Mining Configurable Enterprise Information Systems

M.H. Jansen-Vullers [a] W.M.P. van der Aalst [a] M. Rosemann [b]

[a] *Department of Technology Management, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands*

[b] *Faculty of Information Technology, Queensland University of Technology, 126 Margaret Street Brisbane Qld 4000, Australia*

Abstract

Process mining is the extraction of a process model from system logs. These logs have to meet minimum requirements, i.e. each event should refer to a case and a task. Many system logs do not meet these requirements, and therefore it is not possible to use process mining for process optimization or delta analysis. This paper shows an alternative process mining procedure for logs containing data on the frequency that process steps have been executed. To be able to mine such logs we apply Configurable Event-driven Process Chains (C-EPCs). If a C-EPC is available, we propose a method to mine the process. If only a classical reference model (i.e. an EPC) is available, we propose a method to first derive the C-EPC through mining and then analyse the process. This approach enables us to do process mining in the context of ERP systems such as the SAP solutions.

Key words: Data and Process Re-engineering, Data Mining, Knowledge Discovery, Reference Models, Enterprise Information Systems

1 Introduction

Many organizations are confronted with problems resulting from badly implemented enterprise information systems like workflow management systems, case-handling systems or Enterprise Resource Planning Systems. They observe that the system frustrates the normal way of handling processes. Maybe the system is not able to support the process as it should be carried out, or maybe the business process as

Email addresses: `m.h.jansen-vullers@tm.tue.nl` (M.H. Jansen-Vullers), `w.m.p.v.d.aalst@tm.tue.nl` (W.M.P. van der Aalst), `m.rosemann@qut.edu.au` (M. Rosemann).

implemented in the enterprise information system is not known or accepted by the employees [34]. As a result, the actual business process may diverge from the processes as implemented in the enterprise information system. To improve such a situation, it would be helpful to know the actual steps when carrying out a particular process. Process mining can do this as the actual process is taken as a starting point [6].

Other areas where process mining might contribute are reconfiguration or upgrades of configurable systems. When implementing a system, configuration enables an organization to map functionality of the system onto the business processes. However, when an enterprise information system is in use for some time, typically quite a number of updates and extensions have been added to the system. Documentation, if present at all, is outdated. Furthermore, the business process might have been changed as well. When mining the actual processes from the old system, the result can be used to implement the new system, without requiring too much effort from scarce resources [32]. Apart from finding the actual process or making a delta analysis, process mining allows for performance analysis by explicitly defining the executed processes.

Process mining is well-described in the scientific literature, see e.g. [6,8,11,17–19,28,29,40–42,46–48]. The goal of process mining is to extract information about processes from transaction logs [6]. Process mining requires logs from enterprise information systems such as workflow management systems, case-handling systems and Enterprise Resource Planning Systems. These logs may record several attributes, e.g. the case that is being handled, the task and event types, the user that carried out the task etc. It is not necessary that all information is available in the log, but the reference to both cases and tasks is required.

We first describe a number of case studies that we carried out in the area of process mining (Section 1.1), which leads us to the research statement of this paper (Section 1.2).

1.1 Case studies

We have successfully applied our mining techniques in several organizations. In this section, we briefly show some results for these organizations.

The first application is based on the log of a workflow system, supporting the processes of a Dutch governmental institution responsible for fine-collection [2]. [1] A case (process instance) is a fine that has to be paid. There may be more fines related

[1] The name of the organization is not given for reasons of confidentiality. We want to thank L. Maruster, R. Dorenbos, H.J. de Vries, H. Reijers, and A. in 't Veld for their valuable support.

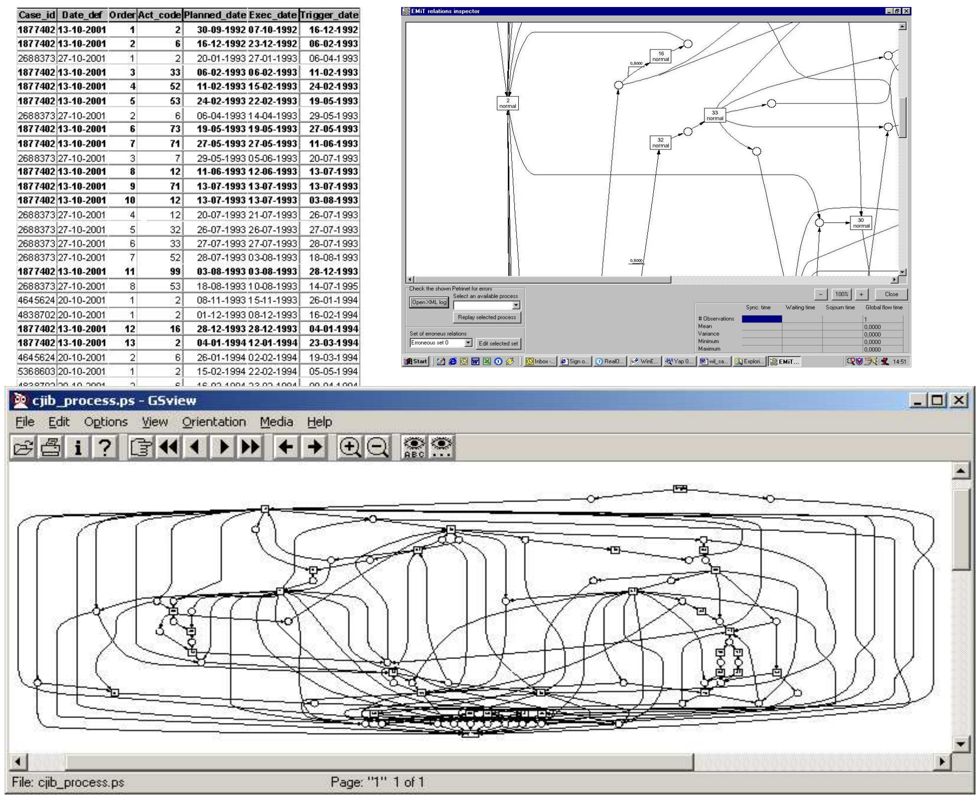

Case_id	Date_def	Order	Act_code	Planned_date	Exec_date	Trigger_date
1877402	**13-10-2001**	**1**	**2**	**30-09-1992**	**07-10-1992**	**16-12-1992**
1877402	**13-10-2001**	**2**	**6**	**16-12-1992**	**23-12-1992**	**06-02-1993**
2688373	27-10-2001	1	2	20-01-1993	27-01-1993	06-04-1993
1877402	**13-10-2001**	**3**	**33**	**06-02-1993**	**06-02-1993**	**11-02-1993**
1877402	**13-10-2001**	**4**	**52**	**11-02-1993**	**15-02-1993**	**24-02-1993**
1877402	**13-10-2001**	**5**	**53**	**24-02-1993**	**22-02-1993**	**19-05-1993**
2688373	27-10-2001	2	6	06-04-1993	14-04-1993	29-05-1993
1877402	**13-10-2001**	**6**	**73**	**19-05-1993**	**19-05-1993**	**27-05-1993**
1877402	**13-10-2001**	**7**	**71**	**27-05-1993**	**27-05-1993**	**11-06-1993**
2688373	27-10-2001	3	7	29-05-1993	05-06-1993	20-07-1993
1877402	**13-10-2001**	**8**	**12**	**11-06-1993**	**12-06-1993**	**13-07-1993**
1877402	**13-10-2001**	**9**	**71**	**13-07-1993**	**13-07-1993**	**13-07-1993**
1877402	**13-10-2001**	**10**	**12**	**13-07-1993**	**13-07-1993**	**03-08-1993**
2688373	27-10-2001	4	12	20-07-1993	21-07-1993	26-07-1993
2688373	27-10-2001	5	32	26-07-1993	26-07-1993	27-07-1993
2688373	27-10-2001	6	33	27-07-1993	27-07-1993	28-07-1993
2688373	27-10-2001	7	52	28-07-1993	03-08-1993	18-08-1993
1877402	**13-10-2001**	**11**	**99**	**03-08-1993**	**03-08-1993**	**28-12-1993**
2688373	27-10-2001	8	53	18-08-1993	10-08-1993	14-07-1995
4645624	20-10-2001	1	2	08-11-1993	15-11-1993	26-01-1994
4838702	20-10-2001	1	2	01-12-1993	08-12-1993	16-02-1994
1877402	**13-10-2001**	**12**	**16**	**28-12-1993**	**28-12-1993**	**04-01-1994**
1877402	**13-10-2001**	**13**	**2**	**04-01-1994**	**12-01-1994**	**23-03-1994**
4645624	20-10-2001	2	6	26-01-1994	02-02-1994	19-03-1994
5368603	20-10-2001	1	2	15-02-1994	22-02-1994	05-05-1994

Fig. 1. A fragment of the log of a Dutch governmental institution responsible for fine-collection and the corresponding process mining result.

with the same person. However, each fine corresponds to an independent case. This process has the particularity that as soon as the fine is paid, the process stops. In total there are 99 distinct activities which can be either manually or automatically executed. We selected the fines information for 130136 cases. We constructed the process log and we applied to this log our process discovery method that can handle noisy data [7,27].

Figure 1 (top-left) shows a fragment of the log containing 130136 cases. This log is generated by an information system specifically constructed for the Dutch governmental institution. The top-right screen shows a screenshot of our mining tool EMiT while analyzing the log. The bottom screenshot shows the whole process obtained through application of the process mining techniques. The discovered models have been inspected by the domain experts. They concluded that our discovered models were able to grasp the important aspects of the process. Moreover, the discovered models revealed aspects that are often questioned when discussing the process model. These experiences showed that process discovery can provide useful insights into the current practice of a process and highlight difference between the actual process and the prescriptive/descriptive model [27].

The second application of our mining techniques is based on a log coming from a case handling system, i.e. FLOWer (the workflow product of Pallas Athena). This

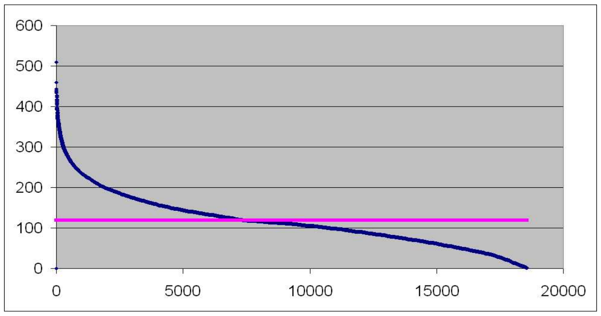


Fig. 2. Throughput time Medical cases in ‘objections on the Disablement Insurance Act’

system is used to support business processes of the Employee Insurance Implementing Body (Uitvoering Werknemersverzekeringen, or UVW) in the Netherlands. In this project we made a download of the relevant FLOWer tables and converted them to the common XML format [24]. Note that in FLOWer, the emphasis is on the data-elements in stead of on the control flow. For the conversion, this results in restrictions on the capacity of tools and systems, but also on the data that can be mined. We adapted our tools, first developed to mine workflow systems, in such a way that we were able to mine case handling systems as well.

The UWV project showed that process mining in a case handling system added value to the business processes. We analyzed several throughput times of the business process ‘objections’. Using the FLOWer downloads and the PROM framework, we extracted, amongst others, Figure 2. The throughput time in days is on the vertical axis, the number of cases on the horizontal axis. The majority of cases has a throughput time of less than 100 days, the horizontal line shows the legal deadline after which a case can be adjourned. Apart from the number of cases that is finished before/after the deadline, also the shape of the curve is of interest: before the deadline the curve is round (many cases are finished just before the deadline), after the deadline the curve is hollow (only a few cases are finished just after the deadline). It shows the type of management control, i.c. the percentage ‘in time’ is measured, but also that employees are able to influence the workflow. We made such analyzes for a number of processes, each time resulting in different aspects of the workflow.

In addition to implementations of systems based on a predefined process model, several other types of systems exist which can be classified as ‘Process Aware Information Systems (PAIS)’. An important type of such systems are Enterprise Resource Planning Systems (ERP), such as SAP solutions, SSA Global $_{\text{LN}}$ or Peoplesoft. On the one hand, these systems record a lot of data related to cases and activities, and on the other hand the additional value of obtaining a process model can have much more impact given the fact that no explicit process model is available yet.

An example of a SAP R/3 project is one for financial processes of the shared service center of a large Dutch industrial company [2] . This company aimed to discover their processes to use it as an input for configuration of their new SAP release. This project was based on the ERP-ReDoc$^{\mathrm{HRW}}$ Solution, a commercial tool for re-documentation of SAP solutions [32]. At first, we extracted data from the SAP system: frequency of transaction execution, custom adjustments and organizational functions with the help of ABAP reports. In the second step, we analyzed the data with the Reverse Business Engineer (RBE). In the third step the process structure has been generated with the help of the ASAP Question and Answer Database. In the last step, the data has been transferred from ASAP into the SAP R/3 reference model of ARIS. As a result, those parts of the reference model that are actually in use were activated. This information is used as an input for the upgrade of the financial information system, and secondly, it has been used to improve their business processes. It turned out that several 'old' transactions were frequently used, whereas several 'new' transactions were hardly used. It clearly demonstrated that the new way of working that had been implemented some time ago, has not been adopted by workers at the shop floor. The project lasted for about 4 weeks and required about 15 working days for consulting.

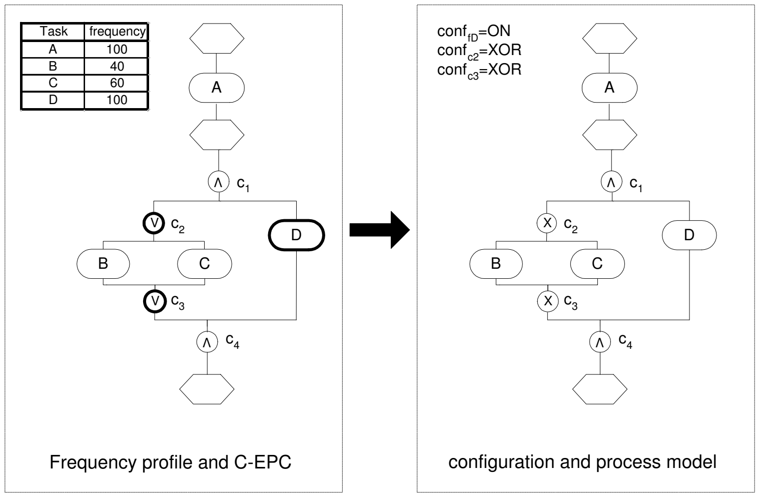
These case studies show that in many situations there is a discrepancy between the predefined process (i.e., a descriptive or prescriptive process model) and the real process (i.e., the process model obtained through mining). From the viewpoint of business alignment these discrepancies are of particular interest since they may indicate a misfit between the information system (based on an unrealistic or incorrect descriptive or prescriptive process model) and the real business process. What is lacking at this point in time, is a process mining approach for ERP systems that doesn't require customer and consultant involvement as was required for the SAP project described above, which is less dependent of the actual information system and which is more or less comparable to the effort in the mining process for workflow and case handling systems.

1.2 Research statement

To apply our current process mining techniques, we need a reference to both a case and a task. Our first mining efforts in SAP showed that this requirement is not necessarily met when dealing with ERP systems. The last case study in section 1.1 uses a tool based on the ASAP Question and Answer database, thus avoiding this problem.

This paper focuses on mining actual processes from incomplete logs, this is without both case and task reference. An example of such an incomplete log is a frequency

[2] We kindly thank Karel Bastiaanssen for sharing his valuable ideas and practical input on process mining in ERP environments. He was responsible for this project, as well as several other RBE projects in the manufacturing and process industry.



Task	frequency
A	100
B	40
C	60
D	100

Fig. 3. Process mining based on a frequency profile and an EPC (for elaboration of this example we refer to Section 4)

profile, based on the frequency a transaction has been executed, e.g. task A has been carried out 100 times and task B 40 times (see left-hand side of Figure 3). As a result, pure process mining, this is without any prior information about the process structure, is not possible. We need some additional information in the form of process models such as best practices or reference models [43,44]. We discuss how reference models (especially C-EPCs [36]) can contribute to mine actual processes. In Figure 3 this process is illustrated for a frequency profile and a reference model represented by a C-EPC.

In practice, configurable reference models are not available yet. ERP systems such as SAP R/3 are based on a set of configurable business processes. However, the reference models in which these are organized are merely a general repository in stead of a configurable process model. To bridge the gap between configurable processes represented by non-configurable process models towards a situation in which configurable process are represented by configurable process models, we make use of the configurable variant of EPCs: C-EPCs. Therefore, we include more general reference models, such as EPCs. Together with the frequency profile, a C-EPC and the actual process model can be determined. It should be taken into account that the derived C-EPC is merely based on one particular log and might differ from C-EPCs based on other logs. In Figure 3 this process is illustrated for a frequency profile and a reference model represented by an EPC.

An extension of the previous approach is to take *n* logs, resulting in a set of *n* C-EPCs. Subsequently, this set of C-EPCs can be consolidated into a more general and more accurate C-EPC. The C-EPC can be improved by rules, e.g. a particular configuration setting in one part of the model implies a particular setting in another part of the model.

Ultimately, the C-EPC can be extended by guidelines. Apart from the reference model and *n* logs, additional meta data is required, such as cost and performance data of the process, size of the company or the industry sector. Another extension can be the use of additional data in the log (time and resource of each event is known).

The paper is structured as follows. In the next section we present the background of the research: process mining, its applicability to configurable enterprise information systems and the role reference models might play mining these systems. In our approach, we make extensive use of reference models, and especially EPCs and C-EPCs. The semantics of these modelling languages should be defined unambiguously, therefore EPCs and C-EPCs are formally described in Section 3. Readers already familiar with this material could skip this section.

In Section 4 we elaborate the first research question: taking a C-EPC and a frequency profile as a starting point we derive the configuration and the actual process model. We show how to apply existing software packages to support solving our mining problems; these software packages are based on Integer Programming techniques. In Section 5 we reformulate our approach in such a way that it indeed corresponds to an Integer Programming problem. In Section 6 we take an EPC and a frequency profile as a starting point, taking into account that configurable process models are not yet available in practice yet. We derive the configuration and the actual process model, and as a side-effect we also derive the configurable process model. We show that the solution here is a variant of the solution of the first research question. This paper concludes with a section on related work, a brief summary and future work.

