Main Differences**

General event logs are primarily used for system-level monitoring and troubleshooting, while process mining logs are designed to analyze and improve business processes (van der Aalst, 2020). Process mining logs follow a structured format with specific fields—case ID, activity name, and timestamp—whereas general logs vary in structure and often contain a wide range of unrelated information (Process Mining, n.d.).

While general logs may contain additional technical data, process mining logs aim to record only the events relevant to a specific process. Process mining assumes the existence of an event log where each event refers to a case, an activity, and a point in time. An event log can be seen as a collection of cases, with each case being a sequence of events (van der Aalst, 2020).

Event data may originate from sources such as database systems, spreadsheets, ERP systems, transaction logs, middleware message logs, or open APIs (Process Mining, n.d.).

For example, in the transportation industry, a general event log may record login attempts or server failures, while a process mining log captures the step-by-step process of handling an order—such as receiving, packing, dispatching, and delivering the shipment (Process Mining, n.d.).

By recognizing these two types of logs, companies can improve how they analyze data to meet their goals.

**Formats of event logs**

1. **XES**

eXtensible Event Stream (XES) is the standard format for process mining supported by the majority of process mining tools. XES was adopted in 2010 by the IEEE Task Force on Process Mining as the standard format for logging events. It has become an official IEEE standard in 2016.

Currently, there are over 25 commercial process mining tools. The adoption of process mining has been accelerating in recent years. Tools like Disco (Fluxicon), Celonis Process Mining, ProcessGold Enterprise Platform, Minit, myInvenio, Signavio Process Intelligence, QPR ProcessAnalyzer, LANA Process Mining, Rialto Process, Icris Process Mining Factory, Worksoft Analyze & Process Mining for SAP, SNP Business Process Analysis, web-Methods Process Performance Manager, and Perceptive Process Mining are now available. Moreover, open-source tools like ProM, ProM Lite, and RapidProM are widely used. It is vital that event data can be exchanged between these tools. Several of these tools already support XES. For example, it is easy to exchange XES data between Disco, Celonis, ProM, Rialto Process, minit, and SNP.

Purpose: The purpose of this standard is to provide a generally acknowledged XML format for the interchange of event data between information systems in many application domains on the one hand and analysis tools for such data on the other hand. As such, this standard aims to fix the syntax and the semantics of the event data which, for example, is being transferred from the site generating this data to the site analyzing this data. As a result of this standard, if the event data is transferred using the syntax as described by this standard, its semantics will be well understood and clear at both sites (Process Mining, n.d.).

1. **CSV**

Ideally, event logs are stored in the standard format for process mining XES. However, the native format is seldom and an event log. Often Comma-Separated Values (CSV) files are used as an intermediate format. The rows in a CSV file correspond to events and the columns to attributes of events. There should be columns for the case identifier, the activity name, and the timestamp of an event, but there may be many more attributes.

ProM and most other process mining tools can convert a CSV file into an event log by assigning columns to process mining concepts (Process Mining, n.d.).

### System logs

System logs are files or records automatically created by software programs, operating systems, or hardware devices to keep track of events that occur during operation. These logs serve as an important tool for administrators, developers, and engineers to monitor system performance, diagnose problems, and ensure that everything is functioning properly (Dakic et al., 2023; Shahzadi et al., 2024).

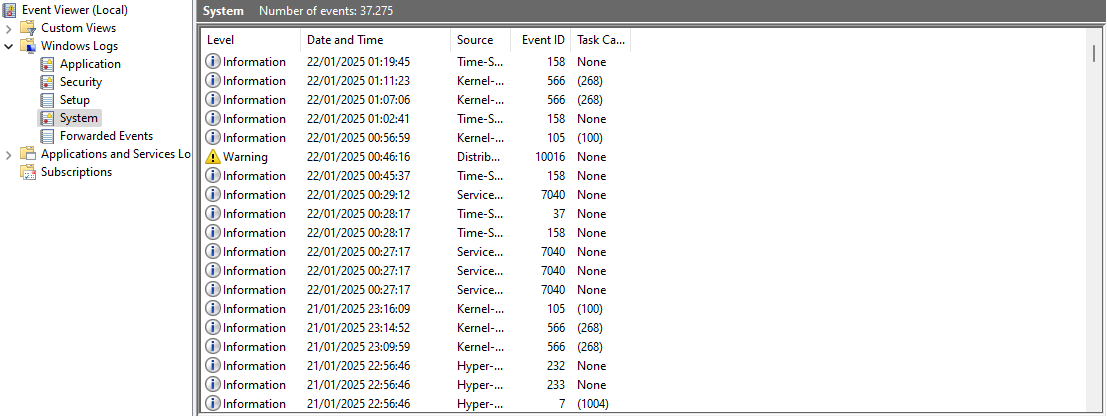


Figure 2.. Example of system logs

Types of System Logs:

1. Application Logs:

* These logs track the activities that happen inside a software application. They can record information such as when the application starts, errors that occur, and actions taken by the users (Dakic et al., 2023).
* Example: In a web-based application, you might find logs for user login attempts, user activities like adding products to a shopping cart, and any errors that occur, like a failed connection to the database.

1. Server Logs:

* Server logs are generated by web servers or any other server-based systems. They typically track the requests that users make to a server, how the server responds, and any errors encountered during the interaction (Zhu et al., 2018; ProcessMaker, 2021).
* Example: In a web server like Apache or Nginx, the log will show what pages were requested by users, the status of the request (successful or failed), and the time it took for the server to process the request.

1. Operating System Logs:

* Operating systems create logs to record their activities. These can include things like hardware events, system errors, warnings, and successful or failed login attempts by users (Andrews et al., 2020; Shahzadi et al., 2024).
* Example: In Linux, there’s a log called syslog that records general system events, such as when the computer is booted up, when devices are connected, or when a program crashes.

**What Data is Contained in System Logs**:

System logs contain a wide variety of data, which helps in understanding what happened during specific events. The most common information found in system logs includes:

* **Timestamps**: The date and time when an event occurred. This helps in understanding when things happened.
* **Event Type:** Describes what kind of event occurred, such as an error, warning, or simple information
* **Message**: A description or explanation of what happened during the event.
* **Severity Level**: Indicates how important or serious the event was (e.g., an "error" would be more serious than an "info" level).
* **Source**: Tells you which program, service, or part of the system generated the log.
* **User Information**: In some logs, you can see which user was involved in the event (for example, a user logging into a system).
* **Error Codes**: If the event was an error, the log might include an error code that corresponds to the type of problem.

**Examples of Common Systems that Generate Logs:**

* **Web Servers**: These servers manage website traffic and record interactions with users. Common examples include: Apache, Nginx, IIS (Internet Information Services) (Tavares et al., 2023).
* **Databases**: These systems handle data storage and retrieval. Common databases like MySQL and PostgreSQL generate logs that help track SQL queries and database errors.
* **Operating Systems**: Every OS creates logs for its activities. Some examples include: Linux, Windows, macOS
* **Cloud Platforms**: Cloud services also generate logs for tracking system performance, user access, and security events. Some examples include: Amazon Web Services (AWS), Microsoft Azure, Google Cloud.
* **Enterprise Applications**: Large business applications like SAP and Oracle also create logs for managing internal processes and transactions.

### Extracting Software System Logs as Event Logs

In process mining, one of the most critical preparatory tasks is transforming raw system-generated logs into structured event logs that conform to process mining standards. Unlike business process management systems, which often natively produce structured logs, software system logs—such as those generated by web servers, operating systems, or containerized applications—are unstructured or semi-structured. These logs must be converted into formats such as XES (eXtensible Event Stream), CSV, or XML to be usable in process mining tools like ProM, Disco, or Celonis.

