Matching Observed Behavior and Modeled Behavior: An Approach Based on Petri nets and Integer Programming

Wil M.P. van der Aalst

Department of Technology Management
Eindhoven University of Technology
P.O.Box 513, NL-5600 MB Eindhoven, The Netherlands
w.m.p.v.d.aalst@tm.tue.nl

Abstract. Inspired by the way SAP R/3 and other transactional information systems log events, we focus on the problem to decide whether a process model and a frequency profile "fit" together. The problem is formulated in terms of Petri nets and an approach based on integer programming is proposed to tackle the problem. The integer program provides necessary conditions and, as shown in this paper, for relevant subclasses these conditions are sufficient. Unlike traditional approaches, the approach allows for labelled Petri nets with "hidden transitions", noise, etc.

Keywords: Process mining, Reference models, Petri nets, Integer programming, SAP R/3, Marking equation, Conformance testing.

1 Introduction

For many processes in practice there exist models. These model are *descriptive* or *prescriptive*, i.e., they are used to describe a process or they are used to control or guide the system. A typical example are the so-called reference models in the context of Enterprise Resource Planning (ERP) systems like SAP [24]. The SAP reference models are expressed in terms of so-called Event-driven Process Chains (EPCs) [23] describing how people should/could use the SAP R/3 system. Similarly models are used in the workflow domain [1], but also in many other domains ranging from flexible manufacturing and telecommunication to operating systems and software components [27]. In some domains these models are referred to as *specifications* or *blueprints*. In reality, the real process may deviate from the modeled process, e.g., the implementation is not consistent with the specification or people use SAP R/3 in a way not modeled in any of the EPCs.

Clearly, the problem of checking whether the modeled behavior and the observed behavior match is not new. However, when we applied our process mining techniques [4] to SAP R/3 we where confronted with the following interesting problem: The logs of SAP do not allow

for monitoring individual cases (e.g., purchase orders). Instead SAP only logs the fact that a specific transaction has been executed (without referring to the corresponding case). Hence, tools like the SAP Reverse Business Engineer (RBE) report on the frequencies of transaction types and not on the cases themselves. These transactions can be linked to functions in the EPCs, but, as indicated, not to individual cases. Moreover, some functions in the EPC do not correspond to a transaction code, and therefore, are not logged at all. This raises the following interesting question: *Do the modeled behavior (i.e., the EPC) and the observed behavior (i.e., the transaction frequencies) match?*

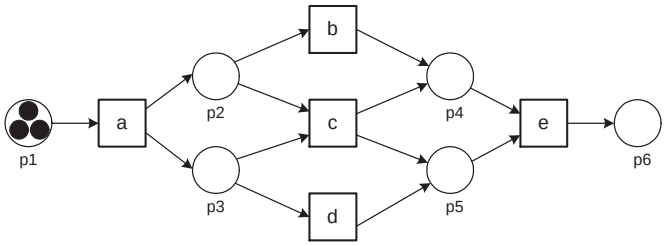


Fig. 1. A Petri net.

The problem of checking whether the modeled behavior and the observed behavior match is not only relevant in the context of SAP. In a wide variety of applications only frequencies are being recorded and/or it is impossible to link events to specific cases. Therefore, we consider an abstraction of the problem. Consider a *Petri net* with some initial marking [28, 29] and a *frequency profile* which is a partial function indicating how many times certain transitions fired. Consider for example the marked Petri net shown Figure 1. A frequency profile fp could be $fp(a) = 3$, $fp(b) = 2$, $fp(c) = 2$, $fp(d) = 2$, and $fp(e) = 3$, thus indicating the number of times each transition occurred. However, the modeled behavior (i.e., the marked Petri net) and the observed behavior (the frequency profile fp) do not match. It is easy to see that $fp(b)+fp(c)$ cannot exceed $fp(a)$ since b and c depend on the tokens produced by a. Now consider another frequency profile fp: $fp(a) = 3$, $fp(b) = 2$, $fp(d) = 2$, and $fp(e) = 3$, i.e., the number of times c occurred is unknown. Now the modeled behavior and the observed behavior match, i.e., the observed transition frequencies are consistent with the Petri net model. Moreover, it is clear that in this situation c occurred precisely once.

