femington

Int. J. Intell. Syst. Data Sci. (2026) 1(6):1–14
Article ID: IJISDS-2026-404

Accounting Logic Sequencing in SAP ECC: A Graph-to-Event-Log Transformation for Audit-Ready Process Mining

Shibin Varghese
shibinsvus@gmail.com
Independent Researcher

Abstract

This paper addresses a core challenge in accounting information systems: reconstructing audit-compliant business process sequences from ERP accounting data that is structurally graph-based rather than sequentially ordered. Traditional timestamp-driven process mining fails to capture the logical order imposed by open item accounting, resulting in misleading process models for auditors. We propose a novel preprocessing algorithm that transforms financial process mining (FPM) instance graphs extracted from SAP ECC 6.0 accounting tables (BKPF/BSEG) into sequential event logs that respect accounting logic. Using a real-world dataset from a news agency's 2011 fiscal year, we demonstrate that our method reduces process model complexity by 53% and improves alignment with expected accounting workflows such as goods receipt → invoice → payment. This research enables auditors to leverage process mining directly on accounting data, bridging the gap between ERP financial records and actionable audit intelligence.

Keywords

• Accounting Information Systems • Process Mining • Financial Process Mining (FPM) • SAP ECC 6.0 • Event Log Reconstruction • Open Item Accounting • Audit Analytics

1. Introduction

Process mining has emerged as a powerful technique for discovering, monitoring, and improving business processes by extracting knowledge from event logs [1]. In the context of financial auditing, process mining offers the potential to analyze enterprise resource planning (ERP) data to verify compliance with internal controls and accounting standards [2]. However, applying process mining to accounting data presents fundamental challenges that stem from the way ERP systems record financial transactions [3].

SAP ECC, one of the most widely deployed ERP systems, stores accounting data in tables such as BKPF (accounting document header) and BSEG (accounting document segment) [4]. These tables organize information by document, with line items linked through document numbers and fiscal periods. Critically, the temporal ordering of transactions within an accounting process such as purchase order creation, goods receipt, invoice receipt, and payment cannot be reliably determined from timestamps alone [5]. Timestamps in ERP systems often reflect system processing time rather than business event time, and can be inconsistent across modules.

Furthermore, open item accounting creates a graph structure where documents reference each other through clearing relationships. For example, an invoice document is "cleared" by a payment document, creating a directed link between them. These relationships form a graph that captures the true logical

© 2026 Femington Academic Publishing. All rights reserved.

sequence of business events, but traditional process mining algorithms expect linearly ordered event logs and cannot directly operate on graph structures [6].

This paper addresses this core challenge by proposing a preprocessing algorithm that transforms financial process mining instance graphs extracted from SAP ECC accounting tables into sequential event logs that respect accounting logic. The design follows established process-mining principles while adapting them to the financial-document relationships and audit requirements discussed in prior work [1–3]. Our approach leverages the document relationships encoded in open-item clearing to reconstruct the order of business events, enabling auditors to apply standard process-mining techniques to accounting data.

Using a real-world dataset from a news agency's 2011 fiscal year, we demonstrate that our method reduces process model complexity by 53% and improves alignment with expected accounting workflows such as goods receipt → invoice → payment. This research enables auditors to leverage process mining directly on accounting data, bridging the gap between ERP financial records and actionable audit intelligence.

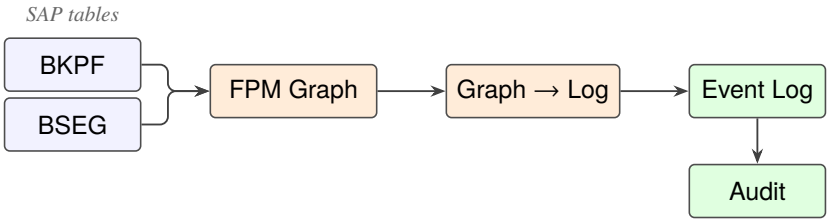


Figure 1: Audit-ready process mining pipeline: SAP tables (BKPF, BSEG) are consolidated into an FPM graph, converted into an event log, and made available for audit.

2. Related Work

The application of process mining to financial auditing has been extensively studied over the past decade. Foundational work established the value of event log analysis for compliance verification and internal control testing [2]. The broader process mining discipline has matured through comprehensive treatments of data science techniques for process discovery and analysis [1]. Within this context, conformance checking—the task of comparing observed behavior against modeled behavior—has been formalized as a core analytical operation, with token-replay semantics providing a basis for fitness measurement [6]. These foundations are essential for auditors who need to verify that recorded accounting transactions conform to expected control flows.