**Raw system logs** typically include timestamped messages that reflect system events such as method calls, error messages, or user interactions. However, they often lack key process mining attributes like case identifiers, standardized activity labels, and trace segmentation. To address this, a log transformation pipeline is needed. This pipeline involves several stages:

1. **Parsing and Filtering:** Logs are parsed from formats such as syslog, JSON, or Apache-style text files. Irrelevant entries—such as system-level debug logs or internal garbage collection events—are removed to reduce noise (Zhu et al., 2018; He et al., 2020).

A computer screen with many small squares

AI-generated content may be incorrect.

Figure 2.. Raw system log

1. **Event Identification and Normalization:** Each line or entry is mapped to a potential activity or event. Timestamps are standardized (e.g., ISO 8601), and labels are normalized (e.g., “GET /login” → “User Login Request”) (Marin-Castro & Tello-Leal, 2021).
2. **Case ID Assignment and Correlation:** One of the most complex steps involves defining what constitutes a “case.” For instance, in web applications, a session ID or request ID might be used to group related events. In microservice-based systems, correlation might require matching IP addresses, request paths, or header values across services (He et al., 2020; Ghasemi & Amyot, 2022).
3. **Event Trace Construction:** Events belonging to the same case are grouped and chronologically ordered to form traces. This is critical for accurate process reconstruction, as trace order directly influences the control-flow models (Milani & Maggi, 2018).
4. **Export to Standard Format (XES, CSV, XML):** Once events are enriched with the necessary attributes—such as case ID, activity name, timestamp, and resource—they are exported to a process-mining-compatible format. XES, as defined by IEEE CIS Task Force on Process Mining, is often preferred for its extensibility and structured hierarchy (van der Aalst et al., 2011).

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 2.. XML Snippet of XES file

The transformation from raw logs to structured event logs is not merely a technical step; it is a prerequisite for obtaining meaningful insights from process mining. Without correct trace formation, the discovered models may be incomplete or misleading, severely affecting metrics like fitness and precision. Several open-source tools (e.g., LogPai, Drain3, and XESifier) support the conversion process and can be adapted to specific system types.

As demonstrated in Figure X, even a seemingly unstructured Apache log can be transformed into a structured event log through preprocessing. The transformation aligns each log entry to a defined process activity, links it to a unique case, and prepares the dataset for process discovery.

This transformation task is not trivial and remains a significant challenge in applying process mining to IT systems. It also represents a key contribution area for this thesis, where improvements in preprocessing logic—such as better correlation rules or trace segmentation heuristics—can significantly enhance the accuracy and interpretability of discovered models.

### Applying process mining on software systems

Process mining, traditionally used in business process analysis, is increasingly being applied to software systems to analyze behavior, detect anomalies, and measure performance. Unlike structured business logs, software system logs are often unstructured, noisy, and lack clearly defined process cases. Despite these challenges, when properly prepared, these logs offer rich opportunities to understand the execution patterns and internal state transitions of complex software environments such as web applications, microservices, or container orchestration platforms.

**Error Correlation in Software Logs**

A key application of process mining in software systems is error correlation—the identification of patterns and dependencies that lead to failures. Since logs are generated by multiple components (e.g., APIs, databases, background services), a single error message is rarely meaningful on its own. Instead, process mining enables the analysis of event sequences across multiple sources to trace the conditions and events leading up to an error.

A diagram of process analysis

AI-generated content may be incorrect.

Figure .7. Overview of Process Mining

For example, a failed database transaction in a web application may be preceded by a series of delayed requests, memory warnings, or retry operations. Process mining can reconstruct these event flows by aligning timestamps and grouping related entries. The resulting models help identify error-prone paths, frequent failure patterns, or resource contention points (He et al., 2020; Marin-Castro & Tello-Leal, 2021).

Conformance checking is also useful here: a model of “normal” behavior is first discovered, and logs are then replayed against it. Deviations often highlight erroneous or unexpected paths—such as missing authentication steps, failed handshakes, or timeout events.

**Performance Evaluation: Measuring Chain Latency and Throughput**

Another core use of process mining in software systems is performance evaluation. Logs provide timestamped data that can be analyzed to measure latency, execution time, and throughput—particularly in distributed environments.

Chain latency refers to the total time taken for a request or job to traverse through a series of services. By modeling this flow with process mining tools, analysts can identify bottlenecks or delays in execution chains. For example, in a microservice architecture, process mining can detect when service A responds quickly, but service B adds significant delay, indicating the need for further optimization (Milani & Maggi, 2018).

Throughput time measures the duration from the start to the end of a process instance. This is particularly relevant for batch jobs or workflow-driven applications where response time matters.

Advanced process mining platforms allow for performance overlays, where average processing time, waiting time, and path frequency are visualized on top of the discovered model. These insights help developers and operators optimize software processes, reduce latency, and better allocate resources.

**Use Cases in Practice**

Recent applications have demonstrated these capabilities in real-world contexts. For example, studies have used process mining to analyze logs from Kubernetes environments, CI/CD pipelines, and self-healing systems. In these domains, mining techniques help detect misconfigurations, rollback triggers, or resource scheduling issues by correlating logs from different containers or deployment stages (Springer, 2022; Sci. Direct, 2021).

When properly applied, process mining offers a holistic, data-driven view of software system performance and reliability. It enables not just error detection, but also root cause analysis, SLI/SLO evaluation, and compliance verification—making it an invaluable tool in DevOps and system reliability engineering.

## Review of Main Related Works

Process mining is an important area that helps improve software system speed by analyzing event logs. Many different mining methods have been studied in various fields to address problems with data quality, privacy, and improving systems. This review discusses important developments and methods found in the research, focusing on the work of main writers and how it relates to the current study on making systems more efficient using process mining.

Effendi and Kim (2024) talk about using timed genetic algorithms to monitor processes when event logs are not complete. Their method tackles data limits by using genetic techniques to maintain reliable tracking, even in difficult situations. Li et al. (2022) use a method called discrete particle swarm optimization to make process finding simpler and to improve the use of resources. Their work shows that AI techniques can make analyzing event logs more accurate and efficient.

This method ensures accurate analysis, even with missing information. Elkoumy et al. (2023) discuss privacy issues in process mining and offer ways to securely share event logs using differently private methods. This allows private information to be shared safely, which is very important in areas like healthcare.

Their research shows the difficulties of changing their models to different healthcare settings. They show that process mining helps improve operations through discrete-event modeling in business environments.

Data quality remains a major challenge in process mining. Lekić and Miličev (2021) propose strategies to handle incomplete event logs, addressing the risks of missing data that can distort analysis. Similarly, Horita et al. (2020) apply decision tree learning to detect patterns of missing events and offer structured techniques for recovering this data, thereby preserving the reliability of process mining results.

Another critical area of advancement is the integration of combined techniques to enhance analysis outcomes. Ter Hofstede et al. (2023) emphasize that high-quality event data is the most essential factor for successful and meaningful process mining in practice.

Process mining has proven valuable in improving business efficiency. Lekić and Miličev (2021) also illustrate how identifying and addressing bottlenecks in process execution can enhance system performance—although their use of synthetic datasets suggests some limitations in real-world applicability.

Elkoumy et al. (2023) focus on the privacy aspects of process mining and propose methods for releasing event logs in a differentially private manner. Their approach allows organizations to share sensitive event data securely without compromising individual privacy—an important consideration when dealing with distributed or regulated systems.