In the remainder we will focus on this problem and propose an approach based on *Integer Programming* (IP) [33, 36]. Using a marked Petri net and a frequency profile, an IP problem is formulated to check whether the modeled behavior and the observed behavior match and, if so, the frequencies of transitions not recorded in the profile are determined. First, we introduce some preliminaries, i.e., process mining, Petri nets, and integer programming, and discuss related work. Then we focus on the core problem and formulate the IP problem. We demonstrate the applicability of our approach using an example. Moreover, we show in more detail why the problem is relevant in the context of SAP and apply the approach to an SAP process model. Finally, we conclude the paper by summarizing the results and discussing future work.

2 Preliminaries

This section presents some preliminaries needed in the remainder of the paper. We first discuss the concept of *process mining* and then introduce the two techniques used in this paper: *Petri nets* and *Integer Programming*. Finally, we present some related work.

2.1 Process mining

The research reported in this paper is part of our work on process mining [4, 5, 14, 35]. The goal of process mining is to extract information about processes from transaction logs [4]. We typically assume that it is possible to record events such that (i) each event refers to an *activity* (i.e., a well-defined step in the process), (ii) each event refers to a *case* (i.e., a process instance), (iii) each event can have a *performer* also referred to as *originator* (the person executing or initiating the activity), and (iv) events have a *timestamp* and are totally ordered.[1] Table 1 shows an example of a log involving 19 events, 5 activities, and 6 originators. In addition to the information shown in this table, some event logs contain more information on the case itself, i.e., data elements referring to properties of the case.

Event logs such as the one shown in Table 1 are used as the starting point for mining. We distinguish three different perspectives: (1) the process perspective, (2) the organizational

[1] Note that in Table 1 we abstract from *event types*, i.e., we consider activities to be atomic. In real logs events typically correspond to the start or completion of an activity. This way it is possible to measure the duration of activity and to explicitly detect parallelism. Moreover, there are other event types related to failures, scheduling, delegations, etc. For simplicity we abstract from this in this paper. However, in our process mining tools we take event types into account.

case id	activity id	originator	timestamp
case 1	activity A	John	9-3-2004:15.01
case 2	activity A	John	9-3-2004:15.12
case 3	activity A	Sue	9-3-2004:16.03
case 3	activity B	Carol	9-3-2004:16.07
case 1	activity B	Mike	9-3-2004:18.25
case 1	activity C	John	10-3-2004:9.23
case 2	activity C	Mike	10-3-2004:10.34
case 4	activity A	Sue	10-3-2004:10.35
case 2	activity B	John	10-3-2004:12.34
case 2	activity D	Pete	10-3-2004:12.50
case 5	activity A	Sue	10-3-2004:13.05
case 4	activity C	Carol	11-3-2004:10.12
case 1	activity D	Pete	11-3-2004:10.14
case 3	activity C	Sue	11-3-2004:10.44
case 3	activity D	Pete	11-3-2004:11.03
case 4	activity B	Sue	11-3-2004:11.18
case 5	activity E	Clare	11-3-2004:12.22
case 5	activity D	Clare	11-3-2004:14.34
case 4	activity D	Pete	11-3-2004:15.56

Table 1. An event log.

perspective and (3) the case perspective. The *process perspective* focuses on the control-flow, i.e., the ordering of activities. The goal of mining this perspective is to find a good characterization of all possible paths, e.g., expressed in terms of a Petri net or Event-driven Process Chain (EPC). The *organizational perspective* focuses on the originator field, i.e., which performers are involved and how are they related. The goal is to either structure the organization by classifying people in terms of roles and organizational units or to show relation between individual performers (i.e., build a social network). The *case perspective* focuses on properties of cases. Cases can be characterized by their path in the process or by the originators working on a case. However, cases can also be characterized by the values of the corresponding data elements. For example, if a case represents a replenishment order it is interesting to know the supplier or the number of products ordered.

