FROM RELATIONAL DATABASE TO VALUABLE EVENT LOGS

FOR PROCESS MINING PURPOSES: A PROCEDURE

Mieke Jans | mieke.jans@uhasselt.be | Hasselt University, Belgium

Abstract

The art of structuring event data in such a way that it fully empowers process mining analysis, is currently only within reach after labor and time intensive trial and error. This paper reports on a procedure that can be followed to build such an event log for process mining purposes and aims to accelerate this learning curve. It intends to create awareness of the decisions an event log builder takes and its related consequences. The procedure consists out of seven steps and is written in the form of a manual for the event log builder. The report is based on nine years of process mining experience in both academics and industry by the author.

universiteit hasselt

KNOWLEDGE IN ACTION

1 Introduction

If a process is supported by a process-aware information system, extracting an event log that complies with the format prerequisites for process mining can be trivial. However, information systems are very often based on a relational database structure. In those cases, one can literally speak of 'building the event log'. These relational databases traditionally hold a huge amount of data, on a broad spectrum of related documents and on different levels of granularity. Yet this data still has to be converted into a minable event log in which events are related to a single instance and on one level of granularity. In a first step, the architecture of the event log has to be decided upon. In a second step, the event log is built by feeding the architectural frame with data. This procedure focuses on the architecture-step, not on the operational build-step, nor on the technological minable format in which the event log needs to be converted (preferable XES).

2 Objectives of the procedure

The procedure is a sequence of steps that could be followed during the architecture-step of creating an event log for process mining, starting from a relational database. The design of the procedure, together with a running example, seeks to meet following objectives: it will

1. be possible to be employed by process analysts with limited knowledge of process mining, i.e. the analyst is familiar with the concept of process mining and the prerequisites of an event log, but may not have any experience in conducting a process mining analysis herself.
2. increase the analyst's understanding of the decisions and their consequences, related to the choice of process instance, activities and attributes.
3. provide the process analyst with a practical example or sufficient background information in order to conduct the approach in a consistent manner.

3 Procedure

The architecture building phase starts with step 1, being preceded by a preparatory step 0. The process analyst is the person that carries out the procedure. He should take the lead in this approach, guiding the stakeholders through the steps. It is the process analyst's responsibility to safeguard the correct application of the approach and to guide the stakeholders through the decisions to be taken. The process analyst has to provide the stakeholders with enough information to take well-informed decisions, but has to be cautious not to overload them with too many technicalities.

Step 0: State process and primary goal → stakeholder identification

As preparatory step, the analyst should state the process to be mined and the goal of the analysis, according to the project sponsor. There are two categories of goals: efficiency and compliance. In this step, it is important to decide what the main emphasis of the analysis will be, the 'must have' output of the project. In reality, often both goals of efficiency and compliance are combined. Still, there is a primary goal that we recognize as a 'must have', and a secondary goal that can be seen as a 'nice to have'. For example, an internal auditor might have a primary goal of assuring compliance (the 'must have'), but sees extra insights in efficiency as a 'nice to have'. A process owner, on the other hand, will probably aim for efficiency gains as a

'must have', taking the compliance assurance for 'nice to have'.

Based on the preparatory step, the process analyst identifies the stakeholders and schedules a meeting to go through the steps of the approach.

Running example. *We will use a running example throughout the steps of the approach for demonstration purposes. The running example is inspired by real life contexts and database constructs. The example analysis comprises a traditional procure-to-pay process, analysed by an internal auditor whose primary goal is compliance. Consequently, the stakeholders of this project are the overall responsible of the procure-to-pay (P2P) process (the process owner), the IS domain expert that is acquainted with the underlying P2P system (SAP), and the internal auditor. A meeting is scheduled by the process analyst, inviting these three persons.*

Step 1: Reconfirm primary goal and identify process cornerstones

According to this approach, the following stakeholders should be involved in the architecture building phase: (a) the process owner of the process under investigation, and (b) the information systems domain expert that is acquainted with the supporting information system of the process at hand. Optionally, (c) a third party project sponsor (in case this is not the process owner or the IS domain expert) can be involved in this phase.

To apply the approach, a meeting with the process analyst and the identified stakeholders should be set up to go through all the steps, starting from step 1. This meeting should be attended by all invitees at the same time (in contrast to splitting the meeting in several 'sub-meetings'), in order to guarantee a cross-over of all those person's expertise and a well-functioning of the approach. An important aspect of this meeting is a general understanding of the process and its underlying business.

At the start of the meeting, the goal, as understood by the process analyst, should be reconfirmed by the stakeholders. To facilitate communication in later phases, three to five key questions should be listed. The questions are example questions of what the stakeholders expect to have the answer to, at the end of the process mining analysis. The formulation of these questions forces all team members to turn to concrete process aspects and assures everyone is on the same line when they speak about their goal. In addition, implicit project expectations are converted into explicit expectations. This assists the process analyst in managing expectations towards realistic project outcomes, a key factor for project success. The questions could be formulated according to the SMART format, where you aim to formulate your questions in such a way that they are Specific, Measurable, Attainable, Relevant, and Time-bounded.

After the goal confirmation, the first step requires an identification of the process cornerstones. A process cornerstone is a key activity in the process according to different stakeholders. In a business process, these activities are often tied to transactions that are executed on documents (for example signing an invoice), while in operational processes this is not so often the case. The goal that was set, can influence the selection of process cornerstones in terms of scope setting. This should be kept in mind.

The output of step 1 is

- an agreed-upon goal must-have of efficiency or compliance,
- a set of key example questions that the stakeholders aim to have an answer on at the end of the process mining project, and
- a list of process cornerstones.

Running example. *The meeting conversations could yield the following output:*

- *Goal: compliance*
- *Key questions:*
  - *'Is there Segregation of Duty (SOD) between the Purchase Order (PO) creation and the Goods Receipt?'*
  - *'Is there SOD between the first and second level of approval?'*
  - *'Does an invoice always stem from a PO?'*
  - *'Is there a new approval, after someone alters the PO?'*
- *Process cornerstones:*
  - *Create a Purchase Requisition (PR)*
  - *Create a Purchase Order (PO)*
  - *Approve PO*
  - *Receive Goods*
  - *Book Invoice*
  - *Add PO line (This cornerstone comes forward as activity that could influence process execution and is added to the list of cornerstones)*

Step 2: From cornerstones to key tables

In contrast to the first step, the second step turns towards a more technical discussion. To this end, it is important to provide the stakeholders at this point in time a clear vision on the end goal of the meeting. It is suggested to keep the explanation as simple as possible, and therefore the process analyst is recommended to only use two graphics to explain the concept of process mining and the desired format and terminology of an event log (Figure 1 and Table 1 respectively).