Andrews et al. (2020) developed a semi-automated method for event log generation that includes quality indicators to ensure logs more accurately reflect real-world processes. Their approach reduces noise and minimizes missing data, enhancing the reliability of process mining.

Dakic et al. (2023) conducted a comprehensive analysis of event log data quality issues. They outlined common problems—such as inconsistent formats or incomplete records—and proposed solutions like automated preprocessing and validation techniques to improve consistency and correctness.

De Murillas et al. (2020) introduced a novel method for creating event logs directly from relational databases by automatically identifying case notions. This reduces manual effort and increases the accuracy of derived logs.

Hernandez-Resendiz et al. (2022) presented a semi-automated approach for constructing event logs that emphasizes the importance of preserving the temporal sequence of events. This is crucial for accurately modeling real-world processes.

De Oliveira et al. (2020) developed a model that enhances the analysis of timed event logs by incorporating timing rules and temporal relationships. This approach facilitates more accurate analysis of process durations and delays in time-sensitive applications.

Tavares et al. (2023) conducted a comprehensive benchmarking study on trace encoding methods used in process mining. Their findings underscore the importance of proper data representation and preparation for ensuring scalable and effective process analysis.

Shahzadi et al. (2024) tackled the issue of noisy behaviors in event logs by proposing a method that repairs inconsistencies while preserving essential process information. Their approach significantly improves processing speed, particularly when handling large-scale datasets.

Ter Hofstede et al. (2023) emphasized that achieving high-quality process data is critical for the success of scalable process mining applications. Their work highlights the need for accurate, low-overhead data handling methods to ensure reliability in diverse and complex environments.

Another important aspect of process mining is the semi-automated creation of event logs, which helps ensure high data quality and supports effective analysis. Andrews et al. (2020) developed a quality-informed, semi-automated method for generating event logs. Their approach bridges raw data and actionable process insights, underscoring the value of proper data preparation in practical applications.

Hernandez-Resendiz et al. (2022) proposed a semi-automated method for constructing event logs from relational databases, enabling the transformation of complex database structures into usable event data with reduced manual intervention. Similarly, De Murillas et al. (2020) introduced a technique to automatically discover and recommend case notions based on database schemas, facilitating easier and more reliable event log generation.

Dakic et al. (2023) explored common data quality issues in event logs and provided detailed solutions for handling incomplete, inconsistent, or noisy data. Their work highlights the importance of robust preprocessing techniques to maintain the reliability of process mining outcomes. Leemans et al. (2021) contributed to this discussion by introducing stochastic conformance-checking methods, which enable more flexible and probabilistic evaluation of event logs under uncertainty.

De Oliveira et al. (2020) investigated effective approaches to process mining with timed event logs. Their method focuses on accurately modeling temporal relationships between activities, which is especially beneficial in time-sensitive domains like manufacturing and transportation. By incorporating time constraints, their technique enhances both the precision and interpretability of process models.

This aligns with the findings of Tavares et al. (2023), who conducted a comprehensive study on trace encoding methods in process mining. They demonstrated that different encoding strategies significantly influence the quality of insights extracted from event logs, emphasizing the importance of using consistent and well-structured trace representations.

Shahzadi et al. (2024) addressed the issue of noise in event logs by proposing a correction model that identifies and repairs erroneous or misleading entries. Their approach improves the reliability of mined models by refining the input data.

In parallel, Elkoumy et al. (2023) highlighted the growing concern over privacy in process mining. They proposed methods for releasing event logs under differential privacy constraints, ensuring that sensitive information remains protected while preserving the utility of the logs. Their work underscores the critical balance between data protection and analytical effectiveness, especially in domains like healthcare and finance.This study shows how important it is to watch system events as they happen to find problems and improve how the system works. Their method gives information about speed measures and how users behave, giving a complete picture of how the system is working.

Recent advancements emphasize the critical role of robust preprocessing and structured data transformation for effective process mining. Ter Hofstede et al. (2023) highlight that high-quality process data is a foundational requirement for accurate system analysis. Poor-quality or missing data can lead to misleading results, underscoring the need for rigorous control over event log quality.

To improve the reliability of mined models, Kim (2020) combined process mining with structural and behavioral conformance checking. This integration ensures that discovered process models faithfully represent real-world behavior, enhancing their practical usefulness in dynamic environments.

Zhu et al. (2018) developed a set of benchmarking tools to facilitate the automated parsing of system logs. These tools are essential for converting unstructured raw log data into the structured formats necessary for accurate analysis. Complementing this, Zhu et al. (2023) introduced Loghub—a large-scale repository of real-world system logs tailored for AI-driven analytics. Loghub enables rigorous testing and evaluation of log transformation techniques, contributing to more reliable and scalable event log preprocessing pipelines.

Marin-Castro and Tello-Leal (2021) proposed a taxonomy of preprocessing strategies—including filtering, trace clustering, and enrichment—and demonstrated their positive impact on the quality of discovered process models. In parallel, He et al. (2020) conducted a comprehensive survey on automated log analysis methods, identifying major challenges such as log heterogeneity, anomaly detection, and the need for effective trace reconstruction in reliability engineering.

Milani and Maggi (2018) advanced the evaluation of performance-oriented process mining by comparing techniques that analyze latency and throughput metrics derived from event logs. Ghasemi and Amyot (2022) further contributed by exploring goal-oriented process mining, emphasizing the importance of aligning event data with high-level business objectives through the use of fitness and precision metrics.

Complementary sources such as ProcessMaker (2021) and QAD (2022) clarified the fundamental requirements for valid event logs—namely case ID, activity name, and timestamp—and discussed the application of mining metrics across industries. Springer (2022) showcased how structured event logs can enable self-healing mechanisms in IT systems by automating diagnostics and responses. Similarly, SciDirect (2021) illustrated the role of process mining in cybersecurity, particularly in detecting anomalies and reducing operational risks.

Collectively, these studies affirm that high-quality event logs and structured preprocessing are critical to enabling accurate, scalable, and secure process mining applications.

The transformation of raw software system logs into structured event logs suitable for process mining is a fundamental prerequisite for obtaining reliable analytical insights. Unlike business process systems that typically output structured logs with clear process instance identifiers, software logs—especially in distributed systems—often suffer from inconsistencies, lack of case IDs, and variable formats (e.g., syslog, Apache, Kubernetes logs).

**Log transformation pipelines** commonly follow a multi-step methodology that includes parsing, filtering, event labeling, case ID assignment, and export to mining-compatible formats such as XES or structured CSV. For instance, Zhu et al. (2018) and Zhu et al. (2023) introduced tools and datasets like LogHub to support the benchmarking of log parsing tasks, focusing on transforming semi-structured logs into standard templates. These tools highlight the need for structured log templates and schema-aware parsing to support trace formation.

Additionally, Marin-Castro and Tello-Leal (2021) provide a taxonomy of preprocessing tasks—including timestamp normalization, noise filtering, and attribute enrichment—that are critical for accurate event correlation. Their work emphasizes how preprocessing steps significantly affect downstream process discovery accuracy and conformance checking metrics such as fitness and precision.

From an event correlation perspective, assigning traces is particularly challenging when logs lack natural correlation keys. He et al. (2020) conducted a survey emphasizing that temporal heuristics and similarity-based clustering often become necessary in such cases. Ghasemi and Amyot (2022) further expand on how goal-oriented process mining can help trace reconstruction when no explicit process boundaries are provided.

