How active learning and process mining can act as Continuous Auditing catalyst

Peer-reviewed author version

JANS, Mieke & HOSSEINPOUR, Mehrnush (2018) How active learning and process mining can act as Continuous Auditing catalyst. In: International journal of accounting information systems, 32, p. 44-58.

DOI: 10.1016/j.accinf.2018.11.002
Handle: http://hdl.handle.net/1942/28124

How active learning and process mining can act as Continuous Auditing catalyst

Mieke Jans, Mehrnush Hosseinpour

Hasselt University, Martelarenlaan 42, 3500 Hasselt, Belgium

Abstract

In the context of Continuous Auditing, different approaches have been proposed to incorporate data analytics to accomplish a continuous audit environment. Some work suggests the use of data mining, some the use of process mining; some work reports on concrete case studies, where other work presents a conceptual approach. In this paper, we present an actionable framework to address one specific level of continuous auditing: the transaction verification level. This framework combines the techniques of data mining and process mining on one hand, and includes the auditor as a human expert to deal with the typical alarm flood on the other hand. Further, different research opportunities are identified in this context.

Keywords: continuous auditing, internal control testing, process mining, data mining, active learning

1. Introduction

The concepts of Continuous Auditing (CA) were presented almost 30 years ago by Groomer and Murthy (1989) and Vasarhelyi and Halper (1991). The idea of a Continuous Audit involves information systems to automate the audit process, striving for (near) real-time assurance (Li et al., 2016). With the rapid advances in technology and techniques, research on the topic of audit analytics, and not so much Continuous Auditing, has emerged and expanded (Moffitt et al., 2016). Most recently published works on this topic describe both the opportunities and the challenges of combining data analytics with auditing and accounting. Examples of articles on possible roles data analytics can play in auditing can be found in Brown-Liburd et al. (2015); Warren et al. (2015); Krahel and Titera (2015); Titera (2013). Different research questions that are related to aspects of the data and/or the technology itself are discussed. The works of Kuhn Jr. and Sutton (2010) and of Chan and Vasarhelyi (2011) use a different approach: they explore the architecture for Continuous Auditing and relate this

Email address: mieke.jans@uhasselt.be (Mieke Jans, Mehrnush Hosseinpour)

to future research questions.

Two key elements that were, right from the start, part of the principles of Continuous Auditing, are the 'audit by exception' principle and a business process view. Both elements have been subjects of separate research investigations. With regards to the exceptions, the CA pilot implementation of Alles et al. (2006), among others, confirmed the idea of 'discrepancy analysis', coined by Vasarhelyi et al. (2004). The focus on exceptions is a result of alarm floods (too many false positives). These can incapacitate an internal audit department and present a critical problem in the adoption of Continuous Auditing (Li et al., 2016). The challenge of managing the alarm floods was the subject of study in work of Perols and Murthy (2012) and Li et al. (2016), who suggested information fusion and the use belief functions respectively to deal with the exceptions.

The second element, the business process view, has been part of the Continuous Auditing principles since the early beginning (Vasarhelyi et al., 2004). A first experiment on the influence of a process-focused system on auditing was presented by O'Donnell and Schultz Jr (2003). Although this research was not explicitly related to the CA context, the process aspect stayed under the radar for a long time and was only picked up more recently. The expansion of data analysis opportunities within the broader field of auditing presumable triggered this revival. One new type of data analytics that might influence the audit is process mining (Jans et al., 2011). The purpose of process mining is to discover, monitor, and improve real processes (as opposed to assumed processes) by extracting knowledge from event logs in information systems (van der Aalst, 2016). Process mining could be described as data analytics from a process point of view. To date, some initiatives on applying process mining in an auditing setting have been reported (for example Jans et al. (2014); Werner (2017)). It is important to note that the field of process mining is relatively young, so different challenges are still present. Yet, given the outline of the process mining field (well-defined types of input, types of possible analyses, and types of output), it is already possible to reason on how exactly these techniques could be employed in a Continuous Auditing setting.

Where previous research investigated opportunities and challenges of Continuous Auditing, exception management, data mining, or process mining in isolation from each other, we connect these related issues with each other in a concrete manner. In this paper, we aim to link the opportunities that both data mining and process mining present for Continuous Auditing, including the key challenge of dealing with alarm floods. In order to do so, we propose incorporating an active learning mechanism that combines the power of artificial intelligence techniques with the expertise of the human auditor. The auditor functions as an *oracle* to feed a machine learning algorithm that classifies the exceptions. These exceptions are first formulated in a process context. Only later, pure data analysis is added to enrich patterns. To present this holistic, but concrete approach, a framework is proposed.

To ensure an embedding in the Continuous Auditing principles, the framework relates specifically to one of the proposed levels of Monitoring and Control by Vasarhelyi et al. (2004): the transaction evaluation level. This level is primarily linked with the tests of controls procedure. In this paper, we will first explain the underlying principles of the internal control testing procedure and formalize the procedure in an unambiguous way. After formalizing the internal control testing procedure, we present our framework that explains the use of analytics for this procedure, displaying the following three characteristics: 1) a process view is taken, 2) managing the alarm flood is incorporated in the approach, and 3) human interpretation (the professional judgment of the auditor) is used as input for the data mining algorithm to address this alarm flood. The approach to start from an existing procedure and to identify how analytics can support this, builds on the expectation that CA will be adopted by automating tasks in the first place. Only in a later phase it is expected that the procedure itself is reconsidered (Vasarhelyi et al., 2004).

By starting from one specific layer, addressing one audit procedure in depth, and integrating previously identified challenges, the proposed framework contributes to the literature by creating synergies between research initiatives that are related to each other, but were previously conducted isolated from each other. To the knowledge of the authors, this is the first proposal on an actionable Continuous Auditing procedure that incorporates both data mining and process mining techniques, and employs the ideas of an active learning approach as leverage to deal with the alarm flood. This last aspect is highly important, since exception sampling undermines the strength of full-population testing. The main contributions of our framework are:

- the algorithm that classifies transactions into 'OK' or 'Not OK', is not forced to classify all transactions in these two classes, but has the option to remain uncertain
- professional judgement of the auditor is incorporated in the data mining approach, to feed the classifier with new information

Incorporating human intervention is also important in the light of compliance with a new European law that forces algorithms to explain their decisions. Additionally, the new research stream of process mining for auditing purposes framed in a larger context, along with all its current research challenges, is a relevant new assessment.

Although not the main contribution, the formalization of the internal control procedure in a continuous auditing setting can also be valued as an increase of the knowledge base. The task of internal control testing itself has been represented as a generic process in Business Process Model and Notation (BPMN)

specifications. Having a general representation of the process of internal control testing, facilitates research on this topic. Namely, having an overview over business processes, and their 'as is' enactment, enables organizations to identify potential improvements. We include the proposition that this rationale on business processes can be extended to the processes and procedures of auditing as well.

The remainder of the paper is structured as follows. Section 2 starts with related literature on Continuous Auditing, managing alarm floods, and process mining. Section 3 describes the generic process of internal control testing, the procedure we relate to the transactional verification from a business process perspective. Section 4 provides some more background in process mining, since we do not assume the reader to be acquainted with this research field. Next, Section 5 discusses the transactional verification framework in detail. Section 6 highlights potential research alleys that are linked to the framework. We conclude the paper in Section 7.

2. Related Literature

2.1. Continuous Auditing

Preceded by the work of Groomer and Murthy (1989) and advanced by Vasarhelyi and Halper (1991), the concept of Continuous Auditing originated in the 1990s. The ideas of having an (automated) layer that is installed on top of a company's functional system, or an embedded audit module within the application, that supports the auditor in performing the audit on a more frequent and continuous base, have been examined in the succeeding years.

The principles of Continuous Auditing are summarized in the publication of Vasarhelyi et al. (2004), and later updated in Chan and Vasarhelyi (2011). In the first publication the authors state that by the 'electronization' of firms' information flows and integration of business processes, auditing could be transformed into continuous analytic monitoring of business processes. The idea of starting from business processes to enable a continuous audit, has been part of both theoretical frameworks (Vasarhelyi et al., 2004; Chan and Vasarhelyi, 2011) and of some practical implementations (Alles et al., 2008; Borthick, 2012), albeit not truly adopted in use. A more tangible link between Continuous Auditing and a process-view, is present in internal control testing. Based on identified risks within an organization's processes, controls are developed and instantiated to mitigate these risks. These controls are monitored and tested by internal auditors and present the ideal start of a Continuous Auditing implementation.

In 2004, four levels of audit objectives for continuous assurance have been proposed: transactional verification, compliance verification, estimate verification, and judgment verification. It is the first level that relates most to business

processes and holds most promises to start adopting the principles of continuous auditing. By principles, we refer to the major dimensions as stipulated by Chan and Vasarhelyi (2011): 1) more frequent audits, 2) proactive approaches, 3) automated procedures, 4) exception-based work and human judgment where necessary, the auditor as certifier, 5) continuous control monitoring and data assurance, control monitoring and detailed testing simultaneously and on full-population, 6) use of data modeling and analytics for monitoring and testing, and 7) more frequent reporting.

For an extensive overview of the Continuous Auditing background, we refer the reader to the paper of Kuhn Jr. and Sutton (2010), who also presented interesting, related research questions. This paper addresses some of those research questions. Research challenges 12, 13, and 14 address the issue of dealing with a flood of false positives. The authors stated that *"More sophisticated analytics that apply artificial intelligence techniques or other mathematical algorithms may also improve the detection capability of continuous auditing systems and to reduce the overload of false positives."* (Kuhn Jr. and Sutton, 2010, page 107). We discuss this in the next subsection.