Figure 1 could be used to explain the basic concept of process mining, on the link between real life transactions, data that is captured in the supporting information system, and how this is used as input for process mining to produce as-is process models. Given the assumption that the process analyst is acquainted with process mining, it goes beyond the scope of this report to write out a literal explanation of how to introduce the stakeholders to process mining. In case the process analyst finds the picture and brief explanation above shortcoming to execute this step with confidence, this should be seen as an indication that the process analyst's experience with process mining is too low to conduct the event log building phase independently.

To introduce the concept of an event log, along with its assumptions in most process mining algorithms, Table 1 can be used. The table shows a simple example of an event log of a helpdesk

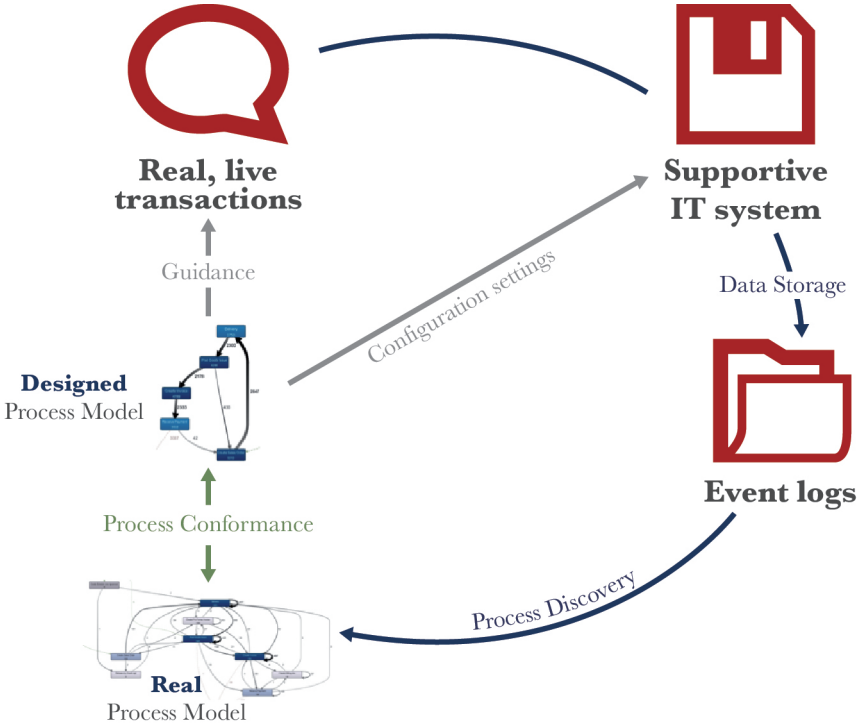

Figure 1: Process mining concept

Case ID	Activity	Timestamp	Resource	Role	Priority	Impact	Finding	Result
1	Create ticket	January 2, 3:15 PM	Sarah	Client	2	Medium		
1	Screen ticket	January 2, 3:32 PM	Li	Junior analyst			Simple	
1	Repair - simple	January 10, 9:45 AM	Li	Junior analyst				Repaired
1	Close ticket	January 10, 11:34 AM	Steve	Assistent				
2	Create ticket	January 2, 4:04 PM	Philip	Client	2	High		
2	Screen ticket	January 2, 4:05 PM	Li	Junior analyst			Complex	
2	Repair - complex	January 3, 1:38 PM	Marie	Senior analyst				Repaired
2	Close ticket	January 4, 9:23 AM	Steve	Assistent				
3	...							

Table 1: Event log example

(possible IT, not necessarily) with two traces, a process that everyone in a business can interpret without a specific background prerequisite. The event log should assist the process analyst in introducing the event log as such, along with the concepts 'case id', 'activity', 'timestamp', and 'attributes'. The difference between case and event attributes should be left out of the discussion at this moment in time, only introducing concepts that are necessary at this point in time. The same holds for the lifecycle transitions that are represented by an activity (for example 'start', 'complete', 'assign', ...).

After introducing the stakeholders to process mining and the concept of an event log, the second step can be taken. In this step, the underlying tables of the cornerstones, listed in step 1, are identified. In business processes, activities –and hence cornerstones- typically refer to actions that relate to documents. Think for instance at 'signing of an order' or 'entering the goods receipt in the system'. In such processes, it is suggested to take an intermediary step from cornerstones to underlying documents. Once the underlying documents are made explicit, the step can be finalized by identifying the tables that capture the timestamp of the action on the document that was listed as a cornerstone. It could be that, depending on the individuals in the meeting, different underlying documents are identified when multiple documents are related. This is not a problem, since they all lead to the same goal: identifying the timestamp of the action that is listed as a process cornerstone. Making the underlying documents explicit is merely a facilitating step.

It is of utmost importance that the process analyst verifies whether the selected timestamp actually captures what is expressed in the cornerstone and nothing related though not similar to it. Going from the identified cornerstones to specific fields in the database that capture related timestamps, might force the architecture to discard or to reformulate previously listed cornerstones. For example 'Receiving goods' might be transferred into 'Enter Goods Receipt in system' because there is no data field that captures the timestamp of actually receiving the goods.

The output of this step is a list of tables that capture timestamps that are related to the cornerstones of step 1.

Running example. *The meeting conversations could yield the following output:*

Cornerstone	Document	Table
Create a Purchase Requisition (PR)	PR	PR header
Create a Purchase Order (PO)	PO	PO header
Approve PO	PO	Change log header
Receive Goods ! Enter GR	PO	PO history
Book invoice	Invoice	Invoice header
Add PO line	PO	Change log line

The list of cornerstones, the output of step 1, is the input for step 2. The related documents are the documents that the stakeholders associate with the transactions that are mentioned in the cornerstones. This might differ from group to group. For instance, this group identified the PO as the related document for a goods receipt, where the IS domain expert narrowed this down to a specific table that captures the timestamp of entering a Goods Receipt in the system. Another group however, might have chosen for the Goods Receipt document itself as underlying document. As mentioned above, this is no problem, since both documents will lead to a timestamp of the cornerstone.

Going from the listed cornerstones to the specific database at hand, forced the architecture to adapt the cornerstone 'Receive Goods' to 'Enter GR'. The remaining cornerstones are all backed with a logged timestamp in the database and remain unchanged.

The list of tables holds tables like 'Invoice header' or 'Change log line'. This kind of tables refer to a parent-child relationship between tables. In these relationships the parent table captures general information of a document, traditionally relating to the header of a document. The child table captures information that differs from line to line on the body of the document. This kind of set-up is very common at ERP-system databases.

Step 3: Identify relationships between key tables

In step 3, again the output of the previous step is taken as input for the current step. Starting from the listed tables in step 2, an entity-relationship diagram[1] is drawn, depicting all underlying relationships of the tables that were mentioned this far. The process analyst might opt for a short explanation of this step by using Figure 2, showing shortly the concepts of an entity-relationship diagram and its notation.