The ProM framework [4, 5, 14] has been developed to extract information from event logs.[2] It offers a wide varieties of so-called "plug-ins". There are mining plug-ins for each of the three perspectives. Figure 2 shows a screenshot of the ProM tool while analyzing the event log shown in Table 1.

[2] The ProM framework can be downloaded from http://www.processmining.org.

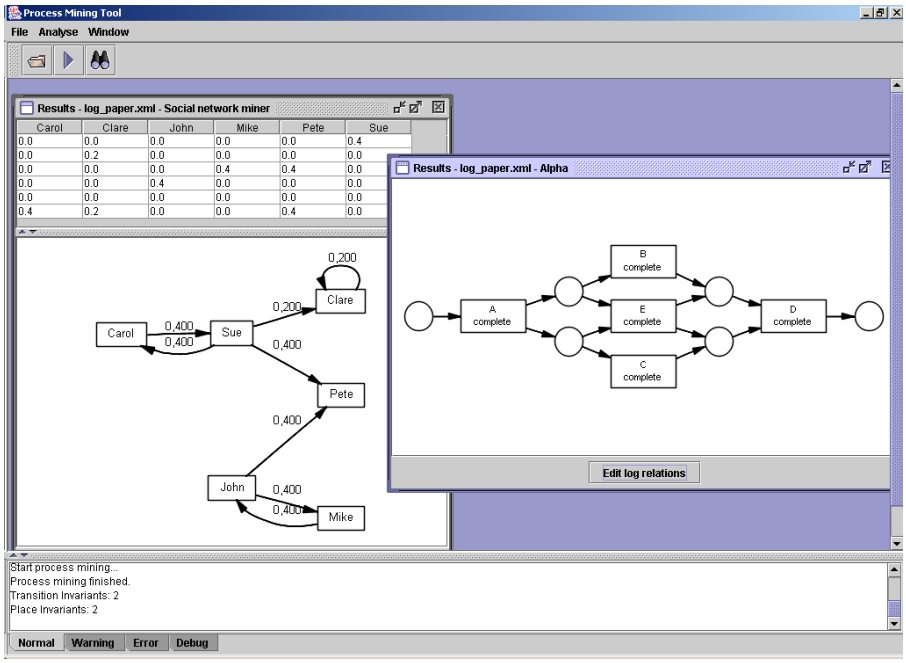


Fig. 2. Some mining results obtained using our ProM tool (see http://www.processmining.org). The results shown are based on the event log shown in Table 1.

The Petri net [13] shown on the right-hand side in Figure 2 is the result of applying the α algorithm plug-in to the event log shown in Table 1. This is one of the five mining plug-ins focussing on the *process perspective*. Note that the event log contains information about five cases (i.e., process instances). The log shows that for four cases (1, 2, 3, and 4) the activities A, B, C, and D have been executed. For the fifth case only three activities are executed: activities A, E, and D. Each case starts with the execution of A and ends with the execution of D. If activity B is executed, then also activity C is executed. However, for some cases activity C is executed before activity B. The α algorithm [5] translates this information into causal dependencies and generates the Petri shown in Figure 2. It is easy to see that this is indeed the most likely process model explaining the behavior observed in the log. The Petri net starts with activity A and finishes with activity D. These activities are represented by transitions. After executing A there is a choice between either executing B and C in parallel or just executing activity E.

ProM also has plug-ins to analyze the *organizational perspective*. An example is shown on the left-hand side in Figure 2. Using the social network mining plug-in [3] a so-called social

network is generated. The social network shown in Figure 2 is based on the transfer of work from one individual to another, i.e., the focus is on relations among individuals (or groups of individuals) based on how work flows through the organization. Consider again Table 1. Although Carol and Mike can execute the same activities (B and C), Mike is always working with John (cases 1 and 2) and Carol is always working with Sue (cases 3 and 4). Probably Carol and Mike have the same role but based on the small sample shown in Table 1 it seems that John is not working with Carol and Sue is not working with Carol. These examples show that the event log can be used to derive relations between performers of activities, thus resulting in a sociogram as shown in Figure 2. The sociogram shows that work is transferred to Pete but not vice versa. Mike only interacts with John and Carol only interacts with Sue. Clare is the only person transferring work to herself.