2 Background

2.1 Process Mining

During the last decade explicit process concepts (e.g. workflow models) have been applied in many enterprise information systems [4,14,22,26]. Workflow Management (WFM) Systems such as Staffware, IBM MQ-Series, COSA, etc. offer generic modelling and enactment capabilities for structured business processes. By making graphical process definitions, i.e. models describing the life-cycle of a particular case (process instance) in isolation, one can configure these systems to support business processes. Many other systems make use of explicit process models. Consider for example Enterprise Resource Planning systems (e.g. SAP, Peoplesoft, Baan and Oracle) or Customer Relationship Management software (Siebel), etc. As pointed out in the introduction, collection of runtime data from such Enterpise Information Systems may contribute to diagnoses, design and redesign of Enterpise Information Systems and business processes. The collection of runtime data and the analysis of these data is called process mining.

For this mining process, the data logs of an enterprise information system are processed by a mining tool, e.g. EMiT, Little Thumb, Process Miner, etc. [6]. The mining tools are generic, i.e. can be applied to all kinds of business processes and all kinds of enterprise information systems. These tools require a common XML format for storing and exchanging the logs. We have developed such a format, which is described by a document type definition (DTD). Both can be downloaded from www.processmining.org. Figure 4 shows that the XML-format connects the transactional systems such as workflow systems, ERP systems, CRM systems etc. The XML-format is then used as input for the mining tools. The goal of using a single format is to reduce the implementation effort and to promote the use of the mining concepts in multiple contexts.

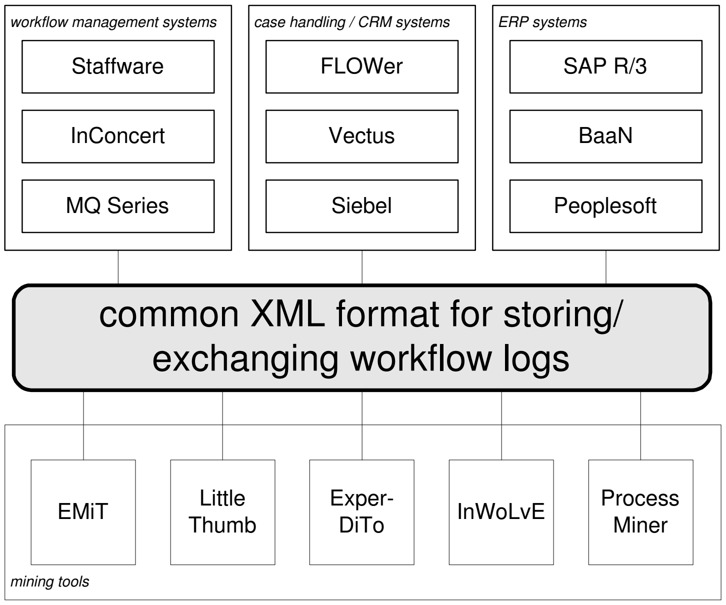


Fig. 4. The XML-format as solver/system independent medium (available from www.processmining.org)

Application of the mining tools require logs in the specified format. Furthermore, we assume that (1) each event refers to a task or well-defined step, (2) each event refers to a case or instance and (3) events are totally ordered. If this is available, mining algorithms can be used to derive models describing the underlying process. The minimum information (i.e. task and case) may be complemented by resource, time, event type etc., thus allowing for the discovery of organizational structures, social networks and performance indicators.

Figure 5 shows an example log of a process from the workflow management system Staffware. This log contains the required data case-id and task-id, and additionally the event type (the task is scheduled, processed or released), the user and a time stamp.

However, for many types of systems the logs may look differently. Although from a pure mining view such a log is incomplete, other data may be available to help mining the actual process model. In the next subsection, we elaborate two alternative approaches to show which information in configurable enterprise information systems might contribute to process mining.

2.2 Alternative approaches

When mining processes in enterprise information systems, the mining process is straightforward if business process management data is available, e.g. like in Figure 5. If this type of information is not available, other data might help. We consider two types of registrations: (1) transaction data stored in the tables and (2) process registrations primarily used and intended for the measurement of database performance.

Enterprise information systems make extensive use of the database that supports the system: each business transaction results in a system transaction that is recorded in the database. The data that is recorded are documents that are created or updated, such as purchase requisitions, purchase orders or scheduling agreements. In relation to these documents all other detailed transaction data is stored in the database. Making use of these data is promising in the context of process mining: the docu-

```
Case 3
Step description       Event               User                  yyyy/mm/dd hh:mm
---------------------------------------------------------------------------------------

                       Start               mhjansen@staffw_          2004/5/6 15:22
A                      Processed To        mhjansen@staffw_          2004/5/6 15:22
A                      Released By         mhjansen@staffw_          2004/5/6 15:22
C                      Processed To        mhjansen@staffw_          2004/5/6 15:22
C                      Released By         mhjansen@staffw_          2004/5/6 15:22
D                      Processed To        mhjansen@staffw_          2004/5/6 15:22
D                      Released By         mhjansen@staffw_          2004/5/6 15:23
                       Terminated                                    2004/5/6 15:24


Case 1
Step description       Event               User                  yyyy/mm/dd hh:mm
---------------------------------------------------------------------------------------

                       Start               mhjansen@staffw_          2004/5/6 15:22
A                      Processed To        mhjansen@staffw_          2004/5/6 15:22
A                      Released By         mhjansen@staffw_          2004/5/6 15:22
B                      Processed To        mhjansen@staffw_          2004/5/6 15:22
B                      Released By         mhjansen@staffw_          2004/5/6 15:22
D                      Processed To        mhjansen@staffw_          2004/5/6 15:22
D                      Released By         mhjansen@staffw_          2004/5/6 15:23
                       Terminated                                    2004/5/6 15:28


Case 2
Step description       Event               User                  yyyy/mm/dd hh:mm
---------------------------------------------------------------------------------------

                       Start               mhjansen@staffw_          2004/5/6 15:22
A                      Processed To        mhjansen@staffw_          2004/5/6 15:22
A                      Released By         mhjansen@staffw_          2004/5/6 15:22
B                      Processed To        mhjansen@staffw_          2004/5/6 15:22
B                      Released By         mhjansen@staffw_          2004/5/6 15:23
D                      Processed To        mhjansen@staffw_          2004/5/6 15:23
D                      Released By         mhjansen@staffw_          2004/5/6 15:24
                       Terminated                                    2004/5/6 15:28
```

Fig. 5. Example log (based on the EPC in Figure 6)

ment numbers can be considered as case-id. The derivation of the tasks that have been executed to create or update these documents is less straightforward, but may be possible. This approach is used in tools such as ARIS Process Performance Manager (PPM) when analyzing process performance of business processes.

A disadvantage of this approach is that it is time consuming and very specific. For each particular process and variant of a process, it is necessary to find the relevant tables and table fields, since the actual configuration of the system may influence the use of tables and fields. To be able to pinpoint the exact table fields that are affected by a particular process, it is necessary to examine whether these fields may differ for particular configurations. Eventually, this may even lead to a mining approach in which the actual process should be known completely before mining the process. In PPM for example, we see that application of the software first requires customization of PPM [21]. In this manual step a consultant discusses the actual process with the process owner, which is input for PPM and thus the performance analysis. The quality of the output of the performance analysis is, amongst others, dependent on the quality of the model and the configuration of PPM.

Another approach originates from the fact that enterprise information systems such as the SAP solutions log data to be able to analyze the database. For such an analysis, the processor workload caused by each transaction carried out in the system is stored. The workload analysis of a particular period summarizes which transactions have been carried out, by whom and how much computing time it consumed. Tools like the SAP Reverse Business Engineer (RBE) make use of this feature and are able to report the transaction frequencies [32,38]. We can apply this approach for process mining when deriving the frequencies of the execution of transactions. Unfortunately, there is no link between case-ids (document numbers) and these frequencies. In this context reference models may contribute. A well-known example of such reference models are Event Driven Process chains (EPC's) which have been developed in a collaborative research project conducted by SAP AG and the IDS Scheer AG [23,39] and which form the basis of the SAP reference models [12]. Also other ERP systems make use of similar reference models, e.g. Baan/SSA Global [45] or Intentia [15]. In this paper, we elaborate on this second approach. Therefore, in the next subsection we focus on the application of reference models for process mining.

2.3 Reference models

Business process management frequently uses models, e.g. for modelling the enterprise, for information system specification or end-user training [9,16,35]. These models may be descriptive or prescriptive. A typical example are reference models in the context of Enterprise Resource Planning systems such as from SAP. The SAP

Task	FP-A	FP-B	FP-C
A	100	100	140
B	100	0	100
C	0	100	40
D	100	100	140

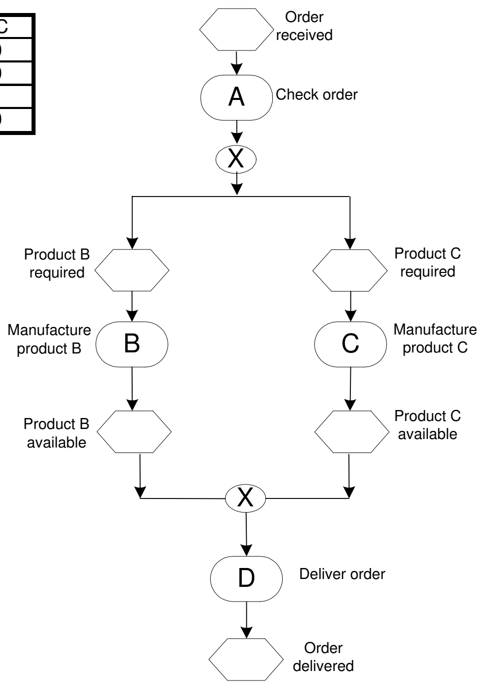


Fig. 6. Example EPC

reference models are expressed in so-called Event-driven Process Chains (EPCs). Figure 6 shows an example of an EPC for internal order handling.

It should be taken into account that reference models for ERP systems such as the SAP solutions are extremely complex because of the complexity of the business processes at one hand, and the fact that these systems are configurable at the other hand. In fact, such a reference model (represented by an EPC) is an ‘upperbound’ of process models that may possibly be implemented in a particular enterprise. Consider for example an ERP system that allows configuration of the purchasing process with respect to the quotation process. The related reference model consists of two branches (procurement with respectively without quotations). Some companies might not implement quotations and it is clear which part of the process model is relevant. Other companies however, might implement the quotation functionality, though leaving implicit whether quotations are required (one branch of the process model is applicable) or quotations may be used which may dependent on some criteria (both branches of the process model are applicable). Because such implicit decisions cannot be derived from such reference models, we call these ‘upperbound’ or ‘maximum’ reference models (for example EPC-Max). Further complexity drivers are the number of (sub)models and the interrelationships between these models.

To handle the complexity that is caused by the possibility to configure a system, a new approach has been developed that intuitively reflects the configurable nature of an ERP system. The representation of this reference modelling language is called configurable EPC (C-EPC) [36]. In this paper we will look at this specific class of reference models. In a C-EPC, there is an explicit distinction between choices made at runtime and choices made at configuration time. The following example illustrates the difference between these two types of decisions.

The left-hand side of Figure 7 shows an example of a C-EPC for transmitting purchase orders. The configurable XOR-connector is used to state that at configuration time it should be decided whether executing both the left and the right branch, i.e. purchasing with or without scheduling agreements, is allowed for a particular company. In the C-EPC this is modelled as configurable XOR-connector, which can be set XOR (i.e. choice at runtime); this is depicted in the middle part of Figure 7 (variant 1). In practice one may also find companies that do not allow purchase order processing based on scheduling agreements. In that case only one path can be selected (i.e. choice made at configuration time); this is depicted in the right-hand part of Figure 7 (variant 2). This example is taken from [36] and is based on the SAP Reference Model Purchasing, version 4.6c. For more details on the C-EPC approach we refer to Section 3.

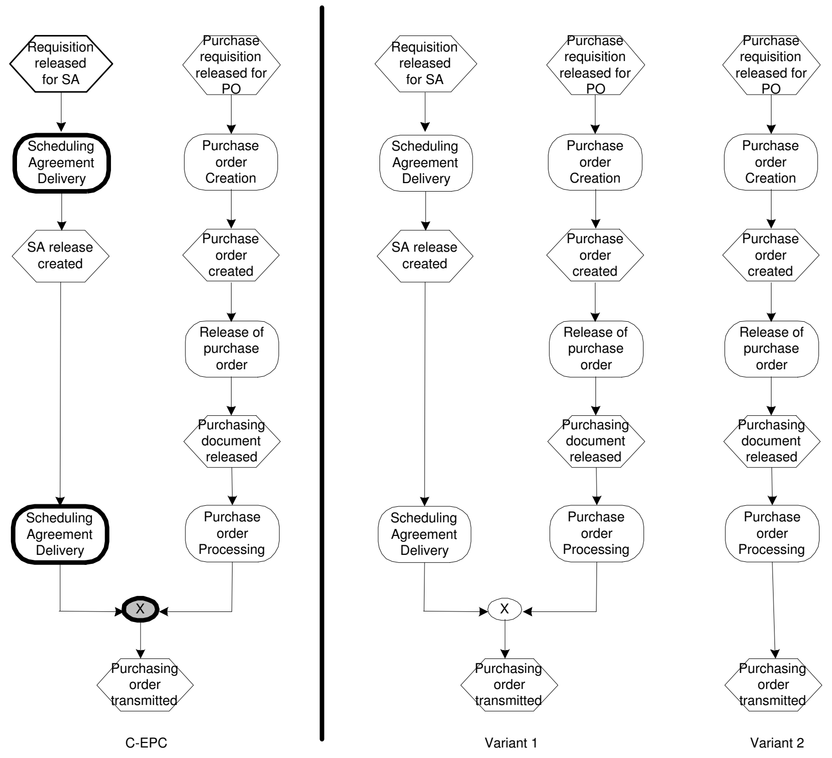

Requisition released for SA
Purchase requisition released for PO
Scheduling Agreement Delivery
Purchase order Creation
SA release created
Purchase order created
Release of purchase order
Purchasing document released
Scheduling Agreement Delivery
Purchase order Processing
X
Purchasing order transmitted
C-EPC
Variant 1
Variant 2

Fig. 7. Example of a C-EPC with XOR-join (SA-scheduling agreement, PO - purchase order)

3 Formalization of EPCs and C-EPCs

3.1 Event-driven Process Chains

An Event-driven Process Chain (EPC) consists of events, functions and connectors. However, not every diagram composed of events, functions and connectors is a correct EPC. For example, it is not allowed to connect two events to each other (cf. [23]). Unfortunately, a formal syntax for event-driven process chains is missing. In this section, we give a formal definition of an event-driven process chain. This definition is based on the restrictions described in [23] and imposed by tools such as ARIS and SAP. This way we are able to specify the requirements an event-driven process chain should satisfy.

Definition 1 (Event-driven process chain (1)) *An event-driven process chain is a five-tuple* (E, F, C, l, A)*:*

- E *is a finite set of events,*
- F *is a finite set of functions,*
- C *is a finite set of logical connectors,*
- $l \in C \to \{\wedge, XOR, \vee\}$ *is a function which maps each connector onto a connector type,*
- $A \subseteq (E \times F) \cup (F \times E) \cup (E \times C) \cup (C \times E) \cup (F \times C) \cup (C \times F) \cup (C \times C)$ *is a set of arcs.*

An event-driven process chain is composed of three types of nodes: events (E), functions (F) and connectors (C). The type of each connector is given by the function l: $l(c)$ is the type ($\wedge$, XOR, or $\vee$) of a connector $c \in C$. Relation A specifies the set of arcs connecting functions, events and connectors. Definition 1 shows that it is not allowed to have an arc connecting two functions or two events. There are many more requirements an event-driven process chain should satisfy, e.g., only connectors are allowed to branch, there is at least one start event, there is at least one final event, and there are several limitations with respect to the use of connectors. To formalize these requirements we need to define some additional concepts and introduce some notation.

Definition 2 (Directed path and elementary path) *Let* EPC *be an event-driven process chain. A directed path* p *from a node* n_1 *to a node* n_k *is sequence* $\langle n_1, n_2, \ldots, n_k \rangle$ *such that* $\langle n_i, n_{i+1} \rangle \in A$ *for* $1 \leq i \leq k-1$. p *is elementary iff for any two nodes* n_i *and* n_j *on* p, $i \neq j \Rightarrow n_i \neq n_j$.

The definition of directed path will be used to limit the set of routing constructs that may be used. It also allows for the definition of C_{EF} (the set of connectors on a path from an event to a function) and C_{FE} (the set of connectors on a path from a function to an event). C_{EF} and C_{FE} partition the set of connectors C. Based on the function l we also partition C into $C_\wedge$, $C_\vee$, and C_{XOR}. The sets C_J and C_S are used to classify connectors into join connectors and split connectors.

Definition 3 (N, $C_\wedge$, $C_\vee$, C_{XOR}, $\bullet$, C_J, C_S, C_{EF}, C_{FE}) *Let $EPC = (E, F, C, l, A)$ be an event-driven process chain.*

- *$N = E \cup F \cup C$ is the set of nodes of EPC.*
- *$C_\wedge = \{c \in C \mid l(c) = \wedge\}$*
- *$C_\vee = \{c \in C \mid l(c) = \vee\}$*
- *$C_{XOR} = \{c \in C \mid l(c) = XOR\}$*
- *For $n \in N$:*
  $\bullet n = \{m \mid (m, n) \in A\}$ is the set of input nodes, and
  $n\bullet = \{m \mid (n, m) \in A\}$ is the set of output nodes.
- *$C_J = \{c \in C \mid |\bullet c| \geq 2\}$ is the set of join connectors.*
- *$C_S = \{c \in C \mid |c \bullet| \geq 2\}$ is the set of split connectors.*
- *$C_{EF} \subseteq C$ such that $c \in C_{EF}$ if and only if there is a path $p = \langle n_1, n_2, \ldots, n_{k-1}, n_k\rangle$ such that $n_1 \in E$, $n_2, \ldots, n_{k-1} \in C$, $n_k \in F$, and $c \in \{n_2, \ldots, n_{k-1}\}$.*
- *$C_{FE} \subseteq C$ such that $c \in C_{FE}$ if and only if there is a path $p = \langle n_1, n_2, \ldots, n_{k-1}, n_k\rangle$ such that $n_1 \in F$, $n_2, \ldots, n_{k-1} \in C$, $n_k \in E$, and $c \in \{n_2, \ldots, n_{k-1}\}$.*

These notations allow for the completion of the definition of an event-driven process chain.