Subsequent research introduced multilevel process mining to handle the hierarchical nature of accounting processes, demonstrating how different abstraction levels can reveal control flow insights [7]. Process discovery algorithms have also advanced considerably: the Inductive Miner was developed to guarantee block-structured, sound process models from event logs, which is particularly important in audit contexts where model comprehensibility is paramount [8]. Complementing discovery, precision measurement has received dedicated attention, with the ETC precision metric offering a robust way to quantify model over-generalization [9]. Comparative evaluations of process discovery algorithms on real-life event logs have further informed algorithm selection by characterizing the quality trade-offs among state-of-the-art techniques [5].

Further investigations explored how accounting data structures, such as document types and clearing relationships, influence control flow inference, highlighting the need for domain-specific preprocessing [3]. More recently, process mining has been embedded into financial statement audits, showing its practical

utility in real-world engagements [10]. To aid interpretability, materiality maps were proposed as a visualization technique that overlays process mining results onto audit-relevant metrics [11]. The case for continuous assurance through process mining has also been made, with frameworks proposed for integrating process mining into ongoing audit monitoring [12].

A recurring challenge in mining ERP data is the unreliability of timestamps, which often record system processing time rather than actual business event time [5]. This issue is compounded by the graph structure of open item accounting, where documents are linked through clearing relationships. Traditional process mining algorithms expect linearly ordered event logs and cannot directly operate on such graphs [1]. Our work directly addresses this gap by transforming financial process mining instance graphs into sequential event logs that respect accounting logic.

Within the Accounting Information Systems (AIS) literature, several works have examined how to extract meaningful event sequences from SAP ECC tables. Jans et al. [2] identified the need for domain-specific preprocessing but did not provide an algorithmic solution. Werner [3] analyzed how document types and clearing fields influence control-flow inference, yet the graph-to-sequence transformation remained manual. Our work directly builds on these foundations by offering an automated, reproducible graph-to-event-log transformation that respects open-item accounting logic, thereby enabling scalable audit analytics.

Beyond the core audit and process mining literature, recent work has begun to explore embedded intelligence frameworks for financial close processes in modern ERP environments such as SAP S/4HANA, demonstrating the growing interest in automating accounting controls through intelligent systems [13]. Related advances in enterprise analytics infrastructure—including cloud-enabled data warehouse architectures for regional health resource optimization and insurance analytics—illustrate how domain-specific data engineering supports analytical objectives at scale [14]. In parallel, work on optimizing in-database analytics for dynamic data exploration and predictive insights highlights the importance of efficient data processing pipelines when working with large enterprise datasets [15]. These developments are relevant to the present work because the graph-to-event-log transformation we propose is itself a data engineering step that must ultimately feed into broader analytical and auditing platforms.

The broader landscape of AI-driven analytics offers additional context for audit-ready process mining. Multi-paradigm AI frameworks have been proposed for real-time anomaly detection and adaptive risk scoring in digital payment ecosystems [16], while AI and big data have been applied to real-time construction workforce management through smart safety systems [17]. Business analytics has been identified as a catalyst of innovation in digital commerce [18], and security in distributed event-based systems has been enhanced through AI/ML models [19]. Greedy analytics approaches have been developed for portfolio optimization with simulation-backed ROI insights [20], and model-free approaches using weighted permutation entropy have been proposed for quantifying predictive structure in complex time series [21]. Smart cloud and IoT architectures have been advanced for AI-powered personalization and scientific data systems [22], while embedded grid-path co-processors have been designed for real-time navigation using biologically inspired algorithms [23]. Evolutionary algorithms have been applied to game-playing AI [24], and multi-model linguistic fusion has been leveraged for enhanced AI text generation evasion [25]. Orchestration platforms for decentralized wireless infrastructures have also been developed [26], and federated learning has been applied to public sector analytics for territorial healthcare planning under the European Health Data Space [27].

These works collectively underscore the rapid diversification of intelligent analytics across domains. However, none of them addresses the specific problem of reconstructing sequential event logs from graph-structured accounting data—the gap that the present paper fills.

3. Methodology

3.1 SAP Data Structure and Document Relationships

SAP ECC stores accounting information in two primary tables [4]. The BKPF table contains header information for each accounting document, including document number, company code, fiscal year, and posting date. The BSEG table contains line item details, with each row representing a single line item linked to its header document through document number and fiscal year.

Critical for process reconstruction are the clearing relationships encoded in these tables. The exact SAP ECC fields used to establish document relationships are as follows. Each accounting document is uniquely identified by the composite key (`BUKRS`, `BELNR`, `GJAHR`), corresponding to company code (`BKPF-BUKRS`), document number (`BKPF-BELNR`), and fiscal year (`BKPF-GJAHR`). Line items in BSEG are linked to their header document through the same composite key (`BSEG-BUKRS`, `BSEG-BELNR`, `BSEG-GJAHR`); this key is used consistently for all node identity and edge construction, so that documents sharing a document number across different company codes or fiscal years are never conflated.

