Event Knowledge Graphs for Auditing: A Case Study

Eva L. Klijn[1][0000−0001−9270−4774], Dennis Preuss[2], Lulzim Imeri[2], Florin Baumann[2], Felix Mannhardt[1][0000−0003−1733−777X], and Dirk Fahland[1][0000−0002−1993−9363]

[1] Eindhoven University of Technology, the Netherlands
{e.l.klijn,f.mannhardt,d.fahland}@tue.nl
[2] Ernst & Young AG, Switzerland
{dennis.preuss,lulzim.imeri,florin.baumann}@ch.ey.com

Abstract. Due to its potential benefits, process mining has become more and more embedded in financial auditing as an analysis technique to support the auditor in their assessment of the design and operating effectiveness of internal controls executed in financially relevant processes. However, standard process mining solutions for audit are developed under the pretense of a single case notion. As a result, an auditor is presented with models and data visualizations of the process that do not accurately reflect the underlying relationship between accounting and other relevant objects in the process, posing challenges for the auditor in obtaining a precise understanding of the process and related controls. In this case study together with EY, we aim to understand requirements for improving the application of process mining in audit. After first inventorizing the current limitations, we explore on a real-life audit use case provided by EY the benefits of graph-based event data representation using an event knowledge graph, especially considering accounting related objects and events. Discussing these results with auditing experts at EY revealed insights and requirements for a process mining analysis in the context of auditing not documented in the literature before.

Keywords: financial auditing · multiple objects · knowledge graph · querying

1 Introduction

An organization's published financial statements are main trusted sources for economies and capital markets. Many stakeholder rely in their decision making upon the information published [14]. To provide assurance that the financial statements of an organization represent a true and fair view, a *financial audit* is performed. Using process mining (PM) on recorded event data in such audit has been shown to be useful in supporting an auditor in assessing the design and operating effectiveness of companies internal controls, as it provides a comprehensive and faithful view of processes [9].

The application of process mining for auditing and specifically financial auditing has been researched [7–9,12–14]. Using existing process mining tools forces the auditor into an undesired trade-off decision early in the analysis when picking a case identifier to build the event log [7], e.g., in a purchasing process, the case could be chosen at the higher level of purchase order headers or each individual line item of the purchase order. Either choice has drawbacks [7] as flattening the relational source data into a sequential event log causes *convergence* (event duplication) and *divergence* (false behavioral

dependencies) [1] that complicate the audit. Werner et al. [12] avoid the trade-offs and drawbacks by not flattening the data under a particular case, but directly constructing from data attributes a *graph of related events* for financial auditing. However, their resulting graph structure cannot be directly used by process mining solutions. Alternative, graph- and object-centric event data representations have been proposed [1, 4] but their application to auditing in practice has not yet been researched.

This paper reports on a case study conducted together with Ernst & Young (EY) with the objective of exploring the challenges of transitioning from a classical process mining analysis to a graph-based process mining analysis for financial auditing. We selected purchase-to-pay (P2P) as standard process and were provided with an anonymized, non-client attributed data set and process description from a real-life case (Sect. 2). Reflecting on the current use of sequential event logs, auditing experts from EY confirmed the known trade-offs and drawbacks [7] but also raised analytical challenges that future process mining solutions in audit have to overcome, specifically the need for flexible multi-perspective views on events and objects at different granularity levels is not fulfilled (Sect. 3). To explore whether graph-based event data models and visualizations meet these challenges, we transformed the ERP data of the P2P case into an Event Knowledge Graph (EKG) [4, 5] and designed a prototype visualization for auditing in an open-source graph database and visualization software (Sect. 4). We confronted the EY auditing experts with this visualization to obtain feedback whether this representation avoided the trade-offs and addressed the challenges through a more realistic and accurate view of the process from an auditing perspective (Sect. 5). We found that the graph, indeed, makes it easier for auditors to understand how business and accounting objects are interrelated and whether controls are violated. It provides the required flexibility to subset the event data for different objects and, therefore, switch between different perspectives. We reflect on the implications in Sect. 6.

2 Context and Use Case

We first recall financial auditing and its use case in process mining (Sect. 2.1) after which we introduce our use case and the ERP source data (Sect. 2.2).