Definition 4 (Event-driven process chain (2)) *An event-driven process chain $EPC = (E, F, C, l, A)$ satisfies the following requirements:*

- *The sets E, F, and C are pairwise disjoint, i.e., $E \cap F = \emptyset$, $E \cap C = \emptyset$, and $F \cap C = \emptyset$.*
- *For each $e \in E$: $|\bullet e| \leq 1$ and $|e \bullet| \leq 1$.*
- *There is at least one event $e \in E$ such that $|\bullet e| = 0$ (i.e. a start event).*
- *There is at least one event $e \in E$ such that $|e \bullet| = 0$ (i.e. a final event).*
- *For each $f \in F$: $|\bullet f| = 1$ and $|f \bullet| = 1$.*
- *For each $c \in C$: $|\bullet c| \geq 1$ or $|c \bullet| \geq 1$.*
- *C_J and C_S partition C, i.e., $C_J \cap C_S = \emptyset$ and $C_J \cup C_S = C$.*
- *C_{EF} and C_{FE} partition C, i.e., $C_{EF} \cap C_{FE} = \emptyset$ and $C_{EF} \cup C_{FE} = C$.*

The first requirement states that each component has a unique identifier (name). Note that connector names are omitted in the diagram of an event-driven process chain. The other requirements correspond to restrictions on the relation A. Events cannot have multiple input arcs and there is at least one start event and one final event. Each function has exactly one input arc and one output arc. A connector c is either a join connector ($|c \bullet| = 1$ and $|\bullet c| \geq 2$) or a split connector ($|\bullet c| = 1$ and $|c \bullet| \geq 2$). The last requirement states that a connector c is either on a path from an event to a function or on a path from a function to an event. In the remainder of this paper we assume all event-driven process chains to be syntactically correct.

Note that $\{C_J, C_S\}$, $\{C_{EF}, C_{FE}\}$, and $\{C_\wedge, C_{XOR}, C_\vee\}$ partition C, i.e., C_J and C_S are disjoint and $C = C_J \cup C_S$, C_{EF} and C_{FE} are disjoint and $C = C_{EF} \cup C_{FE}$, and $C_\wedge$, C_{XOR} and $C_\vee$ are pair-wise disjoint and $C = C_\wedge \cup C_{XOR} \cup C_\vee$. In principle there are $2 \times 2 \times 3 = 12$ kinds of connectors! In [23] two of these 12 constructs are not allowed: a split connector of type C_{EF} cannot be of type XOR or $\vee$, i.e., $C_S \cap C_{EF} \cap C_{XOR} = \emptyset$ and $C_S \cap C_{EF} \cap C_\vee = \emptyset$. As a result of this restriction, there are no choices between functions sharing the same input event. A choice is resolved *after* the execution of a function, not *before*. In this paper, we will not impose this restriction.

The semantics of EPCs have often been debated in literature. Here we do not contribute to this discussion but simply refer to [1,3,13,25,33,37].

3.2 Configurable EPCs

This section introduces the notion of a *configurable event-driven process chain* (C-EPC). In a C-EPC functions and connectors can be configurable. Configurable functions may be included (ON), skipped (OFF) or conditionally skipped (OPT). Configurable connectors may be restricted at configuration time, e.g., a configurable connector of type $\vee$ may be mapped onto a $\wedge$ connector. Local configuration choices like skipping a function may be limited by configuration requirements. For example, if one configurable connector of type $\vee$ is mapped onto $\wedge$ connector, then another configurable function needs to be included. This configuration requirement may be denoted by the logical expression $c = \wedge \Rightarrow f = ON$. To guide the configuration process there is also a partial order which suggests the order of configuration. Moreover, besides the configuration requirements there may also be configuration guidelines. One can think of configuration requirements as hard constraints and interpret configuration guidelines as soft constraints.

Definition 5 (Configurable event-driven process chain) *A configurable event-driven process chain (C-EPC) is a five-tuple $(E, F, C, l, A, F^C, C^C, O^C)$:*

- *E, F, C, l, and A are as specified in Definition 1 satisfying the constraints mentioned in Definition 4,*
- *$F^C \subseteq F$ is the set of configurable functions,*
- *$C^C \subseteq C$ is the set of configurable connectors,*
- *$O^C \subseteq (F^C \cup C^C) \times (F^C \cup C^C)$ is a partial order over the configurable nodes suggesting the order of configuration,*

Configurable nodes are denoted by thick circles (for configurable connectors) or thick rectangles (for configurable functions).

A configurable function may be configured as included (ON), skipped (OFF) or conditionally skipped (OPT). Configurable connectors are mapped onto a concrete

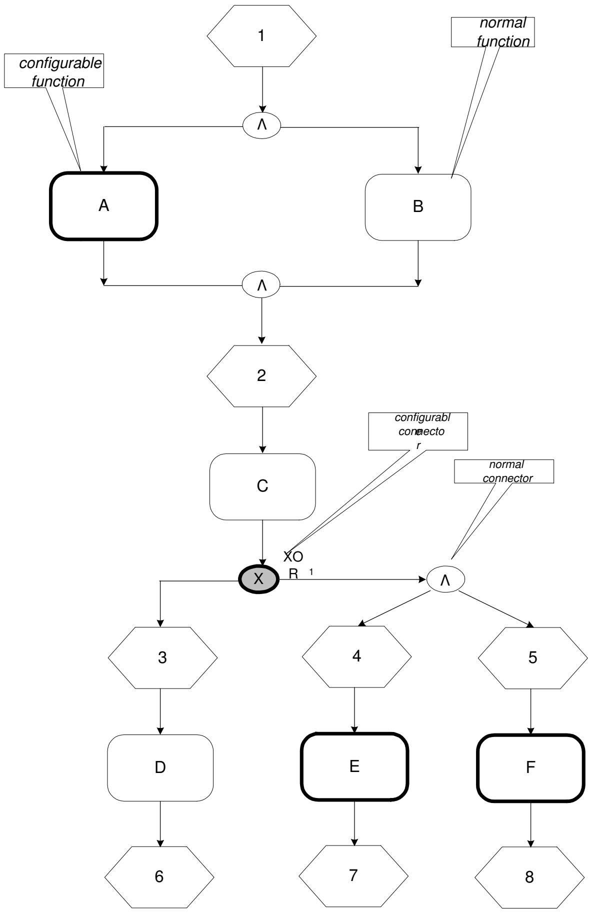


Fig. 8. Example of a configurable EPC

choice for the split or join considered. Clearly, a configurable connector of type $\wedge$ may not be mapped onto a concrete connector of type connector of type $\vee$. The concrete connector should always represent a behavior allowed by the configurable connector, i.e., the configuration process only restricts the possible execution sequences. In case of a configurable connector of type XOR or $\vee$, also only one of the options may be selected, e.g., if a split connector c has an output function f, then $c = SEQ_f$ denotes that function f is always selected.

In Figure 8 there are three configurable functions: A, E, and F. Each of these three functions can be configured as included (ON), skipped (OFF) or conditionally skipped (OPT). The other three functions cannot be configured, i.e., are always ”ON”. There are four connectors and only the XOR connector is configurable. The configurable XOR connector can be set XOR (i.e., a choice at runtime), or select one of the two paths (i.e., at configuration time the left-hand side or right-hand side is selected).

We have to be aware of the interrelationship between (configurable) functions and (configurable) connectors: In Figure 8 the configuration of function A is related to the first AND-split connector. Because of this effect, derivation of the implemented model from a C-EPC is a two stage process. First, the relevant sub-path is identified, second unnecessary branches are cleaned up. In case of the example of Figure 8, if function A is OFF, the left path would be empty (first step). The second step would be to delete the entire left path incl. the connector.

The partial order $\leq^C$ is used to specify which concrete connector type may be used for a given connector type, i.e., $x \leq^C y$ if and only if a connector of type y may be configured to x (e.g., $\wedge \leq^C \vee$ but not $\vee \leq^C \wedge$). The partial order of configurable nodes O^C is not shown in Figure 8.

Definition 6 (Partial ordering $\leq^C$, CT, CTS) *$\leq^C$ defines a partial order on CT $= \{\wedge, XOR, \vee\} \cup CTS$ where $CTS = \{SEQ_n \mid n \in E \cup F \cup C\}$. $\leq^C = \{(\wedge,\wedge),(XOR,XOR),(\vee,\vee),(XOR,\vee),(\wedge,\vee)\} \cup \{(n, XOR) \mid n \in CTS\} \cup \{(n,\vee) \mid n \in CTS\} \cup \{(n,n) \mid n \in CTS\}$.*

Note that $\leq^C = \{(n,n) \mid n \in CT\} \cup (XOR,\vee) \cup \{(n_1,n_2) \mid n_1 \in CTS \wedge n_2 \in \{XOR,\vee\}\}$.

This partial order is motivated by the fact that the configurable connector has to subsume the behavior of the concrete connector. Table 1 illustrates the configuration rules for connectors. This table only describes the overall constraints. Each row corresponds to a configurable connector type (OR^C, XOR^C, AND^C), e.g., an OR^C may be mapped onto an ***OR*** ($\vee$), ***XOR*** ($\times$), ***AND*** ($\wedge$), or ***SEQ*** (SEQ_n for some node *n*).

	OR	XOR	AND	SEQ
OR^C	X	X	X	X
XOR^C		X		X
AND^C			X	

Table 1
Constraints for the configuration of connectors

A configuration maps all configurable nodes onto concrete values like ON, OFF, and OPT for functions and $\wedge$, XOR, $\vee$, and SEQ_n for connectors.

Definition 7 (Configuration) *Let $CEPC = (E, F, C, l, A, F^C, C^C, O^C)$ be a* **C-EPC**. *$l^C \in (F^C \rightarrow \{ON, OFF, OPT\}) \cup (C^C \rightarrow CT)$ is a configuration of $CEPC$ if for each $c \in C^C$:*

- $l^C(c) \leq^C l(c)$
- *if $l^C(c) \in CTS$ and $c \in C_J$, then there exists an $n \in \bullet c$ such that $l^C(c) = SEQ_n$,*
- *if $l^C(c) \in CTS$ and $c \in C_S$, then there exists an $n \in c\bullet$ such that $l^C(c) = SEQ_n$,*

Function l^C maps configurable functions onto values like ON, OFF, and OPT, i.e., $l^C(f) \in ON, OFF, OPT$ for $f \in F^C$. Configurable connectors are mapped onto the set CT, i.e., $l^C(c) \in CT$ for $c \in C^C$. Clearly this mapping should be consistent with Table 1 and the partial order $\leq^C$. Moreover, if $l^C(c) = SEQ_n$, then n should be in the preset (for a join connector) or postset (for a split connector) of c.

Definition 8 (Valid/suitable configuration) *Let $CEPC = (E, F, C, l, A, F^C, C^C, O^C)$ be a C-EPC and l^C a configuration of $CEPC$. l^C is a valid configuration if it satisfies all configuration requirements, i.e., it satisfies all logical expressions in R^C. l^C is a suitable configuration if is valid and it satisfies all configuration guidelines, i.e., it satisfies all logical expressions in G^C.*

Definition 9 (Satisfiable) *Let $CEPC = (E, F, C, l, A, F^C, C^C, O^C)$ be a C-EPC. $CEPC$ is satisfiable if and only if there is valid configuration.*

Up to now we assumed a complete configuration, i.e., l^C is a complete function mapping each configurable node onto a concrete value. However, the configuration process may go through several stages and therefore we also add the notion of a partial configuration. One can think of a C-EPC with a partial configuration as another C-EPC.

Definition 10 (Partial configuration) *Let $CEPC = (E, F, C, l, A, F^C, C^C, O^C)$ be a C-EPC. $l^C \in (F^C \not\rightarrow \{ON, OFF, OPT\}) \cup (C^C \not\rightarrow CT)$*[3] *is a partial configuration of $CEPC$ if for each $c \in C^C \cap dom(l^C)$:*

- *$l^C(c) \leq^C l(c)$*
- *if $l^C(c) \in CTS$ and $c \in C_J$, then there exists an $n \in \bullet c$ such that $l^C(c) = SEQ_n$,*
- *if $l^C(c) \in CTS$ and $c \in C_S$, then there exists an $n \in c\bullet$ such that $l^C(c) = SEQ_n$,*

4 Mining C-EPC's: From C-EPC to EPC

Mining configurable enterprise information systems is hindered by the fact that logs from these systems do not meet the requirements of traditional process mining, e.g. of workflow systems and case handling systems. On the other hand, the advantages of process mining as shown in Section 1.1 are equally applicable to ERP systems as these are to WFM and case handling systems. An alternative approach for process mining in enterprise information systems is to use frequency profiles and reference models, in this particular case represented by C-EPCs.

Problem 1 *Consider a frequency profile FP $\in$ (F $\rightarrow$ N) and a C-EPC $CEPC = (E, F, C, l, A, F^C, C^C, O^C)$. Find configurations $l^C \in F^C \rightarrow (ON, OFF, OPT) \cup (C^C \rightarrow CT)$ and such that the frequency profile and the C-EPC match.*

[3] Note that $f \in X \not\rightarrow Y$ denotes a partial function whose domain $dom(f) \subseteq X$.

We first elaborate an example C-EPC and a number of frequency profiles and show that several alternatives for the resulting configuration and EPC may exist. Next we define a function to show whether a C-EPC, configuration and frequency profile match and we define a function to determine the best configuration out of the matching alternatives. We conclude that a generalized approach to find the best matching EPC may include Integer Programming techniques.

4.1 Searching for configurations

An EPC consists of events, functions and connectors. A C-EPC may additionally contain configurable functions and three different types of configurable connectors (AND^C, XOR^C and OR^C). In this subsection we elaborate an example consisting of a C-EPC (with two configurable OR-connectors and one configurable function as represented in Figure 9) and three frequency profiles, represented in Table 2. Note that we assume complete cases at this point in time.

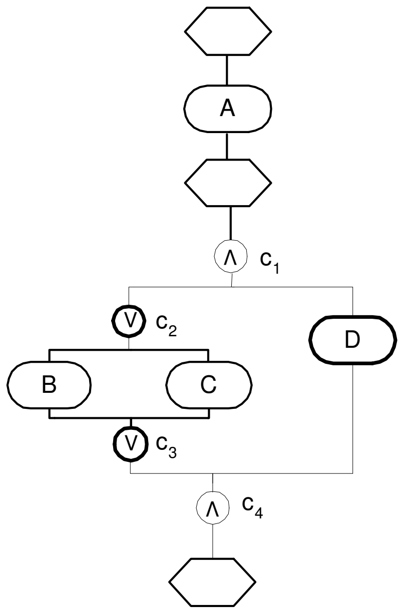


Fig. 9. Example of a configurable EPC

Frequency profile A shows that in this particular case function D and the leftmost branch of the C-EPC have not been executed. Function D may have been configured OFF or OPT: $l^C \in \{(D, OFF), (D, OPT)\}$. In the latter case, D has not been performed at runtime. The explanation why the leftmost branch has not been executed is a bit more complex:

- during configuration time the configurable connectors c_2 and c_3 have been configured SEQ_C, and at runtime function B could not be performed (EPC variant 1 in Figure 10): $l^C = ((OR_{c_2}, SEQ_C), (OR_{c_3}, SEQ_C))$.

function	frequency profile A	frequency profile B	frequency profile C
A	100	100	100
B	0	100	80
C	100	100	60
D	0	100	70

Table 2
Three frequency profiles for the C-EPC in Figure 9

- during configuration time the configurable connectors c_2 and c_3 have been configured *XOR*, and at runtime function B has not been performed (EPC variant 2 in Figure 10): $l^C = ((OR_{c_2}, XOR), (OR_{c_3}, XOR))$.
- during configuration time the configurable connectors c_2 and c_3 have been configured *OR*, and at runtime function B has not been performed (EPC variant 3 in Figure 10): $l^C = ((OR_{c_2}, OR), (OR_{c_3}, OR))$.

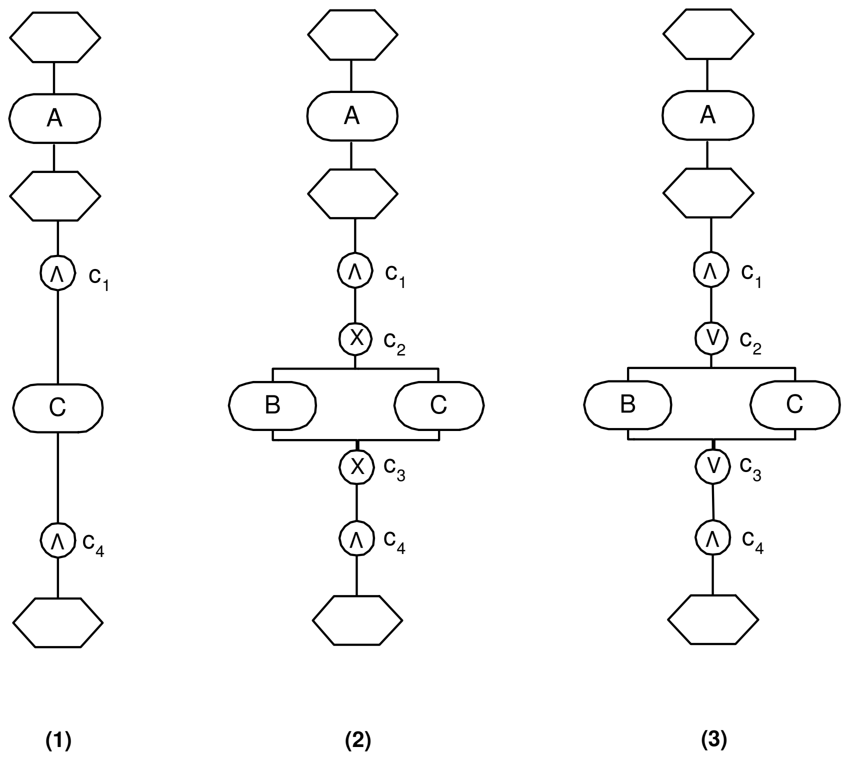


Fig. 10. Resulting EPCs based on the C-EPC in Figure 9 and frequency profiles FP-A, FP-B and FP-C

Frequency profile B shows that functions A and D have been performed the same number of times, which implies that function D should be configured *ON* or *OPT*: $l^C \in \{(D, \mathit{OFF}), (D, \mathit{OPT})\}$. In the latter case, D has been performed each time at runtime. Furthermore, functions B and C have been performed the same number of times. This can be because:

- during configuration time the configurable connectors c_2 and c_3 have been configured *AND*: $l^C = ((OR_{c_2}, AND), (OR_{c_3}, AND))$.
- during configuration time the configurable connectors c_2 and c_3 have been configured *OR* and coincidentally A, B and C have been performed the same number of times: $l^C = ((OR_{c_2}, OR), (OR_{c_3}, OR))$.

