Accounting Data Structure Dependent Control Flow Inference for Process Mining in Financial Audits

Abstract

The increasing integration of computer technology for the processing of business transactions and the growing amount of generated financially relevant data in organizations create new challenges for external auditors who are responsible to assess the true and fairness of financial statements. The availability of digital data opens up new opportunities for innovative audit procedures. Process mining can be used as a novel data analysis technique to support auditors in this context. Process mining algorithms produce process models by analyzing recorded event logs. Contemporary general purpose mining algorithms commonly use the temporal order of recorded events for determining the control flow in mined process models. The presented research shows how data dependencies related to the accounting structure of recorded events can be used as an alternative to the temporal order of events for discovering the control flow. The generated models provide accurate information on the control flow from an accounting perspective and show a lower complexity compared to those generated using timestamp dependencies. The presented research follows a design science research approach and uses three different real world data sets for evaluation purposes.

Keywords: process mining; financial audits; business process modeling; control flow inference; design science research; enterprise resource planning systems

1 Introduction

Data explosion (Krcmar, 2010) and information overflow (van der Aalst, 2011a) are well known phenomena that accompany the increasing integration of information technology into society and business. The availability of large amounts of digital data provides extensive opportunities for novel data analyses. This study focusses on the analysis of large data sets in the context of financial audits.

Financial audits are carried out by external auditors. They are an important control mechanism in order to enable stakeholders - such as investors or financial analysts - to make decisions based on information that provides a true and fair view of the financial position and performance of the reporting entity. The availability of abundant digital data opens up new ways for improving financial audits. However, it also introduces new challenges to the audit profession as it requires additional skills to deal with data that exhibits characteristics which are commonly discussed under the term Big Data (Chen et al., 2012) in academia and society.

Financial year-end audits encompass different audit activities. An important aspect in contemporary risk-based audits is the testing of internal controls over financial reporting (Rittenberg et al., 2008). In order to be able to assess the effectiveness of internal controls the auditor has to gain an understanding of how the internal controls affect the business processes in an organization. The activities carried out in order to process business transactions create the entries on the financial accounts. An auditor requires an understanding of how the audited entity's internal controls, business processes and financial accounts relate to each other in order to be able to carry out a financial audit effectively. The International Standard on Auditing 315(Revised) requires that

> "the auditor shall obtain an understanding of the information system, including the related business processes, relevant to financial reporting (…)" (IFAC, 2012, sec. 18).[1]

[1] Similar requirements can be found in national audit standards such as the Auditing Standard No. 12 (PCAOB, 2010) or the IDW-PS 261 (IDW, 2009)

The rationale to audit business processes, related internal controls and information systems is the assumption that well-controlled processes lead to complete, accurate, valid and authorized postings on the financial accounts.

The audit of internal controls becomes more complex with the increase of computer technology that is used for transaction processing. Processes become more complex, are automated and generate amounts of data which can hardly be handled with traditional audit procedures. The results are large data sets that form the basis for the preparation of financial statements which the auditor has to issue an opinion upon.

Data analytics can be used to deal with large data sets. Yet, they are rarely used in the context of financial audits. Scientific publications related to the analysis of large journal entry data sets are almost absent.[2] The study at hand focusses on process mining as a novel data analysis technique to improve process audits as part of financial statement audits.

Process mining is a specific data analysis technique that uses recorded event log data to provide information about business processes. Process mining algorithms produce models by analyzing the available source event logs. They can potentially be used in the context of financial audits to create reliable process models for audit purposes effectively and efficiently. During the testing of internal controls auditors assess if the business processes that lead to the entries on the financial accounts are well controlled in order to ensure that only complete, accurate, valid and authorized transactions are recorded. Auditors, for example, assess if a procurement process is properly controlled to ensure that material postings on the relevant financial accounts, such as raw materials or trade payables, are correct.

The type of information needed for process audits is contemporarily collected manually in a time-consuming and error-prone way. External auditors traditionally employ manual audit procedures like interviews and the inspection of source documents during process audits. The application of process mining techniques would enable the auditor to generate reliable process models as a source of information automatically. It would make manual data collection procedures for this purpose obsolete. Instead of collecting relevant information via interviews during walkthroughs

[2] Debreceny and Gray (2010) conclude that a large body of literature regarding data mining does exist for various application domains but their study reveals that there is no literature related to data mining on recorded journal entries.

and the inspection of a very limited number of sampled source documents process mining can potentially be used to produce reliable process models automatically by considering the entirety of recorded transactions. Related to the aforementioned example of a procurement process auditors would not need to carry out time-consuming and error-prone interviews. Instead they could use process mining techniques to create models of the procurement process based on the actually recorded event data. Automating the model generation would set free resources that could then be spent on the actual testing of internal controls and detected deviations from standard procedures.

Process mining projects are commonly carried out in different stages (van der Aalst et al., 2012): planning and justification (stage 0), data extraction (stage 1), connecting the event log and creating the control flow (stage 2), the creation of integrated process models (stage 3) and operational support (stage 4). This study focusses on the determination of the control flow (stage 2).

In order to be useful for process audits the mined models have to represent the underlying data correctly and they must be readable to the auditors. The control flow provides information about the structure of a process. It describes the sequence of activities that are carried out within a process. Contemporary general purpose process mining algorithms use the temporal order of events to infer the control flow in mined process models. They require a specific structure of the event log (van der Aalst, 2011a, chap. 4.2) in order to operate correctly. Data from common source systems like Enterprise Resource Planning (ERP) systems usually do not store financially relevant event data to serve as input for general purpose process mining algorithms. Gehrke and Müller-Wickop (2010) introduce a technique that allows to create an event log from ERP source data by exploring specific data relationships between journal entries. Their approach is promising because it uses the universal structure of accounting data which is in principle independent of the ERP system used for transaction processing.

The algorithm introduced by Gehrke and Müller-Wickop (2010) maintains the original data structure but at the cost that the generated event log is not suitable for traditional general purpose mining algorithms. A main difference compared to traditional event logs is the characteristic that traces[3] in these types of event logs are not linear. In traditional event logs all events are strictly

[3] A single execution of a business process is called process instance. They are reflected in the event log as a set of events that are mapped to the same case. The sequence of recorded events in a case is called trace.

chronologically ordered. This means that each event has exactly one or no predecessor and one or no successor. This is not the case for financially relevant data that served as input for this study. The purchase of two different goods, for example, is often paid by a single payment. If an event log containing non-linear traces is used as input for traditional general purpose process mining these create overly complex models that, from an audit perspective, do not provide the necessary information required by the auditor.[4] The questions arise of how an event log, created on source data from ERP systems by exploiting the general structure of journal entries, can be used in order to infer the control flow of mined models? And how can this be done by simultaneously maintaining information on the relationship between business processes and the financial accounts?

This study answers these questions by introducing an alternative approach to infer the control flow. The creation and linkage of the event log itself is demonstrated by Gehrke and Müller-Wickop (2010). The main contribution of the study at hand is the presentation of a method to infer the control flow by exploiting specific data dependencies that relate to the accounting structure between recorded events instead of temporal dependencies. Journal entries recorded in ERP systems that follow a double-entry bookkeeping system and use an open item accounting structure show specific data dependencies. This study shows how these dependencies can be technically exploited to determine the control flow in mined process models. The aim is to provide process models that accurately represent the control flow of the underlying business processes according to the recorded journal entries. The generated models show a lower complexity due to the disentanglement of control flows on the process instance level. The model complexity in this context serves as a technical proxy for the usability of the mined models.

Debreceny and Curtis (2015) summarize that accounting and auditing nowadays almost completely rely on computerized information systems. The Pathways Commission sponsored by the American Accounting Association and American Institute of Certified Public Accountants warns that "if the accounting community continues to (…) not understanding the technology and dynamic business processes that run companies of the 21st century, the accounting profession has the potential to become obsolete" (AAA/AICPA, 2012, p. 68). An important aspect to prevent such a

[4] The event log can be transferred into a strictly linear event log as shown by Müller-Wickop and Schultz (2013) but this approach introduces the duplication of events and neglects the implication for data values that represent the entries on financial accounts..

development is the integration of information systems and the data that is stored in those systems to improve financial audits. Wang and Cuthbertson (2015) emphasize that the understanding and the use of data analytics in audit engagements are still limited. The research presented in this study contributes to the integration of research related to information systems to improve process audits.

The following section two discusses the research methodology that was used for this study. Section three draws on contemporary literature to provide background information on process mining, the role of business processes in financial statement audits and discusses how process mining can be employed in this context. It also provides technical information about the structure of recorded journal entries in ERP systems as well as event log preprocessing and case matching. The latter aspects are important prerequisites for the introduction of the alternative control flow inference method as the main research result in section four. The paper closes with a conclusion and outlook to future research in section five.

2 Methodology

The research presented in this study follows a design science research (DSR) approach (Hevner et al., 2004; March and Smith, 1995; Österle et al., 2010). The reason for choosing such an approach is the proximity of the research question to the practical problems in auditing and the objective to deliver artifacts that are valuable for the application domain.

DSR consists of the phases analysis, design, evaluation and diffusion (Österle et al., 2010). This study was embedded into a larger research project which ran through several research cycles. The overall project was concerned with the research question of how data analysis techniques can be used to support financial audits in general. During the analysis phase empirical investigations were conducted and published by Müller-Wickop et al. (2013) in order to identify the information needs of external auditors during a process audit. This study focusses on the design and evaluation phase of a particular research cycle with the aim to provide a method for the control flow inference of mined process models that can be used to improve financial audits.

The primary research methods for the design of the presented solutions were method engineering (Brinkkemper, 1996) and prototyping. Method engineering is commonly used in information systems science for the systematic design of methods (Wilde and Hess, 2007). A method in this context consists of different parts (method fragments) that can be combined and reused (Harmsen

et al., 1994). A new method can be engineered by combining existing method fragments in a new manner or by developing completely new method fragments. The method fragments described by Gehrke and Müller-Wickop (2010) and the Colored Petri Net specification presented in (Author Citation) served as input for the development of the control flow inference method described in this paper. This was implemented in a software prototype. Prototyping is traditionally used in software engineering (Naumann and Jenkins, 1982). It allows to develop a software artifact that implements the intended core functionality in iterative cycles. Prototyping was used as a research method in this study in order to develop a software artifact which could be used for the evaluation of the designed methods. Both control flow inference methods described in this study (traditional timestamp dependent and alternative accounting data structure dependent) were implemented in a software prototype on the basis of the Financial Process Mining (FPM) algorithm. The FPM algorithm was developed specifically for the context of financial audits (Gehrke and Müller-Wickop, 2010). The software prototype consists of an extraction module and a mining module.

Data Set	**Set 1**	**Set 2**	**Set 3**
Industry	Manufacturing	Media	Retail
Records in BKPF	1,764,773	156,604	92,487
Records in BSEG	7,395,434	559,506	222,901
Mined Process Instances	1,035,805	18,975	40,634
Mined Process Models	841	516	307

Table 1 Overview of Used Data Sets

The evaluation was carried out by conducting a computational simulation experiment.[5] Extracted data from ERP systems served as input for the experiment. The data was provided by companies from three different industries (manufacturing, media and retail) operating in Germany that participated in the research project. The data was extracted from the companies' operational SAP systems and consisted of data extractions of the database tables BKPF and BSEG. BKPF stores the data of recorded journal entries, BSEG of recorded journal entry items. An overview of the

[5] The conducted experiment differed from traditional experiments that are commonly employed in behavioral science-oriented research (Bhattacherjee, 2012). The subject of investigation in the design science-oriented experiments is the prototype itself (Riege et al., 2009) and the outcomes that are produced by its application.

used data sets is provided in Table 1. This data served as input for the prototype. The output were models that were tested by simulation. The mining results were further analyzed using descriptive statistics to evaluate the output quantitatively.

3 Background

3.1 Process Mining

Process mining is a research area that first emerged in the context of software engineering (Cook and Wolf, 1998, 1999). Research on process mining has matured in the past decades with the development of powerful heuristic (Weijters et al., 2006), fuzzy (Günther and van der Aalst, 2007) and genetic (de Medeiros, 2006) mining algorithms. The Process Mining Manifesto (van der Aalst et al., 2012) provides a comprehensive overview of contemporary challenges in process mining. An overview of basic and advanced concepts on process mining can be found in van der Aalst (2011b).