2.1 Process Mining for Financial Auditing

Financial auditing is the process of examining financial statements of an organization with the purpose of providing reasonable assurance that the statements represent a true and fair view [9]. Financial audits are conducted by external auditors; larger organizations also have an internal audit department that assesses the broad scope of functioning of the organization, e.g., its operations or corporate governance.

Due to its potential benefits, process mining (PM) has become more and more embedded in financial auditing as an analysis technique to support the external auditor in their assessment of the design and operating effectiveness of internal controls [14]. This procedure, shaped by the "International Standards on Auditing" (ISA), consists of four main phases: (1) understanding the entity, e.g., the organization, and its environment,

including its internal control (ISA 315), (2) identifying and assessing the risk of material misstatement (ISA 315), (3) auditor's responses to assessed risks (ISA 330), and (4) forming an opinion and reporting on financial statements (ISA 700) [14].

In this study, we aim to understand requirements for improving the application of PM in the *first phase* of an *external* audit. Here, an auditor has to understand the entity and its environment, including the entity's internal control with the aim to asses the *design* and *operating* effectiveness of a control. In general, a control should prevent the risk that a certain type of behavior or transaction results in material financial misstatements; an example of a control is the approval of a purchase order each time a change is made to it. The primary benefit of PM in this phase is that it can produce more comprehensive models and data visualizations of the actual process which allows an auditor to, instead of a sample, visualize everything from the process. If the model/data visualization is "faithful" to reality, the auditor can inspect it to assess whether implemented controls are present or missing, i.e., are designed effectively [9]. If this is the case, i.e., if the auditor concludes the control prevents a significant risk, they test the operating effectiveness of the control. The typical standard of a mined process model in industrial PM solutions is a directly follows graph (DFG) which shows how the different steps in the process, e.g., creation of a PO or creation of an invoice, are sequenced.

2.2 Process, Controls, and Source Data in the Case Study

To assess the properties of current and other data models for an auditor in concluding the effectiveness of a control, EY provided us with an anonymized, non-client attributed data set representing a real-life purchase-to-pay process (P2P). We describe the process, auditing controls, and source data considered in the case study.

Process. A purchase-to-pay (P2P) process is a standard operational process aimed at procuring goods or services for an organization. The specific process handles *documents* of five different types that are created and updated throughout its execution.

The process starts with the creation of a *purchase order* (PO), a document containing one or multiple *purchase order line items* (POLs) detailing the goods or services being procured. From a data and auditing perspective, a PO and POL are considered two distinct types of entities; the PO stores information about, e.g., the vendor, purchasing organization or the total PO value and the POL stores item specific information, e.g., its quantity and price. Once created, the PO is sent to the supplier. When (part of) the goods and/or services, i.e., a number of POLs requested in the PO, are delivered, a *goods receipt* (GR) is created: a document stating that the goods entered the company's warehouse. Receiving and recording a supplier invoice continues the process which generates an *invoice* document (INV): a document in the ERP system recording the financial transaction which serves as a request for payment. Receipt of goods and invoicing both lead to an increase in the company's inventory and liabilities it has towards its suppliers and, as such, both a goods receipt and invoice relate to an *accounting document* (AD). In the end, the accounting document related to the invoice is settled through another accounting document, which is typically the payment to the supplier.

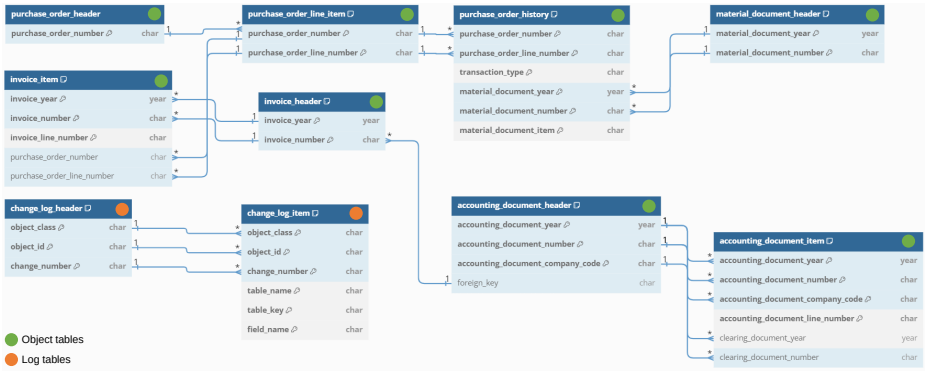

Fig. 1. Simplified database schema from source system provided by EY. Note: the change log item table has an n:1 relationship to each object table (omitted for simplicity).