The analyst might also opt for a more natural explanation on relationships between tables, without talking about for example cardinality symbols, by asking questions. The questions should be formulated like 'Is there for every ... (document B) a ... (document A), or could it

[1]Or another language can be used, like for example UML.

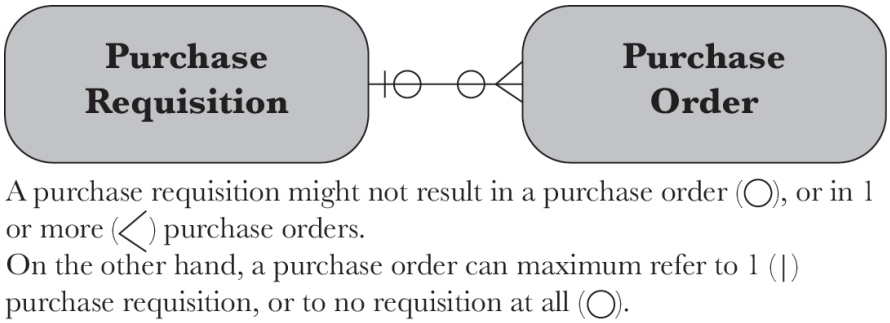

Figure 2: Entity-Relationship diagram concept

also exist without a preceding document?' and 'Could it also be that more than 1 ... (document A)'s result in the same ... (document B)?' etc.

The output of this step will be an entity-relationship diagram that holds, at minimum, the tables that were listed in step 2. By expansion, this diagram will comprise additional tables to make the links between the listed tables so far. The most important part of this step is going through the exercise of discussing the structure of the database and reach a full understanding of how the tables, that hold the document information, are related. The concept of parent-child relationships, as mentioned in the running example in the previous step, is important to pay attention to. This is crucial to understand when selecting the process instance in step 4. Also, the presence of many-to-many relationships should be identified and well-understood by all stakeholders.

Running example. *The meeting conversations could yield the output as depicted in Figure 3:*

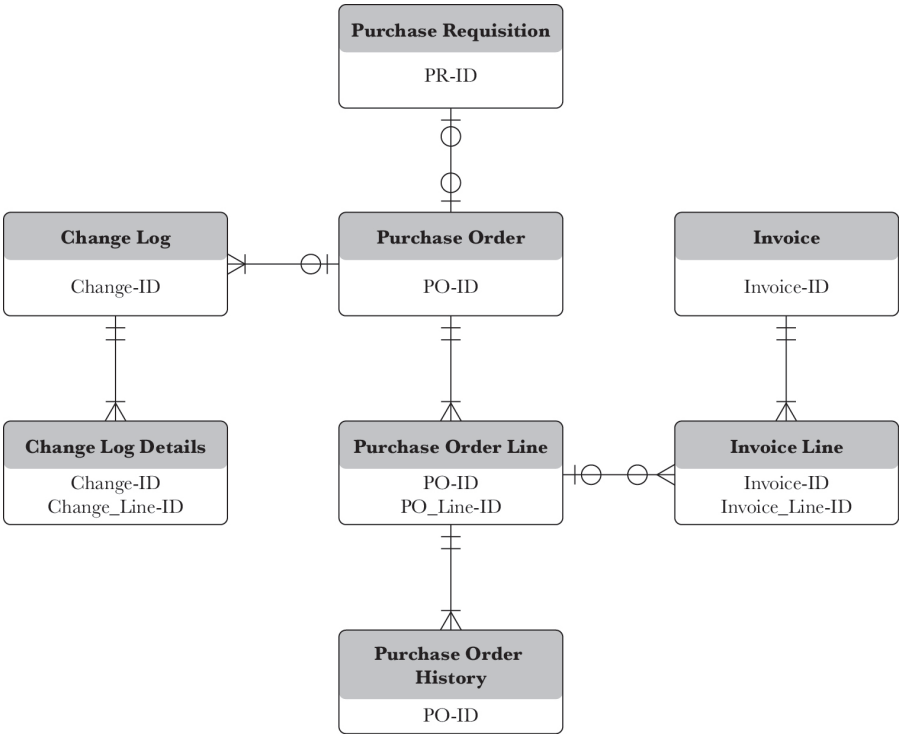

Figure 3: Running example entity-relationship diagram

The tables that were listed in the previous step are taken as a start. Two additional tables are depicted. These tables are the link between the table 'PO header' and 'PO invoice'. The example shows three documents that are represented by tables in a parent-child relationship: the PO, the invoice, and the change log. Further, there is a many-to-many relationship between the PO's and the invoices. This means that one single PO can result in multiple invoices, but likewise, one single invoice can refer to multiple PO's.

Step 4: Select process instance

This step, selecting the process instance, is the most crucial step of the approach, since the interpretation of the following analysis is directly influenced by it. The process instance is the unique piece, document or case –depending on the type of process- that is followed throughout the process. As already mentioned, in business processes, activities are often related to documents and process instances naturally reflect one of those documents. In this light, there are possibly two dimensions related to the decision of process instance selection. Firstly, one has to decide which document to select as process instance: a document that triggers the process, or a document that is created in the last phase of the process? In case the selected document is stored in database tables with a parent-child relationship, a second dimension of the process instance selection imposes itself: which level of granularity of the selected document is chosen? Decisions on these two dimensions should be taken.

Step 4.1: Document selection of process instance

As a start, the documents that carry the transactions of the process need to be identified and listed as candidate process instances. This is in general a simple deduction from the base, designed process model.

Running example. *In our example, the candidate documents would be a PO, a Goods Receipt document, and an invoice. If a PO is selected, then all events related to a single PO will comprise a process instance's trace. This may include several goods receipt entries and several invoices. If an invoice is selected, all events that eventually lead to one invoice will be considered part of that process instance's trace. This may include several POs or only a part of a single PO.*

To select one document out of these candidates, two aspects have to be considered: the goal of the process analysis, and the cardinality of the relations between the candidate documents. Both aspects will be discussed in the two next subsections.

Document selection and goal of process analysis If all process executions start with the same document, because there is technically no other start possible, the start document should be selected as process instance document. However, if a process execution has multiple points of entry, the goal of the analysis (see step 1) influences the document selection.

As explained before, we distinguish two types of business goals of the analysis: efficiency and compliance. If the primary goal is efficiency, it is recommended to select the start document as process instance document. By selecting the start document as process instance, fall-out will be identified. For example, POs or leads that do not result in a final invoice or sale will be identified. Unlike cases that do not result in an end document, cases that do not start with the prescribed start document will however not be identified when the start document is used as process instance. For this reason, selecting the start document in case of a formulated compliance goal is not desirable.