Frequency profile C finally, does not leave any freedom for configuration. Since D has been performed, but less than A, it should be configured *OPT*: $l^C = (D, OPT)$. Note that we assume that the log consists of complete cases without any noise which, in practice, is not necessarily the case. Furthermore, c_2 and c_3 can only be configured *OR*: $l^C = ((OR_{c_2}, OR), (OR_{c_3}, OR))$.

From these examples, we conclude that deriving a configuration based on a C-EPC and a frequency profile is not unambiguous. The remainder of this section is used to show when a particular configuration is allowed and if more than one configuration is allowed, which configuration fits best.

4.2 Matching

Combining a particular C-EPC and frequency profile results in a configuration and related EPC. However, in general, this process may result in a number of configurations and thus several different EPCs. In this subsection we define the function *match* to show whether a C-EPC, configuration and frequency profile do match, in the next subsection we define a function to determine the best configuration out of the matching alternatives.

A C-EPC can be considered a set of concrete EPCs; a particular EPC is determined by its configuration as defined in Definition 7. An EPC/C-EPC is a graph consisting of different nodes. Each node and each arc have a frequency. Through the frequency profile we only know the frequency of functions and not of the other nodes. Assume that the frequency of each node *n* is given by a variable x_n and let f_n be the frequency in the profile if *n* is a function. Consider the following system of equations: $\forall n \in F : x_n = f_n$ and the set of equations generated by arcs. The exact formulation of the set of equations is dependent on (i) the structure of the model, (ii) whether a function is configurable or not (and if applicable its configuration), and (iii) whether a connector is configurable or not (and if applicable its configuration). For all nodes in the model, the applicable equations should be selected from the list below:

- Related to functions:
  - An arc ending in a function has a frequency that is equal to the frequency of the arc starting from that function.

- · All arcs starting or ending in a non-configurable function or event have a frequency that equals the frequency of that function or event.
- · All arcs starting or ending in a configurable function that has been configured ON, have a frequency that equals the frequency of that function.
- · All arcs starting or ending in a configurable function that has been configured OPT, have a frequency that is greater than or equal to the frequency of that function.
- · A configurable function that has been configured OFF, has a frequency 0.

- Related to AND-nodes (connectors or configurations):
  - · An AND-node has a frequency that equals the frequency of each of the arcs starting from that AND-node.
  - · An AND-node has a frequency that is less than or equal to the frequency of each of the arcs ending in that AND-node, and equals the frequency of at least one of the arcs ending in that AND-node.

- Related to XOR-nodes (connectors or configurations):
  - · A XOR-node has a frequency that equals the sum of the frequencies of all arcs starting from that XOR-node.
  - · A XOR-node has a frequency that equals the sum of the frequencies of all arcs ending in that XOR-node.

- Related to OR-nodes (connectors or configurations)
  - · An OR-node has a frequency that is less than or equal to the sum of the frequencies of each of the arcs ending in that OR-node.
  - · An OR-node has a frequency that is greater than or equal to the frequency of each of the arcs ending in that OR-node.
  - · An OR-node has a frequency that is less than or equal to the sum of the frequencies of each of the arcs starting from that OR-node.
  - · An OR-node has a frequency that is greater than or equal to the frequency of each of the arcs starting from that OR-node.

- Related to SEQ$_x$-configurations (split)
  - · A connector that has been configured SEQ$_x$, has a frequency that is equal to the frequency of the arc starting from that connector and ending in node *x*.
  - · An arc starting from a connector that has been configured SEQ$_x$ and ending in any other node than *x* has a frequency 0.
  - · An arc starting from a connector that has been configured SEQ$_x$ and ending in any node has a frequency equal to the frequency of that connector.

- Related to SEQ$_x$-configurations (join)
  - · An arc starting from any node and ending in a connector that has been configured SEQ$_x$ has a frequency equal to the frequency of that connector.
  - · A connector that has been configured SEQ$_x$, has a frequency that is equal to the frequency of the arc starting from node *x* and ending in that connector.

· An arc ending in a connector that has been configured SEQ$_x$ and starting from any other node than *x* has a frequency 0.

We have defined now a C-EPC and a frequency profile, and we are able to define the set of equations that describe this situation. We are ready to define when a configuration matches with a C-EPC and a given frequency profile.

Definition 11 (Match) *Let CEPC =* $(E, F, C, l, A, F^C, C^C, O^C)$ *be a C-EPC,* $l^C \in (F^C \rightarrow \{ON, OFF, OPT\}) \cup (C^C \rightarrow CT)$ *a configuration of C-EPC and FP* $\in (F \rightarrow N)$ *a frequency profile of C-EPC. match:(C-EPC,* $l^C, FP) \rightarrow boolean.$ *If there is a solution to the 'system of equations', match(C-EPC,* l^C*, FP)= true; if there is no solution match(C-EPC,* l^C*, FP)= false.*

In other words, the function match is true iff FP is a possible frequency profile for the EPC that results from C-EPC and l^C. Note that we still have to formalize the system of equations, see Section 5. Consider the example in Figure 9 and frequency profile A. We mentioned three alternative configurations:

(1) $l^C = ((OR_{c_2}, SEQ_C), (OR_{c_3}, SEQ_C), (D, OFF))$
(2) $l^C = ((OR_{c_2}, XOR), (OR_{c_3}, XOR), (D, OFF))$
(3) $l^C = ((OR_{c_2}, OR), (OR_{c_3}, OR), (D, OFF))$

It is easy to verify that the configurations in Section 4.1 match, because the system of equations can be solved for all three configurations and thus match(C-EPC of Figure 9, l^C, FP-A) = true.

4.3 Objective function

In the previous subsection we have defined the function *match* to be able to decide whether a C-EPC, a configuration and a frequency profile actually match. The result of this step is a set of matching configurations. The next step is to decide which of the matching configurations fits best. To be able to do so, we define the objective function of a configuration and minimize this function to find the 'best configuration'. The basic idea is that an EPC should be as specific as possible, still meeting the requirements of the C-EPC. Furthermore, we prefer the configuration of connectors over the configuration of functions.

To define the objective function we refer to the partial ordering in Definition 6 and visualized in the left-hand part of Figure 11. The basic idea for the configuration of connectors is that the concrete connector should be selected conform the partial ordering and that the selected node is as low as possible in the ordering tree. The resulting objective function is depicted in the right-hand part of Figure 11.

The basic idea for configurable functions is that all functions should be configured

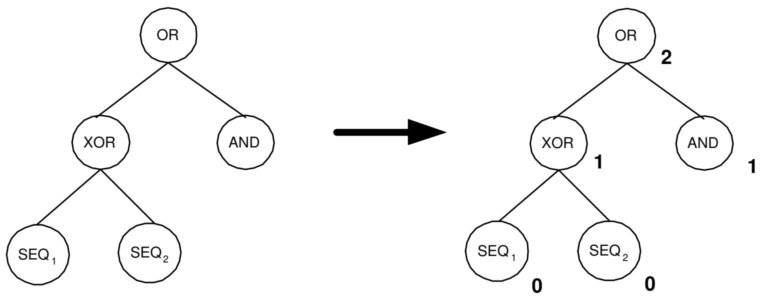


Fig. 11. Objective function for the configuration of connectors

ON, unless this is in conflict with the frequency profile. Furthermore, functions should not be configured *OPT* unless absolutely necessary, because this configuration is not discriminative at all. In Figure 12 the resulting objective function for the configuration of functions is depicted.

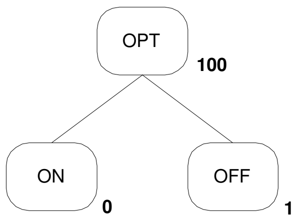


Fig. 12. Objective function for the configuration of functions

The objective function can now be formulated as: minimize the sum of the objective function due to the configuration of all configurable connectors and the objective function due to the configuration of all configurable functions.

For the example in subsection 4.1 we conclude that variant 1 (both connectors configured SEQ_C) and function D configured *OFF* is the best configuration, see Table 3.

1^C	$l^C(c_2)$	$1^C(c_3)$	$1^C(D)$		Objective function
1	SEQ_C	SEQ_C	*OFF*	$\Rightarrow$	0+0+1=1
2	*XOR*	*XOR*	*OFF*	$\Rightarrow$	1+1+1=3
3	*OR*	*OR*	*OFF*	$\Rightarrow$	2+2+1=5

Table 3
Configurations for the C-EPC in Figure 9 and frequency profile FP-A

4.4 Conclusion

In this section we have defined the function *match* to be able to decide whether a C-EPC, a configuration and a frequency profile actually match. The result of this step is a set of matching configurations. The next step is to decide which of the matching configurations fits best. To be able to do so, we defined an objective function of a configuration. Summarizing, if we want to determine an EPC based on a C-EPC and a frequency profile, we go through the following steps:

(1) find all matching configurations
(2) minimize the objective function

The approach that we followed in Section 4.2 (find all solutions of a system of equations) and Section 4.3 (minimize a particular objective function) can be performed in one step by making use of Integer Programming techniques. This has an additional advantage since standard software for solving Integer Programming problems is generally available. In the next section we reformulate the above described approach in such a way that integer programming can be applied. Subsequently we will discuss the required variables, the constraints and the objective function.

5 Formulating the Integer Programming problem

In Section 4 we elaborated process mining based on incomplete logs, i.e. based on a frequency profile. We showed that the concept of process mining is applicable, provided that a configurable reference model (an C-EPC) is available. The approach we developed is a so-called optimization approach: find the optimal solution, given a set of constraints. Many books on operations research focus on such optimization problems. In [30] for example, Murty shows how to solve transportation and production planning problems, making use of linear programming. Solving a linear programming problem requires definition of the set of decision variables, the constraints and the objective function. The constraints are represented by a system of equations, expressed in the previously defined variables. In general, this system of equations has a number of solutions, i.e. each solution assigns a value to the set of variables. Finally, the objective function selects that set of values that fulfills the objective function best.

In linear programming the variables may have any value (integer or real), however in practice real values not necessarily have a meaning. For example, when optimizing the company's profit (objective function) based on the number of production machines (one of the decision variables), the solution 'number of production machines equals 5,4' makes no sense. Production management decides either to invest in 5 machines or in 6 machines. This subclass of linear programming is called Integer Programming.

In this section we apply the Integer programming technique to support process mining based on frequency profiles. We first define the decision variables, then we elaborate the definition of the set of equations and the objective function. The section concludes with an example for a particular C-EPC and frequency profile.

Let CEPC = $(E, F, C, l, F^C, C^C, O^C, R^C, G^C)$ be a C-EPC and FP $\in$ (F $\not\rightarrow$ N) be a frequency profile. A (C-)EPC is a graph consisting of different nodes. Each node and each arc have a frequency. The frequency profile records the frequency of functions and not of the other node types [4]. Consider the following Integer Programming problem.

5.1 Variables

We consider two types of variables: variables to describe the model elements and variables to describe the configuration settings, which are our decision variables. These decision variables are related to the configuration of functions and the configuration of connectors.

All elements in the C-EPC have a frequency.

$$\forall x \in E \cup F \cup C \cup A : \mathit{freq}_x \in N \tag{1}$$

A configurable function can be configured ON, OFF or OPT.

$$\forall f \in F^C : \mathit{conf}_f \in \{ON, OFF, OPT\} \tag{2}$$

All configurable connectors can be mapped onto a concrete connector within CT, provided that the concrete connector is more specific than the configurable connector (cf. partial ordering in Definition 6). Moreover, if $l^C(c) = SEQ_n$, then *n* should be in the preset (for join connectors) or in the postset (for split connectors).

$$\begin{aligned} \forall c \in C^C : \mathit{conf}_c \in \{x \in CT | x \leq^C l(c) \wedge \qquad (3) \\ \textit{if } l^C(c) \in \textit{CTS and } c \in C_J, \textit{there exists an } n \in \bullet c \textit{ such that } x = SEQ_n \wedge \\ \textit{if } l^C(c) \in \textit{CTS and } c \in C_S, \textit{there exists an } n \in c\bullet \textit{ such that } x = SEQ_n\} \end{aligned}$$

[4] We assume that the EPCs are sound, i.e. meet the following three requirements: (i) for each case that is represented in the start event, one and only one representation exists (eventually) in the end event, (ii) when the representation of a case appears in the end event, there is no other representation of this case present in the EPC, and (iii) for each function in the EPC it is possible to move from the start event to a situation in which this function can be executed. Furthermore we assume that the frequency profiles represent complete cases, i.e. the frequency profile does not contain any noise.

We consider several groups of constraints: constraints related to functions, constraints related to concrete, non-configurable connectors (AND, OR and XOR) and to configurable nodes that have been configured (SEQ-, AND-, OR- or XOR-configurations). In this subsection, we show the constraints related to functions and we show how we derived the constraints related to XOR-connectors and XOR-configurations. A complete overview of constraints can be found in appendix A.

5.2.1 Constraints related to functions

All configurable and non-configurable functions have a frequency as recorded in the frequency profile FP.

$$\forall f \in dom(FP) : \mathit{freq}_f = FP(f) \tag{4}$$

An arc ending in a function has a frequency that is equal to the frequency of the arc starting from that function.

$$\forall f \in F\ \forall (x_1, y_1), (x_2, y_2) \in A, (y_1 = x_2 = f) : \mathit{freq}_{(x_1,y_1)} = \mathit{freq}_{(x_2,y_2)} \tag{5}$$

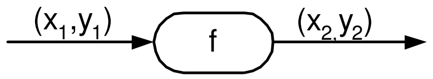


Fig. 13. Illustration of equation 5

All arcs starting or ending in a non-configurable function or event have a frequency that equals the frequency of that function or event.

$$\forall x \in (F \backslash F^C) \cup E\ \forall (y, z) \in A, x \in \{y, z\} : \mathit{freq}_{(y,z)} = \mathit{freq}_x \tag{6}$$

Fig. 14. Two illustrations of equation 6

All arcs starting or ending in a configurable function that has been configured ON, have a frequency that equals the frequency of that function.

$$\forall f \in F^C\ \forall (x, y) \in A, f \in \{x, y\} : \mathit{conf}_f = ON \Rightarrow \mathit{freq}_{(x,y)} = \mathit{freq}_f \tag{7}$$

All arcs starting or ending in a configurable function that has been configured OPT, have a frequency that is greater than or equal to the frequency of that function.

$$\forall f \in F^C \; \forall (x,y) \in A, f \in \{x,y\} : conf_f = OPT \Rightarrow freq_{(x,y)} \geq freq_f \tag{8}$$

A configurable function that has been configured OFF, has a frequency 0.

$$\forall f \in F^C : conf_f = OFF \Rightarrow freq_f = 0 \tag{9}$$

Fig. 15. Two illustrations of equations 7, 8 and 9

5.2.2 *Constraints related to concrete XOR-connectors*

A non-configurable XOR-connector has a frequency that equals the sum of the frequencies of all arcs staring from that XOR-connector.

$$\forall c \in C \backslash C^C, l(c) = XOR : \sum_{\substack{(x,y) \in A \\ x=c}} freq_{(x,y)} = freq_c \tag{10}$$

A non-configurable XOR-connector has a frequency that equals the sum of the frequencies of all arcs ending in that XOR-connector.

$$\forall c \in \{C \backslash C^C | l(c) = \boldsymbol{XOR}\} : \sum_{\substack{(x,y) \in A \\ y=c}} freq_{(x,y)} = freq_c \tag{11}$$

Fig. 16. Illustrations of equation 10 (left) and equation 11 (right)

5.2.3 *Constraints related to XOR-configurations*

Consider a configurable connector. If this connector has been configured a concrete XOR, this connector has a frequency that equals the sum of the frequencies of each of the arcs starting from that connector.

$$\forall c \in C^C : conf_c = \boldsymbol{XOR} \Rightarrow \sum_{\substack{(x,y) \in A \\ x=c}} freq_{(x,y)} = freq_c \tag{12}$$

Consider a configurable connector. If this connector has been configured a concrete XOR, this connector has a frequency that equals the sum of the frequencies of each of the arcs ending in that connector.

$$\forall c \in C^C : conf_c = XOR \Rightarrow \sum_{\substack{(x,y)\in A \\ y=c}} freq_{(x,y)} = freq_c \tag{13}$$

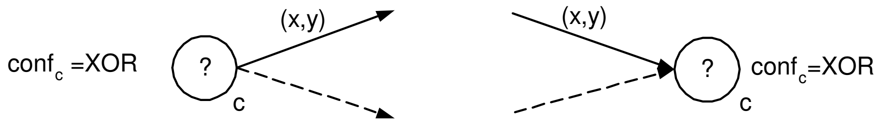


Fig. 17. Illustrations of equation 12 (left) and equation 13 (right)

5.3 Objective function

Up to now, we have formulated the system variables and the decision variables, which enabled us to formulate the constraints of our Integer Programming problem. In this subsection, we define the objective function.