Control. A key control that is assessed in the first phase of the audit for its operating effectiveness is the "*PO approval*", which is in place to prevent the risk of unauthorized procurement leading to potential paying of unauthorized assets or expenses. During the audit, auditors assess whether the control is designed in a way which addresses the above mentioned business risk. For example, if the POL quantity or price is changed then effective control design would trigger re-approval of the entire PO, as the total value of the PO changed. Similar behavior is expected if new procurements, i.e., new POLs, are added to an existing PO.

Source Data. Operational processes under audit, like the P2P process of our use case, track executions through records in a relational database (RDB). Its documents are stored in uniquely identifiable information records that are interrelated via 1:1, 1:n or n:m relations [11]. Fig. 1 shows the (simplified) data schema of the P2P process in our case study defining header and item-level objects for PO and POL (1:n relation), INV and INV item (1:n), AD and AD item (1:n). While PO refers to INV on the line item-level (1:n), INV refers to AD on the header level (n:1). Each creation or update of an object record is recorded as a timestamped *event* in a respective *change log* table (header and item-level). Each change log item table holds the object type as well as the object key that refers to the *object* that was updated (n:1, not shown in Fig.1). The change log header table holds various information like the timestamp and user who performed the change.

3 Challenges of Event Data Representation in Auditing

We reviewed the challenges of extracting event logs from an RDB for a PM-based audit with experts from EY. We confirm trade-offs reported previously, distill resulting analytical challenges, and discuss data models to overcome them.

Event Log Extraction for PM-Based Audit: Choices and Implications In current PM-based audit practice, data from an RDB, e.g., of a P2P process, is first extracted into a *sequential event log* based on a specific object chosen as *case identifier*. Technically, all events (indirectly) related to the case identifier are grouped into a *trace* [2, 5]. The

choice of case identifier depends on the analysis goal and the cardinality of the relations between object in the process [10]. However, each of the choices has known drawbacks for auditing practice [7] that were confirmed by EY's experts.

Current audit practice prefers an *item-level document as case identifier* to construct the event log, e.g., the POL in our P2P data. The trace of one POL contains all event records of any (relevant) change table that is related (via other objects) to the POL record [6]. Any event related to multiple POLs is duplicated (into each corresponding trace), known as *convergence* [1]. In such logs, a single payment event (from an AD header change table, Fig. 1) can be extracted for multiple POLs via two 1:n relationships: the data states more payments than actually happened.

Choosing a *header-level object as case identifier*, e.g., AD header, would in principle allow to analyze behavior on a document level, e.g., an entire payment. However, the resulting log orders events of unrelated objects into a sequence resulting in false behavioral dependencies, known as *divergence* [1]: it would become impossible to derive from the process graph which AD-related events relate to which POL [5].

Finally, the large amount of data kept within a single RDB of a large enterprise requires pre-filtering during extraction, e.g., extract only data of the company within the audit scope. The challenge for an auditor lies in the organization of change logs by object type rather than by company. To ease this, the change log is pre-filtered to changes of relevant objects within a set time frame while object-related tables are filtered based on organizational attributes.

Analytical challenges. The choices in log extraction causes the following analytical challenges that were detailed by the auditing experts, resulting in requirements for future PM-based solutions.

Auditors in practice are not involved in the choices and transformations during event log extraction. They perceive the directly-follows graph (DFG) computed from the log as the truth and are not aware of implications of convergence and divergence. This requires extensive expertise to overcome leading to longer onboarding times for using PM in audit. Also, the more decisions are made in the data transformation, the more has to be explained about the audit result to the client. *Requirement: reduce the decision made in the data transformation.*

However, the main challenges of using a single-case DFG in audit are related to the performing of the audit itself. First, an auditor needs to validate the process graph, which, in practice, is often done by reconciling the figures from the accounting related events back to the balance sheet and income statement movements. However, because of the POL perspective DFG and the fact that often multiple invoices related to POLs are settled through the same payment (i.e., the accounting relevant event), the accounting relevant events, along with the payments values, are duplicated for each of the POLs. As a result, it is more difficult for the auditor to reconcile, as there are now multiple events with the same payment value, but only a single entry in the accounting ledger. *Requirement: Having no convergence/divergence would simplify things for the auditor.*