Process mining has already been successfully applied in the context of internal audits (Jans et al., 2008, 2010a, 2011, 2013, 2014) and financial audits (Author Citation). Several scholars have also used general data mining techniques for auditing (Debreceny and Gray, 2011) and internal fraud detection purposes (Debreceny and Gray, 2010; Jans et al., 2010b). Jans et al. (2014) used the data from a financial service company in order to apply process mining in the context of internal audits. They prepared an event log by exploiting the specific data structure of recorded events of a procurement process and used a general purpose process mining tool for mining. As described in Jans et al. (2013) they created an event log that is suitable to be processed by the Disco software tool (fluxicon, 2015). Their research actually uses existing process mining solutions and applies them to the context of internal auditing. The study at hand chooses a different path. It aims to develop solutions in forms of new methods in order to make the analysis tool fit the source data. By doing so the study contributes to the set of available process mining techniques. It introduces a control flow inference method that exploits the nature of journal entries. We are not aware of any other research that investigates the accounting data structure of events for the inference of the control flow in the context of financial audits.

A fundamental challenge in process mining is the creation of useful process models. Mined process models are often too complex for simple interpretation and analysis. Extremely complex

process models are referred to as *spaghetti* processes in the process mining community due to their graphical characteristics of intertwined arcs in the respective process models (van der Aalst, 2011b). Complexity reduction of process models has been addressed by scholars following different approaches. Advanced mining algorithms such as the Fuzzy Miner (Günther and van der Aalst, 2007) provide functionality to abstract from infrequent events and execution paths and are able to deliver less complex process models. However, infrequent behavior might indeed be relevant for compliance (Becker et al., 2012) and conformance checking purposes (van der Aalst and de Medeiros, 2005). Their omission in the process model might lead to process models that are of little use especially in the context of financial audits. Other scholars suggest abstraction methods such as aggregation and reduction to decrease complexity in process models (Reichert, 2012).

Approaches for organizational mining and social network analysis (Song and van der Aalst, 2008) focus on the resources that interact in a business process and exploit data referring to the relationship between process participants and activities to create models that illustrate organizational structures and social networks. Organizational mining also uses information from the event log other than the temporal order of events for the generation of models but with the objective to discover the interaction of process participants. It therefore differs from the approach used in this study.

The next subsections discuss the relationship between business processes and financial audits. They highlight different aspect that have to be accounted for if process mining is used in the context of financial audits. It also serves as the foundation to illustrate how the suggested procedure can be integrated into the overall audit process.

3.2 Financial Audits and Business Processes

A business process is a set of connected activities that in combination realize a specific business goal (Reichert and Weber, 2012). The audit of business processes and related internal controls is an essential part in contemporary risk-based approaches to audit financial statements (IDW, 2009; IFAC, 2012; PCAOB, 2010). It requires that the auditor has a sufficient understanding about the

relationship between business activities, financial accounts and internal controls.[6] External auditors use process models to gain an understanding of the audited business processes, to identify incomplete, inaccurate, invalid or unauthorized transaction processing and to assess the design effectiveness of internal controls. These process models are usually created using traditional audit procedures like interviews and inspections of available documents. Process models and textual descriptions in current audit practice are prepared manually by using simple general purpose modeling tools such as Microsoft Visio, Powerpoint or Word. These procedures are highly time-consuming and error-prone. They become inefficient or even ineffective if the level of process automation and the volume of processed data increases. Contact persons from the audited entity traditionally serve as an important information source to gain information via interviews about relevant business processes. If these are operated automatically without any or with just limited human interaction the contact persons may not possess the necessary knowledge about relevant business processes anymore and performed interviews might provide little if any reliable information. The high volume of transactions, for example in telecommunication companies, make manual investigations over randomly selected samples very inefficient.

3.3 Process Mining for Financial Audits

Process mining is a relatively novel research area which can potentially help to efficiently produce reliable process models. Process mining techniques can be used to create models in different phases of the audit process. An audit engagement commonly consists of five phases that are illustrated in Figure 1. Process models are particularly relevant in the phases of the design and effectiveness testing of internal controls. External auditors require information about the structure of a process and its impact on the financial accounts in order to decide if it has to be audited from a materiality perspective or if it can be neglected. Such information is also necessary in order to assess if internal controls are adequately implemented in order to prevent incomplete, incorrect, invalid or unauthorized transactions.

[6] The empirical investigation carried out by Müller-Wickop et al. (2013) illustrates that financial accounts, journal entries and internal controls are key concepts for process audits.

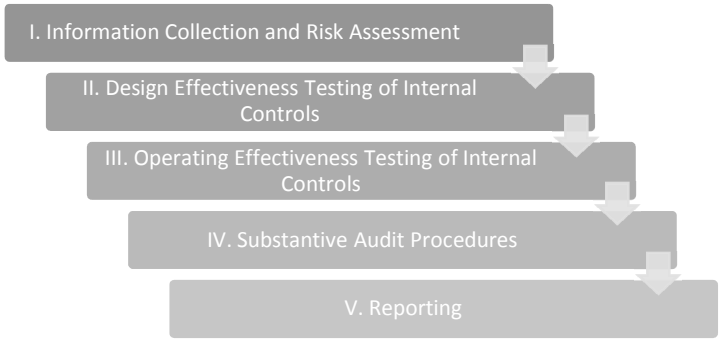

Figure 1 Financial Statement Audit Process

Process mining could be used by auditors to create process models in an automated way to improve the necessary information collection. However, these have to represent the underlying data adequately from an audit perspective. Process models generated via process mining can be assessed according to different quality criteria. It is generally necessary to achieve an adequate compromise between competing criteria depending on the requirements from the application domain. Depending on the used mining algorithm models usually show more behavior than recorded in the event log or less. It is commonly not possible to create models that fulfill all criteria completely with a single model. Rozinat and van der Aalst (2008) discuss four different quality criteria for mined process models - fitness, precision, generalizability and structure - that can be used in order to identify criteria that are important in the context of financial audits. Fitness refers to the ability of a process model to replay the behavior that was recorded in the event log.[7] Precision is the complementary criteria expressing that a process model should not allow behavior that was not recorded in the event log. Generalizability is the opposing criteria that refers to the ability of a process model to abstract from the source event log and to allow additional behavior that is not recorded in the log.[8] Structure, or in other publications also called simplicity (van der Aalst, 2011a, chap. 5.4.3), refers to the structural complexity of a mined model.

[7] A process model has a perfect fitness if it is able to replay all behavior that was recorded in the event log.

[8] The rationale to produce generalized process model is the assumption that a given event log is just an extract of reality that is most likely to be incomplete (van der Aalst, 2011a).

Fitting and precise models are important to prevent false positive and false negative audit results.[9] It is shown in (Author Citation) how perfectly fitting and highly precise process models can be generated by using recorded journal entries. This study investigates in detail how the source data extracted from ERP systems can be used in order infer the control flow of mined models by exploiting the underlying accounting data structure of recorded events. In terms of process model quality criteria this study focusses on the criteria of structure. It refers to the graphical layout of a model and it is related to the principle of Occam's Razor[10]. Ceteris paribus a process model with a lower complexity is superior to a model with a higher complexity.

This study shows how the specific data structure of recorded journal entries can be used to infer the control flow in mined process model. This approach simultaneously creates less complex models in comparison to results that can be achieved by applying timestamp dependent control flow inference.

3.4 Understanding the Structure of Recorded Journal Entries

ERP systems support and automate the operation of business processes. They produce entries on financial accounts when processing financially relevant transactions. Entries in financial accounts exhibit a specific structure that can be used to analyze the business processes that created these entries. Each execution of a financially relevant activity in a business process creates an accounting journal entry that is recorded.

---

[9] Incomplete, inaccurate, invalid or unauthorized transactions might not be represented in the mined process models leading to false positive audit results if the model has a low fitness. Process models might lead to false negative audit results if they show additional behavior (because of a low precision) that is interpreted by the auditor as incomplete, inaccurate, invalid or unauthorized transactions but that in reality did not occur. A low fitness can decrease the audit effectiveness if it leads to false positive audit results. A low precision can decrease the audit efficiency if it leads to false negative audit results.

[10] It refers to the principle that gives "(…) precedence to simplicity: of two competing theories, the simpler explanation of an entity is to be preferred" (Encyclopaedia Britannica, 2010). In terms of process mining "(…) one should look for the 'simplest model' that can explain what is observed in the data set." (van der Aalst, 2011a, p. 90).

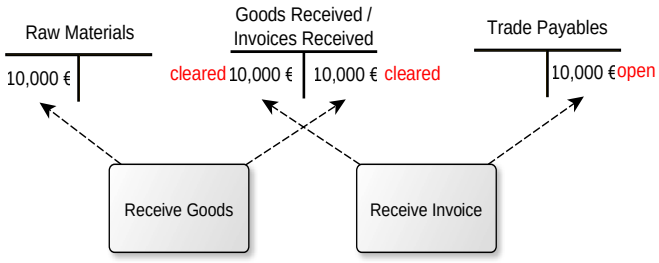

Figure 2 Example Business Process and Financial Accounts

A journal entry consists of at least two journal entry items, one on the debit and one on the credit side of a financial account. Open items on an account are cleared by items belonging to other journal entries. These are created by the execution of activities that belong to the same process instance. This relationship is illustrated in Figure 2. It shows the financial accounts with the corresponding journal entry items and the activities that created them.

This relationship can formally be described using an Entity-Relationship (ER) diagram as shown in Figure 3. Each rectangle represents an entity which is recorded as a table in the source ERP system. For each entity different attributes are recorded. Figure 3 shows all attributes that are relevant for this study. The attribute *TransactionCode* of the entity *Journal Entry,* for example, describes the name of the transaction in the underlying ERP system that is executed in order to process the corresponding activity. The transaction code in SAP ERP systems for the recording of incoming invoices, for example, is *MIRO*. The attribute *JournalEntryItemNr* of entity *Journal Entry Item* is also called *Line Number* in other systems. The abbreviations *PK* and *FK* stand for primary key and foreign key.

The lines between the rectangles represent the relationship types between the entity types. The numbers attached to the lines provide information about the cardinality of the relationship types. For the *'contains'* relationship these cardinalities mean that exactly one entity of the type *Journal Entry* (cardinality *1*) is always related to at least two different or more entities (cardinality *2…N*) of type *Journal Entry Item*. A journal entry represents a single event. When this event occurs a journal entry is created in the source system that contains two or more journal entry items. This relationship is illustrated via the *'contains'* relationship in the ER diagram. Furthermore, an executed transaction clears none, one or many open items which is expressed via the *'clears'* relationship (cardinality *0…N*). On the other hand not all journal entry items are cleared by any other

journal entry (cardinality *0…1*)[11]. The *Receive Invoice* activity illustrated in Figure 2 creates a journal entry that contains two journal entry items, one *10,000 € debit posting* on the *Goods Received / Invoices Received (GR/IR)* account and one *10,000 € credit posting* on the *Trade Payables* account. It also clears the *10,000 € debit posting* on the *GR/IR* account that was posted by activity *Receive Goods*.

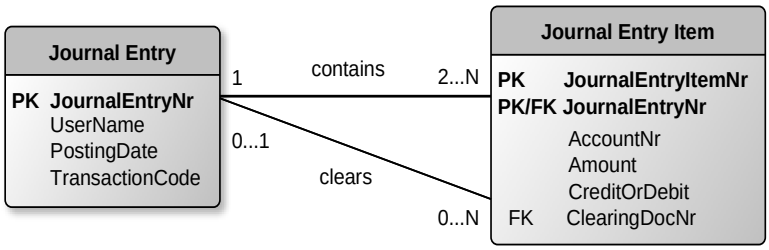

Figure 3 ER Diagram for Accounting Data Structure[12]

The ER diagram in Figure 3 can also be expressed with the relations:

JE = {JournalEntryNr, UserName, PostingDate, TransactionCode}

JEI = {JournalEntryItemNr, JournalEntryNr, PositionNr, AccountNr, Amount, CreditOrDebit, ClearingDocNr}

The attributes *JournalEntryNr* and *ClearingDocNr* in relation *JEI* are foreign keys referring to the primary key *JournalEntryNr* in relation *JE*.

[11] This is the case for those journal entry items that do not follow an open item accounting structure. It is also the case for incomplete process instances where not all open items have yet been cleared.

[12] Figure 3 shows the data structure of the underlying source system. From an accounting perspective a journal entry item is cleared by one or more other items posted on the opposite side of the same account. The used source systems do not link the clearing item directly to the cleared item but instead to the journal entry that created the clearing item. For process mining purposes this difference has no influence on the mining results as the algorithm can be adjusted depending on the actual implementation of data dependencies.