2.2. Managing Alarm Floods

Research has reported already that a continuous auditing system can generate a large volume of 'exceptions' (Debreceny et al., 2003; Alles et al., 2006, 2008; Perols and Murthy, 2012; Li et al., 2016). Currently, human judgment is required to process this information (over)load, leading to suboptimal decision making. On this topic, Chan and Vasarhelyi (2011) state: "*[...] the automation of all traditional audit procedures may not be immediately feasible. Audit procedures requiring complex judgment and professional skepticism will still require manual performance by the auditor [...]*". The work of Perols and Murthy (2012), Swinnen et al. (2011) and of Li et al. (2016) address the issue of excessive alarm floods and propose a solution.

Perols and Murthy (2012) addressed the issue of exceptions by suggesting an information fusion approach. Although the authors do not go into details, their first layer, the monitoring layer, relates to processes. This is also visually expressed in their framework figure (p. 39). The monitoring layer starts from detected exceptions, gathers information on these exceptions, and suggests the use of machine learning algorithms to classify the exceptions. The presented architecture aims to draw meaningful conclusions by processing the exceptions (and their different sources of information).

Li et al. (2016) propose a different framework; one to prioritize exceptions, making use of Dempster-Shafer belief functions. In this framework, the exception detection is limited to applying defined business rules. Based on these rules, the theory of belief functions is used to assign suspicion scores for each transaction that was identified as an exception. The authors followed the arguments of Srivastava and Shafer (1992) that these functions could represent the auditor's

intuitive understanding of audit risk and included an iterative component to bring the exceptions to a manageable number. Next, based on the suspicion scores, the exceptions are ranked and auditors manually investigate the cases with the highest likelihood of being errors or fraud. The auditors classify the exceptions into 'normal', 'erroneous', or 'fraudulent'. The manual classifications are subsequently used as new information to fine-tune the initial confidence levels of the rules. An iterative approach and the intervention of the human auditor is inherent in this framework. This can be seen as an example of active learning.

2.3. Process Mining

In 1991, Vasarhelyi and Halper already stated that "Additional verification procedures have to validate the flow of a transaction to make sure that the sequence of processing corresponds to the process specifications defined in the system." The term 'transaction flow verification' perhaps conveys the underlying idea best: verifying the flow of activities that represent the business transactions. Later, Debreceny et al. expressed how much time they spent on "understanding exactly how the particular business processes are expressed in the accounting information systems" (Debreceny et al., 2003, page 181). The field of process mining, originating around that same period of time, could nowadays provide assistance in this matter.

Although process mining research is booming within the Business Process Management community, little attention has been devoted to it in accounting research. Currently published journal papers on this topic are merely limited to Huang et al. (2009); van der Aalst et al. (2010); Jans et al. (2011, 2013, 2014); Werner (2017), of which only three of these publications are in accounting journals. This contrasts with efforts that are being taken by industry and standard setting bodies. The most visible interest stemming from the latter group is the Rutgers AICPA Data Analytics Research (RADAR) initiative, devoting a considerable amount of their research efforts to process mining (AICPA, 2015). Also, researchers from the Business Process Management spectrum are picking up the topic of combining their process-related view with accounting challenges (for example Schultz (2013); Sonnenberg and Brocke (2014)). Note that it would be wrong to conclude that process mining techniques are already fully matured and of feasible quality to copy-paste to an auditing environment. Process mining algorithms are still evolving, presenting different challenges to the computer science field (like how to deal with noise and incomplete data sets or inconsistency). However, the basis and the outlines of the process mining field are present: the type of input data that is necessary, the classes of algorithms that are feasible, the type of output to be expected. In section 4, a background on these fundamentals is presented.

3. The procedure of internal control testing in continuous auditing- a process representation

The implementation of a continuous auditing system is primarily the responsibility of the internal auditor (Chan and Vasarhelyi, 2011; Li et al., 2016) and internal control testing is assumed to serve as a good starting point for this. Both the focus on high-risk processes and the reporting requirements on the effectiveness of the internal controls contribute to the suitability of this procedure as backbone of a continuous auditing system. Internal control testing is an auditing procedure that corresponds to the first level of continuous assurance and analytical monitoring: transactional verification.

In order to propose a way how analytics can support transactional verification, we first present the generic process of internal control testing. The process is illustrated in Figure 1 and makes abstraction of the level of automation that is applied[1]. This way, the core process of internal control testing is presented, regardless of the maturity level of continuous auditing. The process does however start from a contemporary financial reporting setting, where information systems like Enterprise Resource Planning systems are deployed. Also, the exception-based principle of continuous auditing is taken as a foundation. Having this generic process overview facilitates the identification of opportunities to employ analytics for transaction verification. Again, the underlying assumption is to move to continuous auditing by automating existing manual procedures in a first stage.

Although not incorporated in the process itself, testing internal controls starts with the selection of a business process. As suggested by Chan and Vasarhelyi (2011) in Stage 1, *"the auditor identifies a business process area where continuous auditing can be applied"*, we also start from a selected business process with available data and control settings to test. After the selection, the phases of the depicted process in Figure 1 take place.

- **Understand normative process**
  Once a business process is selected to test the related internal controls, the first step is to get a general understanding of the process. The purpose is to understand the expected process, which is called the normative process. This understanding can be gained through document review, interviews, previous experience with the company, etc. The output of this activity is either implicit -in the auditor's head-, or explicit in the form of a narrative or a process diagram that describes the normative process.

[1]Business Process Modeling and Notation specifications are used as modeling language. This language is chosen for its high level of understandability by non-technical users and its unambiguous interpretation.

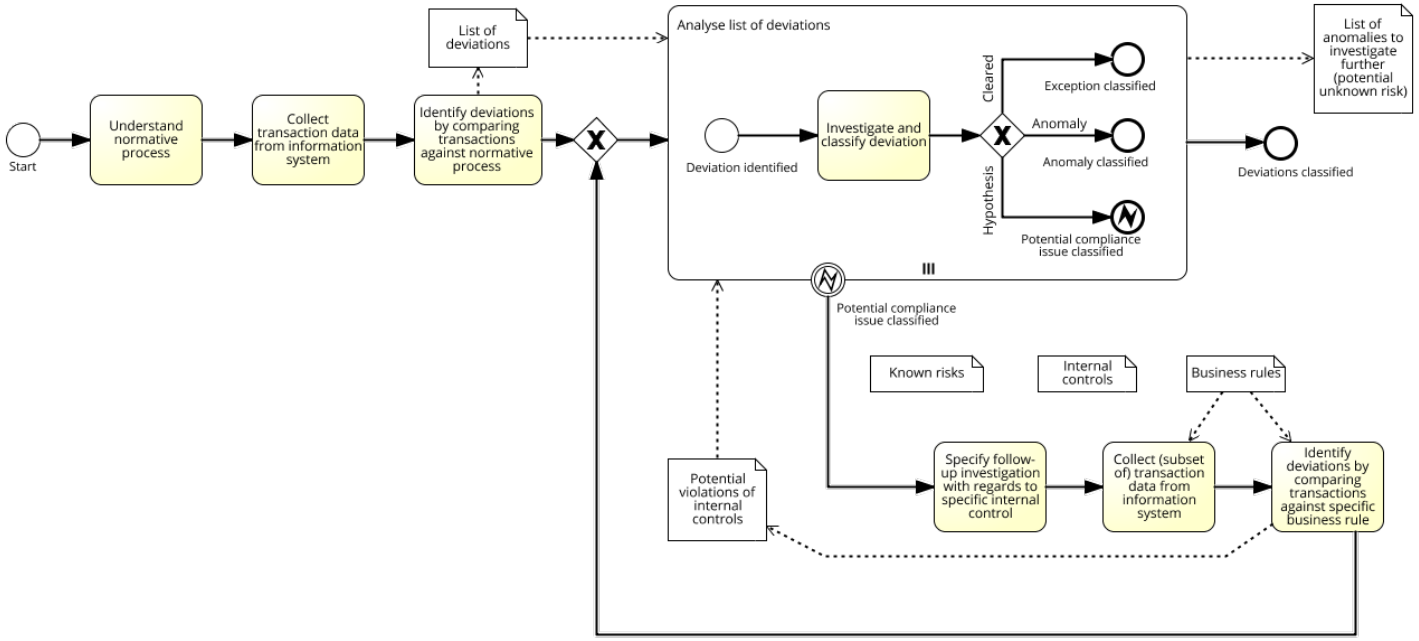

Figure 1: Process of Internal Control testing in BPMN specifications

Aside from understanding the normative process execution, a general understanding of the related process risks must be obtained. The process risks, on their turn, trigger the internal control settings. Although not every single risk and configured control might be disclosed in this first step, understanding the most important risks is an elementary component of understanding the normative process and its controls. Consider a procurement (P2P) process as an example. A simple normative model for this process is illustrated in Figure 2. An example of a process risk can be an execution of a purchase order without any approval. This general level of understanding process risks and their related controls (having an approval strategy in place) are necessary to conduct the procedure. It is important to note that the normative model, like the one in Figure 2, only captures the perspective of activity presence and activity order. Perspectives on who does what, the implied value, or other aspects that do not relate to the presence or order of activities, are not included in this notation. Of course these perspectives also need to be taken into account, but when analyzing processes, these are often only involved in a second phase.