Second, to test the effectiveness of controls, the auditor needs to understand how the process perspective is related to the accounting perspective, in particular, how a process activity can impact objects and the effect this has on the control. (1) This involves inferring from the activities how the process flows between the different business objects

(PO, POL, INV, GR) and accounting objects (AD). These accounting objects are the financial transactions leading to the financial statements. The auditor can only assess the financial impact of an event if they understand the object that the event was involved in. A DFG only shows the flows between the different activities related to the objects but not the objects themselves, nor how these are structurally related. *Requirement: Auditor needs to more easily access the objects related to events.* (2) This also involves understanding at which level of granularity (line-item level or header level), a particular attribute is located and how to relate them correctly, e.g., relating prices in a POL (line-item) to price in AD (header). *Requirement: The auditor needs flexibility in relating objects of different levels*. (3) Different controls concern different levels of granularity. *Requirement: The auditor needs flexibility in changing perspectives on the data*.

Overall, the inherent limitations of flattening event data under a single case identifier forces auditors into undesirable trade-offs and complex decisions. The preference of auditing experts is to avoid these trade-offs altogether and to work with the data "as-is" (no duplication, no false behavioral dependencies) and to view it from different angles (any document, header or item level).

Graph-Based Data Formats in Auditing The problem of convergence and divergence in the context of auditing and its effect on event data representation to the auditor has been addressed before in other works [12,13]. In [13] and later more extensively in [12], the authors proposed a discovery method that exploits the accounting data's structural dependencies by modeling the data as a graph. This enabled to disentangle the control flow on a process instance level providing a more realistic and accurate view of the process from an accounting perspective. However, the graph concepts of [12] are not compatible with standard PM concepts [4].

These and other ideas of a graph-based model for event data [3, 12, 13] were generalized into the model of *event knowledge graphs* (EKGs) [4]. An event knowledge graph is a graph-based data model supported by graph DB systems. It models events and objects in a process as nodes and the relationships between them as edges. Each node or edge is typed with a *label* (nodes can be typed with multiple labels). *Properties* (attribute-value pairs) describe a node or edge further. An event is related to any number of objects it operated on by an edge with label *corr*. Two events related to same object that directly follow each other in time are connected by a *df*-edge (directly-follows). Two objects can be related to each other through structural relationships. Fig. 5 shows an EKG (events are shown orange, df-edges blue, corr edges orange). An EKG can be constructed automatically from event tables [4] and allow advanced query-based analysis and visualization of event data over multiple related data objects [5]. Compared to other object-centric data models, e.g., [1], an EKG allows to model behavioral dependencies between related objects [4], e.g., header and item-level documents. So far, the potential benefits and challenges of using EKGs in an auditing setting have not been analyzed yet.

4 Prototyping a Graph-Based Visualization for Audit

To investigate the potential benefits and challenges of a graph-based approach in the first phase of an external audit, we developed a prototype visualization based on an

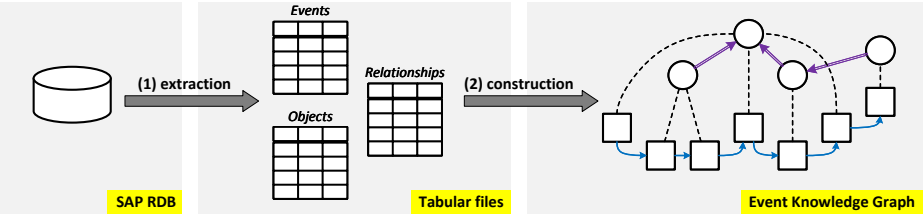

Fig. 2. Data transformation procedure.

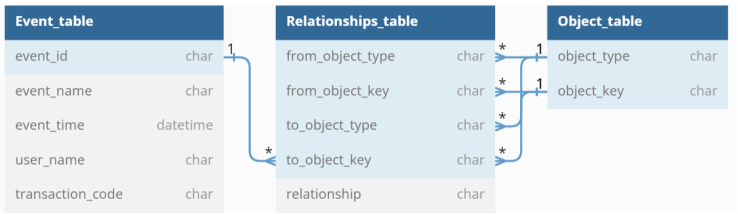

Fig. 3. Tabular data schema after database extraction (Fig. 2 middle).