3.5 Event Log Preprocessing and Case Matching

The application of process mining techniques in real scenarios is commonly conducted in five different stages (van der Aalst et al., 2012). It starts with the planning and justification (stage 0) followed by the data extraction (stage 1). Next, the connection of the event log and the creation of the control flow is established (stage 2). The remaining stages (stages 3 and 4) deal with the creation of integrated process models and operational support.

A traditional event log is essentially a simple table as shown in Table 2. It consists of case IDs, event IDs and data attributes such as the activity name or user who executed the activity. A case represents a single execution of a business process. Each event represent an executed activity of the business process. The recorded event data in common ERP systems is currently not matched to cases. It is perceived as a data source of just medium data quality for process mining purposes (van der Aalst et al., 2012) as the relationship between events and cases is not automatically recorded with the event data itself. It has to be inferred during or after the data extraction. In Table 2 this would mean that the relationship between column *Case ID* and *Event ID* is not yet established. Column *Case ID* would be empty. For the inference of the control flow in the aforementioned process mining stage 2 it is therefore not clear which events recorded in the event log data belong to which case. However, the case matching[13] is a necessary precondition for any process mining algorithm to be able to determine the control flow and to complete stage 2.

The case matching for source data from ERP systems can be carried out by preprocessing the available event data. Gehrke and Müller-Wickop (2010) provide an algorithm in this context that can be used to match events to cases by exploiting the accounting data relationships between journal entries that are shown in Figure 3. The algorithm starts with an arbitrary journal entry $i \in JE$ and searches all journal entry items that belong to this journal entry. The result is a set of journal entry items $JEI_i \subseteq JEI$ where $JournalEntryNr_i \in JEI_i[\text{JournalEntryNr}]$. It then searches the journal entries that cleared the journal entry items posted by i. $CJE_i \subseteq JE$ is the set of journal entries c that cleared journal entry items posted by i where $JournalEntryNr_c \in JEI_i[\text{ClearingDocNr}]$. The algorithm then repeats the forward search of related journal items and clearing journal entries for all elements $c \in CJE_i$. The algorithm stops when all leaves in the graph

[13] Ferreira and Gillblad (2009) call the step of matching recorded events to cases labelling.

are found and then searches backwards to identify the journal entry items that were cleared by i with $CJEI_i \subseteq JEI$ where $JournalEntryNr_i \in JEI[\text{ClearingDocNr}]$. The forward and backward search is repeated until no further related journal entries and journal entry items are found. All journal entries and journal entry items that belong to the same process instance are marked in JE and JEI with the corresponding case ID. A watch list W is used to prevent that the same journal entries and journal entry items are traversed multiple times. The algorithm starts with the next $j \in JE \wedge j \notin W$ when the forward and backward search for i is finished. The result is an event log consisting of two tables where each journal entry and journal entry item is matched to a case.

The algorithm presented by Gehrke and Müller-Wickop can be used to match recorded events in an ERP system to cases. Applying their algorithm reveals an interesting aspect which is central to the study at hand. The event log that can be generated by using their algorithm consists of two tables. This data can be represented as a directed graph. An example of such a graph is shown in Figure 5. This graph represents a single process instance. The data structure of this instance is fundamentally different compared to the data which commonly serves as input for general purpose mining algorithms. Common event logs such as shown in Table 2 are linear. All events that belong to the same case are strictly ordered according to their time of execution (timestamp). This is not the case for the type of data shown in Figure 5. It shows parallelism of events on the process instance level. The reason for this constellation is explained in detail in the next section.

It could now be argued that the event log should be preprocessed in order to linearize it to make it suitable for general purpose mining algorithms. Müller-Wickop and Schultz (2013) present an algorithm that can be used to achieve this. They essentially suggest to cut a process instance that shows parallelism on the process instance level into separate smaller instances. Applying their approach to the example process instance would lead to an event log as shown in Appendix A. It contains four different cases. Essentially one case is created for each individual branch in the original instance. However, this approach creates several undesired side effects. First, several events in the event log are multiplied. This significantly disturbs subsequent quantitative analysis of the mined model. Second, the disintegration neglects the attached data values, in this case the postings on the financial accounts. It remains unclear how these should be treated. A simple multiplication is not possible because as a result the aggregate posting values shown in the mined models would not match the overall posting volume recorded on the financial accounts in the source system an-

ymore. This study therefore maintains the original data structure to produce adequate process models at the cost that the generated event log is not suitable for traditional general purpose mining algorithms anymore as shown in the next section.

To sum up, at this point it is important that the case matching is a necessary precondition to further analyze the source data, but it does not infer the control flow. The control flow defines the sequence of activities in a process and is a fundamental component in process models. The next section describes how it can be determined by using the accounting data relationships illustrated in Figure 3.

4 Accounting Data Structure Dependent Control Flow Inference

The previous section introduced background information crucial for describing the application domain and source data structure. This section investigates how the control flow can be determined by exploiting the aforementioned structure. It starts with the description of an example process instance in the next subsection which is used for illustration in the remainder of this section. The example describes a single execution of a business process to explain the differences and to compare the varying outputs generated by a traditional timestamp dependent approach versus an accounting data structure dependent control flow inference. Figure 4 provides an overview of the concept of abstraction levels that are commonly used in the context of business process management. It is useful to visualize the differences between business processes, business process instances and related model types.

Process mining algorithms usually produce process models that are located on level M1. A business process model is an abstraction of a business process and consists of a set of activity models and execution constraints between them (Weske, 2012).

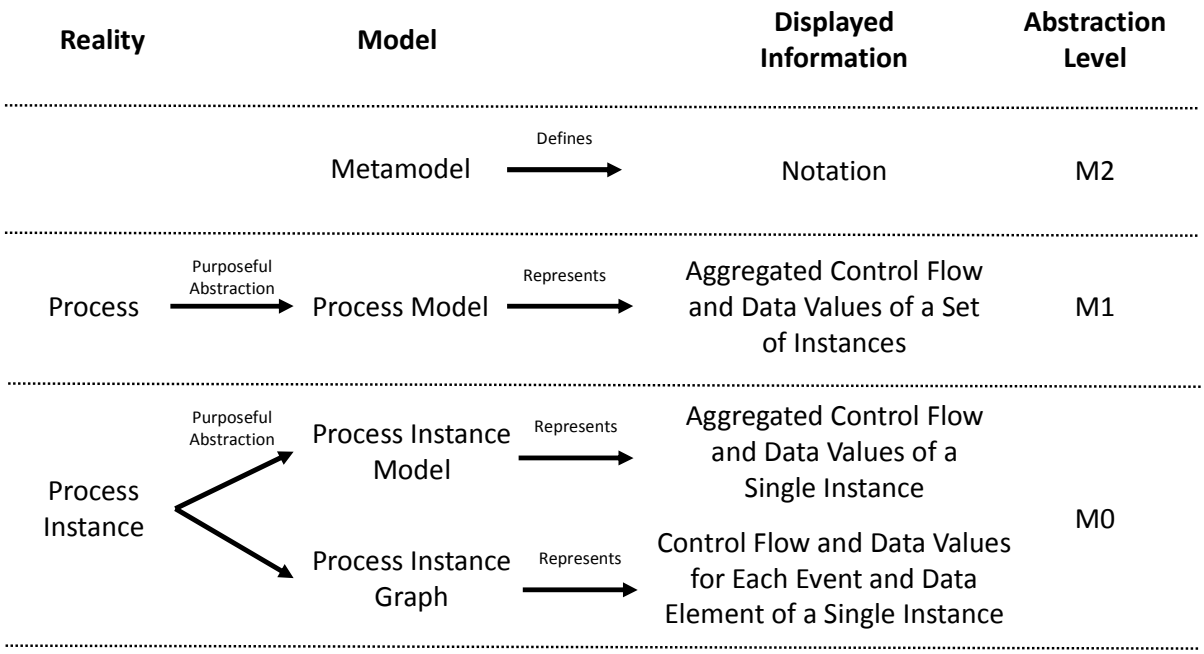

Figure 4 Horizontal Abstraction Levels in Business Process Management adapted from (Weske, 2012, p. 76)

A single execution of a business process is called process instance. Process models represent the behavior of a set of process instances that belong to the same business process. A model representing a single process instance is called process instance model. These models are located on level M0. Process models and process instance models generally include every activity only once. The represented activity models in process instance and process models are already abstractions of a set of executed activities. A process instance graph (van Dongen and van der Aalst, 2005) resides on level M0 as well but provides more details compared to a process instance model. It refers to a single process execution but each event is represented as a single activity in the model.

This study primarily refers to the process instance level with its related process graph and process instance models for ease of illustration. This abstraction level is used to unravel the reasons why both approached, timestamp vs. accounting data structure dependent control flow inference, create different results. However, the same mechanisms and conclusions are also valid for the process model level.

The next subsection 4.1 introduces the example followed by a demonstration of the output using traditional timestamp dependent control flow inference in subsection 4.2. Subsection 4.3 formalizes the specific data dependencies between journal entries necessary to determine the control

flow. Subsection 4.4 deals with specific data constellations observed in the source systems that have to be taken into account whereas subsection 4.5 summarizes the different method fragments and presents the output of the alternative control flow inference. Subsection 4.6 presents evaluations results before it is discussed of how the presented approach can generally be integrated into the overall audit process in subsection 4.7.

4.1 Example Process Instance

Table 2 provides the event log for the example that was derived from a company operating in the manufacturing industry.[14] It shows the event log of only a single process instance which was typical for the procurement process. It was deliberately chosen from data set #1 because it illustrates the average complexity of a mined process model and exhibits specific characteristics of the underlying source data.[15]

Case ID	Event ID (Journal Entry Number)	Timestamp (Posting Date)	Activity (Transaction Code)	User Name
1	0050155443	2010/01/02	Post Received Goods	User 6
1	0050155250	2010/02/08	Post Received Goods	User 7
1	0015975223	2010/02/17	Post Received Invoice	User 5
1	0015975224	2010/02/18	Post Received Invoice	User 5
1	0015975221	2010/02/19	Post Received Invoice	User 5
1	0095348327	2010/02/20	Clear Postings	User 1
1	0095348517	2010/02/21	Clear Postings	User 1
1	0050157332	2010/08/16	Post Received Goods	User 4
1	0015980342	2010/09/03	Post Received Invoice	User 3
1	0012490379	2010/09/04	Payment	User 2
1	0007904673	2010/09/05	Post with Clearing	User 1
1	0095359370	2010/09/07	Clear Postings	User 1

Table 2 Example Event Log

[14]The example is different to examples that are used in common accounting or accounting information systems literature like Considine et al., (2012), Gelinas (2014) or Romney and Steinbart (2008). However, this example was deliberately chosen because it illustrates the common structure of recorded journal entries in contemporary ERP systems.

[15] The average complexity of the mined models was 4.56 transitions per process model for data set #1, median 5, standard deviation 2.07, maximum 15, and minimum 1. The statistics for data sets #2 and #3 were similar.

Figure 5 illustrates the corresponding instance graph that can be generated by using the event log preprocessing and case matching described earlier in this study. The figure uses an extended version of the Business Process Modelling Notation (BPMN). This illustration uses specific symbols as introduced by Mueller-Wickop and Nuettgens (2014) for modelling financial accounts and account entries.[16] It provides information on the activities that were executed, involved financial accounts and posted values. The grey rectangles represent activities that were executed in the process. The BPMN group symbols (dotted colorless rectangles) represent the involved financial accounts. They consist of the account name and number at the top, the account symbol and credit or debit postings. These posting are modelled as BPMN data objects (paper symbols). The color of each of these object signifies if it is a debit or credit posting on a balance sheet (blue and yellow) or profit and loss account (red and green).

The dotted single-headed arrows leading from an activity to a posting denote that the corresponding activity posted a journal entry item on the connected account.[17] The dotted double-headed arrows denote that an entry item was cleared on the respective account by the connected activity.[18] The value of the posted or cleared item is displayed as an inscription for the corresponding data object.

[16] The used software implementation uses Colored Petri Nets (CPN) as a mathematical foundation (Jensen and Kristensen, 2009). The CPN models were manually transformed into BPMN models for illustration purposes in this study.

[17] These arrows represent the *'contains'* relationship illustrated in Figure 3.

[18] These arrows represent the *'clears'* relationship illustrated in Figure 3.