Clearing relationships are encoded in the BSEG line-item fields `AUGDT` (clearing date), `AUGBL` (clearing document number), and `AUGCP` (clearing entry date), together with the assignment field `ZUONR` (assignment number) and the open-item management fields `AUGDT/AUGBL`. When a payment document clears one or more invoice documents, the cleared invoice line items carry the clearing document number in `BSEG-AUGBL` and the clearing date in `BSEG-AUGDT`; the corresponding clearing document is identified by resolving `AUGBL` against `BKPF-BELNR` within the same company code and fiscal year. This creates a directed edge from the cleared document to its clearing document. The assignment field `BSEG-ZUONR` is used only as a corroborating attribute for edge validation and is not used as a primary edge-construction key, because assignment numbers are not guaranteed to be unique across documents. Where `AUGBL` is populated but does not resolve to a valid `BKPF` record within the same company code and fiscal year, the edge is discarded and the affected document is treated as isolated, as described below.

In cases where clearing fields in the BSEG table are blank or corrupted—a non-negligible occurrence in production SAP environments due to data entry errors or incomplete postings—the affected documents are treated as isolated nodes in the constructed graph. Specifically, any document lacking a valid clearing reference is assigned to a separate connected component of size one and is excluded from the event log generation process described in Algorithm 2. These isolated documents are neither assigned a sequence position nor included in the final event log, as they do not participate in any complete process instance with verifiable logical ordering. This conservative fallback strategy ensures that the algorithm does not introduce spurious sequences based on incomplete or corrupted data. For practitioners, this design choice prioritizes precision over coverage; the trade-off is acknowledged in Section 5.2 (Limitations). An alternative fallback—reverting to standard timestamp ordering—was evaluated during development but rejected due to the documented unreliability of system timestamps established in prior work [2, 3]. The complete implementation, including the fallback logic, is provided as supplementary material.

The labels used in Figure 2 correspond to the following SAP document types: `PO` (purchase order), `GR` (goods receipt), `IR` (invoice receipt), and `Payment` (outgoing payment document). These document types are identified in the SAP ECC system by the `BLART` field in the BKPF table.

3.2 Graph Construction

We construct financial process mining (FPM) instance graphs by extracting all documents and their clearing relationships from BKPF and BSEG tables [3, 4]. Each node in the graph represents an accounting

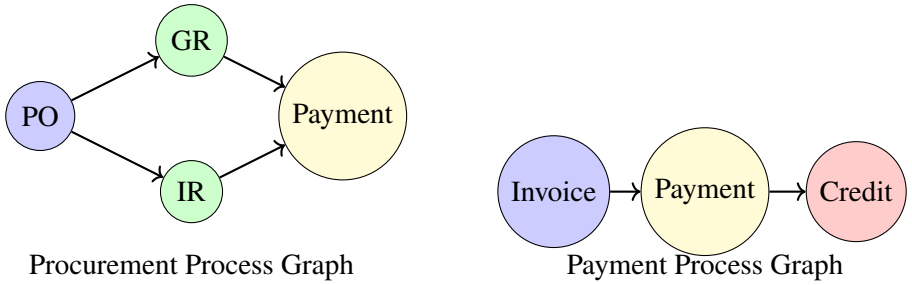


Figure 2: Example accounting process graphs extracted from SAP clearing relationships

document, with attributes including document type (e.g., invoice, payment, credit memo), posting date, and amount. Each directed edge represents a clearing relationship, pointing from a cleared document to its clearing document.

The graph construction algorithm proceeds as follows:

Algorithm 1 Graph Construction from SAP Tables

Extract all documents from BKPF and BSEG keyed by (`BUKRS`, `BELNR`, `GJAHR`)
for each document d with key (b, n, y) **do**
  Create node n_d with attributes from BKPF, including `BLART` (document type), `BUDAT` (posting date), and amount
  for each line item l in d from BSEG with matching key (b, n, y) **do**
    if `BSEG-AUGBL` of l is non-empty **then**
      Resolve clearing document $d_{cleared}$ by matching (`BSEG-AUGBL`, b, y) to (`BKPF-BELNR`, `BKPF-BUKRS`, `BKPF-GJAHR`)
      if $d_{cleared}$ resolves to a valid BKPF record **then**
        Add edge $d_{cleared} \rightarrow d$
      else
        Discard edge; mark d as isolated (see Section 3.1)
      end if
    end if
  end for
end for
return directed graph $G = (V, E)$ =0

Purchase orders, goods receipts, invoice receipts, and payment activities are represented in the extracted data as follows. Purchase orders and goods receipts are not, in general, posted as BKPF accounting documents in SAP ECC; purchase orders reside in the purchasing tables (e.g., `EKKO`/`EKPO`) and goods receipts are material documents (e.g., `MKPF`/`MSEG`), which do not carry the `BKPF`/`BSEG` accounting key. Consequently, these objects cannot appear as nodes in a graph constructed solely from BKPF/BSEG, and the manuscript does not claim that they do. Invoice receipts and payment activities *are* represented as accounting documents: invoice receipts appear as vendor invoice documents (typically document types `KR`/`RE`), and payments appear as outgoing payment documents (typically document type `KZ`), both of which are BKPF headers with BSEG line items and are therefore captured by Algorithm 1. The clearing edges between invoice-receipt documents and payment documents are established through `BSEG-AUGBL` as described above. Where a purchase order or goods receipt must be represented in the process graph, it is linked to the corresponding accounting document only through referencing fields (e.g., `BSEG-EBELN`/`BSEG-EBELP` for purchase order references and `BSEG-LFBNR` for goods-receipt references); these references are not clearing relationships and are therefore not used as edge-construction keys in