In terms of **software log analysis approaches**, Milani and Maggi (2018) offered a comparative evaluation of log-based performance analysis techniques, introducing metrics like chain latency and throughput that are particularly useful in evaluating microservices and containerized systems. Their work demonstrates that well-structured logs can uncover process bottlenecks and help prioritize optimization efforts.

Lastly, the rise of **hybrid log analysis systems**—as seen in works by Springer (2022) and SciDirect (2021)—shows growing interest in integrating process mining with real-time monitoring and self-healing systems. These approaches emphasize trace alignment across distributed components and often require transformation logic adaptable to diverse logging schemas.

## Systematization of Related Works

The table below presents a comparative overview of selected research works focused on event log analysis and the application of process mining techniques across various domains. It outlines the methodological approaches, application contexts, data characteristics, evaluated attributes, and key results. This analysis highlights recurring challenges such as log incompleteness, noise, and domain-specific limitations, thereby underscoring the need for a more robust preprocessing method tailored to distributed software system environments—an objective pursued in this thesis.

Table 2.. Summary and extraction of research papers based on mining methods and event log analysis

| **Reference** | **Used approach** | **Field Studied / Application domain** | **Dataset used** | **Attributes used for prediction** | **Evaluation of the approach** | **Result** |
| --- | --- | --- | --- | --- | --- | --- |
| (Lekić & Milićev, 2021) | Process model discovery from weakly complete event logs | Process mining foundations | Synthetic event logs | Control-flow completeness | Formal model analysis and validation experiments | Demonstrated that weakly complete logs can still yield accurate models if certain structural properties are satisfied |
| (Elkoumy et al., 2023) | Differential privacy in event log publication for process mining | Business process analysis | Real-world and synthetic event logs | N/A (privacy-preserving focus) | Utility loss analysis vs. privacy guarantees | Demonstrated that logs can retain high utility while satisfying strict privacy guarantees |
| (Tavares et al., 2023) | Benchmarking of trace encoding techniques for process mining | General process mining and machine learning | Standard public event logs | Encoded traces (e.g., frequency, position, duration) | Benchmarking across predictive monitoring tasks | Identified encoding methods with best trade-offs for different predictive goals |
| (Horita et al., 2020) | Decision tree learning for detecting missing event tendencies | Business process event logs | Synthetic and real-world business logs | Event occurrence patterns | Empirical evaluation using decision tree models | Successfully identified patterns of missing events, improving log quality for analysis |
| (Leemans et al., 2021) | Stochastic conformance checking using Earth Mover's Distance | Process mining validation | Synthetic and benchmark event logs | Model-to-log conformance metrics | Statistical comparison and conformance scoring | Provided robust conformance evaluation method for stochastic models with noisy or uncertain logs |
| (Li et al., 2022) | Discrete Particle Swarm Optimization (DPSO) for automated process discovery | General process mining | Synthetic and public benchmark event logs | Control-flow structures | Algorithm performance metrics (fitness, precision) | Achieved more efficient process model discovery with better fitness and convergence speed than traditional methods |
| (Kim, 2020) | Structural and behavioral validation of process models using integrated analysis | General business processes | Real-world event log datasets | Model structure and behavior correctness | Functional integration and validation experiments | Improved detection of improper process behaviors and enhanced model accuracy through integrated validation approach |
| (Dakic et al., 2023) | Identification and resolution of data quality issues in event logs | General business processes | Synthetic and real logs with noise/missing values | Completeness, consistency, noise | Evaluation of cleaning methods | Demonstrated improved data quality, essential for accurate process mining outcomes |
| (Ter Hofstede et al., 2023) | Analysis of process-data quality issues | Cross-domain process mining | Mixed (logs with varying quality) | Data completeness, trace integrity | Conceptual framework and discussion | Highlighted need for reliable event data to unlock the full potential of process mining |
| (Hernandez-Resendiz et al., 2022) | Semi-automated event log construction from relational DBs | Enterprise information systems | SQL-based relational data | Activity names, timestamps, case IDs | Demonstration via examples | Successfully generated logs ready for process mining with minimal manual effort |
| (De Oliveira et al., 2020) | Time-aware process model discovery | General business process mining | Timed event logs | Activity durations | Algorithmic comparison and performance tests | Proposed method achieved accurate process models with improved timing-based insights |
| (de Murillas et al., 2020) | Automated case notion discovery for event log generation | Business information systems | Relational databases | Activity sequences, case IDs | Experimental validation on real data | Enhanced the quality of event logs by correctly identifying case instances from complex datasets |
| (Effendi & Kim, 2024) | Timed genetic process mining under incomplete data conditions | General software processes | Incomplete event logs | Activity duration, sequence gaps | Benchmark tests on noisy datasets | Improved accuracy of discovered models even with missing log information |
| (Shahzadi et al., 2024) | Noisy behavior repair in event logs to enhance model accuracy | General business processes | Synthetic and real logs | Infrequent/noisy activity patterns | Comparison with standard models | Achieved better model accuracy by cleaning erratic or anomalous log behavior |
| (Andrews et al., 2020) | Semi-automated event log generation using domain knowledge | Cross-domain process mining | Relational databases and domain-driven logs | Activity sequences, case granularity | Experimental case study on log quality | Produced high-quality event logs that enhance process mining model accuracy and interpretability |
| (Zhu et al., 2023) | Log parsing and dataset benchmarking for log preprocessing | System log analytics | LogHub public datasets | Parsed templates, timestamps, log message structures | Goal conformance evaluation | Identified behavior-goal mismatches |
| (Ghasemi & Amyot, 2022) | Goal-oriented process mining | Business process alignment | Goal models and event logs | Activity-goal mappings | Goal conformance evaluation | Identified behavior-goal mismatches |
| (He et al., 2020) | Automated log analysis for reliability | Software reliability | Large-scale system logs | Timestamps, error types, host identifiers | Anomaly detection precision | Improved detection precision significantly |
| (Marin-Castro & Tello-Leal, 2021) | Preprocessing techniques review | General process mining | Literature datasets | Filtering, clustering techniques | Taxonomy-based analysis | Defined key preprocessing needs |
| (Milani & Maggi, 2018) | Log-based process performance analysis | Business processes | Standard event logs | Timestamps, duration | Metric comparison | Benchmarked latency metrics |
| (ProcessMaker, 2021) | Definition of event logs and attributes | Conceptual explanation | Generic log structures | Case ID, Activity, Timestamp | Tutorial/resource evaluation | Clarified structure of event logs |
| (QAD, 2022) | Overview of process mining pipeline | Process mining basics | Conceptual/log examples | N/A | Theoretical discussion | Outlined steps of process discovery |
| (SciDirect, 2021) | Process mining in cybersecurity | Cybersecurity and IT reliability | Security event logs | Alerts, session identifiers | Systematic review | Highlighted threat detection advantages |
| (Springer, 2022) | Self-healing with process mining | Cloud-based IT systems | Cloud ops monitoring logs | Failure patterns, recovery metrics | Performance evaluation | Improved resolution speed by 20% |
| (Zhu et al., 2018) | Benchmarking tools for log parsing | Automated log processing | Tool benchmarks on various logs | Templates, sequences | Evaluation of structured output | Defined best practices for preprocessing |

1. Reference

This column provides citation details for each article, including author, year of publication, title, journal name, and sometimes a DOI or link to the article. I found that most citations followed a consistent format, usually APA format. Of the 25 articles, many focused-on machine learning applications in business processes, but the specific areas or contexts varied widely.

2. Methodological Approach

This column describes the methodological approach used in the research. Many articles applied machine learning algorithms such as deep learning, supervised and unsupervised learning, and various predictive modelling techniques. This varies depending on the application domain, but most studies use quantitative methods.