As outlined in Section 4.3, the resulting EPC should be as specific as possible, still meeting the requirements of the C-EPC and the frequency profile. Furthermore, if possible, we prefer configuration of connectors over configuration of functions. The objective function can now be formulated as: minimize the sum of the objective function due to the configuration of all configurable connectors and the objective function due to the configuration of all configurable functions. Based on the ideas of Section 4.3 for the configuration of functions and connectors, the resulting objective function is formulated as follows:

Minimize:

$$\sum_{f\in F^C} \begin{cases} 100\ (conf_f = OPT) \\ 1\ (conf_f = OFF) \\ 0\ (conf_f = ON) \end{cases} + \sum_{c\in C^C} \begin{cases} 2\ (conf_c = OR) \\ 1\ (conf_c = AND) \\ 1\ (conf_c = XOR) \\ 0\ (otherwise) \end{cases} \tag{14}$$

Note that to avoid the configuration of functions to OPT as much as possible, we assigned a weight of 100. In situations with a trade-off between configurations with OR/ON combinations at one hand and configurations with AND/OPT combinations on the other hand, we enforce the first combination.

5.4 *Further steps to define the IP-problem*

In the previous subsections, we made some steps to show that finding the best configuration for a given C-EPC and frequency profile can be considered an Integer Programming problem. To be able to use Integer Programming software, we need a number of additional variables. This is necessary because we used three types of constructs that are not allowed in Integer Programming:

(1) $\exists$-constructions (for AND-connectors and AND-configurations);
(2) $\Rightarrow$-constructions (for configurable nodes);
(3) $\{$-constructions (in the objective function).

However, with the help of additional variables, each of these constructs can be reformulated into allowed IP-constructions [30]. Such a translation is rather verbose and mechanical; therefore we omit this step.

5.5 *Example*

In the previous subsections we formulated the derivation of a configuration based on a C-EPC and frequency profile in such a way that it can be considered an Integer Programming problem. In this subsection, we show this approach for a concrete example. Consider the C-EPC shown in Figure 9 and frequency profile FP(f_A) = 100, FP(f_B) = 40, FP(f_C) = 60 and $FP(f_D$ = 80). The C-EPC consists of three functions, of which f_3 is configurable, and of 4 connectors, of which c_2 and c_3 are configurable. The corresponding Integer Programming problem is defined as follows (for explanation of the variables and constraints in this example, see appendix B).

$$min \sum_{f \in F^C} \begin{cases} 100\ (conf_f = OPT) \\ 1\ (conf_f = OFF) \\ 0\ (conf_f = ON) \end{cases} + \sum_{c \in C^C} \begin{cases} 2\ (conf_c = OR) \\ 1\ (conf_c = AND) \\ 1\ (conf_c = XOR) \\ 0\ (otherwise) \end{cases}$$

$$\begin{aligned} s.t.\ & freq_{f_A} = FP(f_A) = 100 \\ & freq_{f_B} = FP(f_B) = 40 \\ & freq_{f_C} = FP(f_C) = 60 \\ & freq_{f_D} = FP(f_D) = 80 \\ & freq_{(e_0,f_A)} = freq_{(f_A,e_1)} \\ & freq_{(c_2,f_B)} = freq_{(f_B,c_3)} \\ & freq_{(c_2,f_C)} = freq_{(f_C,c_3)} \end{aligned}$$

$freq_{(c_1,f_D)} = freq_{(f_D,c_4)}$
$freq_{e_0} = freq_{(e_0,f_A)}$
$freq_{(e_0,f_A)} = freq_{f_A}$
$freq_{f_A} = freq_{(f_A,e_1)}$
$freq_{(f_A,e_1)} = freq_{e_1}$
$freq_{e_1} = freq_{(e_1,c_1)}$
$freq_{(c_2,f_B)} = freq_{f_B}$
$freq_{f_B} = freq_{(f_B,c_3)}$
$freq_{(c_2,f_C)} = freq_{f_C}$
$freq_{f_C} = freq_{(f_C,c_3)}$
$freq_{(c_4,e_2)} = freq_{e_2)}$
$conf_{f_D} = OFF \Rightarrow freq_{f_D} = 0$
$conf_{f_D} = OPT \Rightarrow freq_{f_D} \leq freq_{(c_1,f_D)}$
$conf_{f_D} = ON \Rightarrow freq_{f_D} = freq_{(c_1,f_D)}$
$freq_{(e_1,c_1)} \geq freq_{c_1}$
$freq_{(e_1,c_1)} = freq_{c_1}$
$freq_{c_1} = freq_{(c_1,c_2)}$
$freq_{c_1} = freq_{(c_1,f_D)}$
$freq_{(f_D,c_4)} \geq freq_{c_4} \wedge freq_{(c_3,c_4)} = freq_{c_4}$
$freq_{(c_3,c_4)} = freq_{c_4} \vee freq_{(f_D,c_4)} = freq_{c_4}$
$conf_{c_2} = SEQ_{f_B} \Rightarrow freq_{(c_2,f_B)} = freq_{c_2}$
$conf_{c_2} = SEQ_{f_C} \Rightarrow freq_{(c_2,f_C)} = freq_{c_2}$
$conf_{c_2} = SEQ_{f_B} \Rightarrow freq_{(c_2,f_C)} = 0$
$conf_{c_2} = SEQ_{f_C} \Rightarrow freq_{(c_2,f_B)} = 0$
$conf_{c_2} = SEQ_{f_B} \Rightarrow freq_{(c_1,c_2)} = freq_{(c_2,f_B)}$
$conf_{c_2} = SEQ_{f_C} \Rightarrow freq_{(c_1,c_2)} = freq_{(c_2,f_C)}$
$conf_{c_3} = SEQ_{f_B} \Rightarrow freq_{(f_B,c_3)} = freq_{c_3}$
$conf_{c_3} = SEQ_{f_C} \Rightarrow freq_{(f_C,c_3)} = freq_{c_3}$
$conf_{c_3} = SEQ_{f_B} \Rightarrow freq_{(f_C,c_3)} = 0$
$conf_{c_3} = SEQ_{f_C} \Rightarrow freq_{(f_B,c_3)} = 0$
$conf_{c_3} = SEQ_{f_B} \Rightarrow freq_{(f_B,c_3)} = freq_{(c_3,c_4)}$
$conf_{c_3} = SEQ_{f_C} \Rightarrow freq_{(f_C,c_3)} = freq_{(c_3,c_4)}$
$conf_{c_2} = AND \Rightarrow freq_{(c_2,f_B)} = freq_{c_2} \wedge freq_{(c_2,f_C)} \neq freq_{c_2}$
$conf_{c_3} = AND \Rightarrow freq_{(f_B,c_3)} \geq freq_{c_3} \wedge freq_{(f_C,c_3)} \geq freq_{c_3}$
$conf_{c_3} = AND \Rightarrow freq_{(f_B,c_3)} = freq_{c_3} \vee freq_{(f_C,c_3)} = freq_{c_3}$
$conf_{c_2} = XOR \Rightarrow freq_{(c_2,f_B)} + freq_{(c_2,f_C)} = freq_{c_2}$
$conf_{c_3} = XOR \Rightarrow freq_{(f_B,c_3)} + freq_{(f_C,c_3)} = freq_{c3}$
$conf_{c_2} = OR \Rightarrow freq_{(c_2,f_B)} + freq_{(c_2,f_C)} \geq freq_{c_2}$

$$conf_{c_3} = OR \Rightarrow freq_{(f_B,c_3)} + freq_{(f_C,c_2)} \geq freq_{c_3}$$
$$conf_{c_2} = OR \Rightarrow freq_{(c_2,f_B)} \leq freq_{c_2} \wedge freq_{(c_2,f_C)} \leq freq_{c_2}$$
$$conf_{c_3} = OR \Rightarrow freq_{(f_B,c_3)} \leq freq_{c_3} \wedge freq_{(f_C,c_3)} \leq freq_{c_3}$$

In this example, the solution of the system of equations is $conf_{f_D} = OPT$, $conf_{c_2} \in \{OR, XOR\}$ and $conf_{c_3} \in \{OR, XOR\}$. It is easy to verify that $conf_{f_D} = OPT$, $conf_{c_2} = XOR$ and $conf_{c_3} = XOR$ is the solution of this Integer Programming problem.

6 Mining C-EPCs: From EPC-Max to C-EPC

Process mining from incomplete logs appears to be within reach, as we have shown in the previous sections, provided that a reference model is available. In our approach, we took configurable reference models as a starting point because the system it is representing is configurable as well. From a conceptual viewpoint reference models of configurable ERP systems should indeed be configurable. In practice, these models are not configurable yet and can still be characterized as upper bound or maximal process models. The second step in our process mining research takes this traditional reference model, an EPC-Max, as a starting point. Additionally we have one particular log, i.e. a frequency profile that shows the frequency that particular process steps have been executed. This process results in a configurable EPC and a configuration that fits the log (see Section 6.1). It is evident that the resulting C-EPC and configuration depend on this only log and probably are different when based on another log or multiple logs. This step is elaborated in Section 6.2.

6.1 Deriving 'a' C-EPC

Although the application of non-configurable reference models representing configurable systems is current practice, this is not an ideal situation. In this section, we show an approach to mine a system based on an incomplete log and a non-configurable reference model. The result is, of course, the actual process model. As an intermediary step we use the notion of configurable process models, which additionally results in a useful spin-off: the configurable reference model.

Problem 2 *Consider a frequency profile FP* $\in (F \rightarrow N)$ *and an EPC EPC= (E,F,C,l,A). Find a C-EPC* $CEPC = (E, F, C, l, A, F^C, C^C, O^C, R^C, G^C)$ *and a configuration* $l^C \in (F^C \rightarrow ON, OFF, OPT) \cup (C^C \rightarrow CT)$ *such that the frequency profile, the C-EPC and the configuration match.*

6.1.1 Approach

When mining a process model from an EPC-Max, we start with an intermediary step to find the C-EPC. This may seem to be a step backwards, however, we will demonstrate that the approach developed in Section 4 is also applicable for this type of process mining. Furthermore, we can use the derived configurable reference model as well because it better represents the configurable system than the other reference model did.

We first define the term flexible EPC, this is a C-EPC that resembles the EPC completely, however, all concrete connectors are mapped onto their configurable counterparts and all functions are changed into configurable ones.

Definition 12 (flexible EPC) *Let EPC=(E,F,C,l,A). Then flex(EPC) =* $(E', F', C', l', A', F^C, C^C, O^C)$ *with* $E' = E, F' = F, C' = C, l' = l, A' = A, F^C = F, C^C = C, O^C = O$*.*

We use flexible EPCs as an intermediary result for our mining approach. Recall that this approach included two steps: the matching function and the objective function. The function *match* is used to decide whether a particular configuration, C-EPC (in this case flex(EPC)) and frequency profile match and the objective function to decide for all matching configurations which configuration fits best.

6.1.2 Example

In Figure 18, a simple EPC-Max containing a XOR connector and three frequency profiles are depicted. Frequency profile A is a log of an enterprise that only manufactured product B, frequency profile B is a log of an enterprise that only manufactured product C, whereas frequency profile C is a log of an enterprise that manufactured product B and C, but not in the same production order.

The first step is transform this EPC-Max (left-hand side of Figure 19) into a flexible EPC (right-hand side of Figure 19).

The second step is to find all matching configurations. With reference to Definition 6, the XOR^C-connector can be configured XOR, or SEQ. Since we departed from a XOR-connector in the EPC-Max, at this point we choose to stick to $l^C(c_1) = (XOR, XOR)$ and $l^C(c_2) = (XOR, XOR)$ and we come back on this in the next subsection.

With respect to the functions, we see the following alternatives:

- For frequency profile A, functions A, B and D can be non-configurable, and if configurable these may be configured *ON* or *OPT*. Function C may also be configured *OFF*.

Task	FP-A	FP-B	FP-C
A	100	100	140
B	100	0	100
C	0	100	40
D	100	100	140

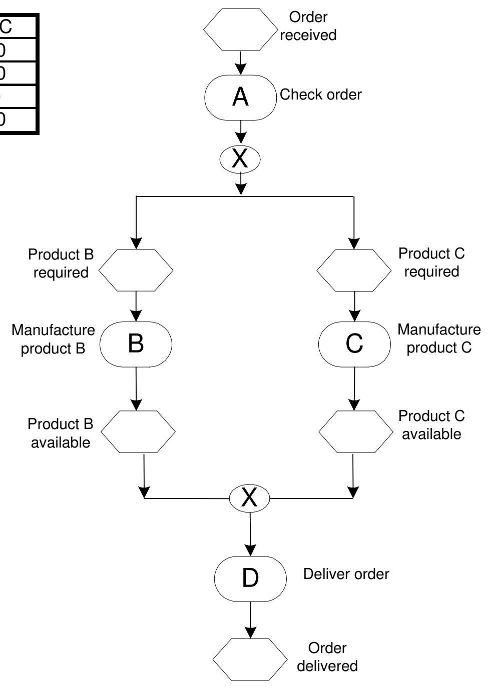


Fig. 18. Example of an EPC-Max reference model and three alternative frequency profiles

- For frequency profile B, functions A, C and D can be non-configurable, and if configurable these may be configured *ON* or *OPT*. Function B may also be configured *OFF*.
- For frequency profile C, all functions can be non-configurable, and if configurable these may be configured *ON* or *OPT*.

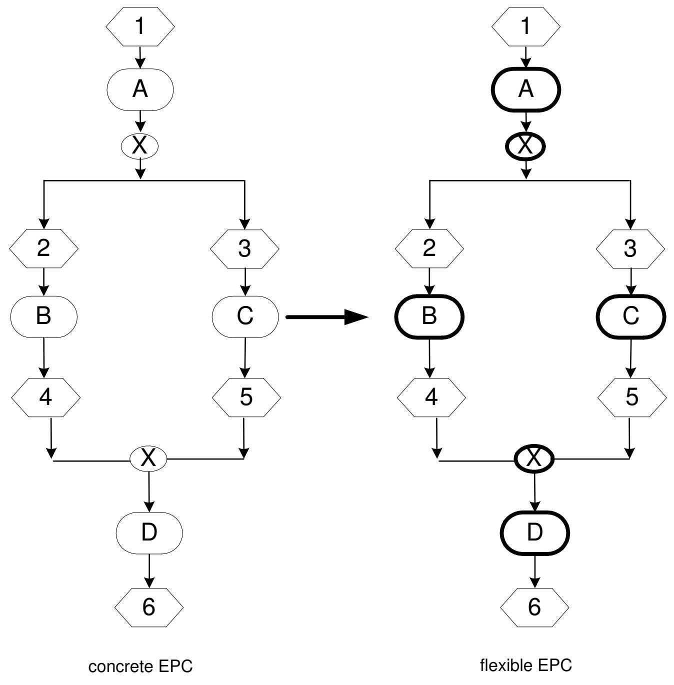


Fig. 19. From EPC to flexible C-EPC

It is easy to verify that *match*(flex(EPC),l^C,FP-A) = true for the configurations in Table 4.

l^C	$l^C(A)$	$l^C(c_1)$	$l^C(B)$	$l^C(C)$	$l^C(c_2)$	$l^C(D)$		Objective function
1	*ON*	XOR	*ON*	*ON*	XOR	*ON*	⇒	0+1+0+0+1+0=2
2	*ON*	XOR	*ON*	*ON*	XOR	*OPT*	⇒	0+1+0+0+1+100=102
3	*ON*	XOR	*ON*	*OPT*	XOR	*ON*	⇒	0+1+0+100+1+0=102
4	*ON*	XOR	*ON*	*OPT*	XOR	*OPT*	⇒	0+1+0+100+1+100=202
5	*ON*	XOR	*ON*	*OFF*	XOR	*ON*	⇒	0+1+0+1+1+0=3
6	*ON*	XOR	*ON*	*OFF*	XOR	*OPT*	⇒	0+1+0+1+1+100=103
7	*ON*	XOR	*OPT*	*ON*	XOR	*ON*	⇒	0+1+100+0+1+0=102
8	*ON*	XOR	*OPT*	*ON*	XOR	*OPT*	⇒	0+1+100+0+1+100=202
9	*ON*	XOR	*OPT*	*OPT*	XOR	*ON*	⇒	0+1+100+100+1+0=202
10	*ON*	XOR	*OPT*	*OPT*	XOR	*OPT*	⇒	0+1+100+100+1+100=302
11	*ON*	XOR	*OPT*	*OFF*	XOR	*ON*	⇒	0+1+100+1+1+0=103
12	*ON*	XOR	*OPT*	*OFF*	XOR	*OPT*	⇒	0+1+100+1+1+100=203
13	*OPT*	XOR	*ON*	*ON*	XOR	*ON*	⇒	0+1+0+0+1+0=102
14	*OPT*	XOR	*ON*	*ON*	XOR	*OPT*	⇒	0+1+0+0+1+100=202
15	*OPT*	XOR	*ON*	*OPT*	XOR	*ON*	⇒	0+1+0+100+1+0=202
16	*OPT*	XOR	*ON*	*OPT*	XOR	*OPT*	⇒	0+1+0+100+1+100=302
17	*OPT*	XOR	*ON*	*OFF*	XOR	*ON*	⇒	0+1+0+1+1+0=103
18	*OPT*	XOR	*ON*	*OFF*	XOR	*OPT*	⇒	0+1+0+1+1+100=203
19	*OPT*	XOR	*OPT*	*ON*	XOR	*ON*	⇒	0+1+100+0+1+0=202
20	*OPT*	XOR	*OPT*	*ON*	XOR	*OPT*	⇒	0+1+100+0+1+100=302
21	*OPT*	XOR	*OPT*	*OPT*	XOR	*ON*	⇒	0+1+100+100+1+0=302
22	*OPT*	XOR	*OPT*	*OPT*	XOR	*OPT*	⇒	0+1+100+100+1+100=402
23	*OPT*	XOR	*OPT*	*OFF*	XOR	*ON*	⇒	0+1+100+1+1+0=203
24	*OPT*	XOR	*OPT*	*OFF*	XOR	*OPT*	⇒	0+1+100+1+1+100=303

Table 4
Configurations for the flex(EPC) in Figure 19 and frequency profile FP-A

The third step is to calculate the objective function. The results for frequency profile A are summarized in the rightmost column of Table 4, for frequency profiles B and

C this can be done in the same way. For all frequency profiles we see that the (XOR, XOR) configuration with all functions configured *ON* fits best.