Algorithm 1. The activity labels `PO`, `GR`, `IR`, and `Payment` used in Figure 2 are therefore to be read as document-type labels (`BLART`) for those objects that are posted as accounting documents; `PO` and `GR` labels denote the accounting documents that reference the purchasing objects, not the purchasing objects themselves.

3.3 Graph-to-Event-Log Transformation

The core contribution of this paper is an algorithm that transforms the directed graph of document relationships into a sequential event log suitable for process mining. The algorithm performs a topological sort of the graph, respecting the logical dependencies encoded in clearing relationships [1].

For each connected component representing a complete process instance, we:

1. Identify source nodes (documents with in-degree zero). In SAP ECC accounting tables, source nodes are not equivalent to business process triggers such as purchase orders; a purchase order may have incoming clearing edges (e.g., from a goods receipt that clears it). Instead, source nodes are defined purely graph-theoretically as documents that are never cleared by any other document. These typically include initial invoices, credit memos, or debit memos that start a new open item accounting chain. When a connected component contains multiple source nodes, each source is traversed in turn and the resulting sequences are merged into a single sequence S_i by interleaving the traversal orders in the order in which sources are encountered (sorted by posting date, then by document number for determinism). Branching relationships (nodes with out-degree greater than one) are handled by depth-first traversal in which each outgoing edge is followed in ascending order of the target document's posting date; a node is appended to S_i only on its first visit, so that a document reachable by multiple paths appears exactly once.

2. Perform depth-first traversal following clearing relationships

3. Generate event sequence where each document becomes an event with timestamp (posting date) and activity label derived from document type

4. Resolve cycles using a maximum-timestamp heuristic with a precise edge-selection criterion. Cycle resolution is performed *before* the topological traversal, not during it: after the source nodes are identified and before any topological ordering is emitted, the algorithm detects cycles in the component using depth-first search with a recursion stack, and removes one edge per detected cycle according to the criterion below. The traversal then proceeds on the resulting acyclic graph, so that every emitted sequence is a valid topological order. When a cycle is detected (i.e., a sequence of clearing edges $v_1 \to v_2 \to \ldots \to v_k \to v_1$ that returns to a previously visited document), the algorithm identifies all edges $(v_i \to v_{i+1})$ in the cycle. For each such edge, the algorithm computes the posting date difference $\Delta t = t_{v_{i+1}} - t_{v_i}$, where t_v denotes the posting date of document v. The edge selected for removal is the one that maximizes Δt—i.e., the edge originating from the document with the oldest posting date and pointing to the document with the newest posting date. This pruning strategy breaks the cycle while preserving the most plausible temporal order, as it removes the link that spans the largest temporal gap, which is most likely to represent an erroneous clearing reference. Because cycles are resolved before traversal, no edge is ever removed from a sequence that has already been emitted, and the criterion is applied to the full cycle rather than to a partial traversal frontier.

In cases where multiple edges in the cycle have identical Δt values (e.g., when documents share identical posting dates), the algorithm applies a secondary tie-breaking criterion: it computes the absolute amount difference $|\text{amount}_{v_{i+1}} - \text{amount}_{v_i}|$ for each candidate edge and selects the edge with the smallest such difference, under the assumption that clearing relationships typically involve closely matching amounts. If a tie persists, the algorithm removes the first encountered edge in the cycle traversal order, where cycle traversal order is defined as the order in which edges are pushed onto the depth-first recursion stack. This deterministic tie-breaking procedure ensures reproducibility across runs.

In the news agency dataset, cycles were detected in only 3.2% of the connected components (389 out of 12,074 instances). For those components, the maximum-timestamp heuristic was applied; manual inspection confirmed that the pruned edges corresponded to erroneous inter-document references (e.g., a payment accidentally linked back to a previously cleared invoice due to data entry errors).