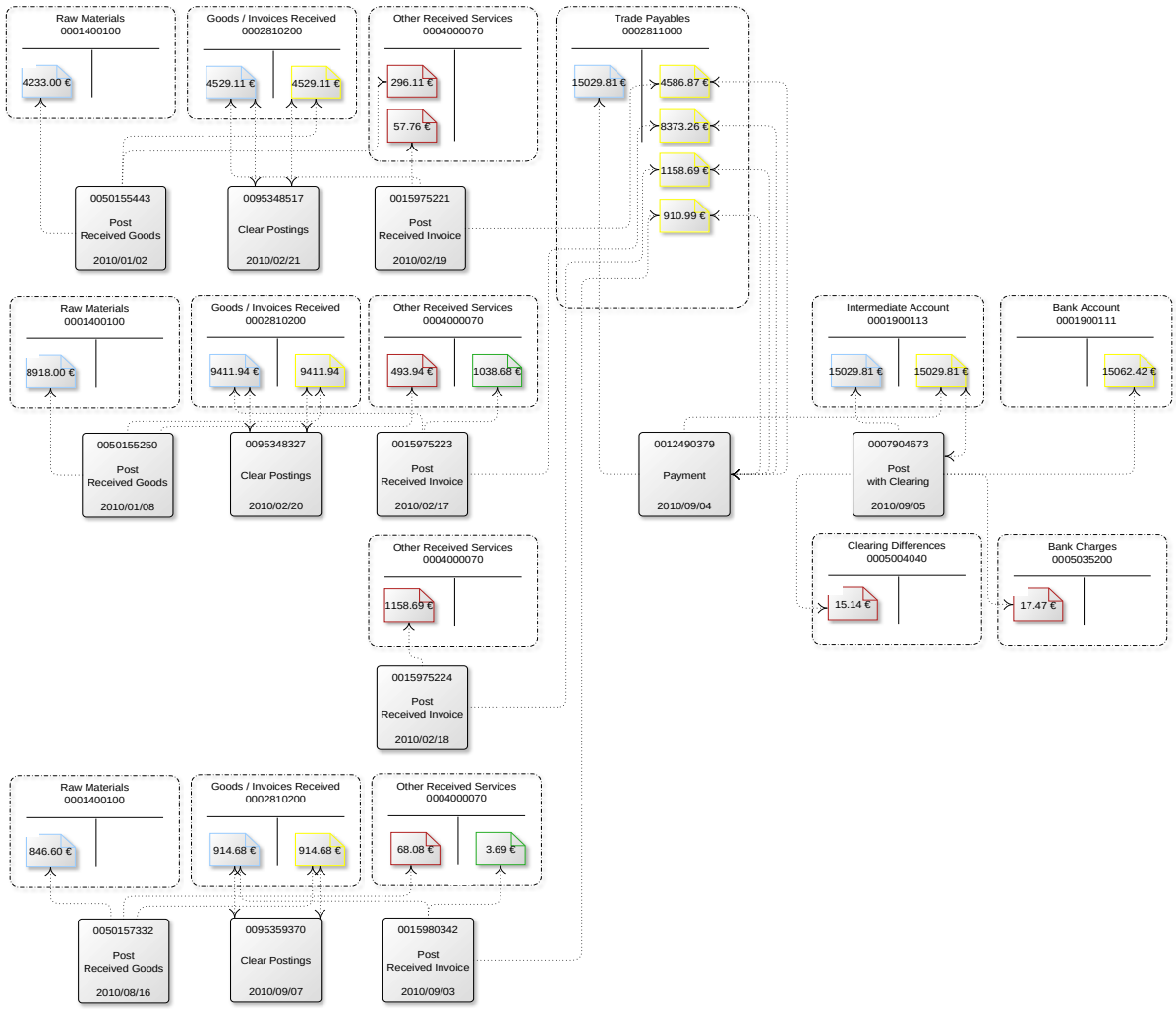

Figure 5 Example Process Instance Graph

The model represents an instance of a purchase process. It shows that three different events occurred for the recording of received goods. The receipt of goods recorded by the activity *Post Received Goods* with the event ID *0050157332*, for example, led to journal entry items posted on the *Raw Materials*, *Goods / Invoices Received* and *Other Received Services* accounts. An invoice was received for each obtained good. An additional invoice processed by the activity with event ID *0015975224* was received with no corresponding recording of received goods. The open items on the related accounts were cleared by using the dedicated activity *Clear Postings*. All received invoices were subsequently cleared by the same payment. Intermediate postings were finally shifted to the final accounts by using the activity *Post with Clearing*.

The diagram provides a detailed overview of the structure of the process instance and illustrates information on financial accounts as well as clearing and posting relationships between account entries and activities that are relevant to reconstruct the logical order of events. It shows more information than represented in Table 2 because it also shows the involved entries on the financial accounts[19]. It also illustrates the specific characteristics of the used source data. Four different sets of events (*Goods Receipt, Invoices Receipt, Clear Postings*) that result in different branches relate to a single event (*Payment*). All of these events refer to a single execution of a business process which ends with the final shifting of account entries from intermediate to final accounts. This structure is different compared to traditional event logs where one event follows exactly one other event within the same case. The source data represented in Figure 5 shows parallelism on the process instance level because the activities in the separate branches were carried out independently from each other. This phenomenon is characteristic for financially relevant events recorded in ERP systems. The observed structure is different from traditional event logs but similar constellations can also be found in very different application domains such as process mining for knowledge sharing processes in online discussion forums (Wang et al., 2014). It is therefore not idiosyncratic for accounting data from ERP systems.

The next subsection analyses the results that are generated if the traditional timestamp dependent control flow inference is used to generate a process instance model based on the used source data.

4.2 Temporal Order

Figure 6 shows the model that is generated if the event log from Table 2 is used as input for inferring the control flow using the timestamps of events.[20] The mining algorithm produces an instance

[19] The complete event log used to generate the model in Figure 5 is included in Appendix B.

[20] The model was created by using the software prototype described in this study. A semantically identical model can also be created by using the event log from Table 2 as input for the general purpose process mining tool Disco (fluxicon, 2015) as shown in Table 7 Journal Entry Item Table

model. It includes every activity only once (contrary to the instance graph in Figure 5) and therefore provides a higher level of abstraction than the model represented in Figure 5.[21]

The mined model shows a different structure than expected when compared to the graph in Figure 5. Figure 5 actually shows four different branches that represent partial execution paths of received goods and invoices that were all paid by the same payment run. The invoice in each branch was received after the receipt of goods was recorded. We would therefore expect a process model that shows the corresponding sequence of *Post Received Goods → Post Received Invoice → Clear Postings → Payment → Post with Clearing*. The model in Figure 6 instead shows many different execution paths due to several short loops and a back loop from the *Clear Postings* activity to *Post Received Goods*.

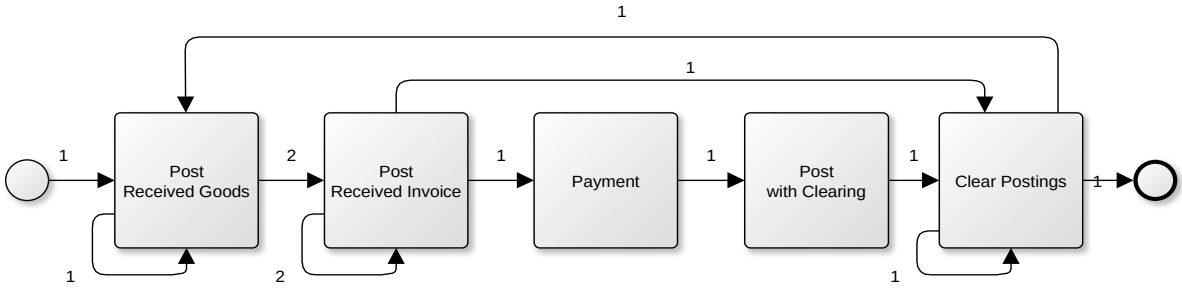

Figure 6 Discovered Process Instance Model Using the Temporal Order of Events

Figure 7 visualizes the reason for the differences.[22] For better comparison the graphical positioning of the activities has been kept constant compared to Figure 5. It shows the temporal dependencies between the different events based on the recorded timestamps. The mining algorithm infers the control flow according to the temporal sequence of events. The event *Post Received Goods* with the event ID *0050155443* was the first event that occurred at the *2010/01/02*.

Appendix C.

[21] The model is not a process model because it only illustrates the behavior of a single process instance.

[22] A semantically identical model can be reproduced by the software Disco if all events are assigned separate activity labels by, for example, substituting the activity labels by a combination of activity label and event ID (please compare Appendix D).

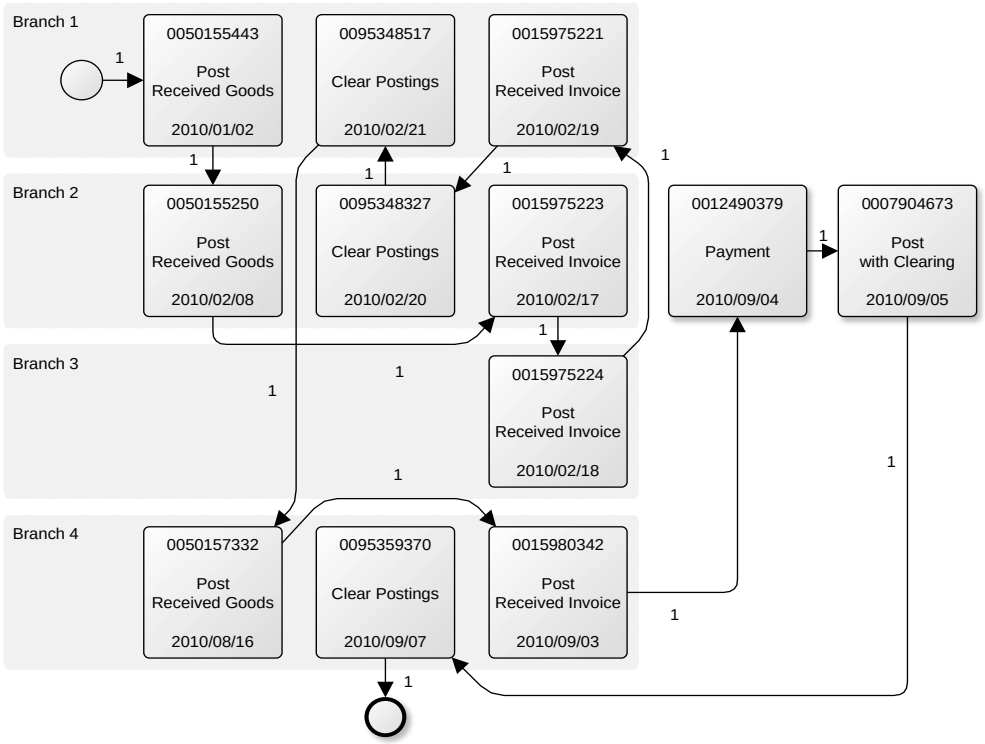

Figure 7 Temporal Sequence

This event was followed by *Post Received Goods* with the event ID *0050155250* at the *2010/02/08*. The algorithm interprets this temporal dependency for reconstructing the control flow from event *0050155443 → 0050155250*. This approach produces a process model that adequately illustrates the temporal order of events. However, due to the parallelism of events on the instance graph level, it is questionable if the inference is useful to understand the represented process.

4.3 Accounting Data Structure Dependent Order

The correct structure in terms of accounting logic can be modelled if the control flow is inferred by analyzing which journal entry item was cleared by another activity using the *'clears'* dependency depicted in Figure 3.