EKG. We constructed an EKG directly from the relational ERP SAP data of the P2P process of Sect. 2 without extracting a classical event log in two steps shown in Fig. 2: first we extracted the data from the RDB into a tabular format anonymizing the data in the process (Sect. 4.1), and then we constructed the EKG in an open-source graph DB and used existing tools to create a prototype visualization (Sect. 4.2).

4.1 Event Data Extraction

Our input is event data stored in an ERP SAP relational database according to the schema shown in Fig. 1 as described in Sect. 2.2). As explained in Sect. 3, production systems contain a lot more data than is required for an analysis use case. Therefore, EY's auditing experts first filtered the data to a relevant subset as in any standard audit.

To enable EKG construction, the ERP data is extracted into events, objects and relationships according to the schema in Fig. 3 as described below. The following objects are taken into consideration: purchase order header, purchase order line item, goods receipt header, invoice header, accounting document line item.

First, the time-stamped records related to the above-mentioned objects are extracted into events by joining the respective object table of the ERP system with the change log [6]. For this study, events were further filtered by timestamp to limit the data to a three-month period that was considered sufficient for the exploration of the concepts. Generally, the scope of the data has, of course, to be evaluated based on the use case of each project. Each event record also includes the foreign key to the object record to which it was joined, so events can be related to objects in a later step. Each activity requires its own SQL query. For some activities, e.g., the posting of the AD, additional object tables are joined to limit the ADs related to INV with POs. All events are inserted into the same table.

Second, we extract from the ERP system's object tables (Fig. 1) all object records for which an event was extracted into the object table of Fig. 3.

Then we build the relationship table by joining tables according to the ERP system's data model (Fig. 1). For example, the purchase order line item is *a child of* a purchase order header, or the invoice header *posts* an accounting document line item. We extract records of such joins into the relationship table of Fig. 3. Also, relationships from extracted events to objects are inserted into the relationship table based on the foreign keys in the event records.

The output of this step is an *object table* with 304 777 objects of 5 different types, an *event table* containing 333 358 events having 10 distinct activities, and, a *relationship table* containing 560 975 relationships. Next, we translate these tables into an event knowledge graph.

4.2 Event Knowledge Graph Construction and Visualization

We constructed the EKG in two steps, thereby extending the original EKG construction procedure of [4]:

Event and Object Import. While [4] only imports event records and then infers objects and relations from event attributes based on domain knowledge, we already have full knowledge of objects and relations from the ERP export of Sect. 4.1 according to the schema of Fig. 3, which we use as follows. We import event records as event nodes (as in [4]) and adapted the import queries to also directly import object records as object nodes with a dedicated label based on the *object_type* specified in the input data. Relationships are imported by querying for each record in the relationships table the source and target node by their type and identifier and then creating a corresponding edge; we used indices on the event and object identifiers to increase performance. Import of the entire dataset required 2.8 minutes on an Intel i5 CPU 1.6 GHz machine with 24GB RAM. The result of this step is an EKG with 304 777 entity nodes, 333 358 event nodes, and 560 975 relationships (corr, and between objects, but no df-edges yet); Fig. 5 shows a sample.

Deriving behavioral relationships. To model the behavior of an object, i.e., the "trace of an object", we derive the df-edges using the query from [4] that (1) retrieves all events $e_1, ..., e_k$ correlated to an object o, (2) orders them by timestamp $e_i.time$ and (3) creates a *df*-edge between each subsequent pair of event nodes e_j, e_{j+1}. In Fig. 5, this results in the df-edge from the *Create POL* to the *Change POL* event related to the top-most POL object.

To model the interactions *between* related objects, we derive df-edges from the perspective of the relation between them. For this, we apply a number of queries from [4]: (1) for each structural relationship in the graph, we first query the objects o_i, o_j involved in the relationship, and, we create a new "derived" object $o_{i \times j}$ and relate it to o_i and o_j via relationships of type *derived* (2) and for each event e correlated to o_i or to o_j, we create a *corr* relationship from e to $o_{i \times j}$; (3) we derive *df*-relationships for each derived object like explained in the previous. For example, in Fig. 5 this results in the df-edge from the *Create POL* to the *Create PO* event between the top-most POL and is parent PO object (marked (3) in Fig. 5).