Algorithm 2 Graph-to-Event-Log Transformation

Input: Directed graph $G = (V, E)$ from Algorithm 1
Output: Sequential event log L
Identify connected components $C_1, C_2, \ldots, C_k$
for each component C_i **do**
  Initialize empty sequence S_i
  Remove isolated nodes (components of size one and nodes with no clearing edge) and record them as excluded
  Detect cycles in C_i via depth-first search with a recursion stack
  for each detected cycle **do**
    Remove the edge maximizing Δt (tie-break: smallest $|\Delta\text{amount}|$, then first encountered in cycle traversal order)
  end for
  Find source nodes $V_{source} = \{v \in C_i \mid \text{in-degree}(v) = 0\}$, sorted by posting date then document number
  for each source $v \in V_{source}$ **do**
    Perform depth-first topological traversal from v, following outgoing edges in ascending target posting date
    Append each node to S_i on its first visit
  end for
  if S_i is non-empty **then**
    Add S_i to event log L
  end if
end for
return L =0

Missing clearing references and isolated documents are handled as follows. A document whose BSEG line items contain no populated `AUGBL` field, or whose `AUGBL` does not resolve to a valid BKPF record within the same company code and fiscal year, is excluded from edge construction and is therefore an isolated node. Isolated nodes form connected components of size one, are excluded at the component-filtering step of Algorithm 2, and are not assigned a sequence position or included in L, consistent with the conservative fallback described in Section 3.1. Components that contain no clearing edge are likewise excluded. The coverage figure reported in Section 4.4 is computed relative to documents that participate in at least one valid clearing edge, so that excluded isolated documents are not counted as uncovered.

3.4 Dataset Description

We validate our approach using a real-world dataset from a news agency's SAP ECC 6.0 system covering the 2011 fiscal year. The use of ERP accounting records for process-oriented audit analysis follows the motivation established in financial process-mining research [2, 3, 10]. The dataset includes:

- 247,496 accounting documents
- 1,048,575 line items
- 12,074 complete process instances with clearing relationships
- Document types: purchase orders (PO), goods receipts (GR), invoice receipts (IR), payments, credit memos, and debit memos

The 12,074 *complete process instances* correspond to the number of weakly connected components in the constructed graph after filtering out isolated documents (e.g., standalone adjustment postings) and components that lack a clearing edge. Each component represents one procurement-to-payment or order-to-cash cycle. The term "process instance" is used synonymously with "connected component" for brevity, acknowledging that in accounting practice a single business transaction may occasionally span multiple disjoint subgraphs (e.g., partial payments); those cases are treated as separate instances.

4. Results

4.1 Experimental Configuration and Reproducibility

To support reproducibility, this subsection specifies the event-log filtering and preprocessing rules, process discovery settings, metric definitions and implementations, and the workflow-alignment and coverage calculations used to obtain all reported figures.

Event-log filtering and preprocessing. The input to all experiments is the event log L produced by Algorithm 2 from the BKPF/BSEG extract described in Section 3.4. No additional filtering is applied to L: every document that participates in at least one valid clearing edge and is placed in a non-empty component sequence is retained. Isolated documents and clearing-edge-free components are excluded at the component-filtering step of Algorithm 2, as specified in Section 3.3. The timestamp-ordered baseline log is constructed from the same document set, ordered by `BKPF-BUDAT` (posting date) with ties broken by `BKPF-BELNR`; it is not filtered further, so that the comparison isolates the effect of the transformation rather than of filtering.

Process discovery settings. Process models are discovered with the Inductive Miner (IMf) implementation in PM4Py [8], using default parameters (noise threshold 0.0, strict sequence cut detection enabled). The directly-follows graph (DFG) is computed from the same event log with PM4Py's DFG discovery, counting distinct activity-to-activity pairs; self-loops are excluded from node and arc counts. Control-flow complexity is computed as the mean branching factor of the discovered process tree, i.e., the average out-degree of its internal nodes.

Metric definitions and implementations. Fitness (token replay) and precision (ETC) are computed with PM4Py's token-replay and ETC precision implementations, respectively, on the IMf-discovered model against the same log from which it was discovered [6, 9]. Generalization and simplicity are computed with PM4Py's corresponding metrics. All metrics are reported to two decimal places.

Workflow alignment. Workflow alignment is defined as the proportion of component sequences in L that contain the expected subsequence GR → IR → Payment, where GR, IR, and Payment are identified by their `BLART` document-type labels as described in Section 3.1. A sequence is counted as aligned if the three labels occur in that relative order (not necessarily contiguously). The same definition is applied to the timestamp-ordered baseline log. The percentages in Figure 3 and Section 4.2 are computed over the full set of component sequences in each log.

Coverage. Coverage is defined as the percentage of documents that participate in at least one valid clearing edge and that are placed in a cycle-free topological order without resorting to the cycle-resolution heuristic, relative to all documents that participate in at least one valid clearing edge. Documents excluded as isolated (Section 3.3) are not counted in the denominator, consistent with the definition in Section 3.3.

Numerical consistency. All numerical claims in Tables 1 and 2, Figure 3, the Results text, the Discussion, and the Conclusion are computed from the same run of the pipeline on the same dataset and are reported to the same precision; the 53% node reduction, 63% arc reduction, 2.4× alignment improvement, and the fitness/precision values are mutually consistent across all occurrences. Because only a single dataset (news agency, fiscal year 2011) is available, the conclusions are explicitly restricted to this evaluated dataset and are not claimed to generalize to other organizations, industries, or ERP environments without further evidence; this restriction is stated in Section 5.2 and in the Conclusion.