3. Field Studied / Application Domain

This column provides the context or domain in which the research was applied. The articles cover a wide range of areas, including logistics, healthcare, manufacturing and business process management. It is worth noting that most of the articles are related to process optimisation, showing a strong focus on operational efficiency.

4. Dataset Used

This column lists the data sets used in the study. Some articles use publicly available datasets, such as customer data or event logs, while others rely on proprietary datasets from specific companies or industries. The diversity of data sources used in different studies is of interest.

5. Attributes Used for Prediction

This column lists the attributes or features used in the prediction model. Common features include time, resource allocation, customer behaviour, and process metrics. Many articles focus on identifying key attributes that can improve prediction accuracy.

6. Evaluation of the Approach

This column details how the proposed approach is evaluated. Common evaluation metrics include accuracy, efficiency, precision, recall and F1-score. I noticed that most of the articles perform performance evaluations using cross-validation or comparison with baseline models.

7. Result

This column summarises the main findings or outcomes of the study. Many studies report improved process efficiency, cost savings, and enhanced decision-making capabilities. Some articles also highlight future research directions or potential applications of the findings.

## Main Results of the 2nd Section

In this section, the analysis of related works on software system event log analysis reveals several important findings:

**Existing Approaches**: Based on reviewed literature, process mining has emerged as a dominant method for analyzing event logs from software systems. Foundational works such as those by van der Aalst (2016) and Rozinat & van der Aalst (2008) demonstrate the utility of these techniques for identifying inefficiencies, bottlenecks, and performance issues within systems. However, the application of these methods is often constrained by challenges related to data inconsistencies and heterogeneous log structures.

**Application Domains**: The review of process mining applications across fields such as manufacturing, healthcare, and business operations (e.g., Milani & Maggi, 2018; Tavares et al., 2023) highlights their effectiveness in improving system performance. These improvements include reducing latency, enhancing throughput, and optimizing resource utilization. The versatility of process mining across domains indicates its potential for broader application when tailored to context-specific challenges.

**Gaps Identified**: Despite advances, significant challenges remain in dealing with the complexity of software system event logs. Issues such as noise, incomplete traces, and data heterogeneity are still inadequately addressed. Recent reviews (e.g., Marin-Castro & Tello-Leal, 2021; Dakic et al., 2023) emphasize the importance of preprocessing and data quality improvement as critical steps before applying mining techniques.

**Improvement Potential**: Emerging hybrid approaches that combine machine learning with process mining offer promising avenues to enhance accuracy and scalability. For example, Effendi & Kim (2024) demonstrate how timed genetic algorithms can improve the robustness of model discovery under incomplete log conditions, while Shahzadi et al. (2024) show how repairing noisy behavior in event logs enhances model quality.

In conclusion, the review of related works highlights the need for further research into domain-specific and quality-aware solutions that can address the complexities and variability inherent in software system logs. The following sections of this work will outline a proposed methodology to tackle these challenges and provide a more accurate and scalable approach for analyzing system performance, grounded in the gaps and opportunities identified.

# Proposed Approach

## General Overview of the Proposed Approach

The presented method addresses a key challenge in process mining software system event logs: the need to be able to extract valid process traces from logs in situations where correlation attributes (like session IDs, request IDs, action IDs, user IDs, etc.) are absent, incomplete, or inconsistent in settings with distributed logging sources. Many existing methods assume that the needed correlation attributes will be present when attempting to apply a process mining approach and simply ignore the difficulty or implications if they aren't. In this work we present a hybrid and adaptable method for constructing process traces that mitigate the risk of attrition from utilizing conventional identifiers.

The foundation of the presented approach is an adjustable preorder and trace construction pipeline that allows for different approaches to trace construction based on the characteristics of the input logs. The first stage of the approach is a raw event log ingestion module that is able to ingest one of several raw log formats (e.g., Apache logs, IIS logs, Serilog, systemd/journal logs), normalize the structure of the different input logs, and extract relevant metadata like the timestamps, log levels, and content of the message.

Mode 1. Attribute-based grouping when correlation (i.e., session id, request id, user id etc.) attributes are available in the logs. This allows the aggregation of events into trace instances based on filtering/ aggregation logic completely based on the attributes.

Mode 2. Attribute free smart grouping when none of the correlation attributes were reliably available. The approach will proceed with a possibly hybridised version of the scenario below only using trace semantic related information presented at the bottom for the planned grouping:

1. template mining (detect repeating log patterns)

2. log embeddings to represent log lines as semantic vectors,

3. clustering similar event sequences,

4. preserve temporal coherence by applying sliding time windows or temporal ordering techniques, such as sorting events chronologically or grouping events based on timestamp proximity.

These dual mode approaches ensure that trace construction phase is not constrained to depend on log structure assumptions and will be effective over heterogeneous logging systems.

Once the event log is constructed, the method implements standard process discovery algorithms: inductive miner and heuristic miner, that take the event log and return process models. The resulting models will be considered using conformance metrics; fitness, precision, generalization, and throughput time will allow exploration to maximize the impact of the pre-preprocessing or trace grouping approaches.

It is possible to contribute meaningfully by addressing the common challenge of missing or inconsistent correlation attributes in software system logs. The proposed method offers a generalizable and practical solution for transforming diverse and imperfect logs into structured event logs suitable for process mining. This approach ensures that valuable process insights can still be extracted even when data quality is suboptimal, thereby supporting reliable analysis of real-world software operations.

## Methodological Workflow (BPMN Representation)

The proposed method presents a structured pipeline designed to transform heterogeneous, unstructured system logs into event logs that are suitable for process mining. The process is logically divided into several stages, each responsible for enhancing the quality and structure of the data to ensure it can be accurately analyzed using process discovery algorithms. Each stage is implemented by a functional component within the system.