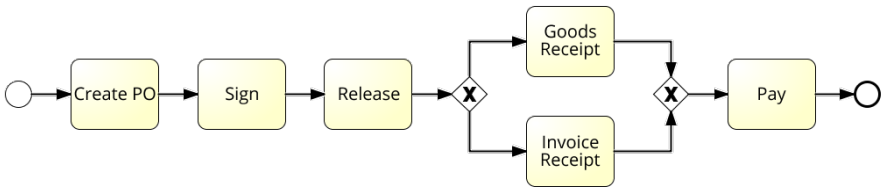

Figure 2: A normative model of a simple procurement process in BPMN specifications

- **Collect transaction data from information system**
  In a second step, transactions of the process under investigation are extracted from the information system.

  A traditional selection of data of the P2P process is a list of some purchase orders and related invoices, whether or not this process is investigated through analytics or manually. Tables 1a and 1b show two short exemplary lists that could be the output of this step. In case an automated audit would take place, these lists would contain all relevant purchase orders and invoices of the period under investigation.

Table 1: Transactional data

Purchase Order							
Nr.	Date	Value	Signed	Released	Goods Received	Invoice	...
1	Sep. 1 2016	23 300	☑	☑	☑	101	
2	Sep. 1 2016	7 329	☑	☑	☑	102	
3	Sep. 1 2016	2 090	☐	☑	☐	103	
4	Sep. 1 2016	4 675	☑	☑	☑	104	
5	Sep. 1 2016	135	☐	☑	☐	105	
6	Sep. 1 2016	4 500	☑	☑	☐	106	
7	Sep. 1 2016	7 210	☑	☐	☑	107	
8	Sep. 1 2016	69 023	☑	☑	☑	108	
9	Sep. 1 2016	12 000	☑	☑	☑	109	
10	Sep. 1 2016	40 329	☑	☑	☑	110	

(a) An example list of a purchase orders for internal control testing

Invoice				
Nr.	Purchase Order	Date	Value	...
101	1	Sep. 12 2016	23 300	
102	2	Sep. 14 2016	7 329	
103	3	Sep. 2 2016	2 090	
104	4	Sep. 20 2016	4 675	
105	5	Sep. 4 2016	135	
106	6	Sep. 25 2016	4 500	
107	7	Sep. 18 2016	3 605	
108	8	Sep. 29 2016	69 023	
109	9	Sep. 19 2016	120 052	
110	10	Sep. 11 2016	40 329	
111		Sep. 15 2016	67	

(b) An example list of invoices for internal control testing

- **Identify deviations by comparing transactions against normative process**
  In a third step, the normative process and the available transaction data are compared with each other. If the transaction, or the process execution that precedes the transaction, is in line with the expected normative model, there is evidence for assurance. If the transaction deviates from what is expected, the deviation is stored in a separate list. Comparing the available transaction data, illustrated in Tables 1a and 1b, with the normative model in Figure 2, it seems that the purchase orders 1, 2, 4, 8, 9, and 10 meet the expectations of the normative model. At least, based on the information that is available in our example, these orders

have been signed, released, goods have been received and an accompanying invoice is present. With regards to following the normative model, these orders do not seem to deviate. The remaining purchase orders are listed as deviations. Also invoice number 111 is listed as deviation, since no related purchase order was presented. We prefer to use the terminology of Depaire et al. (2013) and start from the neutral term 'deviation' when a transaction does not match the normative model.

Take into account that if one would not limit this first investigation to the activities that have been executed, and also take the other information into account, purchase order 9 would be listed as a deviation: the value on the accompanying invoice is slightly higher than on the purchase order, possibly including transportation costs. The scope of elements that are taken into account when executing this third step, depends on the approach of the auditor: whether the auditor includes all elements at once, or splits the investigation in several parts. For reasons of understandability, we limit our example to only checking the process-flow in this step, assuming business rules like 'The value of a purchase order should equal the value of the invoice' are tested separately.

- **Analyse list of deviations**
  Starting from the listed deviations of the previous step, these deviations have to be classified. In our process, we see three possible outcomes, after a closer inspection of a deviation.

  - **Exception**
    A first possibility is that, although the transaction deviates from the normative model, this deviation is in fact also a valid process execution. In our example, invoice 111 deviates from the normative model, as it is not linked to any purchase order. However, although it is not ideal from a risk point of view to procure items without an order, it is an accepted deviation. This might be the case when one orders flowers to thank an employee or some other minor, quick expenses. This deviation is classified as an exception, following the terminology of Depaire et al. (2013): "*Deviations which are accepted and are used to guarantee the necessary flexibility to react fast and operate effectively, are called exceptions.*" Alles et al. use the term 'tolerable exceptions' (Alles et al., 2006, p. 157).
  - **Anomaly**
    A second possibility is that the deviation is classified as an anomaly, '*an undesirable deviation*'. These are deviations for which the auditor, even at closer inspection, cannot formulate a possible explanation for. These deviations are classified as anomalies and will be added to the list of anomalies that warrant further investigation. An example of an anomaly can be a purchase order that has not been released,

like purchase order number 7, but goods have been received, along with an invoice (number 107).

  - **Potential compliance issue**
    The third possibility is that, although the deviation cannot be cleared immediately, the auditor can formulate a hypothesis that would clear this deviation. Returning to our normative procurement model and data sample, purchase orders 3 and 5 will be listed as deviation for missing out a signature. The auditor might however recognize the underlying business rule that states that purchases below 200 do not need to be signed. To reach a final classification on the deviation, the hypothesis 'order value is below 200' needs to be tested. Without proof for accepting the hypothesis, the deviation is classified as a potential compliance issue. In case of this third possibility, the process of internal control testing continues. If no deviations are classified in this category, the internal control testing is finished by reaching the state 'deviations classified'. The output is a list of anomalies that warrant further investigation.

- **Specify follow-up investigation with regards to specific internal control**
  Starting from the deviations that reveal a potential compliance issue, a follow-up investigation is specified. The specifics relate to the potential risk and the configured business rule that, if effective, would mitigate this risk. Turning back to our previous example, the follow-up investigation could relate to the risk of missing a signature for high-value purchases. The configured business rule might be 'If purchase value is less than 200, no signature is requested'. In this case, the follow-up investigation might be specified to test whether all purchases that miss out a signature, indeed have a purchase value of less than 200.

- **Collect (subset of) transaction data from information system**
  Based on the specifications of the follow-up investigation, transaction data has to be collected from the information system again. Depending on the data analytics maturity, this might be a new data extraction phase (in case of a sampling-based approach) or a phase of subsetting previously extracted data (in case of full-population testing). In our example, transactions that relate to a purchase without signature would be collected.

- **Identify deviations by comparing transactions against specific business rules**
  Having the follow-up investigation specified and the relevant transaction data available, transactions are again compared with an 'expected pattern'. This time however, the pattern the transactions are compared with, are in the form of a specific business rule. This is different from the first comparative step, where transactions were compared against a full process. Without turning to specific formats of a process versus a rule, we distinguish between these two artifacts in terms of scope. Where we see 'process'

as a holistic concept, a 'rule' is narrowed down to a few specific characteristics, for example the value of a purchase and the presence/absence of a signature. In our example, the transactions under investigation (the ones without signature, like purchase order 3 and 5) would be tested for having a value that is lower than 200. If this is not the case (like for order number 3), the transaction ends up on the list of deviations again and a new cycle of 'Analyse list of deviations' will follow. The three possibilities of 'exception', 'anomaly', and 'potential compliance issue' are revisited. Continuing on our purchase of higher than 200, but not showing a signature, a new potential compliance issue can be identified, along with a newly formulated hypothesis. For example, 'If the supplier is on the list of 'trusted suppliers', no signature is required'. Then, the procedure continues with testing this hypothesis and so on.

4. Background on Process Mining

Since the mid-nineties, several researchers have been working on techniques for discovering process models out of observed events (van der Aalst et al., 2012). The work of Agrawal et al. (1998), of Lyytinen et al. (1998) and of Cook and Wolf (1998) is recognized as the first process mining introductions in the contexts of work-flow management systems, business process models, and software engineering processes respectively. van der Aalst et al. (2003) provide a more detailed overview of the early work in this domain. In the following subsections, we provide an overview of process mining research that might be linked with internal control testing.

4.1. Event log

The starting point for process mining is an *event log*. The basic assumption is that it is possible to have a set of events such that "(i) each event refers to a task [...], (ii) each event refers to a case [...], and (iii) events are totally ordered." (van der Aalst et al., 2003). Each occurence of an activity, is a unique event. An event log includes more information than only transactional data; it includes data about data, called meta-data. This allows the event log to show temporal dependencies between activities. For example, instead of only knowing that goods have been received (ticked check-box), it is possible to retrieve when and how many times goods have been received. Table 2 is an example of a simplified event log which could have been built from the transactional data in our previous example (Tables 1a and 1b). Note that the event log contains six activities (referring to the rectangles of Figure 2), but 56 unique occurences of the activities (6 + 6 + 4 + 7 + 4 + 5 + 5 + 7+ 6 + 6), called events. In an auditing context, the events in an event log typically refer to transactions that relate to documents, like approving an order. This example log only uses the ordering of the events in time, making abstraction of other characteristics such as value, supplier, and document number.