Besides the "How?" and "Who?" question (i.e., the process and organization perspectives), there is the *case perspective* that is concerned with the "What?" question. The case perspective looks at the case as a whole and tries to establish relations between the various properties (i.e., data) of a case. ProM also allows for the analysis of this perspective (e.g., through the LTL checker plug-in). However, Table 1 does not show any data elements. Therefore, we do not elaborate on this and simply refer to [4, 14].

As Figure 2 shows, an event log such as the one shown in Table 1 can be the starting point of a wide variety of analysis techniques. Unfortunately, these classical forms of process mining *only work if the identities of individual cases are logged*. In reality, like in SAP, often only frequencies of activities are known or the first column in Table 1 is missing (case id's). Therefore, we would like to extend our work on process mining to situations were only frequencies are known as described in the introduction.

2.2 Petri nets

This section introduces the basic Petri net terminology and notations (cf. [29, 12]). Readers familiar with Petri nets can skip this section.

The classical Petri net is a directed bipartite graph with two node types called *places* and *transitions*. The nodes are connected via directed *arcs*. Connections between two nodes of the same type are not allowed. Places are represented by circles and transitions by rectangles.

Definition 1 (Petri net). *A Petri net is a triple* (P, T, F)*:*

- P *is a finite set of places,*
- T *is a finite set of transitions* ($P \cap T = \emptyset$)*,*
- $F \subseteq (P \times T) \cup (T \times P)$ *is a set of arcs (flow relation)*

A place p is called an *input place* of a transition t iff there exists a directed arc from p to t. Place p is called an *output place* of transition t iff there exists a directed arc from t to p. We use $\bullet t$ to denote the set of input places for a transition t. The notations $t\bullet$, $\bullet p$ and $p\bullet$ have similar meanings, e.g., $p\bullet$ is the set of transitions sharing p as an input place. In this paper, we do not consider multiple arcs from one node to another. However, all results can be extended to Petri nets with arcs weights.

Figure 1 shows a Petri net with 5 transitions (a, b, c, d, and e) and 6 places ($p1, \ldots p6$).

At any time a place contains zero or more *tokens*, drawn as black dots. The *state*, often referred to as *marking*, is the distribution of tokens over places, i.e., $M \in P \rightarrow \mathbb{N}$. We will represent a marking as follows: $1'p1 + 2'p2 + 1'p3 + 0'p4$ is the marking with one token in place $p1$, two tokens in $p2$, one token in $p3$ and no tokens in $p4$. We can also represent this marking as follows: $p1 + 2'p2 + p3$. The marking shown in Figure 1 is $p1$. (Note the overloading of notation.) To compare markings we define a partial ordering. For any two markings M_1 and M_2, $M_1 \leq M_2$ iff for all $p \in P$: $M_1(p) \leq M_2(p)$.

The number of tokens may change during the execution of the net. Transitions are the active components in a Petri net: they change the marking of the net according to the following *firing rule*:

(1) A transition t is said to be *enabled* iff each input place p of t contains at least one token.
(2) An enabled transition may *fire*. If transition t fires, then t *consumes* one token from each input place p of t and *produces* one token for each output place p of t.

In Figure 1 transition a is enabled. Firing a results in marking $2'p1 + p2 + p3$. In this marking, three additional transitions (besides a) are enabled (b, c, d). Any of these transitions may fire. However, firing one of these transition will disable one or two other transitions, e.g., firing c will disable both b and d.