Our last step in deriving a definitive C-EPC is to consider whether configurable nodes should remain configurable; i.e. configurable functions that are always *ON* become normal functions and connectors that are mapped onto their identity connector become concrete connectors. All other nodes need to be configurable. Since only one frequency profile is available, after this step indeed no configurable connectors exist.

6.1.3 Further restriction of the C-EPC

The above outlined approach is applicable for decisions that are made at runtime, or at least for frequency profiles from which we cannot conclude that a decision is made at runtime or at configuration time. However, there is an important class of frequency profiles that shows that the decision appears to be made at configuration time. In case a frequency profile is (co-incidently?) more specific than the concrete connector in the EPC-max required (in our example in case of frequency profile A), it is possible to further restrict the flex(EPC) to determine a C-EPC, while still preserving the match condition. This is called ‘overfitting’. This step introduces two questions: (1) what is the scope of the EPC-max, the reference model we started from, and (2) what is de scope of the C-EPC, the reference model that we are deriving. The answer on the first question can very well be a broad scope, e.g., all organizations that might use a SAP solution. The answer on the second question very much depends on the scope of our modelling domain, e.g. a particular branch of industry or only all business units in our company. It is clear that this step requires additional information and cannot be performed automatically.

If we decide to allow further restrictions as described above, we have to pay special attention to (partial) C-EPCs that might include SEQ_n solutions, because SEQ_n is merely a configuration setting instead of a connector type. Basically we have three alternatives:

- The node remains configurable (XOR^C) and the configuration setting is SEQ$_n$;
- Although SEQ$_n$ is a configuration instead of a connector type, we admit concrete SEQ$_n$ connectors;
- If possible, the connector is removed and the graph structure might change.

We use alternative 1 in our approach, because alternative 2 introduces a new node type which is not necessary and alternative 3 complicates the derivation of C-EPCs based on multiple logs. Consequently, this approach may result in one configurable connector (XOR^C) which is configured SEQ_n.

6.1.4 Summary

If we want to determine a C-EPC and configuration based on an EPC-max and a frequency profile, we go through the following steps:

(1) transform the EPC-max into a flexible C-EPC
(2) find all matching configurations
(3) minimize the objective function
(4) restrict the resulting C-EPC if possible

6.2 From EPC-Max to 'the' C-EPC

The mining process can very well be based on a single log; in practice, however, also a number of logs may be available. These logs may come from the same company but covering another period of time, or these may come from other business units or even from competing businesses. The scope of the configurable reference model being developed is dependent on the origin of the set of logs. In case of multiple logs of one particular company, the accuracy of the reference model will increase, in case of multiple logs of competitors, the scope of the reference model will be enlarged.

Problem 3 *Consider a set of frequency profiles and an EPC EPC=(E,F,C,l,A). Find a C-EPC* $CEPC = (E, F, C, l, A, F^C, C^C, O^C, R^C, G^C)$ *and a configuration* $l^C \in (F^C \rightarrow ON, OFF, OPT) \cup (C^C \rightarrow CT)$ *such that the frequency profiles, the C-EPC and the configurations match.*

The third step in our research also takes the traditional reference model, an EPC-Max, as a starting point. Additionally we have a set of logs, i.e. frequency profiles. This process results in a configurable EPC and a configuration that fits all logs. It is evident that the resulting EPC and configuration depend on this set of logs. Although it is more likely to be the correct C-EPC than when based on one particular log, the result may be different when based on another class of logs. For example, the set of logs may originate from a number of business units within an enterprise, but also from a number of enterprises within a particular industry sector (e.g. utilities, automotive, etc.) or across industry sectors.

6.2.1 Searching for the C-EPC

Starting from an EPC-Max containing a (concrete) XOR connector and two logs (frequency profiles), we are constructing a C-EPC that fits both logs. Consider the EPC and frequency profiles A and B of Figure 18. In Section 6.1 we have shown that FP-A results in the C-EPC at the left part of Figure 20 and for FP-B at the middle part of Figure 20. Intuitively this results in a combined C-EPC that is shown at the right part of Figure 20.

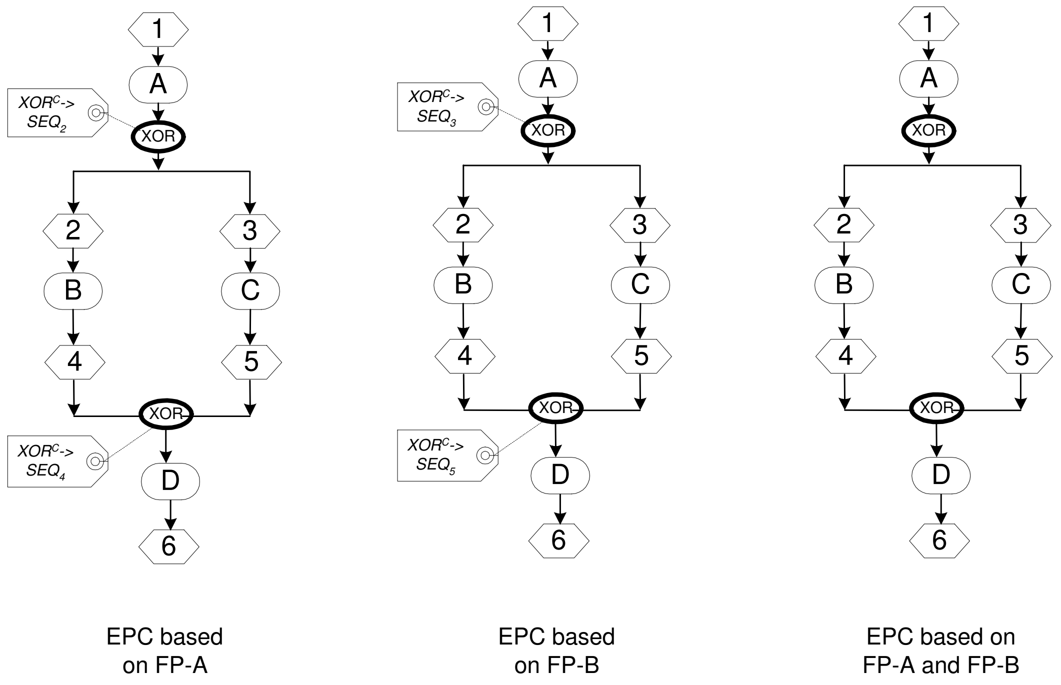


Fig. 20. Aggregation of C-EPCs

6.2.2 *Approach to derive 'the' C-EPC*

To derive a C-EPC that fits *n* logs, we use the approach for 1 log (see Section 6.1) and apply this *n* times. The result is a set of C-EPCs that can be analyzed with respect to differences in configuration settings. In case of a difference between two C-EPCs, the least common multiple is chosen. Again we start with connectors, and within the configuration of connectors we configure the functions. Note that we assume that the C-EPCs have the same graph structure.

The least common multiple for two connectors is based on the idea it should be as specific as possible on the one hand, and cover the scope of both connectors on the other hand. For example, for two OR-nodes this results in a new OR-node; a XOR-node and a SEQ-node result in a XOR^C-node, and an AND-node and a XOR-node result in a OR^C-node. Note that if we combine a XOR and a SEQ, you want to keep the information that other SEQ(s) may never be an option

The least common multiple for functions is defined the same way. If two functions have the same configurations, the common multiple is equal to this. If the configuration of two functions is different, the least common multiple is **F** with $l^C(F) \in \{ON, OPT, OFF\}$. The least common multiple for connectors and functions is depicted in Figure 21.

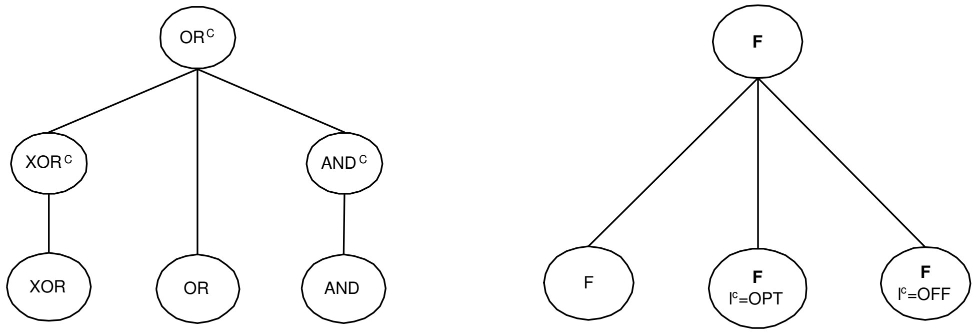


Fig. 21. Least common multiple for connectors and functions

6.2.3 Summary

Summarizing, if we want to determine a C-EPC based on an EPC and n frequency profiles FP_1-FP_n, we apply the approach for one log *n* times, and merge the results by calculating the least common multiple for the resulting C-EPCs and configurations. This results in the following approach:

(1) transform the EPC in a flexible C-EPC
(2) for each FP:
  - find all matching configurations
  - minimize the objective function
  - restrict the resulting C-EPC

(3) determine the least common multiple for the set of C-EPCs

7 Related work

The work described in this paper has been inspired by process mining from event logs [6,8,11,17–20,28,29,40–42,46–48]. The basic idea was to test the process mining techniques developed in workflow and case-handling environments in a more general enterprise information systems environment. We found that in general not all requirements for process mining were met, i.e. (i) each event refers to an activity, (ii) each event refers to a case, and (iii) events are totally ordered. Apart from process mining tools and techniques, we considered Social Network Analysis [10] and its implications for mining Social Networks from event logs [5]. This approach is based on the same requirements as process mining, and additionally requires that each event refers to a performer. For the same reasons, this approach is a good source for inspiration, but in its current form not generally applicable for mining enterprise information systems.

We have high expectations of mining enterprise information systems from a business perspective since it may contribute to (re)configuration and upgrades of such systems at one hand, and it allows for process performance analysis on the other hand.

In the area of (re)configuration and upgrading of ERP systems, a lot work has been done with respect to reference models in general [12] and related to particular ERP systems [15,45]. Furthermore, we already addressed the requirements to have more intuitive, executable and configurable business process models as described in [36,31,35]. Such reference models are very helpful to mine Enteprise Systems in case event logs as described before are not available.

In the area of performance analysis, two types of analysis should be distinguished: analysis based on descriptive models and analysis based on how processes are actually executed. Analysis based on descriptive models can be done, e.g., by making use of mathematical models or simulation models. However, any approach based on descriptive models lack sufficient feedback from real life. On the latter subject we already discussed two tools: Reverse Business Engineer [38,32] and ARIS Process Performance Manager [21]. Both tools have limitations with respect to process mining: these tools still require process knowledge prior to application of the tool, whereas our approach only requires existence of standard reference models. To the knowledge of the authors, no academic publications in this area are available yet.

8 Conclusions and further research

8.1 Conclusions

This paper proposes a solution for process mining from incomplete system logs. Existing mining approaches can be applied when an event log is composed of at least a combination of case-ids and task-ids. In this paper, we discussed how process mining can be applied if this combination of case-ids and task-ids is not available.

The approach that we followed is based on the frequency that a particular transaction has been executed, recorded in a frequency log. Additionally we make use of existing reference models, especially Event-driven Process Chains (EPCs) and the configurable couterpart (C-EPCs).

First we elaborated the idea to derive a configuration of the reference model based on a C-EPC and a frequency profile. To be able to do so, we defined the function *match* to decide whether a C-EPC, a configuration and a frequency profile match, and we defined the objective function to decide which of the matching configurations fits best.

In the approach to decide whether C-EPC, configuration and frequency profile match, the (C)-EPC is considered to be a graph consisting of different nodes; each node and each arc have a frequency. Subsequently we consider a system of equations generated by the arcs, making use of the characteristics of nodes (e.g. 'an arc ending in a function has a frequency that is equal to the frequency of the arc starting from that function' or 'a XOR-node has a frequency that equals the sum of the frequencies of all arcs staring from that XOR-node').

The function match can be applied on the (finite) set of possible configurations and results in the set of configurations that match with the particular frequency profile. The objective function yields the configuration that matches the frequency profile best.

The approach that we followed (find all solutions of a system of equations and then minimize a particular objective function) can be performed in one step by making use of Integer Programming techniques. This has an additional advantage since standard software for solving Integer Programming problems is generally available.

From a conceptual viewpoint reference models of configurable systems should be configurable. In practice, these models are not configurable yet and can be characterized as upper bound or maximal process model. Therefore, the second step in our research takes a frequency profile and a traditional reference model as a starting point.

We showed that we can re-use our approach to find the best matching configuration based on the function *match*, the objective function and Integer Programming techniques. To be able to do so, we need one step in advance, to transform the EPC into a C-EPC. This step results in a configuration, from which we can derive the process model. Additionally we should restrict the C-EPC as much as possible, to be able to reuse the C-EPC as a configurable reference model.

To improve the quality of the derived reference model, the derivation should be based on a number of frequency profiles, in stead of the one log that we used up to now. If we take this into account, we can follow the same approach for each of the frequency profiles, and then determine the least common multiple for the set of C-EPCs.

8.2 Further research

Dependent on the type of system subject to study and the way a particular system has been designed and structured, the approach how to mine the system, may differ. This is caused by the fact that different system designs may lead to different ways to store data. And relevant for process mining, this may or may not allow for the derivation of relations between these data. In this paper we elaborated mining from incomplete logs, making use of configurable reference models. In general, the following phases can be distinguished:

(1) Traditional form of mining assuming a log with case id-s rather than function frequencies. (M1)
(2) Mining based on an EPC-Max and a log allowing for the derivation of function frequencies, resulting in an EPC augmented with frequencies (i.e., functions with frequency zero are greyed out). (M2)
(3) Mining based on an EPC-Max and a log allowing for the derivation of function frequencies, resulting in a C-EPC and a configuration which fits the log. (M3)
(4) Mining based on an EPC-Max and several logs allowing for the derivation of function frequencies, resulting in a C-EPC and a configuration for each log. Note that this allows for the derivation of requirements/guidelines. (M4)
(5) Mining based on an EPC-Max and several logs allowing for the derivation of function frequencies and including meta data, resulting in a C-EPC, a configuration for each log, and requirements/guidelines taking the meta data into account. (M5)
(6) Mining based on a C-EPC and a log allowing for the derivation of function frequencies, resulting in a configuration which fits the log. (M6)

Phases M1 and M2 are current practice already. In this paper, we elaborated three other phases. Section 6.1 contributes to M3, Section 6.2 contributes to M4 and Section 4 to M6. Further research should be done to improve the C-EPC by making use of meta data and resulting in an C-EPC accompanied by guidelines and requirements. Additionally we need further research for process mining in other ERP systems or other modelling methods. This may require conversion of the developed approach to the new situation, or may require a completely new approach.

The approach discussed in this paper is working for the examples as illustrated in Section 5.5 and the appendix. For real-life examples we have to elaborate the Integer Programming problem in such a way that we can use standard Integer Programming software tools. Therefore, we are building software to convert the following constructions to the standard Integer Programming constructions:

(1) $\exists$-constructions (for AND-connectors and AND-configurations);
(2) $\Rightarrow$-constructions (for configurable nodes);
(3) $\{$-constructions (in the objective function).

A second issue that we have to tackle when implementing our approach for real-life examples is the existence of noise: differences in frequencies that cannot be explained by the number of cases alone. Such differences may occur as an effect of the measuring period, when some functions are and some functions are not yet executed for a particular case. Differences may also occur because of noise or contamination in the actual process. To be able to analyze the data in our approach we could introduce a threshold value of say 5%, which means that we consider a value range [x+5%,x-5%] when we find a frequency *x*. Further research should reveal the correct functioning of this method. To be able to do so, we first need to test our approach on real-life cases and then compare the results with the results of conventional methods such as interviews and workshops.

To conclude our program of further research, we like to research the type of logs in current enterprise information systems. In workflow management and case handling systems, we typically found event logs with events referring to both case's and tasks, and frequently also referring to performer, task type or time stamp. In these systems logging on the level of business process allows for business process management. For the type of logs described in this paper, this is not the case, and we developed a solution based on the characteristics of these systems. We made the choice to elaborate on EPCs and CEPCs because we had access to the SAP reference models which are represented by EPCs, and also because SAP is market leader for ERP software. However, in the US for example, SAP does not dominate the market and therefore we will extent our research to other ERP software. We realize that other types of system logs may have different characteristics, and also that these may change over time. We aim to analyze the underlying business processes, regardless of the type of system logs.

A Constraints for the Integer Programming problem

In Section 5 we formulated the integer programming problem enabling us to mine processes from incomplete system logs. First we defined the variables and then defined how to derive the constraints. In this appendix we provide a compete overview of all constraints: constraints related to functions, constraints related to concrete, non-configurable connectors (AND, OR and XOR) and to configurable nodes that have been configured (SEQ-, AND-, OR- or XOR-configurations).