Starting from an event log, it is possible to have a better view on process traces. A trace is a unique sequence of events that is executed to complete the process. In our example in Table 2, a trace consists of all events that are mentioned on one line. As such, purchase orders 1, 2, 4, and 10 share the same trace, just like purchase orders 3 and 5 do.

In order to structure transactional data into an event log, a time stamp for each event must be available, along with a link to a common case. A *case*, or *process instance* as this is called, could be an invoice or an order. It is the unique identifier that ties different events to each other in one process execution (like in one purchase). On top of these minimal requirements, an event log may store additional information. This can refer to the *resource* (a particular person or system) that executed the activity or other *data elements*, like the transaction value. Although the additional information is not mandatory for process mining from a technical point of view, they are most probably crucial from an auditing point of view. Discovering real process executions in terms of event sequences, without details on involved persons, invoice numbers, and transactions for example, will not provide any substantial audit evidence. For reasons of clarity, we refrained from visualizing these additional data elements. We do however assume the availability of these additional data elements in our event log during our discussion.

4.2. Three types of process mining

Although research topics of the process mining field are getting more diverse over the years, the different techniques can still be categorized in 'three types of process mining': *process discovery*, *conformance checking*, and *enhancement* (van der Aalst, 2016).

Process discovery

The first type of process mining, *process discovery*, starts from an event log and reveals the underlying process behavior in the form of a process model. This visual representation definitely has been the key to success for the process mining field. However, to visualize real-life event logs in a representative and understandable map is not straightforward. Different algorithms have been developed to address this task. The α-algorithm of van der Aalst et al. (2004) is considered as the substantial start of process discovering algorithms (Weerdt et al., 2012) [2]. This algorithm has been improved in the following years. Big improvement steps followed by some other process discovery algorithms such as the Heuristics Miner, the Fuzzy Miner, the Genetic Miner, and the Inductive Miner (Weijters et al., 2006; Günther and Van Der Aalst, 2007; De Medeiros

[2]The α-algorithm is shortly explained in appendix

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

Table 2: A selection of first 10 process executions of a procurement process

et al., 2007; Leemans et al., 2013)[3]. It is important to note that, depending on which algorithm you use, you might get different results.

Despite the improvement steps over the last decade(s), a study by Janssenswillen et al. (2017) report on the quality issues of existing discovery techniques. These issues are important for an application domain to take into account when possibly adopting new techniques. The differences between the algorithms are related to different challenges. For example *'Which paths are important enough to visualize*[4], *'To what extent do I need to generalize the observed behaviour (captured in the event log) for possible future process executions?'*[5] and *'How to deal with noise (i.e., outliers and infrequent behavior that does not represent typical behavior)?'*. These questions are addressed in the light of other, more technical constraints that are beyond the scope of auditing research. However, note that dealing with outliers and infrequent behavior is an important characteristic of the process discovery algorithms. The origin can be found in the primary goal of process discovery algorithms: representing the process as it is executed in an understandable manner. Representing *all* observed behavior however, makes it very complex and difficult to understand. Therefore, process discovery algorithms often abstract the less frequent behavior and represent only the typical and mainstream behavior of a process. As will be explained in section 6.2, looking into the infrequent behavior plays an important role for auditors to investigate the possibility of fraudulent or erroneous behavior.

Conformance checking

The second type of process mining, *conformance checking*, deals with comparing an event log with a designed process (Rozinat and van der Aalst, 2008). The main purpose is to check whether the real process (reflected by the event log) is aligned with the designed process, or whether a process model is a valid representation of reality. The former question is stemming from a compliance focus, whereas the latter concern stems from a quality check on the applied discovery algorithm: a qualitative algorithm delivers a process model that nicely represents the information that is captured in the event log. The following paragraphs introduce the different dimensions that are used to qualify whether model and reality are aligned, the two major approaches in conformance checking algorithms, the difference between model-based and rules-based conformance checking, and the different process mining perspectives.

To express a level of conformance, there are four dimensions articulated (van der Aalst, 2016). Given the purpose of the paper however, only the two

[3]All the mentioned algorithms are briefly explained in appendix

[4]Paths are the links and the relationships between tasks that are visual in a process model

[5]Generalization is one of the quality dimensions that could be taken into account while constructing a model from an event log. Generalization assesses to what extent the generated model can reproduce behavior that was **not** in the event log, but might occur in future.

mostly used dimensions will be addressed: *fitness* and *precision*.[6] The *fitness* dimension expresses how much of the observed behaviour (i.e. data in the event log) is captured by - or *fits* - the process model. This dimension is therefore highly related to compliance. For example, does the trace <Create PO - Sign - Release - Goods Receipt - Invoice Receipt- Pay> fit the procurement model in Figure 2? If all traces would fit the model, fitness is said to be 100% and all transactions comply to the model. To check the alignment between observations in both the log and the model, also the opposite direction of conformance might be taken into account: how *precise* is the model? In other words: how much additional behaviour, that is not present in the event log, does the model allow for? Think for example of a process model that only specifies the tasks that need to be executed, but does not articulate any fixed order between them. For example, to create a certain product, the model prescribes to use ingredients A, B, C, and D, in any possible order. As long as real process executions are limited to a combination of these listed tasks, regardless of its sequence, the trace would *fit* the model and deems compliant to the model. However, the *precision* of this model may be very low if the real process executions do not make use of this flexibility. If in reality some form of (informal) procedure is followed and all executions follow path A-B-C-D, the additional behaviour that is allowed for in the model, makes the model less *precise*. Whether or not a model is precise, is mainly of interest when checking the output quality of a discovery algorithm. The *precision* dimension therefore starts from the log, and then checks upon the model. The precision of a model is said to be 100% if all possible paths, according to the model, are actually observed. If this is the case, there exists at least one trace in the event log for every possible execution according to the model. In contrast, the precision of a model is said to be low if it allows much more behaviour than observed in the event log.

For the purpose of internal control testing, conformance checking is mainly approached from the fitness perspective, and not from the precision perspective. The algorithms that check the fitness of logs with regards to models, can be grouped in two main approaches. One is the replay-based approach (most elaborated in the work of Rozinat and van der Aalst (2008)), and the other is the alignment-based approach (examples in Adriansyah et al. (2013) and de Leoni and van der Aalst (2013)).[7] The replay-based approach replays each process execution individually on the process model to check conformance. Whenever the process execution deviates from the model, the algorithm keeps track of this and calcluates a fitness measure. The fitness measure from each trace is calculated and then a weighted mean is provided as a general fitness-measure. This overall

[6] The remaining dimensions (simplicity and generalization) are still a topic of discussion in the research community (see Buijs et al. (2012) and Janssenswillen et al. (2016)) and will not be discussed here.

[7] The interesting work of Munoz-Gama (2014) is not taken into account, since this is related to *precision* and is mostly relevant to check whether a model represents real behaviour. Given the focus of the internal control testing task, we leave this out of our overview.

number indicates how well the behavior in the event log can be replayed on the process model. Considering the model in Figure 2 and the event log in Table 2, for process execution 1, 2, 4, 9, and 10, the fitness is 1 (i.e., 100%) because they are in conformity with the model. However, for process execution 3 and 5, the fitness will be 0.67 (67%), because two out of the six required events (i.e,. 'Sign' and 'Goods Receipt') are missing. Aside from giving one general fitness measure, also a general overview of how many times and at what places the log deviates from the model is presented graphically. So the output is on the level of the process model, with aggregated information on the mismatches between model and executions. An example output could be: "The activity 'Goods Receipt' has been executed 402 times, while, according to the prescribed model, this was not allowed at that phase of the process execution".

The alignment-based approach is a more robust approach than the replay-based approach (Adriansyah et al., 2013). This approach also relies on a comparison of individual process executions with the model, but it first narrows down 'the model' to 'the path of the model that is closest to the process execution'. The output of this approach is also different: it states *for each process execution* whether it contains extra or missing events, compared to the model. So this approach yields detailed output (on the level of the process execution), as opposed to the 'replay-approach' that yields aggregated output (on the level of the model).

Where the previously mentioned conformance checking approaches compare a log with a model, the conformance of a log can also be tested against rules. If the process at hand is very flexible or no normative model is present, a set of predefined rules can be used to check compliance. In a context of internal control testing, there are often plenty of business rules in place. The rules mentioned before, like 'The value of the purchase order must equal the value of the invoice', or 'If the value of a purchase is below 200, no signature is required' are examples of rules that can be checked for compliance. The work of van der Aalst et al. (2005) introduced the first 'rule tester' to check event logs against rules: the *LTL Checker*, referring to the Linear Temporal Logic that the rules need to be formulated in.

A last relevant dimension when discussing conformance checking is the applied process mining perspective. In general, four perspectives are specified: the control-flow, the time, the organizational, and the case perspective (van der Aalst, 2016). When a control-flow perspective is applied, the focus is on the ordering of the activities. For example: is an invoice first booked before it is paid? When a time-perspective is applied, the focus is on time and duration related aspects. For example: how long does it take in general to pay an invoice, after it is booked? When an organizational perspective is applied, the focus is on the persons, roles, functions, or departments that execute the different activities. For example, which functions are involved in booking invoices and are they expected to be involved? At last, when a case perspective is applied, the

focus is on the remaining characteristics of a case. For example, which suppliers are involved in cases that are related to purchase orders on material number x? In the light of conformance checking, the aforementioned techniques that compare a log with a model, apply a control-flow perspective. This implies that a *compliant case* is a case that follows the prescribed order of activities, making abstraction of all other characteristics of the case. The conformance checking techniques that compare a log against rules, mostly also include the other perspectives.