Fig. 4 shows the schema of the resulting EKG.

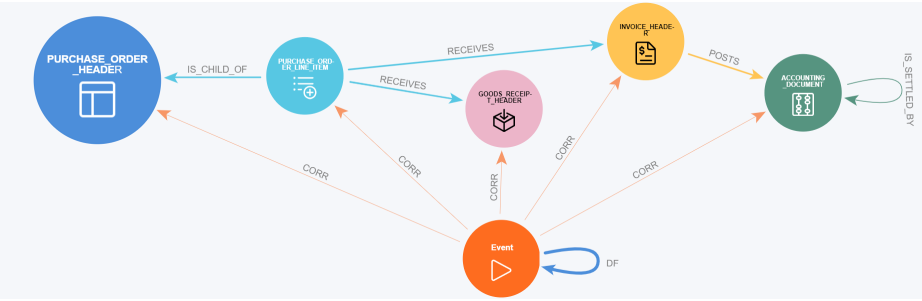

Fig. 4. Graph DB schema after event knowledge graph construction (Fig. 2 right).

Prototype visualization. In order to get feedback on the suitability of a graph-based representation for auditing, we prototyped a visualization of the EKG using Neo4j Bloom on a subset of the data selected by EY. We queried the EKG for all documents and events related to a specific AD that was known to EY to involve the auditing challenges of Sect. 3. We visualized the 5 object types, events, and relations of the schema (Fig. 4, omitting "derived" objects) using distinct colors, icons, and node sizes, and line width to distinguish types. We laid out the graph manually (events placed beneath their related objects, events ordered horizontally according to df-relations).

Fig. 5 shows the chosen subset: it contains two POs (dark blue nodes) related to three POLs (light blue nodes). Each POL receives one INV document (yellow node) that posts to one respective AD (green node); all ADs are settled by a single final AD. There are two additional sets of POLs, INV, ADs (marked (4) and (5) in Fig. 5) whose preceding POs are outside the extracted 3-month window. Each document has related event nodes (orange); the df-edges flow "in parallel" with the document relations and the derived df-edges (marked (1-3) in Fig. 5) show flow between line-item documents related to the same header. Importantly, the graph explicitly shows *each* financial transaction (ADs) of this execution as individual (green) nodes.

Limitations. Data quality issues and choices mode in the extraction step limit our prototype. A difference in timestamp granularity between the creation of POs and POLs results in POLs being created before POs (e.g., (3) in Fig. 5), which does not reflect reality. Limiting just the event extraction by a strict time frame led to objects missing events and missing related objects which occurred outside the time frame, e.g., missing Create POL at Fig. 5 (4) and the INV at Fig. 5 (5) is not related to any POL.

5 Results

Feedback. We confronted the auditing experts at EY with the prototype visualization to receive feedback in relation to the challenges of Sect. 3; their feedback is summarized below.

No event duplication. The experts immediately recognized that no event was duplicated (no convergence) and the graph matches the process description (Sect. 2.2): it correctly describes that its right-most *Settle AD* event is a payment to the 5 INV documents that "flow" into it. Compared to extraction to sequential event logs, where the