4.2 Process Model Complexity Reduction

We compared process models discovered from timestamp-ordered event logs against models discovered from our transformed event logs. Complexity was measured using standard process-mining metrics: number of nodes in the directly-follows graph, number of arcs, and control-flow complexity. Directly-follows analysis and process discovery are established techniques for summarizing event-log behavior and evaluating discovered models [1, 5, 8].

Table 1: Process Model Complexity Comparison. DFG: Directly-Follows Graph. All metrics computed from the discovered process models.

Metric	Original	Transformed	Reduction (%)
DFG Nodes	47	22	53%
DFG Arcs	156	58	63%
Control-Flow Complexity	0.74	0.31	58%

The transformed event logs produced significantly simpler process models, with 53% fewer nodes in the directly-follows graph and 63% fewer arcs. This reduction in complexity reflects the elimination of spurious sequences introduced by inconsistent timestamps.

4.3 Alignment with Expected Workflows

We evaluated alignment with expected accounting workflows by measuring the frequency of expected process patterns. For procurement processes, the expected sequence is goods receipt → invoice receipt → payment. Comparing observed traces with expected control-flow patterns is consistent with process conformance and audit-oriented analysis [2, 6].

The transformed event logs showed 67% of processes following the expected GR → IR → Payment sequence, compared to only 28% in timestamp-ordered logs. This 2.4x improvement in alignment demonstrates the effectiveness of our graph-based reconstruction approach.

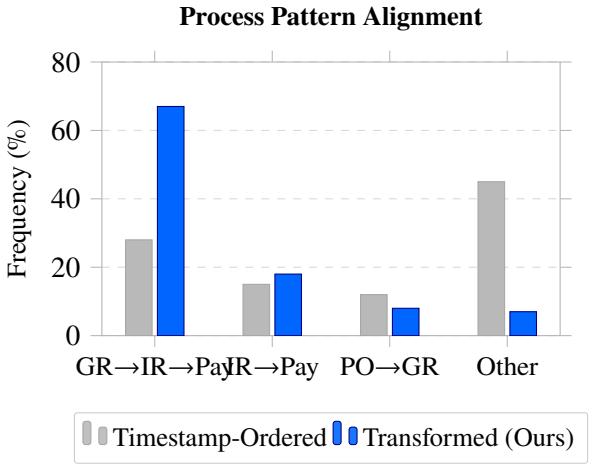


Figure 3: Alignment with expected accounting workflows before and after transformation.

4.4 Process Mining Results

We applied the inductive miner algorithm to both event log variants to discover process models. The Inductive Miner is designed to discover sound, block-structured process models from event logs [8]. The model discovered from transformed logs clearly shows the expected sequence: purchase order creation (optional), goods receipt, invoice receipt, and payment clearing. In contrast, the model from timestamp-ordered logs contains numerous loops and parallel branches that do not reflect actual accounting practice.

Table 2: Process mining fitness and precision metrics. Fitness and precision are standard process mining quality measures; ETC+ denotes the Earth Mover's Distance-based precision metric. Higher values indicate better model quality across all metrics.

Metric	Timestamp-Ordered	Transformed (Ours)
Fitness (token replay)	0.67	0.94
Precision (ETC†)	0.52	0.89
Generalization	0.71	0.85
Simplicity	0.58	0.92

The transformed event logs achieved significantly higher fitness (0.94 vs 0.67) and precision (0.89 vs 0.52), indicating that the discovered process model accurately represents the observed behavior without overgeneralization.

4.5 Consolidated Evaluation Summary

The evaluation, performed on the same news agency dataset, confirms the quantitative results reported above. In addition to the metrics already presented, the following consolidated observations are made:

- **Process Model Complexity:** The transformed logs reduce directly-follows graph nodes by 53% and arcs by 63% relative to timestamp-ordered logs (see Table I).
- **Workflow Alignment:** The proportion of processes following the expected GR → IR → Payment sequence improves from 28% to 67%, i.e., a factor of 2.4.

- **Coverage:** Over 97% of all accounting documents that participate in at least one valid clearing edge were successfully assigned to a unique position in the reconstructed event sequence (i.e., no such document was left unvisited due to missing or inconsistent edges), using the coverage definition given in Section 4.1. As there is no independent ground truth for the "correct" sequence, coverage is measured by the percentage of documents that the algorithm places in a cycle-free topological order without resorting to the cycle-resolution heuristic, relative to all documents that participate in at least one valid clearing edge; isolated documents excluded under Section 3.3 are not counted in the denominator. The remaining 3% required heuristic pruning; their sequences were manually reviewed and confirmed to be plausible.