Enhancement

The third type of process mining, *enhancement*, is concerned with improving an existing model, using the information from the actual behaviour (van der Aalst, 2016). Since it lends itself more to making recommendations in the area of operational efficiency and less to providing assurance, we consider this type outside scope for this paper.

5. Transactional verification framework

In Section 3, the generic procedure of internal control testing is depicted in an unambiguous way. This allows us to present a framework that employs available data and process mining techniques for transactional verification. Based on the challenges that are reported on previous use cases, and on the principles of continuous auditing in general, we postulate three requirements of the framework:

- transaction verification starts from the business process
- managing the stream of alarms is crucial to make the approach actionable
- a human expert is used as an oracle in order to combine the human and computer intelligence to classify transactions

Figure 3 presents our framework. The framework follows a two-stage approach in terms of the applied perspective. In a first stage, only the control-flow perspective is employed. Transactions are viewed in the context of their process execution, and only the sequence of activities is taken into account. Only in the second stage, when additional data is necessary to classify transactions as anomaly or not, the data perspective is added. This perspective brings information like the value, person, etc. into the equation. We continue by discussing the specific phases of the framework in more detail. The first four phases are part of the control-flow perspective. The following last two phases are situated in the data perspective.

Phase 1 As a start, transactional data of the process under consideration is extracted from the relevant relational database. Data from this database is subsequently transformed into an event log. This event log will be used as input

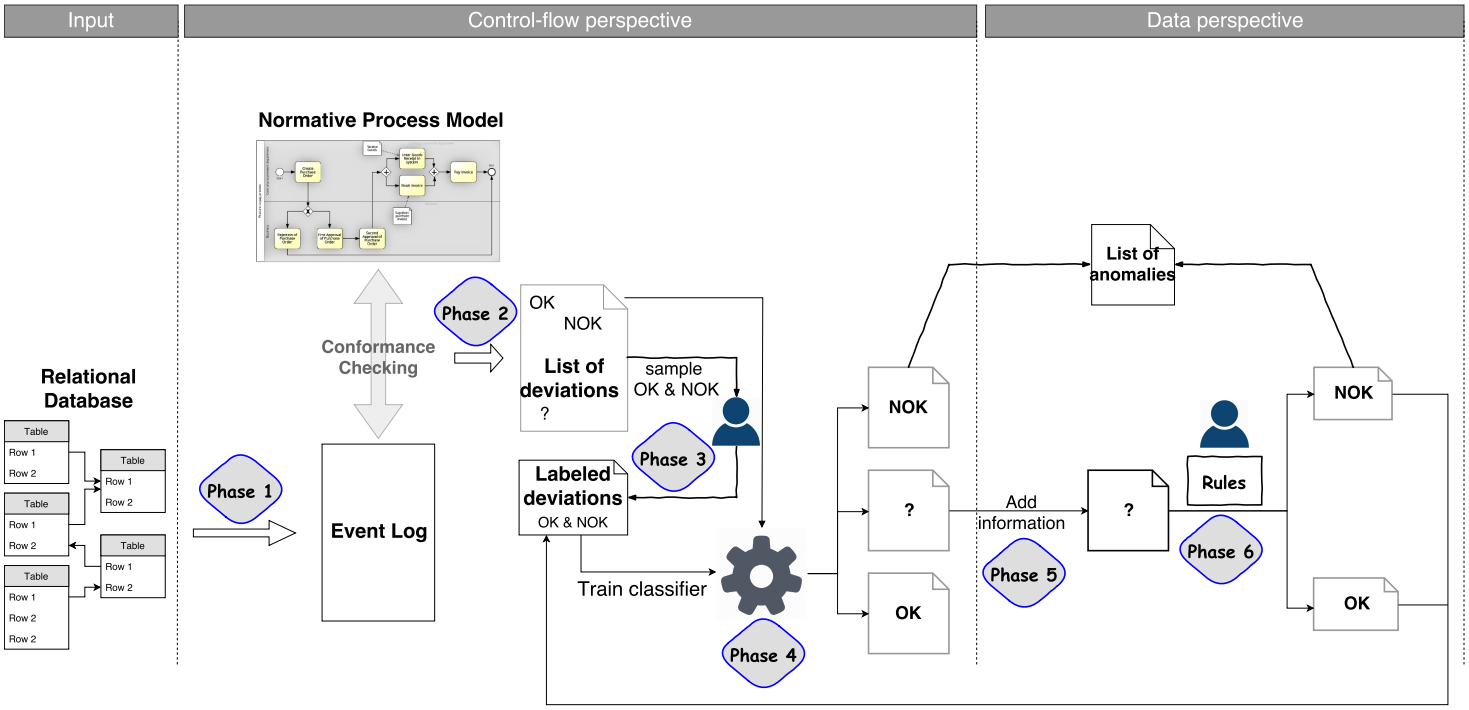

Figure 3: Transactional verification framework

for a conformance checking phase.

Phase 2 In a second phase, a conformance checking algorithm is applied to compare the real transactional data in the event log with the normative process model. The normative process model can be represented in a procedural way, or in a declarative way. The former is a graphical description of *how* process executions should take place (like for example the process in Figure 1), the latter is a description of *what* needs to be executed, along with constraints that are typically formulated in a set of business rules. As long as a process execution respects these constraints, the execution is compliant with the model. The framework presumes to start from a procedural process model that strictly postulates the sequence of activities that needs to be respected. This normative model, along with the event log, is used as input for a conformance checking algorithm. This conformance check only takes into account the order of activities (i.e. control-flow perspective), making abstraction from additional constraints. The output of this phase is a list of deviations in terms of process flows.

The deviations that are identified in this phase are process executions that are different from what is captured in the normative process model. For example an invoice that is not preceded by a purchase order. Since such a model is designed to be general and capturing the *ideal* situation, it is not surprising that also deviations with a logical explanation are captured in this list of deviations. Therefore, this list needs to be processed to filter out true anomalies.

Phase 3 The list of deviations will contain three types of process executions: executions that are perfectly fine, even when deviating from the normative model (OK); executions that under no circumstances can be justified (NOK); and executions for which more information is needed in order to classify them (?). Examples of these three classes are: more approvals than modeled (OK); a payment without an invoice (NOK); a booked invoice without purchase order (?). The third category of deviations are executions that sometimes turn out to be OK, but sometimes also NOK.

Because the list of deviations is based on the control-flow perspective only, the classification of some cases is inconsistent, as captured in the ?-group. The term 'inconsistent' refers to the inconsistent behavior of the final classification the cases belong to: one time a sequence is OK, and another time this same sequence is NOK, depending on additional characteristics. Given the design of our framework, to only start from a control-flow perspective, the inclusion of a classifier that can deal with inconsistency is inherent to this framework. These are called flexible classification models, that support the inclusion of further knowledge coming from human experts. In simpler words, an active learning model for auditing environments should be able to distinguish between undoubtedly safe transactions from the unacceptable ones. However, in case of borderline transactions, the system should delegate the responsibility to the human experts while learning from their rational decisions. Rather than an al-

gorithm that is forced to assign this transaction to one of those two classes (OK or NOK), it pushes this case further to the expert. This behavior resembles the real-world scenario's much better than a binary yes/no scenario. In practice, experts prefer intelligent systems being able to cooperate with them in order to establish a knowledge flow between human being and reasoning machines.

In this third phase, an expert labels examples of the OK and the NOK categories. Deviations that the expert cannot judge on the available input at that moment, are examples of uncertain classes and should remain unlabeled. At the end of this phase, the labeled examples are used to train a flexible classification model. The Dempster-Shafer theory (Shafer and Srivastava, 1990), the three-way decision rules (Yao, 2009), the Rough Set theory (Pawlak, 1998) and the Fuzzy Sets (Zadeh, 1965) seem the perfect starting points to design such intelligent systems.

Phase 4 In the fourth phase, the classifier is fed with the full list of deviations, excluding the labeled sample. The output is again a combination of three 'lists'. The 'OK' list contains cleared deviations that leave the testing procedure. The 'NOK' list goes directly to the list of anomalies for further investigation in a follow-up stage. The deviations that the algorithm was uncertain about proceed to the second stage that employs additional data.

Phase 5 The deviations that could not be classified by the classifier are enriched with additional data. Only here, when extra information is needed to make a final classification, the framework turns to the data perspective.

Phase 6 A human expert will act as the so-called *oracle* and present rules that could clear the control-flow deviation. Take the example of the invoice without a purchase order. The oracle presents the related rule(s) that could justify this deviation. Think for example of the rule that if the supplier is listed as a trusted supplier, no purchase order would be needed. The data concerning the supplier of the deviating process trace is added to the process-related data and the deviation is tested against that rule. A human expert provides all rules that could possibly justify the deviant behavior and these rules are checked upon in this phase. The output is a final clearance of the deviation, or an additional anomaly on the list to warrant further investigation.

This final classification is also used as additional input for the classifier, that now receives more human-based knowledge to learn from and in the end will produce less and less deviations in the 'uncertain' category in phase 4.

One could ponder on the option to immediately capture all rules from the human expert and program these. We postulate that it is difficult, if even feasible, for the expert to provide an exhaustive set of rules with 100% confidence. By confronting the expert with deviations in the activity sequence, the tacit knowledge on which rule is activated in which activity sequence is extracted

and made explicit. The more a rule is presented in different situations, the higher the confidence level of that rule. In this way, the expertise of the human expert can be captured.