A.1 Constraints related to functions

All configurable and non-configurable functions have a frequency as recorded in the frequency profile FP.

$$\forall f \in dom(FP) : freq_f = FP(f) \tag{A.1}$$

An arc ending in a function has a frequency that is equal to the frequency of the arc starting from that function.

$$\forall f \in F\ \forall (x_1, y_1), (x_2, y_2) \in A, (y_1 = x_2 = f) : \mathit{freq}_{(x_1,y_1)} = \mathit{freq}_{(x_2,y_2)} \quad \text{(A.2)}$$

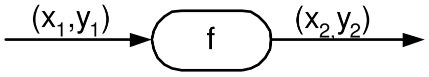


Fig. A.1. Illustration of equation A.2

All arcs starting or ending in a non-configurable function or event have a frequency that equals the frequency of that function or event.

$$\forall x \in (F \backslash F^C) \cup E\ \forall (y, z) \in A, x \in \{y, z\} : \mathit{freq}_{(y,z)} = \mathit{freq}_x \quad \text{(A.3)}$$

Fig. A.2. Two illustrations of equation A.3

All arcs starting or ending in a configurable function that has been configured ON, have a frequency that equals the frequency of that function.

$$\forall f \in F^C\ \forall (x, y) \in A, f \in \{x, y\} : \mathit{conf}_f = \mathit{ON} \Rightarrow \mathit{freq}_{(x,y)} = \mathit{freq}_f \quad \text{(A.4)}$$

All arcs starting or ending in a configurable function that has been configured OPT, have a frequency that is greater than or equal to the frequency of that function.

$$\forall f \in F^C\ \forall (x, y) \in A, f \in \{x, y\} : \mathit{conf}_f = \mathit{OPT} \Rightarrow \mathit{freq}_{(x,y)} \geq \mathit{freq}_f \quad \text{(A.5)}$$

A configurable function that has been configured OFF, has a frequency 0.

$$\forall f \in F^C : \mathit{conf}_f = \mathit{OFF} \Rightarrow \mathit{freq}_f = 0 \quad \text{(A.6)}$$

Fig. A.3. Two illustrations of equations A.4, A.5 and A.6

A.2 Constraints related to concrete AND-connectors

A non-configurable AND-connector has a frequency that equals the frequency of each of the arcs starting from that AND-connector.

$$\forall c \in C \backslash C^C, l(c) = \mathit{AND}\ \forall (x, y) \in A, x = c : \mathit{freq}_{(x,y)} = \mathit{freq}_c \quad \text{(A.7)}$$

A non-configurable AND-connector has a frequency that is less than or equal to the frequency of each of the arcs ending in that AND-connector, and equals the frequency of at least one of the arcs ending in that AND-connector.

$$\forall c \in C \backslash C^C, l(c) = AND\ \forall (x,y) \in A, y = c : \mathit{freq}_{(x,y)} \geq \mathit{freq}_c \tag{A.8}$$

$$\forall c \in C \backslash C^C, l(c) = AND\ \exists (x,y) \in A, y = c : \mathit{freq}_{(x,y)} = \mathit{freq}_c \tag{A.9}$$

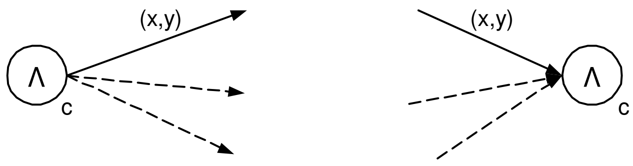


Fig. A.4. Illustrations of equation A.7 (left) and equations A.8 and A.9 (right)

A.3 Constraints related to concrete XOR-connectors

A non-configurable XOR-connector has a frequency that equals the sum of the frequencies of all arcs staring from that XOR-connector.

$$\forall c \in C \backslash C^C, l(c) = XOR : \sum_{\substack{(x,y) \in A \\ x = c}} \mathit{freq}_{(x,y)} = \mathit{freq}_c \tag{A.10}$$

A non-configurable XOR-connector has a frequency that equals the sum of the frequencies of all arcs ending in that XOR-connector.

$$\forall c \in \{C \backslash C^C | l(c) = \mathit{XOR}\} : \sum_{\substack{(x,y) \in A \\ y = c}} \mathit{freq}_{(x,y)} = \mathit{freq}_c \tag{A.11}$$

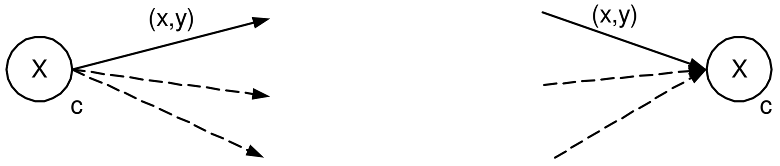


Fig. A.5. Illustrations of equation A.10 (left) and equation A.11 (right)

A.4 Constraints related to concrete OR-connectors

A non-configurable OR connector has a frequency that is less than or equal to the sum of the frequencies of each of the arcs ending in, respectively starting from that

OR-connector.

$$\forall c \in C \backslash C^C, l(c) = OR : \sum_{\substack{(x,y) \in A \\ c \in \{x,y\}}} \mathit{freq}_{(x,y)} \geq \mathit{freq}_c \tag{A.12}$$

A non-configurable OR-connector has a frequency that is greater than or equal to the frequency of each of the arcs ending in, respectively starting from that OR-connector.

$$\forall c \in C \backslash C^C, l(c) = OR \; \forall (x,y) \in A, c \in \{x,y\} : \mathit{freq}_{(x,y)} \leq \mathit{freq}_c \tag{A.13}$$

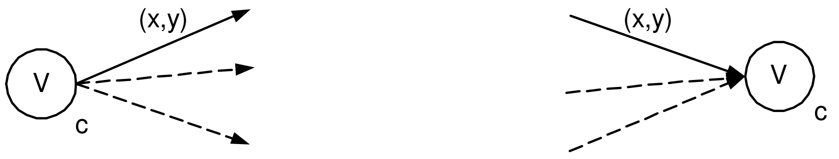


Fig. A.6. Illustrations of equations A.12 and A.13

A.5 Constraints related to SEQ$_x$-configurations

A connector that has been configured SEQ$_x$, has a frequency that is equal to the frequency of the arc starting from that connector and ending in node *x*.

$$\forall c \in C^C, x \in E \cup F \cup C, (c,x) \in A : \mathit{conf}_c = SEQ_x \Rightarrow \mathit{freq}_{(c,x)} = \mathit{freq}_c \tag{A.14}$$

An arc starting from a connector that has been configured SEQ$_x$ and ending in any other node than *x* has a frequency 0.

$$\forall c \in C^C, x \in E \cup F \cup C, (c,x) \in A \; \forall y \in c\bullet, y \neq x : \\ \mathit{conf}_c = SEQ_x \Rightarrow \mathit{freq}_{(c,y)} = 0 \tag{A.15}$$

An arc starting from any node and ending in a connector that has been configured SEQ$_x$ has a frequency equal to the frequency of that connector.

$$\forall c \in C^C, x \in E \cup F \cup C, (c,x) \in A \; \forall z \in \bullet c : \\ \mathit{conf}_c = SEQ_x \Rightarrow \mathit{freq}_{(z,c)} = \mathit{freq}_c \tag{A.16}$$

A connector that has been configured SEQ$_x$, has a frequency that is equal to the frequency of the arc starting from node *x* and ending in that connector.

$$\forall c \in C^C, x \in E \cup F \cup C, (x,c) \in A : \mathit{conf}_c = SEQ_x \Rightarrow \mathit{freq}_{(x,c)} = \mathit{freq}_c \quad \text{(A.17)}$$

An arc ending in a connector that has been configured SEQ$_x$ and starting from any other node than *x* has a frequency 0.

$$\begin{aligned}&\forall c \in C^C, x \in E \cup F \cup C, (x,c) \in A \; \forall y \in \bullet c, y \neq x : \\ &\qquad \mathit{conf}_c = SEQ_x \Rightarrow \mathit{freq}_{(y,c)} = 0\end{aligned} \quad \text{(A.18)}$$

An arc starting from a connector that has been configured SEQ$_x$ and ending in any node has a frequency equal to the frequency of that connector.

$$\begin{aligned}&\forall c \in C^C, x \in E \cup F \cup C, (x,c) \in A \; \forall z \in c\bullet : \\ &\qquad \mathit{conf}_c = SEQ_x \Rightarrow \mathit{freq}_{(c,z)} = \mathit{freq}_c\end{aligned} \quad \text{(A.19)}$$

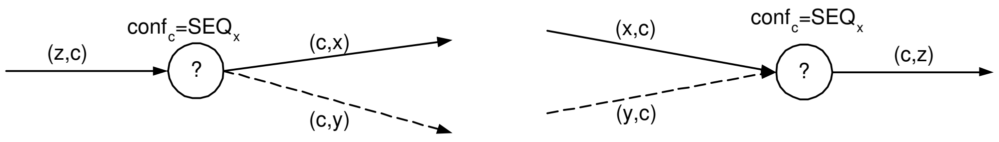


Fig. A.7. Illustrations of equations A.14, A.15 and A.16 (left) and equations A.17, A.18 and A.19 (right)

A.6 Constraints related to AND-configurations

Consider a configurable connector. If this connector has been configured a concrete AND, this connector has a frequency that equals the frequency of each of the arcs starting from that connector.

$$\forall c \in C^C, (x,y) \in A, x = c : \mathit{conf}_c = \mathit{AND} \Rightarrow \mathit{freq}_{(x,y)} = \mathit{freq}_c \quad \text{(A.20)}$$

Consider a configurable connector. If this connector has been configured a concrete AND, this connector has a frequency that less than or equal to the frequency of each of the arcs ending in that connector.

$$\forall c \in C^C, (x,y) \in A, y = c : \mathit{conf}_c = \mathit{AND} \Rightarrow \mathit{freq}_{(x,y)} \geq \mathit{freq}_c \quad \text{(A.21)}$$

$$\forall c \in C^C \; \exists (x,y) \in A, y = c : conf_c = AND \Rightarrow \mathit{freq}_{(x,y)} = \mathit{freq}_c \quad \text{(A.22)}$$

(x,y)
conf_c=AND
?
c
(x,y)
?
conf_c=AND
c

Fig. A.8. Illustrations of equation A.20 (left) and equations A.21 and A.22 (right)

A.7 Constraints related to XOR-configurations

Consider a configurable connector. If this connector has been configured a concrete XOR, this connector has a frequency that equals the sum of the frequencies of each of the arcs starting from that connector.

$$\forall c \in C^C : conf_c = XOR \Rightarrow \sum_{\substack{(x,y)\in A \\ x=c}} freq_{(x,y)} = freq_c \tag{A.23}$$

Consider a configurable connector. If this connector has been configured a concrete XOR, this connector has a frequency that equals the sum of the frequencies of each of the arcs ending in that connector.

$$\forall c \in C^C : conf_c = XOR \Rightarrow \sum_{\substack{(x,y)\in A \\ y=c}} freq_{(x,y)} = freq_c \tag{A.24}$$

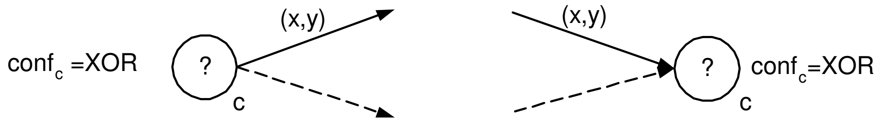


Fig. A.9. Illustrations of equation A.23 (left) and equation A.24 (right)

A.8 Constraints related to OR-configurations

Consider a configurable connector. If this connector has been configured a concrete OR, this connector has a frequency that is less than or equal to the sum of the frequencies of each of the arcs ending in , respectively starting from that connector.

$$\forall c \in C^C : conf_c = OR \Rightarrow \sum_{\substack{(x,y)\in A \\ c\in\{x,y\}}} freq_{(x,y)} \geq freq_c \tag{A.25}$$

Consider a configurable connector. If this connector has been configured a concrete OR, this connector has a frequency that is greater than or equal to the frequency of each of the arcs ending in , respectively starting from that connector.

$$\forall c \in C^C, (x,y) \in A, c \in \{x,y\} : conf_c = OR \Rightarrow freq_{(x,y)} \leq freq_c \tag{A.26}$$

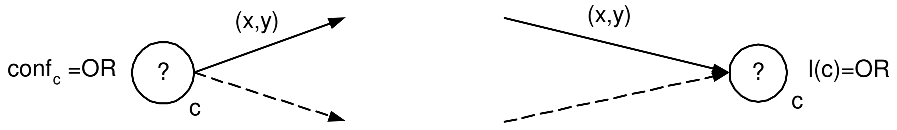


Fig. A.10. Illustrations of equations A.25 and A.26

B Example Integer Programming problem

In Section 5.5 we defined the Integer Programming problem for the C-EPC shown in Figure 9 and the frequency profile FP(f_A) = 100, FP(f_B) = 40, FP(f_C) = 60 and FP(f_D = 80). In this appendix we explain which variables and constraints have been applied, and we define the solution space and the solution. The references refer to the equations in Section 5.

B.1 Variables

B.1.1 Variables related to frequencies (equation 1)

- For events: $\mathit{freq}_{e_0}, \mathit{freq}_{e_1}, \mathit{freq}_{e_2} \in N$
- For functions: $\mathit{freq}_{f_A}, \mathit{freq}_{f_B}, \mathit{freq}_{f_C}, \mathit{freq}_{f_D} \in N$
- For connectors: $\mathit{freq}_{c_1}, \mathit{freq}_{c_2}, \mathit{freq}_{c_3}, \mathit{freq}_{c_4} \in N$
- For arcs: $\mathit{freq}_{(e_0,f_A)}, \mathit{freq}_{(f_A,e_1)}, \mathit{freq}_{(e_1,c_1)}, \mathit{freq}_{(c_1,c_2)}, \mathit{freq}_{(c_2,f_B)}, \mathit{freq}_{(c_2,f_C)}, \mathit{freq}_{(f_B,c_3)}, \mathit{freq}_{(f_C,c_3)}, \mathit{freq}_{(c_3,c_4)}, \mathit{freq}_{(c_1,f_D)}, \mathit{freq}_{(f_D,c_4)}, \mathit{freq}_{(c_4,e_2)}$

B.1.2 Variables related to configurations (equations 2 and 3)

- $\mathit{conf}_{f_D} \in \{ON, OFF, OPT\}$
- $\mathit{conf}_{c_2} \in \{OR, XOR, AND, SEQ_{f_B}, SEQ_{f_C}\}$
- $\mathit{conf}_{c_3} \in \{OR, XOR, AND, SEQ_{f_B}, SEQ_{f_C}\}$

B.2 System of equations

B.2.1 Constraints related to the frequency profile (equation 4)

- $\mathit{freq}_{f_A} = FP(f_A) = 100$
- $\mathit{freq}_{f_B} = FP(f_B) = 40$
- $\mathit{freq}_{f_C} = FP(f_C) = 60$
- $\mathit{freq}_{f_D} = FP(f_D) = 80$

B.2.2 Constraints related to non-configurable functions (equation 5)

- $\mathit{freq}_{(e_0,f_A)} = \mathit{freq}_{(f_A,e_1)}$
- $\mathit{freq}_{(c_2,f_B)} = \mathit{freq}_{(f_B,c_3)}$
- $\mathit{freq}_{(c_2,f_C)} = \mathit{freq}_{(f_C,c_3)}$
- $\mathit{freq}_{(c_1,f_D)} = \mathit{freq}_{(f_D,c_4)}$

B.2.3 Constraints related to non-configurable functions and events (equation 6)

- $\mathit{freq}_{e_0} = \mathit{freq}_{(e_0,f_A)}$
- $\mathit{freq}_{(e_0,f_A)} = \mathit{freq}_{f_A}$
- $\mathit{freq}_{f_A} = \mathit{freq}_{(f_A,e_1)}$
- $\mathit{freq}_{(f_A,e_1)} = \mathit{freq}_{e_1}$
- $\mathit{freq}_{e_1} = \mathit{freq}_{(e_1,c_1)}$
- $\mathit{freq}_{(c_2,f_B)} = \mathit{freq}_{f_B}$
- $\mathit{freq}_{f_B} = \mathit{freq}_{(f_B,c_3)}$
- $\mathit{freq}_{(c_2,f_C)} = \mathit{freq}_{f_C}$
- $\mathit{freq}_{f_C} = \mathit{freq}_{(f_C,c_3)}$
- $\mathit{freq}_{(c_4,e_2)} = \mathit{freq}_{e_2)}$

B.2.4 Constraints related to configurable functions (equations 7, 8 and 9)

- $\mathit{conf}_{f_D} = \mathit{OFF} \Rightarrow \mathit{freq}_{f_D} = 0$
- $\mathit{conf}_{f_D} = \mathit{OPT} \Rightarrow \mathit{freq}_{f_D} \leq \mathit{freq}_{(c_1,f_D)}$
- $\mathit{conf}_{f_D} = \mathit{ON} \Rightarrow \mathit{freq}_{f_D} = \mathit{freq}_{(c_1,f_D)}$

B.2.5 Constraints related to concrete AND-connectors (equations A.7, A.8 and A.9)

- $\mathit{freq}_{(e_1,c_1)} \geq \mathit{freq}_{c_1}$
- $\mathit{freq}_{(e_1,c_1)} = \mathit{freq}_{c_1}$
- $\mathit{freq}_{c_1} = \mathit{freq}_{(c_1,c_2)}$
- $\mathit{freq}_{c_1} = \mathit{freq}_{(c_1,f_D)}$
- $\mathit{freq}_{(f_D,c_4)} \geq \mathit{freq}_{c_4} \wedge \mathit{freq}_{(c_3,c_4)} = \mathit{freq}_{c_4}$
- $\mathit{freq}_{(c_3,c_4)} = \mathit{freq}_{c_4} \vee \mathit{freq}_{(f_D,c_4)} = \mathit{freq}_{c_4}$

B.2.6 Constraints related to concrete XOR-connectors (equations 10 and 11)

NA.

B.2.7 Constraints related to concrete OR-connectors (equations A.12 and A.13)

NA.