If the goal is compliance, the end document could be selected in case of a one-to-one relationship between the start and end document. Starting from the selected end documents, the preceding documents can be related to the process instance via back-tracing. That way, all different entry points will be taken into account and a process instance's trail will be reconstructed, as long as it is related to an end document. An invoice that is booked as end document, but which is not preceded by a PO for example, will be shown. That is why this approach is suggested for a compliance goal. The downside is that process executions that did not reach the end document yet, are falling out of scope. Applying these guidelines implies a trade-off between identifying fall-out and identifying non-compliance.

Selecting the end document as process instance when the goal is compliance, is not always the best solution. In case there is a many-to-one relationship between the start document and the end document (i.e. multiple start documents could result in one end document), an alternative solution should be employed. This solution, that also counters the trade-off between fall-out and non-compliance, is to use a dynamic process instance. A dynamic process instance is a process instance that, depending on the individual case, may have a different underlying document as process instance. For example: if a PO is present, the process instance is that PO, else it is the invoice. In case of such a dynamic process instance, both fall-out and non-compliance, or combinations will be detected. This dynamic process instance uses the first document that is present for an individual case, as process instance. That way, the many-to-one relationship will be reduced to multiple one-to-one relationships. The disadvantage of this approach, however, is a more difficult interpretation of the analysis results in the next phase. 'A case' can refer once to a PO, and in another case to an invoice. This will make communication and analysis more difficult. It is up to the process analyst to verify whether there is a real need for this dynamic process instance. A key factor of success in the use of such a process instance, is a clear understanding of this artificial set-up by the stakeholders.

Running example. *In case of an efficiency-inspired project, the PO would suit best as process instance. In case of a compliance-driven project, the invoice might be a better candidate. Particularly since the invoice is the direct link to financial reporting, mostly the subject that needs to be provided assurance on in compliance-driven projects. In case a dynamic process instance is used, a 'purchase', the case, will be represented by a PO in case a PO is present, or by an invoice, when no PO is present. Given that a goods receipt document can only exist after creating a PO, only the PO and the invoice will be valid representations of the dynamic instance.*

Document selection and cardinality between candidate documents In case the candidate documents show cardinalities that are different than one-to-one, the process analyst should be aware of the consequences of choosing one document over the other document. Having many-to-many relationships can be disentangled to one-to-many and many-to-one relationships. Both the many documents-to-one instance relationship and the many instances-to-one document relationship hold their own consequence.

In case of a relationship where multiple documents can be related to a single process instance, the event log will link a repetition of transactions on these other documents to a single instance. The resulting traces will contain repetitions or self-loops of these activities, representing the many-to-one relationship correctly. In case of a relationship where multiple process instances can be related to a single other document, however, the event log will artificially duplicate the transaction(s) on that other document (under the assumption that these activities are included in the event log). This would result in a misrepresentation of reality. In case of a many-to-many relationship between two documents, one has to decide for which document an artificial

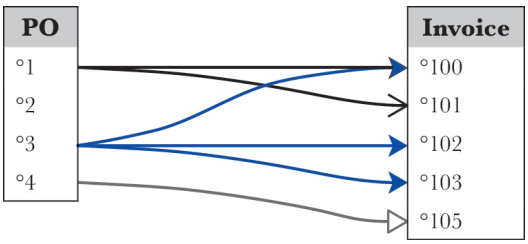

Figure 4: Many-to-many example

Case ID	Activity	Document Nr.
1	Create PO	PO 1
1	Book Invoice	Inv 100
1	Book Invoice	Inv 101
2	Create PO	PO 2
3	Create PO	PO 3
3	Book Invoice	Inv 100
3	Book Invoice	Inv 102
3	Book Invoice	Inv 103
4	Create PO	PO 4
4	Book Invoice	Inv 105

Table 2: Event log with PO selected as process instance

duplication of the related transactions is least harmful or more in line with the goal of analysis. Specifically, if the analysis also relates to the load of resources, duplication of events is not appropriate and might present a biased picture. Notice that, although the decision relates to the selection of a *document*, the consequences are on the level of the related *transactions* that take place on the involved documents.

Running example. *Figure 4 shows an example combination of POs and invoices, like they could appear in our running example, based on the Entity-Relationship diagram of Fig.3. There is a many-to-many relationship between PO and invoice in this example. In case a PO is selected as the process instance, this will lead to the events, listed in the event log in Table 2 (making abstraction of all other activities). The two consequences, explained before, can be seen as follows:*

- *The traces of case 1 and 3 reveal a self-loop on the activity 'Book Invoice'. This is a result of the 'one-to-many' relationship between the document that represents the process instance, the PO, and the related document, the Invoice. Notice that in case multiple events would be related to the invoice, for example also 'Pay invoice', these will all result in multiple events in a single trace. As such, the event log will uncover the repetition of the same activity(-ies) 'Book invoice' and/or 'Pay invoice' on a single PO, adequately representing reality at the level of the process instance.*
- *The activity of booking invoice nr. 100 is mentioned twice in the event log, although this activity only took place once. This is a result of the 'many-to-one' relationship between the document that represents the process instance and the related document. Two process instances, PO 1 and PO 3, refer to the same invoice. Since the PO is the process instance, followed throughout the lifecycle, the event of booking the invoice is artificially multiplied. This poses a discrepancy between reality -invoice 100 is only booked once- and the number of times an activity is included in the event log (twice). This effect does not have to pose a great problem, it only should be taken into consideration when analyzing results. It is mostly important for avoiding an interpretation of the summary statistics on activity frequency.*

If, in this example, the invoice would have been selected as process instance, similar consequences would hold, only starting from a different point of view. A self-loop on the activity 'Create PO' would show up, uncovering the creation of multiple POs, preceding the booking of a single invoice. Further, the activity 'Create PO' would be artificially multiplied, because of the many-to-one relationship between PO and invoice.

To take many-to-many relationships and its consequences into account in the event log architecture, it is important to turn to the key questions and the goal of the project. What

exactly do you wish to get an answer to? For example, a financial auditor is primarily interested in whether invoices were preceded by an approved PO. Therefore, selecting the invoice as the process instance, avoiding a possible artificial multiplication of financial documents, is highly recommended.

Step 4.2: Granularity level selection: parent or child

In case the document, as selected in step 4.1, is represented by a parent-child relationship in the database (see step 3), one has to decide which level of the document should serve as a process instance. Will the document as a whole be followed throughout the process, or will it be divided in sub-parts to follow separately throughout the process?

Firstly, this decision is based on the formulated key questions. If the key questions refer to documents as a whole, or to information that is stored on a header level of a document (like customer involved, region,...), less granularity should be chosen. If the key questions on the other hand refer to information that is typically of finer granularity (f.e. the amount of an invoice, which is typically information that is stored on a line, not in the header), then the finer-grained level of information should be considered.