6. Continuous transactional verification - Research opportunities

Revisiting the framework as described in Section 5, continuous auditing opportunities and the link to process and data mining research can be made in several steps. The argument we make is that by having a streamlined process of an audit procedure, and knowing the specifics of how to optimize and automate this process, continuous auditing can be reached. The process of internal control testing was outlined before. In this section, we give an overview of outstanding research questions that, if answered, bring auditing closer to continuous auditing. Given the close ties to the field of Business Process Management, some related research questions in this area are also presented. We structure the research opportunities around the different phases that were discussed before.

6.1. Phase 1 – Building the event log

In the context of testing internal controls over a process that affects financial reporting, a continuous auditing environment collects transaction data of the full process. This is a wider collection than the financial transactions, the journal entries, alone. As show-cased in Jans et al. (2014), all timestamped process activities leading to a financial transaction can be taken into account in a process mining investigation.

The main challenge is situated in preparing the data for a process mining analysis: building the event log. As described by González López de Murillas et al. (2015) and González López de Murillas et al. (2016), turning data from a relational database (like an ERP system is using) into an event log, is not straightforward. During event log building, decisions need to be taken on the view on the process. Opting for one specific view, might rule out another view. These decisions therefore have a crucial impact on the type of analyses that are possible afterwards. Current research only investigates this topic from a technical viewpoint, like the work of Calvanese et al. (2015) and Lu et al. (2015). However, the impact of these decisions on the analytical procedures needs to be investigated more thoroughly. The work of Jans et al. partly addresses this (Jans et al., 2017; Jans, 2017).

6.2. Phase 2 – Identify deviations by comparing the event log with the normative process model

Creating the normative process model of the process under investigation

In our framework, the decision on which process to examine thoroughly is taken before the transaction verification starts. In a future continuous auditing

environment, where controls monitoring and detailed testing occurs simultaneously (Chan and Vasarhelyi, 2011), the auditor gets an overview of how the key business processes are running. Ideally, this overview might be in a process model for each process, easily and quickly to understand. Based on this overview, resources can be allocated to the process with the highest risk assessment. In order to get this overview, process discovery algorithms would be suited. To date however, depending on which algorithm is used, one data set will result in different process models. As explained before, this is linked to the underlying emphases these algorithms place (higher fitness, higher precision, more general...?). Future research could examine which process discovery technique's underlying assumptions are -most- compatible with the risk assessment task. Or perhaps new algorithms, dedicated to this task, need to be developed (see for example Pika et al. (2016)). Further, in this context, it is crucial to remember that most process discovery algorithms filter out infrequent behavior. This characteristic needs to be taken into account when linking discovery with audit tasks. Another research alley for getting an overview of the key processes is to create metrics to measure the level of potential risk in a process. This could in turn enable a more efficient resource allocation of the audit.

Once a process is selected, a normative process model must be created to check logged behavior against. When creating such a normative process model, there must be an understanding of the process first. Understanding the normative process can have multiple types of input and output. Possible inputs are (tacit or explicit) experiences of the auditor, oral or written process narratives, graphical process models, or a list of business rules. This will depend on the maturity of the company. In a continuous auditing setting, the output of this activity is a formal process representation. This can be either in the form of a procedural model, or in the form of a declarative model. In the framework, we presented to formalize this process in a procedural way. Future research can investigate however whether one format is more or less suited than the other format in a continuous auditing environment.

Artificial intelligence can play an important role to reach a formal process representation at the end of this activity. Natural Language Processing techniques that are capable of interpreting unstructured data that describe a normative process and pour it into a process representation can be further investigated. Some work on this narrative-model translation and on the alignment is already been done by Friedrich et al. (2011) and van der Aa et al. (2015) for example. In a full continuous auditing environment, this formal process representation is present at the start of the auditing procedure. The task 'Understand normative process' is then fully in line with the key concern of the auditor: assessing the risks and the installed controls of the process under investigation.

Identify deviations by comparing transactions against normative process model

Comparing real transactions against a normative process, is what conformance checking is about. Depending on the format of the normative model (procedural versus rules), different process mining techniques are available, as described in Section 4. There are however multiple research questions linked to this step. In order for these techniques to be applicable in a continuous auditing environment, the required input and output specifications of these techniques need to be adapted to the auditing environment. In other words: if research proves that the normative process -for the goal of continuous auditing- is best formalized in a BPMN process model (the process model in Figure 2 is illustrated in BPMN notation), the conformance checking technique should be able to compare an event log with BPMN models. Second, how is a 'deviation' defined? Is it for example defined at the level of a complete process execution ('this process execution deviates from the model') or at the level of an activity ('this Goods Receipt should not take place in this sequence, according to the model')?

In the field of process mining, 'deviance mining' is described in Nguyen et al. (2014). Future research on this topic can investigate what type of deviations are 'typical' for an auditing setting and whether or not certain groups of deviations exist. If the 'list of deviations' could be compressed to a 'list of deviation types', this might be a first step in making full-population testing feasible (Hosseinpour and Jans, 2016). For example, instead of having a list of 10 000 deviations, the output could be a list of certain types of deviations (for example 'an activity is missing'). Each type again could then be a collection of different subtypes (an approval is missing, a goods receipt is missing, and so forth).

6.3. Phase 3 – Labeling sample deviations as OK or NOK

A key piece of the proposed active learning framework relies on human intervention. In a first step, this refers to labeling sample deviations as OK or NOK. These instances comprise the knowledge to train the classification model, which should be capable of supporting inconsistency. The main feature of the envisaged classification model is its ability to provide a *noncommitment decision* in situations of uncertainty. This means that the classifier will inform the user that, based on the information it currently has, no consistent decision can be taken.

While labeling sample deviations as OK or NOK might not seem to offer many theoretical research opportunities, the whole system relies on this step. Due to the fact that the labeling step could become tedious, we identify the design of an Automated Knowledge Engineer as the most urgent need. In that way, an intelligent system could guide and support the auditor in the labeling process. Time and effort and effort can be saved by employing analytics, but the decision will rely on the knowledge of the auditor. Another interesting research direction is related to the data distribution; it could be that the number of OK instances largely exceeds the number of NOK ones, or vice versa (depending on

the complexity of the normative model). If this turns out to be true, then we will obtain a highly imbalanced dataset, so building a qualitative classification model may become challenging. As an alternative, we can design (under the strict supervision of auditors) generative models to create new synthetic instances resembling the NOK ones. Elaborating on new generative models from different data sources emerge as innovative research topics as well.

6.4. Phase 4 – Classifier that is able to deal with inconsistency

In principle, any classifier based on the Rough Set Theory can be used to implement the classification model. This theory is able to approximate a (presumably rough) set of objects using two exact sets referred to as the lower and the upper approximations. The former set comprises the certain information, while the latter contains the possible information. The difference between the possible evidence and the certain evidence defines the boundary region, i.e. the situations on which the auditor must act. The Rough Cognitive Networks (Nápoles et al., 2016; Nápoles et al., 2017, 2018) are classification models that use this underlying principle under the umbrella of the *three-way decision rules*. That is why we recommend such models. However, any classifier implementing a third option of non-commitment can be adopted. In this regard, also the Dempster-Shafer theory, as suggested in Shafer and Srivastava (1990) and in Srivastava and Shafer (1992), could be investigated.

6.5. Phase 5 – Collecting additional data from the transactions that were classified as uncertain

This phase is merely an operational phase, that holds fewer scientific challenges. The related research questions on this phase are similar to the first phase, where the possibilities of event log building are presented.

6.6. Phase 6 – Oracle presents rules that deviant transactions can be checked against

The auditor (the oracle) feeds the framework with her expertise: hypotheses that might explain deviating behaviour. This can be in the form of a possible set of rules. Comparing real transactions against these specific rules is a particular form of conformance checking. In contrast to the conformance check in phase 2, this step is limited to testing deviating transactions against rules, and not against a model. This drives this step to the area of algorithms like the *LTL Checker* and related work. As mentioned before, techniques that compare real behavior against rules, are capable of taking more characteristics into account than only the order of activities. Further research could investigate which type of characteristics are used by auditors to test rules and assess risk, and whether these characteristics can be dealt with by current techniques. Also, which language for rules is most suited to be applied by auditors to express their knowledge in a formal way?

7. Conclusion

The concept of continuous auditing has been analyzed in previous research, and the application of data mining and process mining has been investigated in different studies. The main drawback of these applications, as currently reported, is the existence of a large number of false positives. This keeps continuous auditing from full adoption in practice. In this paper, we presented a framework to fully automate the first level of continuous auditing, the transaction verification. The basic continuous auditing principle to start from a process point of view is respected, hence a process mining approach is applied as a start. However, this is subsequently complemented by a data mining approach. The suggested data mining approach uses the three-way decision principle on which three outcomes are possible: acceptance, rejection or noncommitment.

The paper first presented an overview of related literature and a generic approach of internal control testing. This procedure was chosen as an instantiation of the transaction verification level. Also, a process mining background was presented. Next, our transaction verification framework was presented and discussed in detail. In a last phase, the framework was revisited in the light of research options, suggesting future research opportunities.