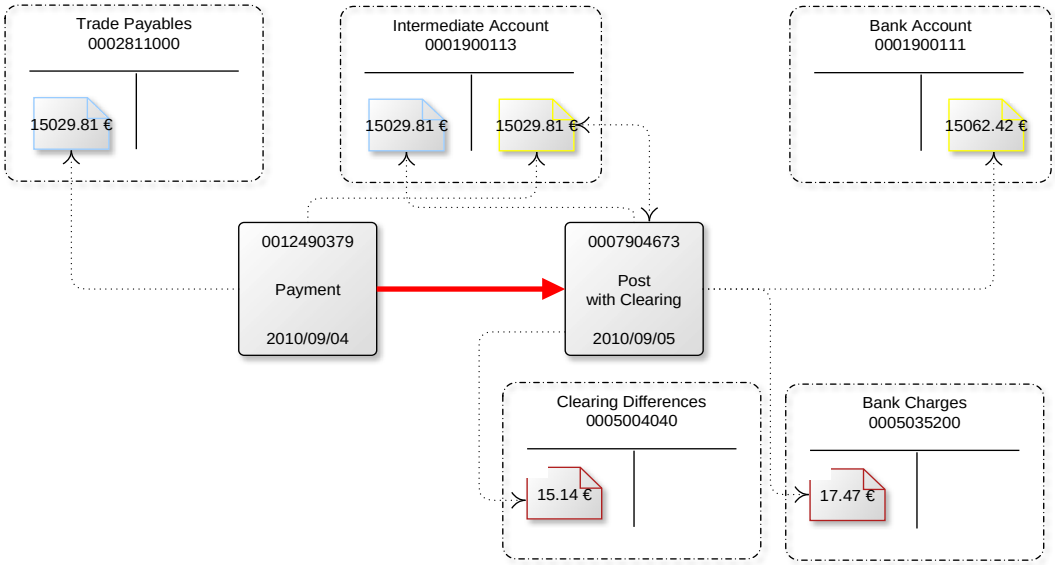

Figure 8 Accounting Data Structure Dependency

Figure 8 provides an illustration of this approach. The activity *Payment* posted a credit item with the value of *15,029.81 €* on the *Intermediate Account*. This item was cleared by the activity *Post with Clearing* with a debit posting on the same account. The accounting data structure dependent order for these activities can therefore be derived as *Payment* → *Post with Clearing*.

The accounting data structure dependencies of an activity α can formally be expressed as ${\lambda^d}_\alpha(I^d{}_\alpha, O_\alpha)$ (Sun and Zhao, 2013) with

I_α: is the input for α

O_α: is the output for α

d: denotes the type of data dependency

They can be calculated as follows:

If activity A with $JournalEntryNr_A \in JE[JournalEntryNr]$ posted $Item_A$ with $JournalEntryItemNr_A \in JEI[JournalEntryItemNr]$ then $Item_A \in O_A$.

If activity B with $JournalEntryNr_B \in JE[JournalEntryNr]$ cleared any posted journal entry item then the selection $JEI_B[ClearingDocNr{:}JournalEntryNr] \neq \emptyset$.

If $JournalEntryItemNr_A \in JEI_B[JournalEntryItenNr]$ then a *'clears'* relationship exists between $Item_A$ and activity B. This implies $Item_A \in I_B^u$ and the mandatory dependency $A \rightarrow {}_mB$ because of $O_A \cap I_B^u \neq \emptyset$ with $Item_A \in O_A \wedge Item_A \in I_B^u$.

Each mandatory dependency is represented as a control arc as shown in Figure 8 (red arrow). These dependencies can be calculated for all recorded activities in the event log to determine the complete accounting data structure dependent control flow of a process instance.

It is further necessary to identify end and start nodes to create complete models. This is quite trivial as start nodes can be determined by identifying activities that do not have any mandatory incoming dependencies and end nodes do not have any outgoing mandatory dependencies:

S is the set of start nodes with $\alpha^{start} \epsilon S$ if $\{\alpha_i \rightarrow {}_m\alpha^{start} \forall i\} = \emptyset$

E is the set of start nodes with $\alpha^{end} \epsilon E$ if $\{\alpha^{end} \rightarrow {}_m\alpha_i \forall i\} = \emptyset$

4.4 Clearing Deadlocks

A problem arises with activities that do not post any items but only clear items posted by other activities. This constellation occurs when the open items that were created by one activity are not directly cleared by the activity that creates the clearing items but instead by another dedicated exclusive clearing activity which does not create any postings itself.

An activity α is called an exclusive clearing activity if $JournalEntryNr_\alpha \in JEI[ClearingDocNr] \wedge JournalEntryNr_\alpha \notin JEI[JournalEntryNr]$. The resulting constellation, which we call a clearing deadlock, is illustrated in Figure 9. It shows that the open credit posting created by the activity *Post Received Goods* on the *Goods / Invoices Received* account was not directly cleared by the following activity *Post Received Invoice* although this created the corresponding debit posting on the same account. Instead the exclusive clearing activity *Clear Postings* was actually used to match the cleared and clearing journal entry items. This type of constellation provides additional flexibility in matching cleared and clearing postings which might explain why it occurs frequently in the used data set. [23]

[23] Exclusive clearing activities were identified in 2.01 % of the instances from data set #1, 7.87 % from data set #2 and 20.66 % from data set #3.

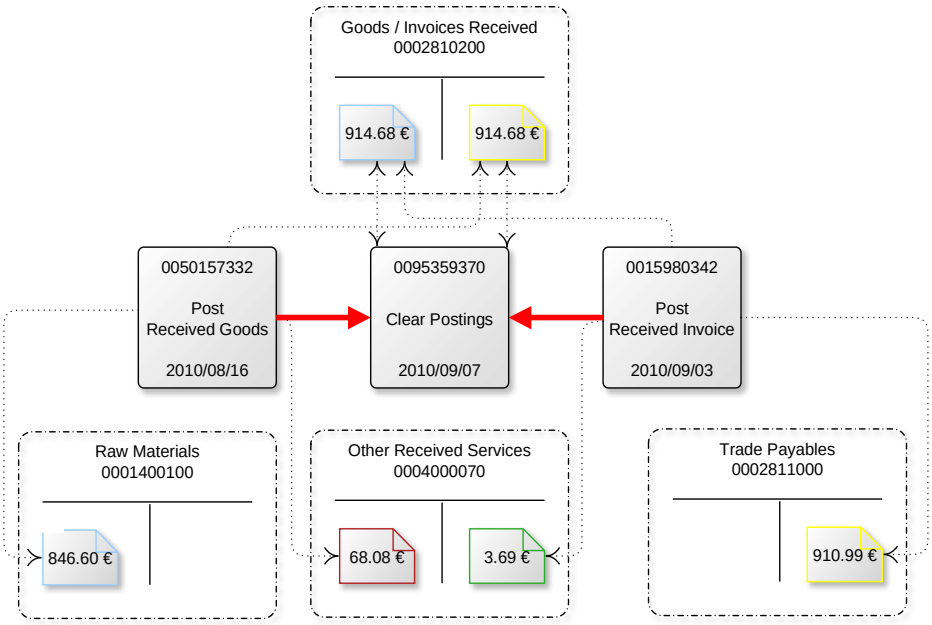

Figure 9 Clearing Deadlock

Due to the fact that the *Clear Postings* activity did not create any other posted item the accounting data structure dependent control flow ends at this activity and it would be defined as an end node. This is a problem because the clearing activity is actually not an end node. In reality the process does not end before the activity *Post with Clearing*. Such a constellation cannot be neglected because the additional end nodes would imply invalid information about the actual process structure.

This outcome can be prevented by solving identified deadlocks by using graph transformations (Rozenberg, 1997). Clearing activities clear items from two or more other activities. At least one of these activities must have posted an additional item that was cleared by another activity different from the clearing activity for a deadlock to appear. Otherwise the clearing activity would represent a valid end node and such a constellation would not be considered a clearing deadlock.

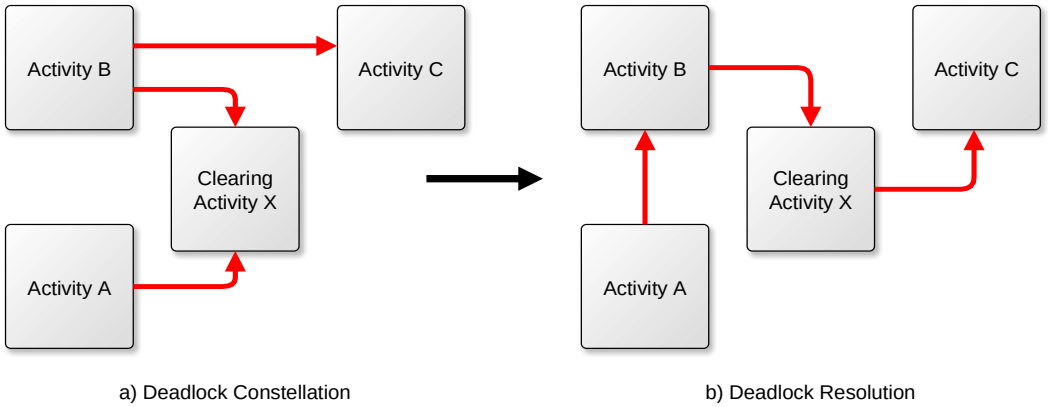

Figure 10 Deadlock Resolution

Figure 10a) provides an illustration of a typical constellation. Items from activities A and B were cleared by the exclusive clearing activity X. One or more additional items from activity B were also cleared by activity C. Because activity A has no further outgoing control arcs it is reasonable to assume that it is logically ordered before activity B and that the process continued with activity C after A, B and X took place. The related sub-graph can be substituted by an amended sub-graph as illustrated in Figure 10b).

4.5 Output of the Accounting Data Structure Dependent Sequencing

The formerly explained procedures for (1) defining causal dependencies, (2) removing clearing deadlocks and (3) defining start and end nodes can be combined to produce a process instance model. Figure 11 shows its output for the example event log. It visualizes the effect of the accounting data structure dependent control flow inference. The positioning of the activities has been kept constant compared to Figure 5 and Figure 7. The first, third and fourth branch show the control flow sequence *Post Received Goods → Post Received Invoice → Clear Postings*. The second branch only consists of the sequence *Post Received Invoice*. All branches follow the subsequent sequence *Payment → Post with Clearing*.

The illustrated instance therefore contains the control flow paths:

α: *Post Received Goods → Post Received Invoice → Clear Postings → Payment → Post with Clearing*

β: *Post Received Invoice → Payment → Post with Clearing*

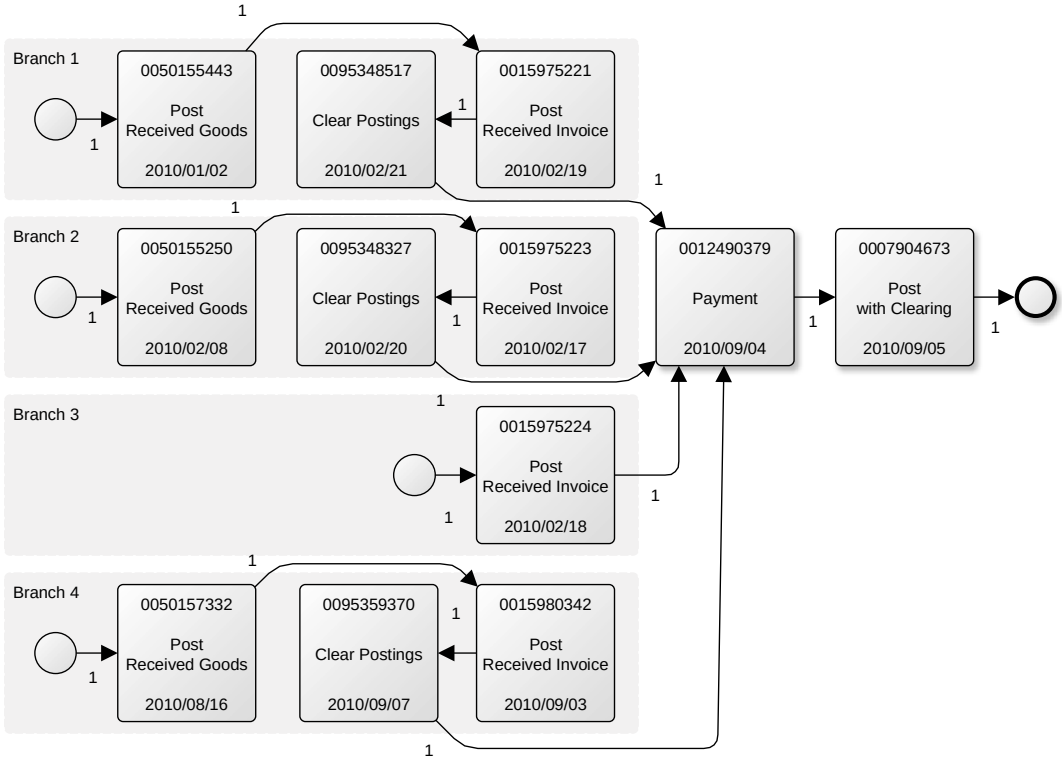

Figure 11 Accounting Data Structure Dependent Sequence

Figure 12 shows the corresponding process instance model. In contrast to the previous models Figure 12 shows the complete BPMN model including the financial accounts and journal entry items. In comparison to the process instance graph shown in Figure 11 the process instance model in Figure 12 only shows each activity type once. As a result the different branches present in Figure 11 have been merged.[24] The numbers on the control flow arcs indicate the frequency of how often the represented control flow was inferred based on the data represented by the process instance.