To take the ultimate decision, two scenarios and their event log consequences should be clear. These are the mix of header level process instances with activities on a more granular level, and the mix of lower level process instances with activities on header level. These scenarios are discussed in the following paragraphs, followed by the final decision at hand: which level of granularity should be selected?

The mix of process instances at parent level with activities on child level Often, a mix of granularity among the activities is present: some activities refer to transactions that relate to complete documents, while other activities refer to transactions on sub-parts of documents. Differently stated: some activities are transactions on the level of the parent, others are on the level of the child. A goods receipt, for example, typically refers only to a line of a PO that has been received (child-level), while an approval is mostly provided on a complete document (parent-level). If a process instance is selected on the parent-level, it is important to decide on the aggregation function of activities that relate to activities at the child level, a finer-grained level of information. For example, if a PO is selected as process instance to follow throughout the procurement process, activities that refer to a PO as a whole do not pose a problem. In a process execution as designed, one process instance will relate to exactly one activity of creating this PO and one activity (or two if desired) on approving this PO. The activities that relate to subsets of this process instance on the other hand require more attention. If a goods receipt for example is entered for each line of a PO, and a PO can have an unlimited number of lines, does the event log needs to include the goods receipt of all individual lines? This will yield traces like 'Create PO – Approve PO – Goods Receipt – Goods Receipt - ... - Goods Receipt - ...', resulting in self-loops on the activity at the lower granularity level. This will provide the analyst with information like 'there are a lot of goods receipt activities on one PO'. However, it will not capture whether there are inefficiencies on line level for example. There will be no insights in whether this self-loop is ascribed to a single goods receipt for each line or to partial deliveries on one line. This is due to the level of the chosen process instance, which dictates on which level information is gained.

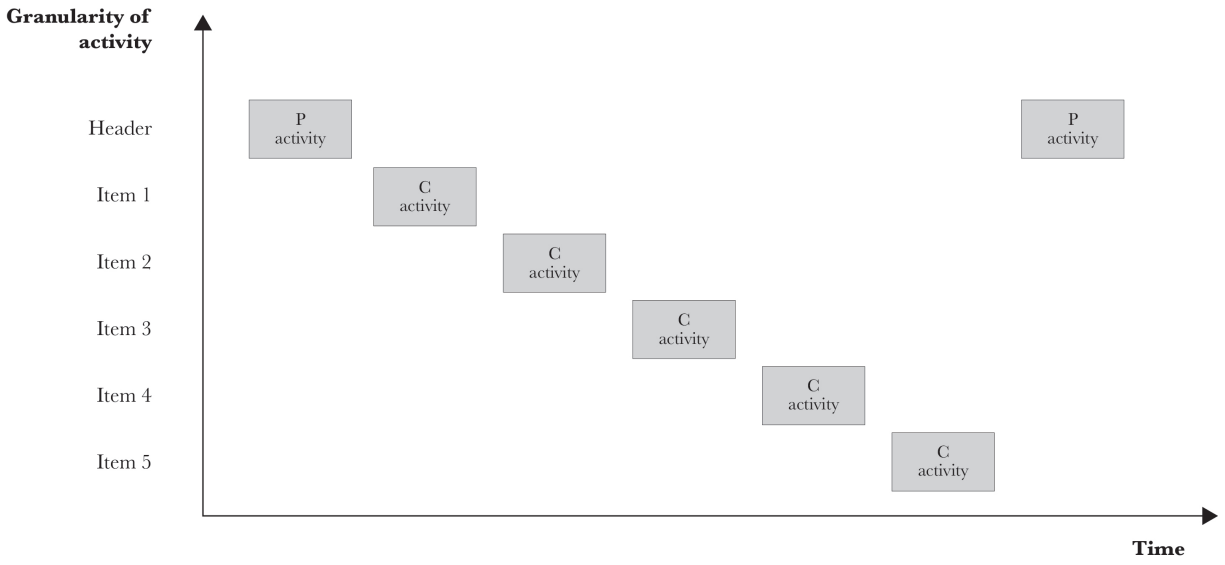

Figure 5: Header and line level activities example 1

Take for example Figure 5, a graphical representation of what could happen in reality. In reality, one activity at header level (P activity), five similar activities on five children (C activities), and again one activity at header level have been executed. If one would select the process instance at header level (in our example a PO as process instance), this would result in one trace with a sequence of <P, C, C, C, C, C, P>. There is no insight on whether this trace represents the situation like depicted in Figure 5 or a situation like depicted in Figure 6.

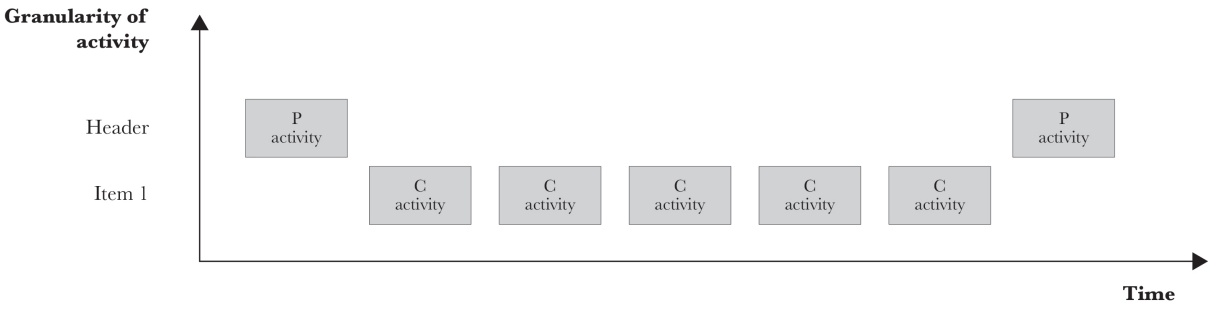

Figure 6: Header and line level activities example 2

In essence, this is a similar situation to the one-to-many situation that was described in the step of document selection. As a result, an important decision in this situation is whether all these finer granularity level activities should be part of the event log, or would an aggregation suffice too.

The mix of process instances at child level with activities on parent level The opposite of the previous situation also has its own characteristics. What are the consequences of choosing a process instance at a lower level of granularity? Take the same example of a PO with the activities 'Create PO' and 'Approve PO' on the highest level of granularity, and 'Goods Receipt' on a lower level. In case the lower level, a PO line, is taken as a process instance, all higher level activities have to be multiplied. This is again the same principle as with the one-to-many table relationships. A PO with for example ten lines, representing ten process instances, could yield ten identical traces of 'Create PO – Approve PO – Goods Receipt'. At first sight, 30 activities seem to be registered (ten traces of three activities). However, only 12 activities are registered: one 'Create PO', one 'Approve PO', and ten times 'Goods Receipt'. Exactly like the previously described parent-child relationships, the activities that relate to the higher level

of granularity are multiplied and should be dealt with cautiously when analyzing the results. Returning to the example, depicted in Figure 5, choosing a process instance at the lower level of granularity, would yield an event log of five identical traces <P, C, P>, instead of one trace with a repetition on I. The example in Figure 6 on the other hand, would result in a different event log. The log would only contain one trace <P, C, C, C, C, C, P>, revealing unambiguously the repetition of the same activity on one single item.