This is not the first framework that is presented to incorporate data analytics in a continuous auditing setting. However, our framework distinguishes itself from previously presented frameworks on a number of aspects. Our framework

- incorporates the process view in a concrete fashion, relating to existing process mining techniques and how these could be employed
- combines process mining and data mining techniques to truly enable full-population testing, including managing the alarm flood
- presents the use of three-way decision rules, the Rough Set theory and the Fuzzy Sets theory to allow classifiers to deal with uncertainty, contributing to a more intelligent classifier and leaving space for the last aspect:
- leverages the human expertise of the auditor to increase the efficiency of the classification algorithm

The possibility of the algorithm to remain uncertain, along with the combination of the machine learning and the human expert feed, ultimately leads to a combined machine-human intelligence, capable of dealing with all transactions.

While various research questions are implied by the framework, the most important research topics, according to us, are:

- For the purpose of gaining an understanding and testing the controls of the process under investigation: is there a difference in suitability between a process model representation (procedural) versus a set of rules (declarative), describing the process?

- What are the implications of having different possible event logs out of one database? Are there formats that are better suited for control testing than others? Is there a difference in audit risk between the different formats?
- Are internal controls best tested against a process model first, and in a second phase against business rules (a two-stage approach like in the suggested framework), or immediately against an exhaustive set of business rules?
- Which approaches or techniques can be used to turn the task of processing deviations feasible?
- How to effectively design an Automated Knowledge Engineer supporting the process of labeling sample deviations?

A final contribution is that we discuss concrete process and data mining characteristics, possibilities, and concerns by starting from a specific framework. This contrasts with research that presents a case study, or with an introduction of potential added value of certain techniques on a high level (like in Jans et al. (2013)). Where the former goes more in detail but leaves out potential research questions in a wider setting, the latter does not go enough into detail.

Like every study, also our study holds some limitations. Although an extensive literature review has been conducted, the selection of concepts and level of detail was mostly guided by the presented framework. We attempted to construct a generic process for all internal control testing procedures, regardless of the level of maturity on continuous auditing to build our framework. Still, different authors might have come up with a different process. We feel confident though, that other process descriptions will cover the same grand lines and that a slightly different process would not have impacted our framework or would have resulted in different potential research questions. Further, future research could involve a pilot implementation of the presented framework as a validation.

References

Adriansyah, A., Munoz-Gama, J., Carmona, J., van Dongen, B. F. and van der Aalst, W. M. P. (2013), *Alignment Based Precision Checking*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 137–149.

Agrawal, R., Gunopulos, D. and Leymann, F. (1998), *Mining process models from workflow logs*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 467–483.

AICPA (2015), 'Rutgers and aicpa unveil data analytics research initiative'.

Alles, M., Brennan, G., Kogan, A. and Vasarhelyi, M. A. (2006), 'Continuous monitoring of business process controls: A pilot implementation of a continuous auditing system at siemens', *International Journal of Accounting Information Systems* **7**(2), 137–161.

Alles, M. G., Kogan, A. and Vasarhelyi, M. A. (2008), 'Putting continuous auditing theory into practice: Lessons from two pilot implementations', *Journal of Information Systems* **22**(2), 195–214.

Borthick, A. F. (2012), 'Designing continuous auditing for a highly automated procure-to-pay process', *Journal of Information Systems* **26**(2), 153 – 166.

Brown-Liburd, H., Issa, H. and Lombardi, D. (2015), 'Behavioral implications of big data's impact on audit judgment and decision making and future research directions', *Accounting Horizons* **29**(2), 451–468.

Buijs, J. C. A. M., van Dongen, B. F. and van der Aalst, W. M. P. (2012), *On the Role of Fitness, Precision, Generalization and Simplicity in Process Discovery*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 305–322.

Calvanese, D., Montali, M., Syamsiyah, A. and Aalst, W. M. P. v. d. (2015), Ontology-Driven Extraction of Event Logs from Relational Databases, *in* 'Business Process Management Workshops', Lecture Notes in Business Information Processing, Springer, Cham, pp. 140–153.

Chan, D. Y. and Vasarhelyi, M. A. (2011), 'Innovation and practice of continuous auditing', *International Journal of Accounting Information Systems* **12**(2), 152–160.

Cook, J. E. and Wolf, A. L. (1998), 'Discovering models of software processes from event-based data', *ACM Transactions on Software Engineering and Methodology (TOSEM)* **7**(3), 215–249.

de Leoni, M. and van der Aalst, W. M. P. (2013), *Aligning Event Logs and Process Models for Multi-perspective Conformance Checking: An Approach Based on Integer Linear Programming*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 113–129.

De Medeiros, A. K. A., Weijters, A. J. and van der Aalst, W. M. P. (2007), 'Genetic process mining: an experimental evaluation', *Data Mining and Knowledge Discovery* **14**(2), 245–304.

Debreceny, R., Gray, G. L., Tham, W.-L., Goh, K.-Y. and Tang, P.-L. (2003), 'The development of embedded audit modules to support continuous monitoring in the electronic commerce environment', *International Journal of Auditing* **7**(2), 169–185.

Depaire, B., Swinnen, J., Jans, M. and Vanhoof, K. (2013), A process deviation analysis framework, *in* M. Rosa and P. Soffer, eds, 'Business Process Management Workshops: BPM 2012 International Workshops, Tallinn, Estonia, September 3, 2012. Revised Papers', Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 701–706.

Friedrich, F., Mendling, J. and Puhlmann, F. (2011), Process model generation from natural language text, *in* H. Mouratidis and C. Rolland, eds, 'Advanced Information Systems Engineering: 23rd International Conference, CAiSE 2011, London, UK, June 20-24, 2011. Proceedings', Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 482–496.

González López de Murillas, E., Aalst, W. M. P. and Reijers, H. A. (2015), Process mining on databases: Unearthing historical data from redo logs, *in* R. H. Motahari-Nezhad, J. Recker and M. Weidlich, eds, 'Business Process Management: 13th International Conference, BPM 2015, Innsbruck, Austria, August 31 – September 3, 2015, Proceedings', Springer International Publishing, Cham, pp. 367–385.

González López de Murillas, E., Reijers, H. A. and van der Aalst, W. M. P. (2016), Connecting databases with process mining: A meta model and toolset, *in* R. Schmidt, W. Gudria, I. Bider and S. Guerreiro, eds, 'Enterprise, Business-Process and Information Systems Modeling: 17th International Conference, BPMDS 2016, 21st International Conference, EMMSAD 2016, Held at CAiSE 2016, Ljubljana, Slovenia, June 13-14,2016 , Proceedings', Springer International Publishing, Cham, pp. 231–249.

Groomer, S. M. and Murthy, U. S. (1989), 'Continuous auditing of database applications: An embedded audit module approach', *Journal of Information Systems* **3**(2), 53.

Günther, C. W. and Van Der Aalst, W. M. P. (2007), Fuzzy miningadaptive process simplification based on multi-perspective metrics, *in* 'Business Process Management', Springer, pp. 328–343.

Hosseinpour, M. and Jans, M. (2016), Categorizing identified deviations for auditing, *in* P. Ceravolo, C. Guetl and S. Rinderle-Ma, eds, 'Proceedings of the 6th International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2016)', number 1757 *in* 'CEUR Workshop Proceedings',

pp. 125–129.
URL: *http://ceur-ws.org/Vol-1757*

Huang, S.-M., Yen, D. C., Hung, Y.-C., Zhou, Y.-J. and Hua, J.-S. (2009), 'A business process gap detecting mechanism between information system process flow and internal control flow', *Decision Support Systems* **47**, 436–454.

Jans, M. (2017), 'Auditor choices during event log building for process mining', *working paper – Hasselt University* .

Jans, M., Alles, M. G. and Vasarhelyi, M. A. (2014), 'A field study on the use of process mining of event logs as an analytical procedure in auditing', *Accounting Review* **89**(5), 1751–1773.

Jans, M., Alles, M. and 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), 1–20.

Jans, M., Soffer, P. and Jouck, T. (2017), 'Learning to build a valuable event log', *working paper – Hasselt University* .

Jans, M., van der Werf, J. M., Lybaert, N. and Vanhoof, K. (2011), 'A business process mining application for internal transaction fraud mitigation', *Expert Systems with Applications* **38**(10), 13351–13359.

Janssenswillen, G., Donders, N., Jouck, T. and Depaire, B. (2017), 'A comparative study of existing quality measures for process discovery', *Information Systems* **71**, 1–15.

Janssenswillen, G., Jouck, T., Creemers, M. and Depaire, B. (2016), Measuring the quality of models with respect to the underlying system: An empirical study, *in* M. La Rosa, P. Loos and O. Pastor, eds, 'Business Process Management: 14th International Conference, BPM 2016, Rio de Janeiro, Brazil, September 18-22, 2016. Proceedings', Springer International Publishing, Cham, pp. 73–89.

Krahel, J. P. and Titera, W. R. (2015), 'Consequences of big data and formalization on accounting and auditing standards', *Accounting Horizons* **29**(2), 409–422.

Kuhn Jr., J. R. and Sutton, S. G. (2010), 'Continuous Auditing in ERP System Environments: The Current State and Future Directions', *Journal of Information Systems* **24**(1), 91–112.

Leemans, S. J. J., Fahland, D. and van der Aalst, W. M. P. (2013), Discovering block-structured process models from event logs-a constructive approach, *in* 'Application and Theory of Petri Nets and Concurrency', Springer, pp. 311–329.

Li, P., Chan, D. Y. and Kogan, A. (2016), 'Exception prioritization in the continuous auditing environment: A framework and experimental evaluation.', *Journal of Information Systems* **30**(2), 135 – 157.