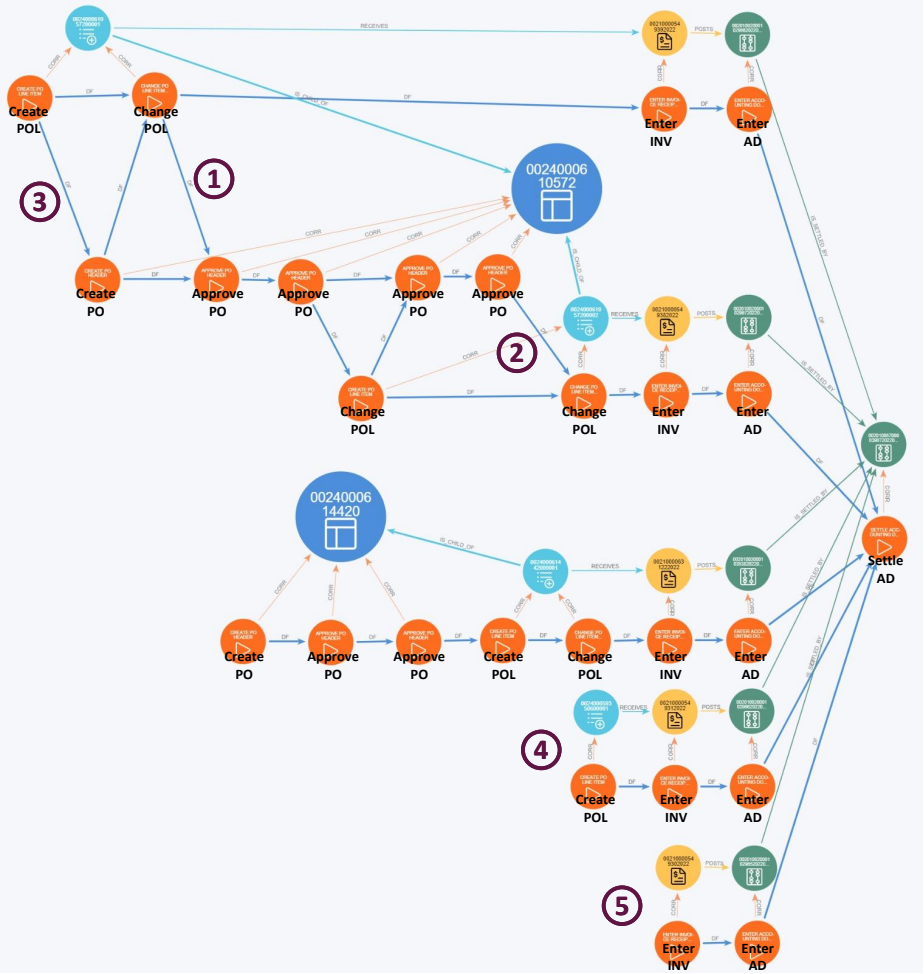

Fig. 5. Prototypical example of two PO objects and related objects.

same *Settle AD* event would be extracted into 5 different traces, the auditor no longer has to spend time to reconcile multiple traces. Also, seeing how every object trace in the graph eventually ends up in the same payment enables a different view and understanding of the process and the auditing task.

Multiple levels of granularity related. As each event is clearly associated to a particular object on a particular level (item or header), experts find it easier to understand that approvals to POL changes happen at the header level, while a POL change happens at the item level. The df-relations between objects of different levels (via "derived objects"), reveal which PO approval relates to (i.e., follows) which POL changes giving a larger-scale understanding missing in a POL-oriented event log (see Sect. 3). As a result it is easier to assess whether there is a violation of this particular control. For example in Fig. 5 (1) we see an instance where this control is working, i.e., a POL change is later approved on the PO level, whereas Fig. 5 (2) shows a violation of this control, i.e.,

a POL is changed and, without further approval, an invoice is entered and eventually paid.

Layout shows the process along business and accounting objects. By laying out object nodes in parallel with the df-edges "forward in time", the graph shows how multiple POLs (blue nodes) "flow into" multiple accounting documents (green node) stating multiple related financial transactions. In contrast, using classical event logs, the auditor has the burden to mentally reconstruct these relations from the source data.

Overall, the ability to read the graph from different perspectives (objects at header and item-level granularity) *opens up the possibility for header-level oriented auditing analysis* that considers larger-scale patterns without suffering from convergence / divergence errors.

Brainstorming. Reflecting on the properties of the prototype led the auditing experts to generate further potential benefits and requirements for graph-based models in auditing which we summarize next.

Subsetting in a graph allows an auditor to quickly get to the level of detail that is tied to certain objects relevant for a control, e.g., POL amounts, vendor, company codes; in logs the relation between objects and attributes for subsetting is more implicit. But the graph potentially enables further use cases: Subsetting based on an object would allow an auditor to assess controls specific to that object, e.g., invoice approval. Subsetting based on interactions between different objects would allow an auditor to assess controls across those objects, e.g. the control of the PO approval which involves both the PO and POL object.

Aggregation wrt. particular objects, relations, or patterns would enable auditors to understand controls on a higher-level of granularity, e.g., by summarizing item-level events to the related header level.