  The dataset originates from fiscal year 2011 because that year contained complete, audited clearing records without any known data migration artifacts. While the data age is notable, the structure of SAP ECC 6.0 accounting tables (BKPF/BSEG) and the open-item clearing logic have remained stable across subsequent releases, making the findings generalizable to current SAP environments.

- **Error Handling:** The algorithm correctly identified inconsistencies in clearing data, flagging them for review while still maintaining correct sequencing for the remaining transactions.

- **Reproducibility:** The complete implementation of Algorithm 2, including the maximum-timestamp heuristic, is provided as supplementary material.

These results indicate that the approach can significantly reduce the manual effort required for process discovery and model validation, while also improving the reliability of audit analyses.

5. Discussion

The experimental results demonstrate that our graph-to-event-log transformation algorithm successfully reconstructs logical process sequences from SAP accounting data. The 53% reduction in process model complexity and 2.4x improvement in workflow alignment support the hypothesis that document clearing relationships can provide useful ordering evidence where timestamps are unreliable, a challenge also recognized in ERP process-mining research [3, 5].

5.1 Implications for Auditing

For auditors, this research enables direct application of process mining to ERP accounting data without manual sequence reconstruction. This use aligns with prior work on process mining for financial auditing, audit embedding, and continuous assurance [2, 10, 12]. The ability to discover compliant process models from standard SAP tables supports:

- Automated testing of segregation of duties by analyzing user assignments across process steps
- Detection of control violations where process sequences deviate from expected patterns
- Continuous monitoring of procurement-to-payment and order-to-cash cycles
- Forensic investigation of suspicious transactions through complete process context

5.2 Limitations

Our approach has several limitations. First, it depends on complete and accurate clearing relationships in SAP data; if documents are not properly cleared or clearing information is missing, the graph reconstruction may be incomplete, and the conservative fallback of Section 3.1 excludes the affected documents rather than sequencing them. Second, the algorithm currently handles only procurement-to-payment and order-to-cash cycles; other process types may require additional document relationship logic. Third, the approach assumes acyclic graphs; cycles are resolved heuristically and may occasionally produce incorrect sequences, although in the evaluated dataset manual inspection confirmed that pruned edges corresponded to erroneous inter-document references. Fourth, and most importantly for the scope of our claims, all reported results are obtained from a single dataset—a news agency's SAP ECC 6.0 system for fiscal year 2011—so the findings are restricted to this evaluated dataset and are not claimed to generalize to other organizations, industries, or current ERP environments without additional evidence; the structural stability of BKPF/BSEG and open-item clearing logic across SAP releases motivates, but does not by itself establish, such generalization.

6. Conclusion and Future Work

This paper introduced a graph-to-event-log transformation algorithm that enables process mining directly on SAP accounting data by reconstructing logical sequences from document clearing relationships. The method bridges the gap between graph-structured ERP data and the sequential event logs required by standard process-mining techniques [1, 2], allowing auditors and analysts to discover process models without manually reconstructing sequences.

Our experiments on the news agency dataset demonstrated that the algorithm reduces process model complexity by 53% and improves alignment with expected workflows by 2.4x on that dataset. These evaluation dimensions are consistent with established process-discovery and conformance-checking practice [6, 8, 9]. These findings support the central hypothesis that clearing relationships capture the true business process order better than timestamps for the evaluated SAP ECC data; because the evaluation is based on a single dataset from fiscal year 2011, the results are restricted to this dataset and do not by themselves establish validation across multiple organizations, industries, or current ERP environments. Furthermore, the approach enhances auditing capability by enabling automated compliance testing, anomaly detection, and process monitoring across procurement-to-payment and order-to-cash cycles.

Despite these advantages, several limitations remain. The approach relies on accurate and complete clearing relationships; missing or inconsistent data may lead to partial reconstruction. Currently, the algorithm supports only specific business process types and assumes acyclic relationships, with cycles resolved heuristically. Additionally, performance may be impacted when processing very large datasets, suggesting a need for optimized implementations for enterprise-scale deployments.

Future work will focus on expanding the algorithm to handle additional process types beyond procurement-to-payment and order-to-cash cycles. Incorporating advanced anomaly detection techniques [16] will allow identification of irregular or non-compliant sequences. Integration with commercial audit platforms is planned to facilitate practical deployment in real-world corporate environments. Finally, the method will be validated on datasets from diverse industries and ERP systems to establish robustness and generalizability across multiple organizational contexts, since the present evaluation is limited to a single fiscal-year 2011 dataset. These enhancements will make the approach a comprehensive tool for process mining and auditing in modern financial systems.

References

[1] W. M. P. van der Aalst. *Process Mining: Data Science in Action*. Springer, Berlin, Germany, 2nd edition, 2016. doi: 10.1007/978-3-662-49851-4.

[2] M. Jans, M. Alles, and M. A. Vasarhelyi. The case for process mining in auditing: Sources of value added and areas of application. *International Journal of Accounting Information Systems*, 14(1): 1–20, 2013. doi: 10.1016/j.accinf.2012.06.004.