Lu, X., Nagelkerke, M. Q. L., van de Wiel, D. and Fahland, D. (2015), 'Discovering interacting artifacts from erp systems (extended version)', *BPM Reports* **1508**.

Lyytinen, K., Mathiassen, L., Ropponen, J. and Datta, A. (1998), 'Automating the discovery of as-is business process models: Probabilistic and algorithmic approaches', *Info. Sys. Research* **9**(3), 275–301.

Moffitt, K. C., Richardson, V. J., Snow, N. M., Weisner, M. M. and Wood, D. A. (2016), 'Perspectives on past and future ais research as the journal of information systems turns thirty.', *Journal of Information Systems* **30**(3), 157 – 171.

Munoz-Gama, J. (2014), Conformance checking and diagnosis in process mining, PhD thesis, Universitat Politecnica de Catalunya - BarcelonaTech.

Nápoles, G., Falcon, R., Papageorgiou, E., Bello, R. and Vanhoof, K. (2017), 'Rough cognitive ensembles', *International Journal of Approximate Reasoning* **85**, 79–96.

Nápoles, G., Grau, I., Papageorgiou, E., Bello, R. and Vanhoof, K. (2016), 'Rough cognitive networks', *Knowledge-Based Systems* **91**, 46–61.

Nápoles, G., Mosquera, C., Falcon, R., Grau, I., Bello, R. and Vanhoof, K. (2018), 'Fuzzy-rough cognitive networks', *Neural Networks* **97**, 19–27.

Nguyen, H., Dumas, M., Rosa, M., Maggi, F. M. and Suriadi, S. (2014), Mining business process deviance: A quest for accuracy, *in* R. Meersman, H. Panetto, T. Dillon, M. Missikoff, L. Liu, O. Pastor, A. Cuzzocrea and T. Sellis, eds, 'On the Move to Meaningful Internet Systems: OTM 2014 Conferences: Confederated International Conferences: CoopIS, and ODBASE 2014, Amantea, Italy, October 27-31, 2014, Proceedings', Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 436–445.

O'Donnell, E. and Schultz Jr, J. J. (2003), 'The Influence of Business-Process-Focused Audit Support Software on Analytical Procedures Judgments', *Auditing: A Journal of Practice & Theory* **22**(2), 265–279.

Pawlak, Z. (1998), 'Rough set theory and its applications to data analysis', *Cybernetics & Systems* **29**(7), 661–688.

Perols, J. L. and Murthy, U. S. (2012), 'Information fusion in continuous assurance', *Journal of Information Systems* **26**(2), 35 – 52.

Pika, A., van der Aalst, W. M. P., Wynn, M. T., Fidge, C. J. and ter Hofstede, A. H. M. (2016), 'Evaluating and predicting overall process risk using event logs', *Information Sciences* **352353**, 98–120.

Rozinat, A. and van der Aalst, W. M. P. (2008), 'Conformance checking of processes based on monitoring real behavior', *Inf. Syst.* **33**(1), 64–95.

Schultz, M. (2013), Enriching Process Models for Business Process Compliance Checking in ERP Environments, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 120–135.

Shafer, G. and Srivastava, R. (1990), 'The Bayesian and belief-function formalisms: A general perspective for auditing', *Auditing: A Journal of Practice & Theory* **9**, 110–137.

Sonnenberg, C. and Brocke, J. v. (2014), 'The missing link between BPM and accounting: Using event data for accounting in process-oriented organizations', *Business Process Management Journal* **20**(2), 213–246.

Srivastava, R. P. and Shafer, G. R. (1992), 'Belief-Function Formulas for Audit Risk', *Accounting Review* **67**(2), 249–283.

Swinnen, J., Depaire, B., Jans, M. and Vanhoof, K. (2011), A process deviation analysis, a case study, *in* F. Daniel, K. Barkaoui and S. Dustdar, eds, 'Business Process Management Workshops', Vol. 99 of *Lecture Notes in Business Information Processing*, Springer Berlin Heidelberg, pp. 87–98.

Titera, W. R. (2013), 'Updating audit standard–enabling audit data analysis', *Journal of Information Systems* **27**(1), 325–331.

van der Aa, H., Leopold, H. and Reijers, H. A. (2015), Detecting inconsistencies between process models and textual descriptions, *in* H. R. Motahari-Nezhad, J. Recker and M. Weidlich, eds, 'Business Process Management: 13th International Conference, BPM 2015, Innsbruck, Austria, August 31 – September 3, 2015, Proceedings', Springer International Publishing, Cham, pp. 90–105.

van der Aalst, W. M. P. (2016), *Process mining: Data science in action*, Springer, Heidelberg.

van der Aalst, W. M. P., Buijs, J. and Van Dongen, B. (2012), Towards improving the representational bias of process mining, *in* 'Data-Driven Process Discovery and Analysis', Springer, pp. 39–54.

van der Aalst, W. M. P., de Beer, H. T. and van Dongen, B. F. (2005), *Process Mining and Verification of Properties: An Approach Based on Temporal Logic*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 130–147.

van der Aalst, W. M. P., Dongen, B. F. v., Herbst, J., Maruster, L., Schimm, G. and Weijters, A. J. M. M. (2003), 'Workflow mining: a survey of issues and approaches', *Data & Knowledge Engineering* **47**(2), 237–267.

van der Aalst, W. M. P., van Hee, K. M., van der Werf, J. M. and Verdonk, M. (2010), 'Auditing 2.0: Using process mining to support tomorrow's auditor', *Computer* **43**(3), 90–93.

van der Aalst, W. M. P., Weijters, T. and Maruster, L. (2004), 'Workflow mining: Discovering process models from event logs', *IEEE Transactions on Knowledge and Data Engineering* **16**(9), 1128–1142.

Vasarhelyi, M. A. and Halper, F. B. (1991), 'The continuous audit of online systems.', *Auditing: A Journal of Practice & Theory* **10**(1), 110 – 125.

Vasarhelyi, M., Alles, M. and Kogan, A. (2004), 'Principles of Analytic Monitoring for Continuous Assurance', *Journal of Emerging Technologies in Accounting* **1**, 1–21.

Warren, J. J. D., Moffitt, K. C. and Byrnes, P. (2015), 'How big data will change accounting', *Accounting Horizons* **29**(2), 397–407.

Weerdt, J. D., Backer, M. D., Vanthienen, J. and Baesens, B. (2012), 'A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs', *Information Systems* **37**(7), 654 – 676.

Weijters, A. J. M. M., van Der Aalst, W. M. P. and De Medeiros, A. K. A. (2006), 'Process mining with the heuristics miner-algorithm', *BETA Working paper series TU/e* **166**, 1–34.

Werner, M. (2017), 'Financial process mining - accounting data structure dependent control flow inference', *International Journal of Accounting Information Systems* **25**, 57–80.

Yao, Y. (2009), *Three-Way Decision: An Interpretation of Rules in Rough Set Theory*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 642–649.

Zadeh, L. (1965), 'Fuzzy sets', *Information and Control* **8**(3), 338 – 353.
URL: *http://www.sciencedirect.com/science/article/pii/S001999586590241X*

Appendix A. Process Discovery Algorithms

α-algorithm

The α-algorithm is the first algorithm that was developed to discover a model from an event log. It scans the process traces in an event log and takes the sequence of activities into account. More specifically, it registers only consecutive activities in the event log and visualizes the pattern it finds. To this end, the algorithm only considers direct relationships between activities: which activities occur directly after, before, or at the same time with another activity?

Four types of relations between activities in the event log are defined: i) direct succession, when activity a is directly followed by activity b, ii) causality, when activity b directly follows activity a, but activity a never follows activity b, iii) parallel, when activity a follows directly activity b, but at another place in the event log, activity b follows directly activity a, and iv) choice, when neither activity a directly follows activity b, nor activity b directly follows activity a. Note that the ordering relations ii) and iii) are sub-collections of i). Based on the identified relations between all pairs of activities in the log, the process model is created. The relations don't take any frequency into account. This means that if there is ever in the log an activity x followed by an activity z, even only once, there will be an arc going from x to z in the process model.

Heuristics miner

The Heuristics miner (Weijters et al., 2006) is an improvement of the α-miner, which takes the frequency of process executions into account. Considering the frequency of process executions helps to generate a model which filters out noise and infrequent behavior. The algorithm can provide an option for the user to define a dependency threshold. Then, the model is constructed only based on the activities' relations whose frequencies are higher than the predefined threshold.

Fuzzy miner

The Fuzzy miner (Günther and Van Der Aalst, 2007) is a process discovery technique that is developed using the concept of a road map as a metaphor. Like road maps, the fuzzy process map is constructed in such a way that it 1) aggregates information that has lots of details, 2) makes abstraction of less important behavior, 3) puts visual emphasis on information that may be most important, and 4) allows the user to customize the map. As a result, this miner is suitable to deal with complex, unstructured, and large event logs. This miner is the basis of commercial process mining tooling like Disco of Fluxicon.

Genetic miner

The Genetic miner (De Medeiros et al., 2007) is also developed to tackle noise in the data and infrequent behavior. This miner is designed in a way to mimic the process of evolution in biological systems. It applies genetic operators, iteratively, to find the fittest model to the log amongst all possible generated process models. The computation time for this algorithm can be very high.

Inductive miner

The Inductive miner (Leemans et al., 2013) is an improvement of the α- and Heuristics miner which guarantees to generate a logical process model. It constructs process trees (a different process model notation) by repeatedly splitting the event log based on the likelihood of its events. The process trees can then be used to construct the process model.