Level of granularity decision Turning back to the essence of step 4.2, a decision on the level of granularity of the process instance needs to be taken. To take this decision in a founded matter, the key questions, as formulated in step 1, should be taken as input. From each question, the related documents (that capture the activities of relevance to the question) and the associated level of granularity should be listed. Based on this list, a decision on granularity level needs to be taken. As a general rule, a parent level is recommended. In case the key questions refer to activities at child-level, there needs to be assurance that answers can still be formulated when using aggregations of activities at child-level. A clear aggregation function should be stated for all activities that relate to a finer granularity than the selected process instance: are all those activities captured in the event log, only the first one, the first one after each interrupting activity, or an alternative condition?

For these discussions, a true understanding of the concepts above is crucial, and the process analyst could use Figure Figure 5 to introduce the stakeholders to these aspects. Instead of using 'P activity' and 'C activity', a parent and child activity of the process at hand should be chosen to make it more concrete.

Exceptional situation: child level activity triggers parent level activity One particular situation requires special attention: when an activity at a finer level of granularity triggers an activity of a higher level. If this construct is present, and the assurance of this construct is part of the analysis, the finer level of granularity, the child, needs to be selected as process instance. Notwithstanding this rule, this selection results in undesired traces.

Take a look at the example in the context of our procurement process in Table 3. The activity 'Add line' is an activity at a finer level of granularity. If this activity takes place, it triggers the activity 'Approve PO', an activity at parent level. In the example, a PO is originally created with four lines. After approval, the goods receipts of all separate lines take place. In parallel, a fifth line is added. This activity triggers a new approval of the PO. While the invoices are received for the first four lines, the goods receipt of the 5th line takes place in parallel.

If the event log would follow a PO at the parent level and include all activities at a lower level –i.e. using no aggregation function–, the trace of this specific PO would look like follows: <Create PO, Approve PO, Goods Receipt, Goods Receipt, Add line, Goods Receipt, Goods Receipt, Approve PO, Receive Invoice, Receive invoice, Receive invoice, Goods Receipt, Receive Invoice, Receive invoice>. If one would analyze this trace, questions like 'Why is there a goods receipt before the PO is approved?' raise. In reality, the interventions of the activity 'Add line' and 'Goods Receipt', relating to line 5, are in parallel with the flow of the first 4 lines. By following a process instance at parent level, one cannot uniquely link these parent level activities to the child activity, but at least they all appear on one trace. It provides the possibility to use this 'all-capturing' trace and rely on assumptions about the triggering characteristic of certain activities (like 'Add line' here) to explain odd occurrences of the responding activity (like 'Approve PO'). Selecting the process instance at the finer level of granularity on the other hand, splits the activities at child level to different traces and no connection can be made anymore

Activity at parent level	**Activity at child level**		
Create/Approve PO	*Goods Receipt*	*Invoice Receipt*	*Add line*
Create PO (with 4 lines)			
Approve PO			
	Goods Receipt of line 1		
	Goods Receipt of line 2		
			Add line 5
	Goods Receipt of line 3		
	Goods Receipt of line 4		
Approve PO			
		Receive Invoice of line 1	
		Receive Invoice of line 2	
		Receive Invoice of line 3	
	Good Receipt of line 5		
		Receive Invoice of line 4	
		Receive Invoice of line 5	

Table 3: Example triggering activity

between the triggering activity in one trace and the response activity that is copied to all related traces. Therefore, a process instance at parent level is preferred over a process instance at child level, even if it does not provide full assurance either.

The final output of step 4 is an identification of the table and the field in the database that captures the unique id of the selected process instance.

Running example. *The decision on which document to select as a basis for a process instance, is linked to the goal of compliance in this example, since not all purchases are system technically forced to start with the creation of a PR (hence no unique start activity). Following the guidelines of step 4.1 and taking the many-to-one relationship between the start document (the PR) and the end document (the invoice) into account, a dynamic process instance is selected. This boils down to a PR as process instance in case there is a PR, a PO as process instance in case the process execution started with a PO, and an invoice as process instance if neither a PR, nor a PO exists for a specific invoice.*

Given that the PO and invoice-related data are captured in two times two tables in our database, the level of granularity has to be decided upon too. There are two options: to follow a PO/invoice as a whole throughout the process, or to follow separate lines. To guide this decision, the key questions from step 1 are retaken:

- *'Is there Segregation of Duty (SOD) between the Purchase Order (PO) creation and the Goods Receipt?'*
- *'Is there SOD between the first and second level of approval?'*
- *'Does an invoice always stem from a PO?'*
- *'Is there a new approval, after someone alters the PO?'*

The underlying activities with their involved documents, including their granularity level of the activity that is mentioned, are:

Create PO and Receive goods	*→ PO header – PO line*
Approve PO	*→ PO header*
Create invoice and create PO	*→ Invoice header and PO header*
Approve PO and Add PO line	*→ PO header – PO line*

Creating an invoice that is stemming from a PO is listed on header level of granularity, since creating both an invoice and a PO are activities on header level. However, we know from the relational database, that the link between these two documents is created on a lower level, with a many-to-many relationship. Following the general principle of taking the header level, and checking whether it is possible to answer the key questions, leads us to the following thinking exercise.

Firstly, the nature of the dynamic process instance, given the underlying relationships, needs to be clear. Since there is a one-to-one relationship between a PR and a PO, we can leave out the PR of the discussion. In case a PO header level process instance would be selected (if the case starts with a PR and/or PO), this could result in traces with multiple invoices (for example 'Create PR – Create PO – Book Invoice – Book Invoice – Book Invoice'). In case the invoice has no preceding PR/PO, and the process instance is, as a consequence, the invoice at header level, there will be maximum one activity of 'Book Invoice' in such a trace, and the different 'Book Invoice' activities will be captured in separated traces. This difference does not need to pose a problem, but the event log builders should be aware of these dynamics.

In order to answer the first key question, while starting from a header level of a PO, it is recommended to retain all activities on goods receipts of all PO lines that are related to that PO (so no aggregation function). The second question relates to approvals, activities that are linked to a PO at its highest level. As a result, there is a good match between the selection of a process instance at header level and this question. The third question also can be answered easily in the header construct of process instances. It just requires a selection of all traces that start with an activity like 'Book Invoice' to filter out the invoices without a preceding PO. The last question is comparable to the first one, that also involves a mixed level of granularity. If all activities that capture an addition of a PO line are taken into account in the event log, it will be possible to answer this question. Beware that if a lower level of granularity was chosen, the issues on lower level activities that trigger header level activities would become active.