B.2.8 Constraints related to SEQ$_x$-configurations, split (equations A.14, A.15 and A.16)

- $conf_{c_2} = SEQ_{f_B} \Rightarrow freq_{(c_2,f_B)} = freq_{c_2}$
- $conf_{c_2} = SEQ_{f_C} \Rightarrow freq_{(c_2,f_C)} = freq_{c_2}$
- $conf_{c_2} = SEQ_{f_B} \Rightarrow freq_{(c_2,f_C)} = 0$
- $conf_{c_2} = SEQ_{f_C} \Rightarrow freq_{(c_2,f_B)} = 0$
- $conf_{c_2} = SEQ_{f_B} \Rightarrow freq_{(c_1,c_2)} = freq_{(c_2,f_B)}$
- $conf_{c_2} = SEQ_{f_C} \Rightarrow freq_{(c_1,c_2)} = freq_{(c_2,f_C)}$

B.2.9 Constraints related to SEQ$_x$-configurations, join (equations A.17, A.18 and A.19)

- $conf_{c_3} = SEQ_{f_B} \Rightarrow freq_{(f_B,c_3)} = freq_{c_3}$
- $conf_{c_3} = SEQ_{f_C} \Rightarrow freq_{(f_C,c_3)} = freq_{c_3}$
- $conf_{c_3} = SEQ_{f_B} \Rightarrow freq_{(f_C,c_3)} = 0$
- $conf_{c_3} = SEQ_{f_C} \Rightarrow freq_{(f_B,c_3)} = 0$
- $conf_{c_3} = SEQ_{f_B} \Rightarrow freq_{(f_B,c_3)} = freq_{(c_3,c_4)}$
- $conf_{c_3} = SEQ_{f_C} \Rightarrow freq_{(f_C,c_3)} = freq_{(c_3,c_4)}$

B.2.10 Constraints related to AND-configurations (equations A.20, A.21 and A.22)

- $conf_{c_2} = AND \Rightarrow freq_{(c_2,f_B)} = freq_{c_2} \wedge freq_{(c_2,f_C)} \neq freq_{c_2}$
- $conf_{c_3} = AND \Rightarrow freq_{(f_B,c_3)} \geq freq_{c_3} \wedge freq_{(f_C,c_3)} \geq freq_{c_3}$
- $conf_{c_3} = AND \Rightarrow freq_{(f_B,c_3)} = freq_{c_3} \vee freq_{(f_C,c_3)} = freq_{c_3}$

B.2.11 Constraints related to XOR-configurations (equations 12 and 13)

- $conf_{c_2} = XOR \Rightarrow freq_{(c_2,f_B)} + freq_{(c_2,f_C)} = freq_{c_2}$
- $conf_{c_3} = XOR \Rightarrow freq_{(f_B,c_3)} + freq_{(f_C,c_3)} = freq_{c3}$

B.2.12 Constraints related to OR-configurations (equations A.25 and A.26)

- $conf_{c_2} = OR \Rightarrow freq_{(c_2,f_B)} + freq_{(c_2,f_C)} \geq freq_{c_2}$
- $conf_{c_3} = OR \Rightarrow freq_{(f_B,c_3)} + freq_{(f_C,c_2)} \geq freq_{c_3}$
- $conf_{c_2} = OR \Rightarrow freq_{(c_2,f_B)} \leq freq_{c_2} \wedge freq_{(c_2,f_C)} \leq freq_{c_2}$
- $conf_{c_3} = OR \Rightarrow freq_{(f_B,c_3)} \leq freq_{c_3} \wedge freq_{(f_C,c_3)} \leq freq_{c_3}$

B.3 Solution of the system of equations

B.3.1 System variables

- For events:
  - $\mathit{freq}_{e_0} = 100$
  - $\mathit{freq}_{e_1} = 100$
  - $\mathit{freq}_{e_2} = 100$
- For functions:
  - $\mathit{freq}_{f_A} = 100$
  - $\mathit{freq}_{f_B} = 40$
  - $\mathit{freq}_{f_C} = 60$
  - $\mathit{freq}_{f_D} = 80$
- For connectors:
  - $\mathit{freq}_{c_1} = 100$
  - $\mathit{freq}_{c_2} = 100$
  - $\mathit{freq}_{c_3} = 100$
  - $\mathit{freq}_{c_4} = 100$
- For arcs:
  - $\mathit{freq}_{(e_0,f_0)} = 100$
  - $\mathit{freq}_{(f_0,e_1)} = 100$
  - $\mathit{freq}_{(e_1,c_1)} = 100$
  - $\mathit{freq}_{(c_1,c_2)} = 100$
  - $\mathit{freq}_{(c_2,f_1)} = 40$
  - $\mathit{freq}_{(c_2,f_2)} = 60$
  - $\mathit{freq}_{(f_1,c_3)} = 40$
  - $\mathit{freq}_{(f_2,c_3)} = 60$
  - $\mathit{freq}_{(c_3,c_4)} = 100$
  - $\mathit{freq}_{(c_1,f_3)} = 100$
  - $\mathit{freq}_{(f_3,c_4)} = 100$
  - $\mathit{freq}_{(c_4,e_2)} = 100$

B.3.2 Decision variable c_2

Connector c_2 is a configurable OR-split. To be able to find possible alternatives for the configurable OR-connector, we apply the following rule from predicate logic: A ⇒ B is equivalent with ¬ B ⇒ ¬ A.

- Checking SEQ_{f_B} : $\mathit{freq}_{(c_2,f_B)} = 40 \neq 100 \Rightarrow \mathit{conf}_{c_2} \neq SEQ_{f_B}$ (equation A.16)
- Checking SEQ_{f_C} : $\mathit{freq}_{(c_2,f_C)} = 60 \neq 100 \Rightarrow \mathit{conf}_{c_2} \neq SEQ_{f_C}$ (equation A.16)
- Checking XOR: $\mathit{freq}_{(c_2,f_B)} + \mathit{freq}_{(c_2,f_C)} = 40 + 60 = \mathit{freq}_{c_2}$ (equation 12). From

the fact that the right part is true, we cannot conclude whether $conf_{c_2}$= XOR.

- Checking AND:$freq_{(c_2,f_B)} \neq freq_{c_2} \wedge freq_{(c_2,f_C)} \neq freq_{c_2} \Rightarrow conf_{c_2} \neq AND$ (equation A.20)
- Checking OR:
  · $freq_{(c_2,f_B)} + freq_{(c_2,f_C)} = 40 + 60 \geq freq_{c_2}$ (equation A.25)
  · $freq_{(c_2,f_B)} \leq freq_{c_2} \wedge freq_{(c_2,f_C)} \leq freq_{c_2}$ (equation A.26)
  From the fact that the right part is true, we cannot conclude whether $conf_{c_2}$= OR.

From this overview we conclude $conf_{c_2} \in \{$OR, XOR$\}$

B.3.3 Decision variable c_3

Connector c_3 is a configurable OR-join. To be able to find possible alternatives for the configurable connector c_3, we apply the following rule from predicate logic: A $\Rightarrow$ B is equivalent with $\neg$ B $\Rightarrow$ $\neg$ A.

- Checking SEQ_{f_B} : $freq_{(f_B,c_3)} = 40 \neq 100 \Rightarrow conf_{c_3} \neq SEQ_{f_1}$ (equation A.17)
- Checking SEQ_{f_C} :
  $freq_{(f_C,c_3)} = 60 \neq 100 \Rightarrow conf_{c_3} \neq SEQ_{f_2}$ (equation A.17)
- Checking XOR: $freq_{(f_B,c_3)} + freq_{(f_C,c_3)} = 40 + 60 = freq_{c3}$ (equation 13). From the fact that the right part is true, we cannot conclude whether $conf_{c_3} = XOR$.
- Checking AND: $freq_{(f_B,c_3)} \neq freq_{c3} \wedge freq_{(f_C,c_3)} \neq freq_{c3} \Rightarrow conf_{c_3} \neq AND$ (equation A.22)
- Checking OR:
  · $freq_{(f_B,c_3)} + freq_{(f_C,c_3)} = 40 + 60 \geq freq_{c3}$ (equation A.25)
  · $freq_{(f_B,c_3)} \leq freq_{c3} \wedge freq_{(f_C,c_3)} \leq freq_{c3}$ (equation A.26)
  From the fact that the right part is true, we cannot conclude whether $conf_{c_3}$= OR.

From this overview we conclude $conf_{c_3} \in \{$OR, XOR$\}$

B.3.4 Decision variable f_D

To be able to find possible alternatives for the configuration of function f_3, we apply the following rule from predicate logic: A $\Rightarrow$ B is equivalent with $\neg$ B $\Rightarrow$ $\neg$ A.

- $conf_{f_D} = OFF \Rightarrow freq_{f_3} = 0$. Since $freq_{f_3} = 80 \neq 0$ this results in $conf_{f_D} \neq OFF$ (equation 9).
- $conf_{f_D} = OPT \Rightarrow freq_{f_D} \leq freq_{(c_1,f_D)}$ (equation 9). From the fact that $80 \leq 100$, the right part is true, and we cannot conclude whether $conf_{f_D} = OPT$ or not.
- $conf_{f_D} = ON \Rightarrow freq_{f_D} = freq_{(c_1,f_D)}$. Since $freq_{f_D} \neq freq_{(c_1,f_D)}$ this results in $conf_{f_D} \neq ON$ (equation 7).

From this overview we conclude $conf_{f_D} = OPT$ because $freq_{f_3} = 60 \wedge freq_{(c_1, f_D)} = 100 \Rightarrow conf_{f_3} \neq OFF \wedge conf_{f_D} \neq ON$

B.3.5 Solution

The system of equations resulted in a number of possible solutions. The objective function is used to select the solution that we consider 'best', see Section 4.3 The objective function is to minimize the sum of the values due to the configuration of all configurable functions (i.e. f_D) and the values due to the configuration of all configurable connectors (i.e. c_2 and c_3), see equation 14. In this small example it is easy to verify that the objective function yields $conf_{f_D}$ = OPT, $conf_{c_2}$ = XOR and $conf_{c_3}$ = XOR.

References

[1] W.M.P. van der Aalst. Formalization and Verification of Event-driven Process Chains. *Information and Software Technology*, 41(10):639–650, 1999.

[2] W.M.P. van der Aalst. Business Alignment: Using Process Mining as a Tool for Delta Analysis. In J. Grundspenkis and M. Kirikova, editors, *Proceedings of the 5th Workshop on Business Process Modeling, Development and Support (BPMDS'04)*, volume 2 of *Caise'04 Workshops*, pages 138–145. Riga Technical University, Latvia, 2004.

[3] W.M.P. van der Aalst, J. Desel, and E. Kindler. On the Semantics of EPCs: A Vicious Circle. In M. Nüttgens and F.J. Rump, editors, *Proceedings of the EPK 2002: Business Process Management using EPCs*, pages 71–80, Trier, Germany, November 2002. Gesellschaft für Informatik, Bonn.

[4] W.M.P. van der Aalst, J. Desel, and A. Oberweis, editors. *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2000.

[5] W.M.P. van der Aalst and M. Song. Mining Social Networks: Uncovering interaction patterns in business processes. In M. Weske, B. Pernici, and J. Desel, editors, *International Conference on Business Process Management (BPM 2004)*, Lecture Notes in Computer Science, pages ??–?? Springer-Verlag, Berlin, 2004.

[6] W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M. Weijters. Workflow Mining: A Survey of Issues and Approaches. *Data and Knowledge Engineering*, 47(2):237–267, 2003.

[7] W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering*, pages ??–??, 2003.

[8] R. Agrawal, D. Gunopulos, and F. Leymann. Mining Process Models from Workflow Logs. In *Sixth International Conference on Extending Database Technology*, pages 469–483, 1998.

[9] J. Becker, M. Kugeler, and M. Rosemann (eds.). *Process Management*. Berlin et al., 2003.

[10] R.S. Burt and M Minor. *Applied Network Analysis: A Methodological Introduction*. Sage, Newbury Park CA, 1983.

[11] J.E. Cook and A.L. Wolf. Event-Based Detection of Concurrency. In *Proceedings of the Sixth International Symposium on the Foundations of Software Engineering (FSE-6)*, pages 35–45, 1998.

[12] T. Curran and G. Keller. *SAP R/3 Business Blueprint: Understanding the Business Process Reference Model*. Upper Saddle River, 1997.

[13] J. Dehnert and P. Rittgen. Relaxed Soundness of Business Processes. In K.R. Dittrich, A. Geppert, and M.C. Norrie, editors, *Proceedings of the 13th International Conference on Advanced Information Systems Engineering (CAiSE'01)*, volume 2068 of *Lecture Notes in Computer Science*, pages 157–170. Springer-Verlag, Berlin, 2001.

[14] L. Fischer, editor. *Workflow Handbook 2001, Workflow Management Coalition*. Future Strategies, Lighthouse Point, Florida, 2001.

[15] T. Forsberg and J. Vikstroem G. Roenne. Process modeling in erp projects - a discussion of potential benefits. Technical report.

[16] J. A. Gulla and T. Brasethvik. On the challenges of business modeling in large scale reengineering projects. In *Proceedings of the 4th International Conference on Requirements Engineering*, Schaumburg, Ill.

[17] J. Herbst. Dealing with Concurrency in Workflow Induction. In U. Baake, R. Zobel, and M. Al-Akaidi, editors, *European Concurrent Engineering Conference*. SCS Europe, 2000.

[18] J. Herbst. *Ein induktiver Ansatz zur Akquisition und Adaption von Workflow-Modellen*. PhD thesis, Universität Ulm, November 2001.

[19] J. Herbst and D. Karagiannis. An Inductive Approach to the Acquisition and Adaptation of Workflow Models. In M. Ibrahim and B. Drabble, editors, *Proceedings of the IJCAI'99 Workshop on Intelligent Workflow and Process Management: The New Frontier for AI in Business*, pages 52–57, Stockholm, Sweden, August 1999.

[20] J. Herbst and D. Karagiannis. Integrating Machine Learning and Workflow Management to Support Acquisition and Adaptation of Workflow Models. *International Journal of Intelligent Systems in Accounting, Finance and Management*, 9:67–92, 2000.

[21] IDS Scheer. *Measure, Analyse and Optimise your Business Process Performance! - ARIS Process Performance Managemr (ARIS PPM) whitepaper*, 2003.

[22] S. Jablonski and C. Bussler. *Workflow Management: Modeling Concepts, Architecture, and Implementation*. International Thomson Computer Press, London, UK, 1996.

[23] G. Keller, M. Nüttgens, and A.W. Scheer. Semantische Processmodellierung auf der Grundlage Ereignisgesteuerter Processketten (EPK). Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 89 (in German), University of Saarland, Saarbrücken, 1992.

[24] W. Kunst. Proefboringen voor Process Mining - Het toepassen van Process Mining in de praktijk van het UWV (in Dutch). Master's thesis, Eindhoven University of Technology, Eindhoven, 2004.

[25] P. Langner, C. Schneider, and J. Wehler. Petri Net Based Certification of Event driven Process Chains. In J. Desel and M. Silva, editors, *Application and Theory of Petri Nets 1998*, volume 1420 of *Lecture Notes in Computer Science*, pages 286–305. Springer-Verlag, Berlin, 1998.

[26] F. Leymann and D. Roller. *Production Workflow: Concepts and Techniques*. Prentice-Hall PTR, Upper Saddle River, New Jersey, USA, 1999.

[27] L. Maruster. *A machine learning approach to understand business processes* . PhD thesis, Eindhoven University of Technology, Eindhoven, 2003.

[28] L. Maruster, A.J.M.M. Weijters, W.M.P. van der Aalst, and A. van den Bosch. Process Mining: Discovering Direct Successors in Process Logs. In *Proceedings of the 5th International Conference on Discovery Science (Discovery Science 2002)*, volume 2534 of *Lecture Notes in Artificial Intelligence*, pages 364–373. Springer-Verlag, Berlin, 2002.

[29] M.K. Maxeiner, K. Küspert, and F. Leymann. Data Mining von Workflow-Protokollen zur teilautomatisierten Konstruktion von Prozemodellen. In *Proceedings of Datenbanksysteme in Büro, Technik und Wissenschaft*, pages 75–84. Informatik Aktuell Springer, Berlin, Germany, 2001.

[30] K.G. Murty. *Operations research: deterministic optimization models*. Englewood Cliffs: Prentice Hall, 1995.

[31] P. Fettke and P. Loos. Classification of reference models - a methodology and its application. *Information Systems and e-Business Management*, 1(1):35–53, 2003.

[32] Ch. Reiter. *SAP ExpertenReport - Modellbasierte Analyse und Redokumentation von SAP Enterprise Solutions*. WCM Online (http://www.newmediasales.com/), 2003.

[33] P. Rittgen. Modified EPCs and their Formal Semantics. Technical report 99/19, University of Koblenz-Landau, Koblenz, Germany, 1999.

[34] C. Rolland and N. Prakash. Bridging the gap between organisational needs and erp functionality. *Requirements Engineering*, 5:180, 2000.

[35] M. Rosemann. Using reference models within the enterprise resource planning lifecycle. *Australian Accounting Review*, 10:19, 2000.

[36] M. Rosemann and W.M.P. van der Aalst. A Configurable Reference Modelling Language. QUT Technical report, FIT-TR-2003-05, Queensland University of Technology, Brisbane, 2003.

[37] F. Rump. *Geschäftsprozessmanagement auf der Basis ereignisgesteuerter Prozessketten*. Reihe Wirtschaftsinformatik, Teubner Verlag, Germany, 1999.

[38] SAP AG. *Reverse Business Engineering - training material nr 50046489*, 2001.

[39] A.-W. Scheer. *Business Process Modelling*. 3rd edition, 2000.

[40] G. Schimm. Process Mining elektronischer Geschäftsprozesse. In *Proceedings Elektronische Geschäftsprozesse*, 2001.

[41] G. Schimm. Process Mining linearer Prozessmodelle - Ein Ansatz zur automatisierten Akquisition von Prozesswissen. In *Proceedings 1. Konferenz Professionelles Wissensmanagement*, 2001.

[42] G. Schimm. Process Miner - A Tool for Mining Process Schemes from Event-based Data. In S. Flesca and G. Ianni, editors, *Proceedings of the 8th European Conference on Artificial Intelligence (JELIA)*, volume 2424 of *Lecture Notes in Computer Science*, pages 525–528. Springer-Verlag, Berlin, 2002.

[43] L. Silverston. *The Data Model Resource Book, Volume 1, A Library of Universal Data Models for all Enterprises*. revised edition, 2001.

[44] L. Silverston. *The Data Model Resource Book, Volume 2, A Library of Data Models for Specific Industries*. revised edition, 2001.

[45] M. Verbeek. *On Tools & Models, in: Dynamic Enterprise Innovation - Establishing Continuous Improvement in Business*. Baan Business Innovation, 3rd edition, 1998.

[46] A.J.M.M. Weijters and W.M.P. van der Aalst. Process Mining: Discovering Workflow Models from Event-Based Data. In B. Kröse, M. de Rijke, G. Schreiber, and M. van Someren, editors, *Proceedings of the 13th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC 2001)*, pages 283–290, 2001.

[47] A.J.M.M. Weijters and W.M.P. van der Aalst. Rediscovering Workflow Models from Event-Based Data. In V. Hoste and G. de Pauw, editors, *Proceedings of the 11th Dutch-Belgian Conference on Machine Learning (Benelearn 2001)*, pages 93–100, 2001.

[48] A.J.M.M. Weijters and W.M.P. van der Aalst. Workflow Mining: Discovering Workflow Models from Event-Based Data. In C. Dousson, F. Höppner, and R. Quiniou, editors, *Proceedings of the ECAI Workshop on Knowledge Discovery and Spatial Data*, pages 78–84, 2002.