The aggregation that is necessary to create a process instance model results in two additional paths:

γ: *Post Received Goods → Post Received Invoice → Payment → Post with Clearing*

δ: *Post Received Invoice → Clear Postings → Payment Post → with Clearing*.

[24] For a detailed description of how the merging can be done please refer to (Author Citation).

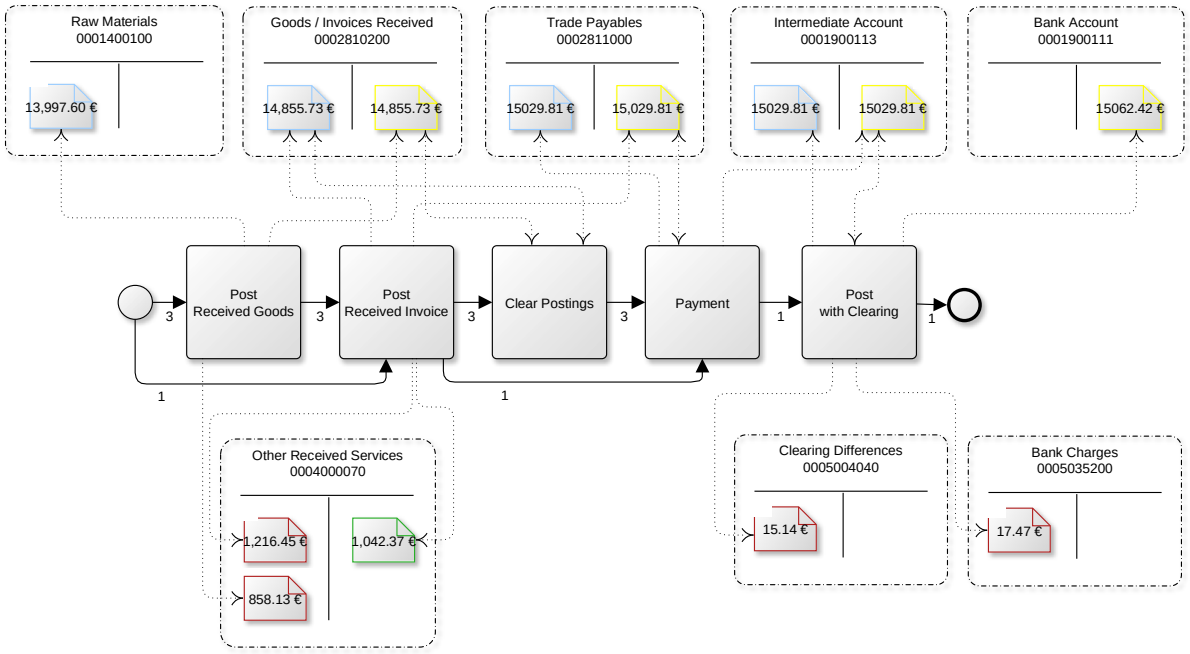

Figure 12 Discovered Process Instance Model Using Accounting Data Structure Dependencies

The model in Figure 12 is less complex in terms of structure than the model in Figure 6. Both models show the same number of activities but the model from Figure 12 shows 8 control flow arcs (bold single headed arrows) whereas the model in Figure 6 shows 11 control flow arcs.

Both approaches introduce execution paths that are not reflected in the original data. This means that the precision of both models is not perfect. The model in Figure 12 introduces 2 additional execution paths γ and δ compared to the underlying process instance graph shown in Figure 5. The model shown in Figure 7 shows exactly one path. The model created using timestamp-based control flow inference shown in Figure 6 shows 16 different paths.[25] The high number of additional paths for the timestamp dependent control flow inference is due to the additional control flow arcs and loops in the model as discussed before.

[25] These numbers were calculated by counting the different execution paths without considering the cardinality of the individual control arcs.

This subsection has summarized the method for inferring the control flow by exploiting the accounting data structure for recorded events and presented the outcome of its application by using a single example. The following subsection focusses on the complexity of the mined models and provides a quantitative analysis for the complete data sets used for this study.

4.6 Evaluation

As introduced in the introductory section this study deals with the research questions of:

(1) How an event log, created on source data from ERP systems by exploiting the general structure of journal entries, can be used in order to infer the control flow of mined models?

(2) And how can this be done by simultaneously maintaining information on the relationship between business processes and the financial accounts?

The method described in the previous subsections illustrates how the control flow can be determined by exploiting the data structure of journal entries. Figure 12 illustrates the outcome for the example used in this study. It shows the relationship between the different process activities, the financial accounts and the posted values.

The aim of the evaluation described in this section is to provide quantitative data for the overall data sets that served as input for this study. In order to assess if the control flow inference worked properly for the whole data set, produced models were tested for specific model characteristics. A model was considered as being mined correctly if it represented the control flow and posting behavior according to the represented real business process. To assess the model correctness mined models were tested for soundness (van der Aalst and Stahl, 2011) by using the academic software Renew (University of Hamburg, 2015). The testing included the assessment of proper completion, option to complete, absence of dead transitions, and safeness for all places except account places.

The second part of the evaluation dealt with the inspection of how the models mined by using the accounting data structure dependent control inference method differ from models mined using the traditional timestamp dependent control flow inference. It is assumed that mined models using the new method are less complex. This should be the case because the accounting data structure dependent control flow inference prevents the generation of loops in the process instance models that are the results of intertwining control flows of parallel branches that are created by timestamp dependent control flow inference.

The data sets described in Table 1 served as input for both, the timestamp dependent control flow inference and the accounting data structure dependent control flow inference. The same algorithm was used - once configured for timestamp dependent inference and once configured for accounting data structure dependent inference - for the evaluation to ensure the comparability of the generated output.

The output was analyzed quantitatively in a second step in order to compare the complexity of the produced models. Several metrics exist to measure the complexity of process models (Rozinat et al., 2008). The metric *structural appropriateness* measures the complexity of a process model by counting the number of included tasks (Rozinat and van der Aalst, 2008). *Structural precision* and *structural recall* measure the amount of causality relations that a mined model has in common with a reference model. The metrics *duplicates precision* and *duplicates recall* measure how many duplicate tasks a mined model has in common with a reference model (de Medeiros, 2006). The metrics *structural precision, structural recall, duplicates precision,* and *duplicates recall* are not suitable for measuring the complexity in our experimental setup because they require a reference model for comparison. The metric *structural appropriateness* only considers the number of modeled activities. Using the data dependencies of activities for process mining has no effect on the number of represented activities because only the sequence of activities is different which is modelled via control arcs. We therefore used the number of control arcs as a suitable complexity measure for the study at hand.

The analysis of the outputs revealed that a major part of the mined models from our data sets represent trivial process instances that consist of only one activity. Such trivial instance models do not differ in complexity if the temporal or accounting data structure dependent order of events is used because the sequence is always the same if just one activity is involved. We therefore focused on nontrivial models (consisting of more than four activities) and complex models (consisting of more than seven activities) assuming that the complexity reduction is higher for more complex models.

Data Set	Nontrivial Models (# of Activities > 4)					
	Timestamp # of arcs per model		Accounting Data Structure # of arcs per model		reduction in %	
	mean	max	mean	max	mean	max
1	9.67	43.00	9.55	21.00	**1.32**	51.16
2	12.25	194.00	11.26	125.00	**8.12**	35.57
3	12.23	158.00	10.70	71.00	**12.47**	55.06

Table 3 Complexity Reduction for Nontrivial Models

Data Set	Complex Models (# of Activities > 7)					
	Timestamp # of arcs per model		Accounting Data Structure # of arcs per model		reduction in %	
	mean	max	mean	max	mean	max
1	24.89	43.00	16.33	21.00	**34.39**	51.16
2	23.17	194.00	20.19	125.00	**12.88**	35.57
3	32.90	158.00	21.70	71.00	**34.04**	55.06

Table 4 Complexity Reduction for Complex Models

Table 3 and Table 4 summarize the evaluation results for the used data sets.[26] Table 3 shows the mean and maximum values of control arcs represented in mined nontrivial models. The values are provided in separate columns for the timestamp dependent and the accounting data structure dependent control flow inference. The columns on the right-hand side show the reduction in complexity. The figures reveal that the average complexity reduction is modest ranging from 1.32 to 12.47 percent. Table 4 shows the results for complex instance models. The reduction is significantly higher for complex models ranging from 12.88 to 34.04 percent. The complexity reduction for the most complex model was 51.16 percent for data set 1, 35.57 percent for data set 2 and 55.06 percent for data set 3.

The evaluation results confirm that the presented control flow method can be used to create correct process models. These adequately model the control flow that has been recorded in the source systems by exploiting the accounting data structure dependency between recorded events. The mined models are significantly less complex if they are nontrivial. The model complexity can

[26] A random sample consisting of 100,000 instances was used for data set 1 due to computational constraints.

be seen as a technical proxy for the usability of the mined models. Ceteris paribus process models with a lower complexity are superior to models with a higher complexity. Under this aspect the evaluation results confirm that the presented control flow method is superior to the timestamp dependent control flow inference for the used type of source data.

However, it is questionable if a reduced model complexity is indeed an aim in itself or if it has to be embedded into the application context. The presented type of evaluation does not assess if the presented approach will actually add value to real process audits carried out by external auditors. This type of evaluation would require different evaluation methods such as field studies or field experiments. The study at hand focusses on the evaluation if the presented solutions work properly from a technical perspective which is a fundamental prerequisite before they can be tested with alternative evaluation methods such as field studies or experiments. This type of evaluation is intended for future research.

Another limitation of the presented results is the fact that it only refers to the data that is stored in the SAP database tables BKPF and BSEG. This means that just those events are considered that produce entries on the general ledger balance sheet as well as profit and loss accounts. Auditors are usually also interested in process activities that do not necessarily lead to entries on the financial accounts such as the creation of purchase requisitions or orders. These can be relevant because internal controls might be in place that ensure that only correct requisitions and orders are created that later on in the process lead to the processing of the received goods, invoices and payments. Those events that do not create entries on the financial accounts are not covered by the presented approach so far.

4.7 Integration into to the Audit Process

This subsection deals with the question of how the presented mining and control flow inference approach can be integrated into the existing auditing process. As discussed in section 3.2 the audit process consists of different phases. Process mining can potentially be used to improve audit procedures especially in the design and operating effectiveness testing of internal controls. Here the auditor requires information about how business processes are supported by information systems, how they relate to the financial accounts, and how they are controlled. Using process mining in

these phases enables the auditor to get a reliable picture of the relevant business processes. Reliability is ensured because the designed models base on actually recorded event data. This is not the case for traditional modelling techniques that are used by auditors as these primarily refer on information received in interviews or the manual inspection of a small number of source documents.

The suggested solution provides the advantage that it does not only model the control flow but also shows the postings on the financial accounts as demonstrated in Figure 12. This information can be used to assess which processes are material and which are not. The advantage of the proposed solution is the fact that the mined models show the structure of the underlying processes correctly from an accounting perspective and that they are less complex.

Table 1 shows the number of mined process instances and process models for the used evaluation data sets. The study at hand focusses on the process instance level. In (Author Citation) it is shown how these can be merged to perfectly fitting and highly precise process models. The used mining algorithm merges such process instances that show exactly the same control flow. The generated models differ from the individual process instance models because the process models represent the posting volume of all represented process instances.

The resulting number of process models may still seem to be high for audit purposes. However, when analyzing the mined process models derived from data set #1 it turns out that a significant number of process models just represent trivial or very infrequent process variants. Of the overall 841 process models in data set #1 145 represent trivial process instances consisting of only one or two activities. Of the remaining process model, 413 process models represent just one process instance. These are very infrequent process variants. Just 17 from the 841 process models actually represent non-trivial processes with more than 100 instances. These are the models that are particularly interesting from a process audit perspective.

In the overall audit process the proposed mining approach could be used to first receive an overview of the processes that exist in an organization and how these relate to the financial accounts. The generated models can be used as an additional information source, to confirm management expectations about how processes are structured and to plan and to carry out the design and operating effectiveness testing of internal controls. The auditor can identify those business processes that represent a material number of process executions and postings on the financial accounts. These can then be tested by assessing internal controls relevant for these processes.

Those processes that are trivial or infrequent but represent material postings (outliers) can be inspected via targeted testing. Trivial and infrequent process instances that are not material can be tested via statistical sampling.