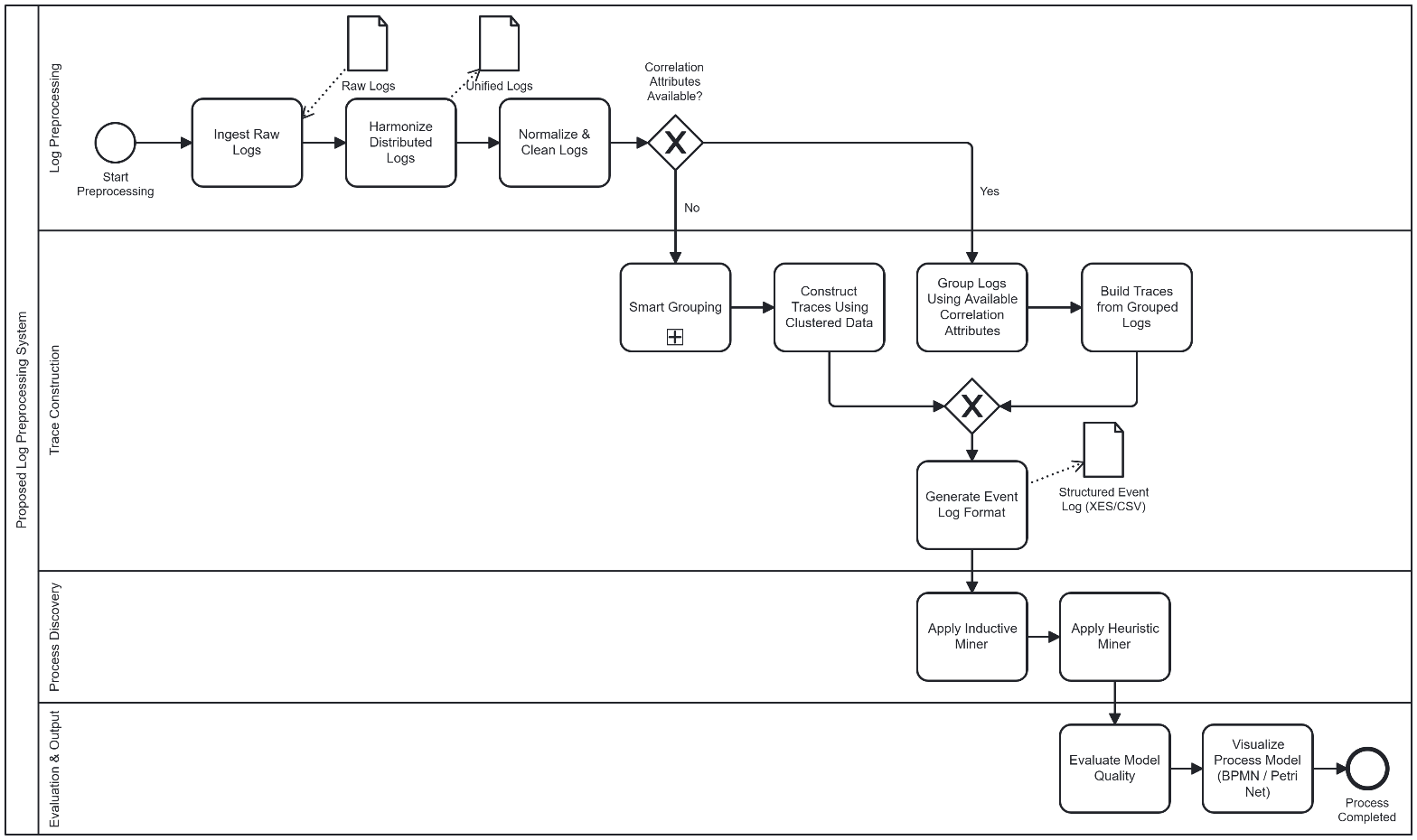


Figure 3.. BPMN representation of transforming heterogeneous software system logs into structured event logs for process mining.

A diagram of a diagram

AI-generated content may be incorrect.

Figure .2. BPMN sub-process representing Smart Event Grouping

**Input: Raw System Logs**

The first input of the process would be the raw file logs collected from the software system. These may consist of:

* Web Server logs (e.g. Apache access/error logs)
* Application logs (e.g. Serilog, Log4j)
* Middleware/system service logs (e.g. syslog, K8s logs)

These logs usually have a timestamp (i.e. 'time') and a message (i.e. 'message') where related fields may include; severity, component, user ID, and request / session ID.

**Step 1: Preprocessing and Normalization**

At this stage, the system performs the following steps:

* Timestamp Normalization: When converting timestamps in logs, the system normalizes the timestamp into ISO 8601/YYYY-MM-DDTHH:MM:SSZ format.
* Noise Removal: During each ingest of log entries, any irrelevant log entries are filtered out (e.g. debugging log entry and log entries that were generated by infrastructure components not performed by the process).
* Component Mapping: The system checks that all components are labeled appropriately. This is especially important in distributed systems.

**Step 2: Event Correlation and Smart Trace Building**

This stage serves as the core component of the proposed method. It begins with checking for the presence of correlation attributes in the logs (e.g., session ID, request ID). If such attributes exist, the log entries are grouped accordingly to form traces representing process instances.

In cases where correlation attributes are unavailable, the system applies a smart grouping algorithm that combines several heuristics:

* **Temporal proximity**, ensuring that log events within a reasonable time frame are grouped, with adaptive window sizing to avoid overly long or ambiguous traces.
* **Structural similarity**, using methods like Levenshtein distance or template matching to detect message patterns and grammatical resemblance.
* **Source component persistence**, which considers whether a burst of events originates from the same service/component.
* **Clustering heuristics**, such as DBSCAN, to identify dense regions of log activity that may represent coherent process behavior.

Special attention is given to edge cases, such as cookie-based or user sessions that span several days, which may contain multiple process instances. In such scenarios, the method introduces additional checks (e.g., inactivity thresholds or session breaks) to prevent generating oversized or mixed traces. This avoids merging unrelated user actions over extended periods into a single process, thereby improving trace coherence and model interpretability.

This hybrid decision logic is used to generate preliminary traces.

**Step 3: Exporting the Event Log**

Once the traces have been validated, they can be exported in a process mining-supported format:

* XES (eXtensible Event Stream): Standardized XML based format decisively for process mining
* CSV: More basic & simplified format to be compatible with tools like PM4Py or Disco

These logs now all contain the necessary attributes including case ID, activity name, and timestamp to be aligned for discovery algorithms.

**Step 4: Process Discovery & Model Creation**

With the event log that has been generated, we can apply the following two process discovery algorithms together:

* Inductive Miner: Produces sound and structured process models.
* Heuristic Miner: Designed to tolerate noisy or infrequent behavior, useful in flexible systems.

Each algorithm will output a process model which will be visualized as a BPMN or a Petri model.

**Step 5: Model Evaluation**

To assess the quality of the discovered models, the following conformance metrics are used:

Table 3.. Evaluation Metrics For Software System Event Log Analysis

|  |  |
| --- | --- |
| **Metric** | **Description** |
| Fitness | Measures how well the process model can reproduce the behavior seen in the event log. |
| Precision | Evaluates how much of the behavior allowed by the model is actually seen in the log. |
| Generalization | Assesses the model’s ability to handle future, unseen but valid behaviors. |
| Simplicity | Indicates how simple and interpretable the process model is. |

Models are compared with and without preprocessing to assess the impact of the proposed method.

**Output: Validated and Interpretable Process Models**

The final output consists of:

* High-quality, noise-free event logs
* Validated process models with improved conformance
* Diagnostic insights (e.g., bottlenecks, latency, throughput) derived from model analysis

## Advantages of proposed Method

The detailed method described in this chapter offers several advantages in dealing with the challenges associated with the conversion of heterogeneous software system logs into structured event logs for use in process mining applications. These advantages further contribute to both the theoretical evolution and practical implementation of detailed explorations of process mining in convoluted real-world software situations:

**Flexibility across log types:**

This proposed method is different from traditional methods in that it doesn't call for any fixed schema or unique correlation parameters (session ID, user ID) in order to implement the transformation and analysis of logs, or when converting (more accurately, transforming) data sets into contextually-structured event logs. The proposed method supports a wide range of log types (e.g., Apache logs, IIS logs, syslog, Serilog) by employing a configurable field mapping mechanism. This allows the system to adapt to various log schemas with minimal manual effort, enabling users to standardize different field names (e.g., timestamp, activity, component) into a consistent structure through a simple configuration file. As a result, heterogeneous log sources can be effectively preprocessed and transformed into event logs suitable for process mining without extensive customization.

**Smart tracing construction without correlated identifying information:**

The proposed method based mechanism took from the hybrid-based correlation - which would be rule-based filtering, as well similarity-based-based connecting messages - in trace construction of the process instances. The architecture environment integrated in the method to allow disambiguating the individual cases of correlation with no identifying information because it advanced heuristics based on the proximity of timestamps to similar frameworks, similar source components, and similar / manual behavioral representation heuristics.