Based on the underlying relationships between PR, PO, and invoice, and based on the formulated key questions, a dynamic process instance at the header level of granularity was selected. The unique identifiers will be found in the table 'PR header', field 'PR number', otherwise in table 'PO header', field 'PO number', and else in table 'Invoice header', field 'Invoice number'.

Please note that if the key questions were more focused on the alignment of separate PO line items, the received goods, and the booked invoice lines, it probably would have been recommended to use a lower level of granularity as process instance. The process analyst should communicate clearly that, by choosing this high level of granularity, it will be more difficult to gain insights in separate order lines, their related receipts and booked invoice amounts for example. On the other hand, aggregated information of these lines can be captured in the attributes (see later).

Document number	Creator	Timestamp	Doc type	...
00001	Chris	April 29, 2015	PO	
00002	Amber	April 29, 2015	PO	
00003	Keira	April 29, 2015	Framework Agreement	
...				

Table 4: Purchase Documents

Step 5: Select activities

After selecting the process instance, the activities can be selected. The following algorithm should be followed:

1. **Initial set of activities** = set of cornerstones with their identified timestamps, as identified in step 2.
2. **Verified initial set of activities** = initial set of activities, provided that each element can be related to the selected process instance, or in case of a different level of granularity, the aggregation function that should be applied is expressed.
3. **Set of candidate activities** = verified initial set of activities, enhanced with all other timestamps of the tables, as identified in step 3, along with the activity description these timestamps capture.
4. **Pruning step**: starting from the set of candidate activities, discard the activities that are not of interest to the current process analysis, given the selected goal and scope.
5. **Possible additional activities** in case of attribute-dependent timestamps: check whether other attribute values hold interesting activities.
6. **Final set of activities** = set of candidate activities, minus the activities that are discarded in the pruning step, plus the additional activities that are selected in the fifth step.

The fifth step might require some further explanation. The attribute-dependent timestamps relate to timestamps that capture a specific activity, only if another field (an attribute) holds a specific value. Take for example Table 4, that holds information on purchase documents. During previous steps, it might have been that the activity 'Create Purchase Order' was proposed. On the question whether there was a timestamp that captured this cornerstone, the IS expert would probably have answered that the field 'Timestamp' in this specific table captures indeed the activity of creating a purchase order, on condition that the field 'Doc type' holds the value 'PO'. This is an example of what we call 'an attribute-dependent timestamp'. The fifth step of the procedure requires a closer look to all other possible values of that 'attribute' the timestamp depends upon to capture a specific activity. In this case, 'Doc type' also holds a value 'Framework Agreement' and might also hold other values. Consequently, the stakeholders have to decide whether to include the activity 'Create Framework Agreement' in their event log or not. The underlying assumption of this procedure, is that no value-adding or important activities are missed if one starts from the cornerstones, as identified by the stakeholders. The stakeholders, representing both the business and the IT dimension of the process, provide insights from what they perceive as most important steps in the process. This is enhanced with other timestamps in related tables, capturing possibly missed key activities. The probability that, by using this approach, key activities are still missing in the event log, is presumed to be close to zero.

The output of this step is a list of tables and timestamp fields, along with the name of the activity that these timestamps represent. In case of an attribute-dependent timestamp, the attribute value of interest should be specified. In case of an aggregation function (for example: only the first activity in its kind per process instance), this should be stated too. If nothing is stated, no aggregation will be executed in the next phase of event log building.

Running example. *Going through the algorithm for our running example, could lead to the following outputs:*

1. *Initial set of activities*

Table	Field	Activity
PR header	CreationDate	Create PR
PO header	CreationDate	Create PO
Change log header	SystemTime	Approve PO
PO history	Timestamp	Enter GR
Invoice header	BookingDate	Book invoice
Change log line	SystemTime	Add PO line

2. *Verified initial set of activities*

Table	Field	Activity	Verification
PR header	CreationDate	Create PR	Link to process instance (PR) = OK
PO header	CreationDate	Create PO	Link to process instance (PR/PO) = OK
Change log header	SystemTime	Approve PO	Link to process instance (PR/PO) = OK Attribute `action' must equal `approve'
PO history	Timestamp	Enter GR	Link to process instance (PR/PO) = OK Attribute `document' must equal `GR' No aggregation function
Invoice header	BookingDate	Book invoice	Link to process instance (PR/PO/Invoice) = OK
Change log line	SystemTime	Add PO line	Link to process instance (PR/PO) = OK Attribute `eld' must equal `POLine' No aggregation function

3. *Set of candidate activities, including five extra activities (in bold) in comparison to previous step*

Table	Field	Activity	Verification
PR header	CreationDate	Create PR	Link to process instance (PR) = OK
PR header	**ClearingDate**	**Create PO**	**Link to process instance (PR) = OK**
PO header	CreationDate	Create PO	Link to process instance (PR/PO) = OK
PO header	**ChangeDate**	**Last change PO**	**Link to process instance (PR/PO) = OK**
Change log header	SystemTime	Approve PO	Link to process instance (PR/PO) = OK Attribute 'action' must equal 'approve'
PO history	Timestamp	Enter GR	Link to process instance (PR/PO) = OK Attribute 'document' must equal 'GR' No aggregation function
Invoice header	BookingDate	Book invoice	Link to process instance (PR/PO/Invoice) = OK
Invoice header	**SystemDate**	**Enter Invoice**	**Link to process instance (PR/PO/Invoice) = OK**
Invoice header	**InvoiceDate**	**Supplier Creates Invoice**	**Link to process instance (PR/PO/Invoice) = OK**
Invoice header	**ClearingDate**	**Pay Invoice**	**Link to process instance (PR/PO/Invoice) = OK** **Attribute 'reference' must be empty**
Change log line	SystemTime	Add PO line	Link to process instance (PR/PO) = OK Attribute 'field' must equal 'POLine' No aggregation function

4. *Pruning step*
   The following activities are not withheld:

   - *Create PO, retrieved from the timestamp in table 'PR header'.*
   - *Last change on PO, because it has no added value to the stakeholders.*

5. *Possible additional activities, stemming from the attribute-dependent timestamps*
   The field 'SystemTime' of table 'Change log header' is up till now only used to distill information on approving PO's. Other values of the attribute 'action', that are selected by the stakeholders are 'reject' and 'forward'. This leads to two extra activities: 'Reject PO' and 'Forward PO'. The same reasoning is applied on the activity 'Add PO line', leading to the incremental activity 'Increase value PO line'. No further activities of interest were identified in this step.