The approach presented in this study aims to discover processes in an organization and to provide information of how these relate to the financial accounts. Models as shown in Figure 12 illustrate how processes are structured from an accounting perspective and how they relate to the financial accounts. This is a valuable information during the audit process according to empirical studies (Müller-Wickop et al., 2013). However, in its current state the presented solutions are not designed to automate the testing itself. The presented accounting structure dependent control flow inference is useful to model correct process models that shows the structure of the underlying processes from an accounting perspective. However, this can just be seen as a step towards the automation of process audits and it does not answer all questions that are relevant during such audits. For example, the presented approach uses the data dependencies inherent to accounting entries to infer the control flow. The time perspective is therefore neglected. During the operating effectiveness testing of internal controls it might be important to know if a certain time sequence of events was followed or not. If, for example, an invoice was received and paid before the received goods were recorded this might be an indication of a compliance violation, or it could just be attributable to the specific characteristics of the inspected process. In order to assess if violations have actually occurred additional analyses are required. These could include the analysis of time sequence anomalies, violations of segregation of duties or social network analyses.

The presented approach focusses on the discovery of the process models by incorporating the information needs of external auditors and the specific characteristics of the available source data. The manual information gathering and modelling of relevant business processes uses a large amount of audit resources in current audits. Automating these procedures would set free resources that could then be spent on the actual testing of these processes and improve the effectiveness and efficiency of process audits. Solutions that automate the conformance and compliance checking of mined models itself have to be covered in future research.

5 Conclusion and Outlook

External auditors face new challenges with the ongoing integration of information technology for the processing of business transactions. This study investigates how process mining can be used to improve process audits that are an important part of financial audits. It focusses on source data from ERP systems and investigates how this data can be used in a novel way to produce less complex process models that provide accurate information on the control flow from an accounting perspective.

Traditional process mining algorithms use the time dependencies between recorded events in order to infer the control flow. These are not suitable for financially relevant data recorded in ERP systems where recorded events are not yet matched to cases and where these cases are non-linear. This study introduces a novel method to exploit the accounting data structure dependencies of recorded events rather than the temporal order for inferring the control flow. The evaluation shows that sound process models can be generated with this approach that are less complex compared to those produced by timestamp dependent control flow inference.

The provided solution can potentially be employed in financial audits to provide external auditors information about relevant business processes effectively and efficiently. Its application would contribute to the reduction of the current imbalance between automated transaction processing of large data sets on the companies' and manual audit procedures on the auditors' side. It would set free audit resources that are currently spent on documenting standard business processes and related internal controls. These would then be available for the actual audit of the processes and of non-standard transactions which commonly exhibit a higher audit risk. This should lead to overall improved financial statement audits.

The presented method has been evaluated by using extensive real world data that was provided by companies operating in different industries. Although the data sets are extensive they only included event data from SAP systems. It can therefore not be concluded that the results are also valid for other ERP systems. However, due to the fact that the chosen methods exploit the generic structure of accounting entries which need to be supported by all information systems used for accounting it is very likely that they are also applicable for other systems. The evaluation demonstrated that the designed methods work correctly and achieved a significant reduction of model

complexity. However, it did not provide information if the presented solutions will also be accepted and are useful in real world organizational settings. Additional research in the form of field studies or experiments will be conducted in future research to address this aspect.

Scholars and professionals have pointed out that it is necessary to integrate research related to information systems for accounting and auditing purposes to keep pace with the technological progress (AAA/AICPA, 2012; Debreceny and Curtis, 2015). The research results presented in this study are a step towards the development of computerized process audit procedures. Forthcoming research will deal with the question of how transactional data like journal entries can be combined with control-related data from ERP systems in order to be able to assess in an automated manner if a specific business process was well-controlled or not. This will enable auditors to rely on the control mechanisms that are implemented in operational information systems and to get audit comfort about relevant business processes by using automated audit procedures.

This study focusses on the application domain of financial audits. It demonstrates how information on the structure of accounting data can be used to infer the control flow in process models. Similar requirements also exist in other application scenarios. Wang et al. (2014), for example, faced difficulties in using traditional process mining algorithms for mining knowledge sharing processes in online discussion forums because of concurrency of events in process instances. Using an alternative control flow inference approach by considering domain specific data dependencies might also improve mining results in similar application scenarios.

References

AAA/AICPA, 2012. The Pathways Commission - Charting a National Strategy for the Next Generation of Accountants. AAA/AICPA, Sarasota, FL.

Author Citation, n.d. Blinded for peer review.

Becker, J., Delfmann, P., Eggert, M., Schwittay, S., 2012. Generalizability and Applicability of Model- Based Business Process Compliance-Checking Approaches – A State-of-the-Art Analysis and Research Roadmap. BuR - Business Research 5, 221–247.

Bhattacherjee, A., 2012. Social Science Research: Principles, Methods, and Practices. A. Bhattacherjee, Tampa, Fla.

Brinkkemper, S., 1996. Method Engineering: Engineering of Information Systems Development Methods and Tools. Information and Software Technology 38, 275–280.

Chen, H., Chiang, R.H.L., Storey, V.C., 2012. Business Intelligence and Analytics: From Big Data to Big Impact. MIS Quarterly 36, 1165–1188.

Considine, B., Parkes, A., Olesen, K., Blount, Y., Speer, D., 2012. Accounting Information Systems: Understanding Business Processes. John Wiley & Sons Australia, Ltd, Milton, Qld.

Cook, J.E., Wolf, A.L., 1999. Software Process Validation: Quantitatively Measuring the Correspondence of a Process to a Model. ACM Transactions on Software Engineering and Methodology 8, 147–176.

Cook, J.E., Wolf, A.L., 1998. Discovering Models of Software Processes from Event-based Data. ACM Transactions on Software Engineering and Methodology 7, 215–249.

de Medeiros, A.K.A., 2006. Genetic Process Mining. Eindhoven University of Technology, Eindhoven.

Debreceny, R.S., Curtis, M.B., 2015. Challenges From and To the Senior Editors of the Journal of Information Systems. Journal of Information Systems 29, 1–8.

Debreceny, R.S., Gray, G.L., 2011. Data Mining of Electronic Mail and Auditing: A Research Agenda. Journal of Information Systems 25, 195–226.

Debreceny, R.S., Gray, G.L., 2010. Data Mining Journal Entries for Fraud Detection: An Exploratory Study. International Journal of Accounting Information Systems 11, 157–181.

Encyclopaedia Britannica (Ed.), 2010. The New Encyclopaedia Britannica, 15th ed. ed. Chicago.

Ferreira, D., Gillblad, D., 2009. Discovering Process Models from Unlabelled Event Logs. Business Process Management 143–158.

fluxicon, 2015. Process Mining and Process Analysis - Fluxicon [WWW Document]. URL www.fluxicon.com (accessed 2.2.15).

Gehrke, N., Müller-Wickop, N., 2010. Basic Principles of Financial Process Mining A Journey through Financial Data in Accounting Information Systems, in: Proceedings of the 16th Americas Conference on Information Systems, Lima, Peru.

Gelinas, U.J., 2014. Accounting Information Systems, 10th edition. ed. South-Western, Australia.

Günther, C., van der Aalst, W.M.P., 2007. Fuzzy Mining – Adaptive Process Simplification Based on Multi-perspective Metrics. Business Process Management 328–343.

Harmsen, A.F., Brinkkemper, J.N., Oei, H., 1994. Situational Method Engineering for Information System Project Approaches. University of Twente, Department of Computer Science.

Hevner, A.R., March, S.T., Park, J., Ram, S., 2004. Design Science in Information Systems Research. MIS Quarterly 28, 75–105.

IDW, 2009. IDW PS 261 Feststellung und Beurteilung von Fehlerrisiken und Reaktionen des Abschlussprüfers auf die beurteilten Fehlerrisiken.

IFAC, 2012. ISA 315 (Revised), Identifying and Assessing the Risks of Material Misstatement through Understanding the Entity and Its Environment.

Jans, M., Alles, M., Vasarhelyi, M., 2013. The Case for Process Mining in Auditing: Sources of Value Added and Areas of Application. International Journal of Accounting Information Systems 14, 1–20.

Jans, M., Alles, M., Vasarhelyi, M., 2010a. Process Mining of Event Logs in Auditing: Opportunities and Challenges. Working paper. Hasselt University. Belgium.

Jans, M., Alles, M.G., Vasarhelyi, M.A., 2014. A Field Study on the Use of Process Mining of Event Logs as an Analytical Procedure in Auditing. The Accounting Review 89, 1751–1773.

Jans, M., Lybaert, N., Vanhoof, K., 2010b. Internal fraud risk reduction: Results of a data mining case study. International Journal of Accounting Information Systems 11, 17–41.

Jans, M., Lybaert, N., Vanhoof, K., Van Der Werf, J.M., 2008. Business Process Mining for Internal Fraud Risk Reduction: Results of a Case Study, in: Proceedings of the International Research Symposium on Accounting Information Systems, Paris.

Jans, M., van der Werf, J.M., Lybaert, N., Vanhoof, K., 2011. A business process mining application for internal transaction fraud mitigation. Expert Systems with Applications 38, 13351–13359.

Jensen, K., Kristensen, L.M., 2009. Coloured Petri Nets. Springer.

Krcmar, H., 2010. Informationsmanagement. Springer, Berlin; Heidelberg.

March, S.T., Smith, G.F., 1995. Design and natural science research on information technology. Decis. Support Syst. 15, 251–266.

Mueller-Wickop, N., Nuettgens, M., 2014. Conceptual Model of Accounts - Closing the Gap between Financial Statements and Business Process Modeling, in: Proceedings of the Modellierung 2014. Wien.

Müller-Wickop, N., Schultz, M., 2013. ERP Event Log Preprocessing: Timestamps vs. Accounting Logic, in: Proceedings of the 8th International Conference on Design Science Research in Information Systems and Technology, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 105–119.

Müller-Wickop, N., Schultz, M., Peris, M., 2013. Towards Key Concepts for Process Audits – A Multi-Method Research Approach, in: Proceedings of the 10th International Conference on Enterprise Systems, Accounting and Logistics, Utrecht.

Naumann, J.D., Jenkins, A.M., 1982. Prototyping: the New Paradigm for Systems Development. MIS Quarterly 29–44.

Österle, H., Becker, J., Frank, U., Hess, T., Karagiannis, D., Krcmar, H., Loos, P., Mertens, P., Oberweis, A., Sinz, E.J., 2010. Memorandum on design-oriented information systems research. European Journal of Information Systems 20, 7–10.

PCAOB, 2010. Auditing Standard No. 12 Identifying and Assessing Risks of Material Misstatement.

Reichert, M., 2012. Visualizing Large Business Process Models: Challenges, Techniques, Applications, in: 1st Int'l Workshop on Theory and Applications of Process Visualization, Tallin.

Reichert, M., Weber, B., 2012. Enabling Flexibility in Process-aware Information Systems Challenges, Methods. Springer, Berlin; New York.

Riege, C., Saat, J., Bucher, T., 2009. Systematisierung von Evaluationsmethoden in der gestaltungsorientierten Wirtschaftsinformatik. Wissenschaftstheorie und gestaltungsorientierte Wirtschaftsinformatik 69–86.

Rittenberg, L.E., Schwieger, B.J., Johnstone, K.M., 2008. Auditing: a business risk approach, 6th ed. ed. Thomson/South-Western, Mason, OH.

Romney, M.B., Steinbart, P.J., 2008. Accounting Information Systems, 11th Revised edition. ed. Prentice Hall.

Rozenberg, G., 1997. Handbook of Graph Grammars and Computing by Graph Transformation, illustrated edition. ed. World Scientific Pub Co, Singapore.

Rozinat, A., de Medeiros, A.K.A., Günther, C.W., Weijters, A., van der Aalst, W.M.P., 2008. The Need for a Process Mining Evaluation Framework in Research and Practice, in: Business Process Management Workshops. pp. 84–89.

Rozinat, A., van der Aalst, W.M.P., 2008. Conformance Checking of Processes Based on Monitoring Real Behavior. Information Systems 33, 64–95.

Song, M., van der Aalst, W.M.P., 2008. Towards comprehensive support for organizational mining. Decision Support Systems 46, 300–317.

Sun, S.X., Zhao, J.L., 2013. Formal Workflow Design Analytics Using Data Flow Modeling. Decision Support Systems 55, 270–283.