**Logs contain sufficient quality for process mining:**

Through earlier preprocessing steps such as filtering irrelevant entries, normalizing timestamps, and enriching each message with additional attributes, the pipeline ensures that the logs are transformed into a higher-quality representation suitable for process discovery. Improving the quality of event logs at this stage directly enhances the quality of the process models discovered by mining algorithms. As a result, the models are more likely to capture meaningful behavior, achieve greater completeness, and provide a more accurate representation of the actual system processes.

**Supports varied Process Discovery Algorithms:**

The event logs generated by the proposed method are exported in standard formats commonly supported by the process mining community, such as XES and CSV. These formats ensure compatibility with various process discovery algorithms and tools, making the method independent of any specific process mining technique. As a result, logs can be analyzed using different algorithms—such as Inductive Miner or Heuristic Miner—allowing for comparative evaluations and even the generation of hybrid models. This flexibility improves the reliability and interpretability of the resulting process models, as it mitigates the limitations associated with relying on a single discovery approach.

**Scalable, modular development and process:**

The proposed method is a modular process that could confer the independence to scale any specific component of the preprocessing pipeline process. Thus, allowing for a process and a method that is compatible for real time process/context, and for logging environments where distribution components and processing can be advantageous.

In conclusion, the proposed method provides a generalizable but strong contribution to one of the most persistent issues in process mining, converting heterogeneous software logs as raw use case evidence to high-quality event logs that were suitable for interpretation and analysis. This contribution can be particularly useful in domains like DevOps, IT monitoring, reliability engineering, etc., where structured analysis exists to understand/optimize system behavioral context(s).

## Main Results of 3rd Section

To be added.

# Empirical Study and Results

This chapter describes the evaluation that was conducted to assess the efficacy, flexibility, and robustness of the proposed preprocessing and traces building strategies for software system event logs. The evaluation was perform to show how the method better transformed raw, heterogeneous logs into structured event logs that were suitable for process mining. To create a comprehensive evaluation, both real-world and synthetic datasets were utilized, including logs from Apache servers and from multi-component systems provided by industrial partner sources.

This study describes the definition of a baseline experimental design, the description of the datasets used, and an outline of the tools and metrics that will be used to assess the quality of the process models generated. The key performance indicators fitness, precision, generalization, latency, and throughput time were measured in order to evaluate both the conformance and performance of the process models mined using Inductive and Heuristic Miners. The results demonstrate some of the practical consequences of having effective log preprocessing and intelligent correlation, especially when identifiers and linkage data may be less clear or completely missing.

## Initial Experimental Plan

The first experimental plan is to identify whether the proposed preprocessing and smart trace building method can convert heterogeneous system logs into structured event logs from which processes can be mined. The experimental plan consists of data sources, tooling context, experimental methods, and evaluation methods.

**Data Sources and Log Types**

To maximize generalizability, the experimentation will use real-world and publicly available synthetic logs. The experimentation will have at least two of the following types of system logs:

1. Apache HTTP Server Logs (Access/Error logs): Commonly used in the context of server monitoring, there are a number of publicly available datasets (LogHub) that provide this kind of logs.

2. Custom Synthetic Logs - Multi-component logs that will be created to simulate behavior for a distributed software system (e.g., microservices), without explicit case id's.

Each log format will vary in structure and completeness:

Table 4.. Example characteristics of selected log sources

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Log Type** | **Case ID** | **Timestamp** | **Activity** | **Component** | **Notes** |
| Apache Error Log | No | Yes | Partial | Yes | Real-world examples with missing IDs |
| Simulated Microservice | Varies | Yes | Yes | Yes | Designed to test smart grouping logic |

**Tooling and Environment**

The experiments will be conducted using the following toolset:

* Python: Preprocessing scripts, timestamp normalization, clustering-based trace construction.
* PM4Py: Open-source process mining library, log transformation, discovery algorithms (Inductive Miner/Heuristic Miner) and Quality evaluation.
* Camunda Modeler: Use to visualize discovered process models in BPMN.
* Excel / XES Exporter: Logging into process-mining compatible format.

**Experiment Design**

Each dataset will follow these steps:

1. Baseline Discovery: Use raw logs directly (minimal cleaning) for process discovery.
2. Proposed Method Pipeline:

* Apply full preprocessing
* Perform smart event correlation/ trace construction
* Format logs into structured format (XES/CSV)
* Discover process models using Inductive Miner and Heuristic Miner

1. Comparative Evaluation:

* Mark baseline and improved models using process Quality evaluation metrics.

**Evaluation Metrics**

The discovered models will be evaluated using standard conformance and performance metrics.

**Expected Outcome**

It is anticipated that models produced using the proposed preprocessing and smart trace construction method will demonstrate:

* Higher fitness and precision
* Improved generalization for flexible behavior patterns
* Better performance metrics such as reduced latency and clearer bottlenecks

## Datasets will be used in Experiments

To test the above preprocessing and smart trace construction approach to converting heterogeneous system logs to structured event logs, a number of datasets will be used. The datasets will include both real-world logs as well as publicly available benchmark datasets. Using a number of datasets will involve testing the applicability of the proposed method across various software environments.

**Dataset selection criteria**

The dataset selection criteria includes:

* Heterogeneity: Both structured and unstructured logs will be characterized and used (e.g., Apache logs, Linux syslog, applications logs).
* Lack of case identifiers: To examine some logs that are purposely lacking an explicit process or session identifiers will allow the smart trace construction algorithm to be tested.
* Public documents: Publicity among documents is preferred for the purpose of replicability and transparency in the experiment.
* Real-world context: Logs appear to exist in real operational contexts such as: web servers, distributed systems, or system monitoring platforms.

Table 4.. Summary of datasets will be used in experiments

|  |  |  |  |
| --- | --- | --- | --- |
| **Dataset Name** | **Description** | **Format** | **Challenges Introduced** |
| **Apache Logs** | Real logs from Apache access and error modules | Plain text | No case ID, unstructured activities |
| **HDFS Logs** | Hadoop Distributed File System logs used in failure analysis | JSON | Large volume, service-level interactions |
| **BGL Logs** | BlueGene/L system logs from supercomputers | CSV | No case ID, timestamp misalignment |
| **Custom Synthetic Log** | Simulated multi-service logs based on microservice requests | JSON/CSV | Partial information requires smart correlation |

A computer screen shot of a computer code

AI-generated content may be incorrect.

Figure .. Example: Apache Raw Log (Before Preprocessing)

**LogHub Datasets**

LogHub (Zhu et al., 2018) is a popular benchmark repo for system log analysis and process mining research. It has logs from different real-world and academic environments, including web servers, distributed storage systems, supercomputing systems, and others. The dataset enables to test robustness of the method in different logging structures / scales.

To evaluate the effectiveness of the proposed method in transforming and analyzing software system event logs, several standard process mining metrics will be applied. These metrics provide insights into both the structural quality and behavioral accuracy of the discovered process models, as well as the performance of the underlying system processes. Table 3 outlines the key metrics selected for this study, along with their descriptions.

**Data Preprocessing Requirements**

The collected datasets will require some preprocessing steps:

* **Log parsing:** To extract relevant fields (ie, timestamp, message, component).
* **Template extraction:** Using Drain, Logpai, or similar parsers to extract templates for unstructured logs.
* **Timestamp normalization:** To align logs from disparate sources or machines.
* **Event abstraction:** To map raw messages to higher level activities.

**Justification of Dataset Diversity**

By utilizing logs with different levels of structure, completeness, and semantics, the potential to thoroughly test the adaptability and accuracy of the method increases. Specifically:

* Apache logs will serve as the ideal medium for testing trace construction in the absence of a case ID.
* BGL logs will demonstrate challenges to timestamp-based correlation due to clock skew.
* Synthetic logs will be suitable to validate the method in controlled experimental contexts.