6. *Final set of activities*

Table	Field	Activity	Verification
PR header	CreationDate	Create PR	Link to process instance (PR) = OK
PO header	CreationDate	Create PO	Link to process instance (PR/PO) = OK
Change log header	SystemTime	Approve PO	Attribute 'action' must equal 'approve'
Change log header	SystemTime		Attribute 'action' must equal 'reject'
Change log header	SystemTime		Attribute 'action' must equal 'forward'
PO history	Timestamp	Enter GR	Attribute 'document' must equal 'GR'
Invoice header	BookingDate	Book invoice	
Invoice header	SystemDate	Enter Invoice	
Invoice header	InvoiceDate	Supplier Creates Invoice	
Invoice header	ClearingDate	Pay Invoice	Attribute 'reference' must be empty
Change log line	SystemTime	Add PO line	Attribute 'field' must equal 'POLine'

Step 6: List attributes

In this step, all the attributes are listed for the data analyst that will build the event log in the next phase. To list the attributes, one starts from the output list of the previous step, entailing all tables that are related to the activities. In those tables, characteristics, other than the timestamp, are captured. There are two types of attributes: attributes that contain characteristics of the process instance, and attributes that contain characteristics of the activity on a process instance. The former attributes are case attributes, the latter event attributes. In general, the following rule can be applied: attributes that relate to an activity that can only occur once per process instance (like for instance the creation of the selected process instance), are in se case attributes. The remainder of attributes are event attributes. Depending on the tooling that is used, the distinction between case and event attributes can be made. For example, in ProM Import and XESame (tools to convert data into a minable event log format), case and event attributes are dealt with accordingly. However, some tooling (mostly commercial) expect one flat file (one large table) as event log. In such format, all attributes are treated as event attributes, since a record in such a file relates to an event, and not to a case. This format forces the process analyst to store case information on event level. One could opt to assign the case attributes to one single activity, but in order for this information to always be captured, the analyst should make sure this is a mandatory activity (for example an activity like 'create PO' when PO is selected as process instance). If there is no such mandatory event for each process instance, appearing exactly once per trace, one could create an artificial start event for all cases and relate all case attributes to this artificial start event. The other option is to assign the case attribute values to all events. That way, the information is for sure captured in the event log. The downside is that the summary statistics always need to be filtered to remove the attribute redundancy.

As a last remark, interesting calculated attributes can be listed too. For example, if the selected process instance is at parent level, it might be of interest to aggregate some information that is stored at a lower level. For example, one might desire to include the sum of all invoice amounts that relate to one PO.

The output of this step is a list of case attributes and a list of event attributes. For each attribute, a specification of the table and field that stores this information is given. The event attributes are listed per event type. Optionally, calculated attributes can be listed in a third list with a clear formula how these attributes should be calculated, on which level the input is taken, and on which level the calculated attribute needs to be stored.

Running example. *For the tables, identified in the previous steps, the following list could be revealed as selected attributes:*
Case attributes

With a dynamic process instance, case attributes are very difficult to select. In this example, no case attributes are selected, all attributes are assigned to events.

Event attributes

Activity	Event attributes
Create PR	creator, type of PR, requestor, requestor department
Create PO	creator, supplier, supplier region
Approve PO	approver, function of approver, level of approval
Reject PO	rejecter, function of rejecter, number of rejection, reason of rejection
Forward PO	resource, department forwarded to, person forwarded to, notes
Enter GR	resource, number of units, type of unit, net value
Book invoice	resource, type of invoice, net amount, VAT regime
Enter Invoice	resource, department, reference number to scan
Supplier Creates Invoice	resource, supplier
Pay Invoice	resource, bank account, amount
Add PO line	resource
Increase value PO line	resource, department, absolute value change, relative value change (to calculate)

In a real life example, each identified attribute is accompanied by the field name that captures the information of that attribute.

Step 7: Consider attributes to incorporate in activities

The last step contains a double-check of the listed attributes, whether there might be some attributes that would add more value if the extra information was incorporated in the activity. For example, if a certain attribute only has a limited number n of possible values, it might be beneficial to create n variants of that activity, including the information of the attribute. In case the attribute of interest is an event attribute, only the related event will be multiplied. In case the attribute of interest is a case attribute, all activities can be multiplied with the possible values of that attribute, hereby visualizing different paths for different types of cases in one process model. The main consequence of incorporating information in activities, lies in a different visual level of granularity of the process. It is important to realise that the analysis opportunities as such are not impacted by this step. It only results in possible different process maps in the next phase.

Take for example an activity like 'scan document' with an attribute that captures the type of document (invoice versus order). It might be of interest to recplace the activity 'scan document' by the activities 'scan invoice' and 'scan proposal', immediately taking the extra attribute information into account. Whether this would be of interest to the analysis, is up to the stakeholders to decide, taking into account the goal and key questions this would help to answer or not.

Running example. *In our example, the shareholders could have selected the activity 'Book invoice' to multiply into n different activities, where n stems from the different values of the attribute 'type of invoice'. However, it was decided that this was not yielding extra information in the light of the formulated goal.*

4 Conclusion

This report presents a procedure to build event logs for process mining purposes in a business context. Structuring the data that stems from information systems in a suitable event log, requires a thorough understanding of the underlying data structure and the different log structure options. Along the process of building the architecture of such an event log, different decisions are taken, which all have their impact on analysis possibilities of the process (the event log) afterwards. There are no right or wrong decisions, there are simply consequences that one might want to realise before the analysis phase.

The presented procedure is a written reflection of the author's own experience of building event logs. This experience of building event logs for process mining purposes dates back to 2007 and covers different business and process contexts. The presented procedure is a structured approach that aims to help process mining novices to consciously build event logs. A carefully built log assures the process mining project to both manage expectations and deliver what is expected. It further maximizes the process mining opportunities in the analysis phase.

Author

Mieke Jans is assistant professor at the Business Informatics research group of Hasselt University, Belgium. The topic of her PhD thesis was applying data mining and process mining for internal fraud risk reduction. For her research, she collaborated with a large European financial institution to apply process mining techniques on the procurement process. Building further on this experience, her academic research is mostly positioned on the nexus of process mining and auditing and resulted in international scientific publications (for example in *The Accounting Review* and *International Journal of Accounting Information Systems*). After receiving her Phd in 2009, Mieke Jans started working as a manager (and later as senior manager) at Deloitte Belgium, Enterprise Risk Services. Part of her responsibilities was to set up a process mining service line, both for financial and internal audit, and for operation excellence purposes. In September 2014, returned to academia and took up her present function at Hasselt University. Her research interests are the art of building event logs and the application of process mining techniques in accounting and auditing contexts.

universiteit hasselt

KNOWLEDGE IN ACTION

Campus Hasselt | Martelarenlaan 42 | BE 3500 Hasselt
Campus Diepenbeek | Agoralaan Gebouw D | BE 3590 Diepenbeek
Tel. 011 26 81 00 | www.uhasselt.be