University of Hamburg, 2015. Renew - The Reference Net Workshop [WWW Document]. URL http://www.renew.de/ (accessed 7.14.15).

van der Aalst, W.M.P., 2011a. Process Mining: Discovery, Conformance and Enhancement of Business Processes, 1st Edition. ed. Springer, Berlin Heidelberg.

van der Aalst, W.M.P., 2011b. Process Mining: Discovering and Improving Spaghetti and Lasagna Processes, in: IEEE Symposium on Computational Intelligence and Data Mining (CIDM). pp. 1–7.

van der Aalst, W.M.P., Andriansyah, A., de Medeiros, A.K., Arcieri, F., Baier, T., Blickle, T., Bose, J.C., van den Brand, P., Brandtjen, R., Buijs, J., 2012. Process Mining Manifesto, in: BPM 2011 Workshops Proceedings. pp. 169–194.

van der Aalst, W.M.P., de Medeiros, A.K.A., 2005. Process Mining and Security: Detecting Anomalous Process Executions and Checking Process Conformance. Electronic Notes in Theoretical Computer Science 121, 3–21.

van der Aalst, W.M.P., Stahl, C., 2011. Modeling Business Processes : a Petri Net-oriented Approach. MIT Press, Cambridge, Mass.

van Dongen, B.F., van der Aalst, W.M.P., 2005. Multi-Phase Process Mining: Aggregating Instance Graphs into EPCs and Petri Nets, in: PNCWB 2005 Workshop. pp. 35–58.

Wang, G.A., Wang, H.J., Li, J., Fan, W., 2014. Mining Knowledge Sharing Processes in Online Discussion Forums, in: Proceedings of the 47th Hawaii International Conference on System Science, IEEE, pp. 3898–3907.

Wang, T., Cuthbertson, R., 2015. Eight Issues on Audit Data Analytics We Would Like Researched. Journal of Information Systems 29, 155–162.

Weijters, A., van der Aalst, W.M.P., de Medeiros, A.K.A., 2006. Process Mining with the Heuristics Miner-algorithm. Technische Universiteit Eindhoven, Tech. Rep. WP 166.

Weske, M., 2012. Business Process Management Concepts, Languages, Architectures. Springer, Berlin; New York.

Wilde, T., Hess, T., 2007. Forschungsmethoden der Wirtschaftsinformatik. Wirtschaftsinformatik 49, 280–287.

Appendix A

Case ID	Event ID	Timestamp	Activity
1.1	0050155443	2010/01/02	Post Received Goods
1.1	0015975221	2010/02/19	Post Received Invoice
1.1	0095348517	2010/02/21	Clear Postings
1.1	0012490379	2010/09/04	Payment
1.1	0007904673	2010/09/05	Post with Clearing
1.2	0050155250	2010/02/08	Post Received Goods
1.2	0015975223	2010/02/17	Post Received Invoice
1.2	0095348327	2010/02/20	Clear Postings
1.2	0012490379	2010/09/04	Payment
1.2	0007904673	2010/09/05	Post with Clearing
1.3	0015975224	2010/02/18	Post Received Invoice
1.3	0012490379	2010/09/04	Payment
1.3	0007904673	2010/09/05	Post with Clearing
1.4	0050157332	2010/08/16	Post Received Goods
1.4	0015980342	2010/09/03	Post Received Invoice
1.4	0095359370	2010/09/07	Clear Postings
1.4	0012490379	2010/09/04	Payment
1.4	0007904673	2010/09/05	Post with Clearing

Table 5 Adjusted Preprocessed Event Log

Table 5 shows the results when the example process instance shown in Figure 5 is preprocessed by using the algorithm introduced by Müller-Wickop and Schultz (2013). Figure 13 illustrates the model mined by the general purpose process mining software Disco (fluxicon, 2015) if the event log from Table 5 is used as input. The preprocessing of the event logs multiplies several activities in the model. The data perspective is neglected.

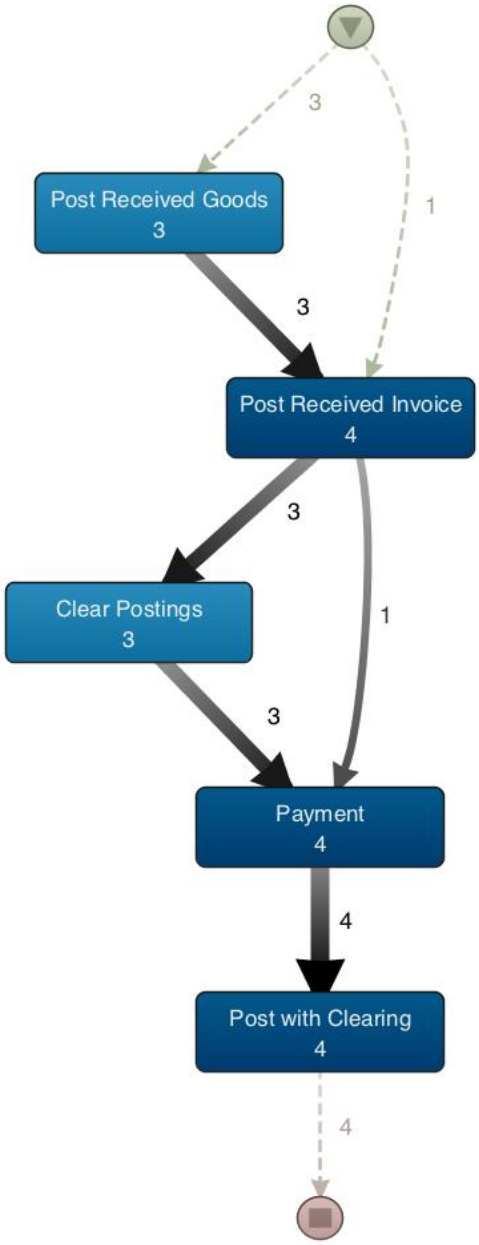

Figure 13 Mined Disco Model for Preprocessed Event Log

Appendix B

Table 6 and Table 7 show the complete event log for the process instance shown in Figure 5. This event log is the output of the method described by Gehrke and Müller-Wickop (2010) which is formalized in section 3.5.

Case ID	JournalEntryNr (BELNR)	TransactionCode (TCODE)	UserName (USNAM)	PostingDate (BUDAT)
1	0050155443	MB01 (Post Received Goods)	User 6	20100102
1	0050155250	MB01 (Post Received Goods)	User 7	20100208
1	0015975223	MIRO (Post Received Invoice)	User 5	20100217
1	0015975224	MIRO (Post Received Invoice)	User 5	20100218
1	0015975221	MIRO (Post Received Invoice)	User 5	20100219
1	0095348327	FB1S (Clear Postings)	User 1	20100220
1	0095348517	FB1S (Clear Postings)	User 1	20100221
1	0050157332	MB01 (Post Received Goods)	User 4	20100816
1	0015980342	MIRO (Post Received Invoice)	User 3	20100903
1	0012490379	F110 (Payment)	User 2	20100904
1	0007904673	FB05 (Post with Clearing)	User 1	20100905
1	0095359370	FB1S (Clear Postings)	User 1	20100907

Table 6 Journal Entry Table

Case ID	JournalEntryNr (BELNR)	JournalEntry-ItemNr (BUZEI)	ClearingDocNr (AUGBL)	Amount (DMBTR)	Account (HKONT)	CreditOr-Debit (SHKZG)
1	0007904673	1		15,062.42 €	0001900111	H
1	0007904673	2		15.14 €	0005004040	S
1	0007904673	3		15,029.81 €	0001900113	S
1	0007904673	4		17.47 €	0005035200	S
1	0012490379	1		15,029.81 €	0002811000	S
1	0012490379	2	0007904673	15,029.81 €	0001900113	H
1	0050155250	1		8,918.00 €	0001400100	S
1	0050155250	2	0095348327	9,411.94 €	0002810200	H
1	0050155250	3		493.94 €	0004000070	S
1	0050155443	1		4,233.00 €	0001400100	S
1	0050155443	2	0095348517	4,529.11 €	0002810200	H
1	0050155443	3		296.11 €	0004000070	S
1	0050157332	1		846.60 €	0001400100	S
1	0050157332	2	0095359370	914.68 €	0002810200	H
1	0050157332	3		68.08 €	0004000070	S

1	0015975221	1	0012490379	4,586.87 €	0002811000	H
1	0015975221	2	0095348517	4,529.11 €	0002810200	S
1	0015975221	3		57.76 €	0004000070	S
1	0015975223	1	0012490379	8,373.26 €	0002811000	H
1	0015975223	2	0095348327	9,411.94 €	0002810200	S
1	0015975223	3		1,038.68 €	0004000070	H
1	0015975224	1	0012490379	1,158.69 €	0002811000	H
1	0015975224	2		1,158.69 €	0004000070	S
1	0015980342	1	0012490379	910.99 €	0002811000	H
1	0015980342	2	0095359370	914.68 €	0002810200	S
1	0015980342	3		3.69 €	0004000070	H

Table 7 Journal Entry Item Table

Appendix C

Figure 14 shows the process instance model created by using the event log from Table 2 as input for the general purpose process mining tool Disco (fluxicon, 2015). It is semantically identical to the model shown in Figure 6.

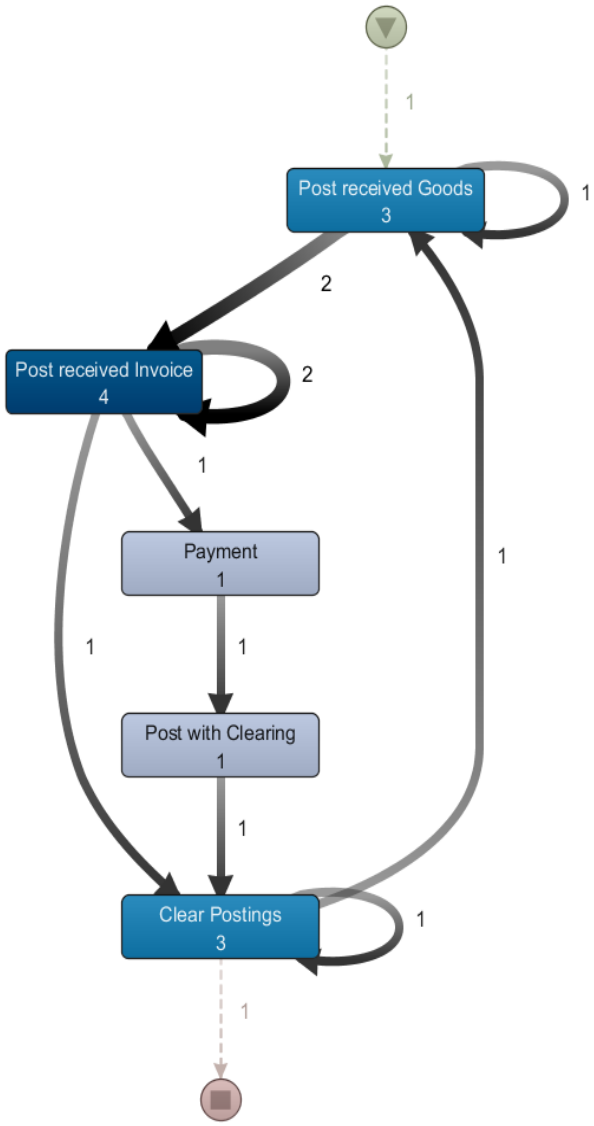

Figure 14 Disco Process Instance Model for Timestamp-based Control Flow Inference

Appendix D

Figure 15 shows the instance graph model for the example process instance which can be generated by using the Disco software. To produce this type of model the TransactionCode and Journal-EntryNr from Table 6 where merged in order to create a unique activity name for each recorded event. The created model is semantically identical to the one shown in Figure 7.

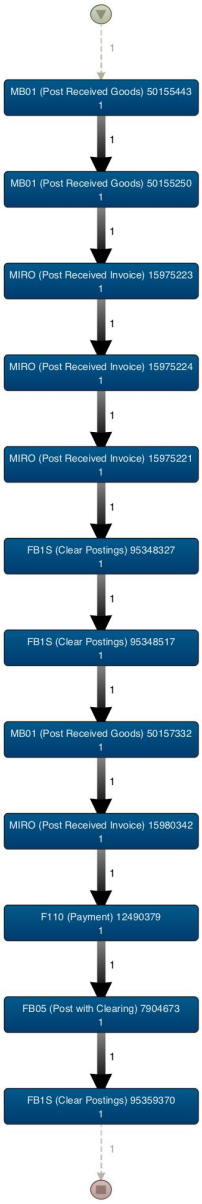

Figure 15 Disco Process Instance Graph Model for Timestamp-based Control Flow Inference