The experiments encompassed here provides varying levels of discrepancies across the experimental base. Therefore, I can confidently proclaim that the whole automated artifact, including methodology, is robust enough to encounter a variety of 'real world' logging situations.

# Conclusion

Based on the performed analysis of existing literature on software log transformation and process mining techniques, the obtained results show that while substantial progress has been made in log parsing, anomaly detection, and process discovery, there remains a significant gap in effective trace construction from heterogeneous and unstructured system logs. The reviewed methods primarily rely on pre-existing correlation attributes or predefined log structures, which are often absent in real-world distributed systems. This implies that current approaches are limited in generalizability and automation, highlighting the need for more adaptive preprocessing methods that can handle incomplete, noisy, or poorly correlated logs.

Based on the developed method and its application to real and synthetic system log datasets, the obtained result shows that the proposed preprocessing and smart trace construction pipeline improves the structure and quality of event logs used in process mining. By combining rule-based filtering, timestamp alignment, and similarity-based event grouping, the system effectively reconstructs meaningful traces even in the absence of explicit correlation attributes. This demonstrates the feasibility of a more flexible and data-driven approach to event log preparation and suggests that further refinement using machine learning or configuration templates could enhance applicability across various log types and domains.

# References

Andrews, R., van Dun, C. G. J., Wynn, M. T., Kratsch, W., Röglinger, M. K. E., & ter Hofstede, A. H. M. (2020). Quality-informed semi-automated event log generation for process mining. *Decision Support Systems*, *132*, 113265. <https://doi.org/10.1016/J.DSS.2020.113265>

Dakic, D., Stefanovic, D., Vuckovic, T., Zizakov, M., & Stevanov, B. (2023). Event Log Data Quality Issues and Solutions. *Mathematics*, *11*(13). <https://doi.org/10.3390/MATH11132858>

de Murillas, E. G. L., Reijers, H. A., & van der Aalst, W. M. P. (2020). Case notion discovery and recommendation: automated event log building on databases. *Knowledge and Information Systems*, *62*(7), 2539–2575. <https://doi.org/10.1007/S10115-019-01430-6>

De Oliveira, H., Augusto, V., Jouaneton, B., Lamarsalle, L., Prodel, M., & Xie, X. (2020). Optimal process minin of timed event logs *Information Sciences*, *528*, 58–7. <https://doi.org/10.1016/J.INS.2020.04.020>

Effendi, Y. A., & Kim, M. (2024).Timed Genetic Process Mining for Robust Tracking of Processes under IncompleteEvent Log Conditions. *Electronics (Switzerland)*, *13*(18). <https://doi.org/10.3390/ELECTRONICS13183752>

Elkoumy, G., Pankova, A., & Dumas, M. (2023). Differentially private release of event logs for process mining. *Information Systems*, *115*. <https://doi.org/10.1016/J.IS.2022.102161>

Hernandez-Resendiz, J. D., Tello-Leal, E., Ramirez-Alcocer, U. M., & Macías-Hernández, B. A. (2022). Semi-Automated Approach for Building Event Logs for Process Mining from Relational Database. *Applied Sciences (Switzerland)*, *12*(21). <https://doi.org/10.3390/APP122110832>

Horita, H., Kurihashi, Y., & Miyamori, N. (2020). Extraction of missing tendency using decision tree learning in business process event log. *Data*, *5*(3), 1–12. <https://doi.org/10.3390/DATA5030082>

Kim, K. P. (2020). Functional integration with process mining and process analyzing for structural and behavioral properness validation of processes discovered from event log datasets. *Applied Sciences (Switzerland)*, *10*(4). <https://doi.org/10.3390/APP10041493>

Leemans, S. J. J., van der Aalst, W. M. P., Brockhoff, T., & Polyvyanyy, A. (2021). Stochastic process mining: Earth movers’ stochastic conformance. *Information Systems*, *102*. <https://doi.org/10.1016/J.IS.2021.101724>

Lekić, J., & Milićev, D. (2021). Weakly complete event logs in process mining. *Computing and Informatics*, *40*(2), 341–367. <https://doi.org/10.31577/CAI_2021_2_341>

Li, G. L., Jing, S. Y., Shen, Y., & Guo, B. (2022). Efficient Discrete Particle Swarm Optimization Algorithm for Process Mining from Event Logs. *International Journal of Computational Intelligence Systems*, *15*(1). <https://doi.org/10.1007/S44196-022-00074-9>

Shahzadi, S., Emam, W., Shahzad, U., Iftikhar, S., Ahmad, I., & Sharma, G. (2024). Enhancement in Process Mining Model by Repairing Noisy Behavior in Event Log. *IEEE Access*, *12*, 82938–82948. <https://doi.org/10.1109/ACCESS.2024.3411089>

Tavares, G. M., Oyamada, R. S., Barbon, S., & Ceravolo, P. (2023). Trace encoding in process mining: A survey and benchmarking. *Engineering Applications of Artificial Intelligence*, *126*. <https://doi.org/10.1016/J.ENGAPPAI.2023.107028>

Ter Hofstede, A. H. M., Koschmider, A., Marrella, A., Andrews, R., Fischer, D. A., Sadeghianasl, S., Wynn, M. T., Comuzzi, M., De Weerdt, J., Goel, K., Martin, N., & Soffer, P. (2023). Process-Data Quality: The True Frontier of Process Mining. *Journal of Data and Information Quality*, *15*(3). <https://doi.org/10.1145/3613247>

Process Mining. (n.d.). Event data for process mining. Retrieved January 21, 2025, from <https://www.processmining.org/event-data.html>

Jieming Zhu, Shilin He, Pinjia He, Jinyang Liu, Michael R. Lyu. [Loghub: A Large Collection of System Log Datasets for AI-driven Log Analytics](https://arxiv.org/abs/2008.06448). IEEE International Symposium on Software Reliability Engineering (ISSRE), 2023

Ghasemi, M., & Amyot, D. (2022). From event logs to goals: A systematic review of goal-oriented process mining. ACM Digital Library. <https://doi.org/10.1007/s00766-018-00308-3>

He, S., Zhu, J., He, P., & Lyu, M. R. (2020). A survey on automated log analysis for reliability engineering**.** arXiv. <https://arxiv.org/abs/2009.07237>

Marin-Castro, H. M., & Tello-Leal, E. (2021). Event log preprocessing for process mining: A review. Information Systems, 103, 101835. <https://doi.org/10.3390/app112210556>

Milani, F., & Maggi, F. M. (2018). A comparative evaluation of log-based process performance analysis techniques. Information Systems, 74, 67–83. <https://arxiv.org/abs/1804.03965>

ProcessMaker. (2021). What are event logs in process mining? ProcessMaker Blog. <https://www.processmaker.com/blog/what-are-event-logs-in-process-mining>

QAD. (2022). What is process mining? QAD Blog. <https://www.qad.com/blog/what-is-process-mining>

SciDirect. (2021). Process mining usage in cybersecurity and software reliability analysis. ScienceDirect. <https://www.sciencedirect.com/science/article/pii/S2590005621000576>

Springer. (2022). Using log analytics and process mining to enable self-healing in IT systems. SpringerLink. <https://link.springer.com/article/10.1007/s10669-022-09859-x>

Zhu, J., He, S., Du, X., & Li, Z. (2018). Tools and benchmarks for automated log parsing. arXiv. <https://arxiv.org/abs/1811.03509>