Recognize impact on controls. If an auditor understands which nodes (objects, events) in the graph relate to a specific control, then the graph enables an auditor to understand which events and objects impact which controls (and whether one event or object impacts multiple controls). For example, adding the user as an "object" allows to observe batch work [5] in turn enabling the auditor to assess whether controls are implemented correctly in all circumstances, e.g., also in case of batch processing 10 subsequent invoice approvals in a short time-frame. This is not possible in a DFG based on a sequential event log.

The approach enables to *inter-link multiple processes* in a single graph, enabling, for instance, to assess whether changes to master data involved in an object impacted a particular control, which is laborious in a classical setting.

6 Conclusion

This paper presents the results of a case study conducted together with EY on the application of graph-based process mining for financial auditing of a real-world P2P process. The difficulties encountered when extracting event logs from the ERP system of the process led to the identification of analytical challenges and requirements for process mining when used in financial auditing. We built a graph-based event representation

using an EKG directly from the ERP system data and evaluated this representation for its suitability to meet the identified requirements.

We found that the graph-based representation shows potential in overcoming the technical and analytical challenges that arise in PM-based audits on case-based event logs. This strongly suggests that future research on PM-based auditing solutions should systematically address these challenges. Another advantage is the flexibility of using open source software like the one used in our case study. This enables companies to focus much more on the use cases and its benefits before investing in a broad process mining software implementation. Analysis and queries can be executed in an open source implementation. This helps to sharpen the audit use cases without the need for a full fledged process mining solution implementation / roll-out.

Our findings are limited by the exploration of a single process and data set, suggesting further studies for robustness. Moreover, the prototype visualization was based on a manually created layout that significantly helped conveying the structure of the event data. Substantial research in better visualizations in process mining is required for automation.

References

1. van der Aalst, W.M.P.: Object-centric process mining: Dealing with divergence and convergence in event data. In: SEFM 2019. LNCS, vol. 11724, pp. 3–25. Springer (2019)
2. Accorsi, R., Lebherz, J.: A practitioner's view on process mining adoption, event log engineering and data challenges. In: Process Mining Handbook, LNBIP, vol. 448, pp. 212–240. Springer (2022)
3. Berti, A., van der Aalst, W.M.P.: Extracting multiple viewpoint models from relational databases. In: SIMPDA 2018 & 2019. LNBIP, vol. 379, pp. 24–51. Springer (2019)
4. Esser, S., Fahland, D.: Multi-dimensional event data in graph databases. J. Data Semant. **10**, 109–141 (2021)
5. Fahland, D.: Process mining over multiple behavioral dimensions with event knowledge graphs. In: Process Mining Handbook, LNBIP, vol. 448, pp. 274–319. Springer (2022)
6. Fahland, D., de Leoni, M., van Dongen, B.F., van der Aalst, W.M.P.: Behavioral conformance of artifact-centric process models. In: BIS 2011. LNBIP, vol. 87, pp. 37–49. Springer (2011)
7. Jans, M.: Auditor choices during event log building for process mining. J. Emerg. Technol. Account. **16**(2), 59–67 (Aug 2019)
8. Jans, M., Alles, M.G., Vasarhelyi, M.A.: A field study on the use of process mining of event logs as an analytical procedure in auditing. The Accounting Review **89**(5), 1751–1773 (May 2014)
9. Jans, M., Eulerich, M.: Process mining for financial auditing. In: Process Mining Handbook, LNBIP, vol. 448, pp. 445–467. Springer (2022)
10. Jans, M., Soffer, P.: From relational database to event log: Decisions with quality impact. In: Business Process Management Workshops. LNBIP, vol. 308, pp. 588–599. Springer (2017)
11. Lu, X., Nagelkerke, M., van de Wiel, D., Fahland, D.: Discovering interacting artifacts from ERP systems. IEEE Trans. Serv. Comput. **8**(6), 861–873 (2015)
12. Werner, M.: Financial process mining - accounting data structure dependent control flow inference. Int. J. Account. Inf. Syst. **25**, 57–80 (2017)
13. Werner, M., Gehrke, N.: Multilevel process mining for financial audits. IEEE Trans. Serv. Comput. **8**(6), 820–832 (2015)
14. Werner, M., Wiese, M., Maas, A.: Embedding process mining into financial statement audits. Int. J. Account. Inf. Syst. **41**, 100514 (2021)