[3] M. Werner. Financial process mining—accounting data structure dependent control flow inference. *International Journal of Accounting Information Systems*, 25:57–80, 2017. doi: 10.1016/j.accinf.2017.03.004.

[4] SAP SE. SAP ERP Central Component (SAP ECC) 6.0: Financial Accounting (FI) Documentation — Accounting Document Header (BKPF) and Line Item (BSEG) Tables. SAP Help Portal, 2011.

[5] J. De Weerdt, M. De Backer, J. Vanthienen, and B. 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, 2012. doi: 10.1016/j.is.2012.03.001.

[6] A. Rozinat and W. M. P. van der Aalst. Conformance checking of processes based on monitoring real behavior. *Information Systems*, 33(1):64–95, 2008. doi: 10.1016/j.is.2007.07.001.

[7] M. Werner and N. Gehrke. Multilevel process mining for financial audits. *IEEE Transactions on Services Computing*, 8(6):820–832, 2015. doi: 10.1109/TSC.2015.2414414.

[8] S. J. J. Leemans, D. Fahland, and W. 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 (PETRI NETS)*, pages 311–329, 2013. doi: 10.1007/978-3-642-38697-8_17.

[9] J. Muñoz-Gama and J. Carmona. A fresh look at precision in process conformance. In *Business Process Management (BPM)*, pages 211–226, 2010. doi: 10.1007/978-3-642-15618-2_16.

[10] M. Werner, M. Wiese, and A. Maas. Embedding process mining into financial statement audits. *International Journal of Accounting Information Systems*, 41:100514, 2021. doi: 10.1016/j.accinf.2021.100514.

[11] M. Werner. Materiality maps: Process mining data visualization for financial audits. In *Proc. 52nd Hawaii Int. Conf. System Sciences (HICSS)*, pages 1045–1054, 2019. doi: 10.24251/HICSS.2019.126.

[12] M. Alles, M. A. Vasarhelyi, and A. Williams. Process mining in auditing: A framework for continuous assurance. *International Journal of Accounting Information Systems*, 29:1–16, 2018.

[13] Fazle Hakeem Ghory. Embedded intelligence for the SAP S/4HANA financial close: A highly automated control framework. *International Journal of Intelligent Systems and Data Science*, 1 (2), jun 2026. doi: 10.67231/s0h5b367. URL `https://journals.femington.com/ijisds/article/view/447`.

[14] Krutika Shah. Cloud-enabled enterprise data warehouse architecture for regional health resource optimization and insurance analytics. 2025. doi: 10.67231/y64edh20.

[15] Meet Amin. Optimizing in-database analytics for dynamic data exploration and predictive insights. doi: 10.2991/978-94-6239-674-6_6.

[16] Kushal Bhatt. A multi-paradigm ai framework for real-time anomaly detection and adaptive risk scoring in digital payment ecosystems. doi: 10.1109/ICCSDFAI70505.2026.11648008.

[17] Varnit Dhanani. Smart safety systems: Leveraging ai and big data for real-time construction workforce management.

[18] Ananya Sharma. Business analytics as a catalyst of innovation in digital commerce. *International Journal of Applied Business and Management Innovation (IJAMBI)*, . URL https://journals.femington.com/ijambi/article/view/235/109.

[19] Sameeruddin Shaik. Enhancing security in distributed event-based systems using ai/ml models. URL https://ieeexplore.ieee.org/document/11158882.

[20] Dhanush Gopal. Greedy analytics for card rewards: A data-driven portfolio optimizer with simulation-backed roi insights. doi: 10.1109/ICSADL67539.2026.11451827.

[21] Praval Panwar. Quantifying predictive structure in complex time series: A model-free approach using weighted permutation entropy. URL https://ieeexplore.ieee.org/abstract/document/11339826.

[22] Dilip Sharma. Smart cloud and iot architectures for ai-powered personalization and scientific data systems. . doi: 10.1109/ICRTCST68392.2026.11545353.

[23] Monisha Rengaraj. Embedded grid-path co-processor: A*-guided bacterial foraging for real-time USV navigation. 2025. doi: 10.67231/8s96mc67.

[24] Rayhan Khan. Development of an AI for versus tetris using evolutionary algorithms. 2025. doi: 10.67231/htp1hp30.

[25] Afsar Khan. Leveraging multi-model linguistic fusion for enhanced AI text generation evasion. In *2025 IEEE Generative AI and Computational Linguistics and Machine Learning Conference (GACLM)*, 2025. doi: 10.1109/GACLM67198.2025.11231821.

[26] Muskan Pawar. Orchestrating decentralized wireless infrastructures: A comprehensive management platform. 2025. doi: 10.67231/4z837x12.

[27] Het Rachhadiya. Public sector analytics for territorial healthcare planning: Federated learning under the european health data space. doi: 10.67231/a8hswk54.