Given a Petri net (P, T, F) and a marking M_1, we have the following notations:

- $M_1 \xrightarrow{t} M_2$: transition t is enabled in marking M_1 and firing t in M_1 results in marking M_2
- $M_1 \rightarrow M_2$: there is a transition t such that $M_1 \xrightarrow{t} M_2$
- $M_1 \xrightarrow{\sigma} M_n$: the firing sequence $\sigma = t_1 t_2 t_3 \ldots t_{n-1}$ leads from marking M_1 to marking M_n via a (possibly empty) set of intermediate markings $M_2, ...M_{n-1}$, i.e., $M_1 \xrightarrow{t_1} M_2 \xrightarrow{t_2} ... \xrightarrow{t_{n-1}} M_n$

A marking M_n is called *reachable* from M_1 (notation $M_1 \xrightarrow{*} M_n$) iff there is a firing sequence σ such that $M_1 \xrightarrow{\sigma} M_n$. Note that the empty firing sequence is also allowed, i.e., $M_1 \xrightarrow{*} M_1$.

To manipulate firing sequences, we introduce the *Parikh vector* $\pi_\sigma \in T \rightarrow \mathbb{N}$, where $\pi_\sigma(t)$ denotes the number of occurrences of transition t in σ.

We use (PN, M) to denote a Petri net PN with an initial marking M. A marking M' is a *reachable marking* of (PN, M) iff $M \xrightarrow{*} M'$. Consider the Petri net shown in Figure 1 with only one token in $p1$. For this initial marking there are 6 reachable markings.

2.3 Integer Programming

Besides Petri nets we use *Integer Programming* (IP) to address the problem of checking whether the modeled behavior and the observed behavior match. An IP problem can be seen as a variant of the classical *Linear Programming* (LP) problem [33, 36]. Therefore, before introducing the IP problem, we briefly introduce the basic idea of an LP problem. First, we define the LP problem. The *standard form* of an LP problem is:

$$\begin{aligned} &\min\ (c_1, c_2, \ldots, c_n)(x_1, x_2, \ldots, x_n) \\ &\text{s.t. } A(x_1, x_2, \ldots, x_n) = (b_1, b_2, \ldots, b_m) \\ &\qquad x_i \geq 0 \quad \text{for all } 1 \leq i \leq n \end{aligned}$$

where $x_1, x_2, \ldots, x_n$ are n *variables* forming a (unknown) vector $(x_1, x_2, \ldots, x_n)$, A is a matrix of known coefficients, and $(c_1, c_2, \ldots, c_n)$ and $(b_1, b_2, \ldots, b_m)$ are vectors of known coefficients. The expression $(c_1, c_2, \ldots, c_n)(x_1, x_2, \ldots, x_n)$ takes the product of two vectors and is called the *objective function*. The equations formed by $A(x_1, x_2, \ldots, x_n) = (b_1, b_2, \ldots, b_m)$ are called the constraints. All these entities must have consistent dimensions.[3]. Note that n is the number of variables and m is the number of constraints. The goal is to minimize the objective function while respecting the constraints.

[3] Technically, one should add transpose symbols to vector/matrix multiplications.

Although all linear programs can be put into the standard form, in practice it may not be necessary to do so. For example, although the standard form requires all variables to be non-negative it is possible to rewrite $k \leq x_i \leq l$ into the standard form by using two new variables $x_k = x_i - k$ and $x_l = l - x_i$ and require $x_k \geq 0$ and $x_l \geq 0$. Similarly, inequalities in the constraints can be replaced by equalities by introducing explicit slack variables. The simplex method was the first method developed to solve LP problems. A much more efficient (polynomial time) algorithm was found by Karmarkar in 1984 [22].

For many applications the assumption that the variables are continuous is unrealistic. In many practical applications, some variables will denote decisions, e.g., $x_i = 0$ or $x_i = 1$ rather than any value between 0 and 1. In an Integer Programming (IP) problem the variables are integers, i.e., it is like an LP problem but now x_i should be *integer* for all $1 \leq i \leq n$. Unfortunately, the IP problem can no longer be solved in polynomial time and one needs to resort to computationally expensive methods like branch and bound [33, 36].

In some cases it is useful to consider the *LP relaxation* of an IP problem. In this case the objective function and constraints are the same but the integer variables are replaced by appropriate continuous variables and constraints. For example $x_i = 0$ or $x_i = 1$ is then replaced by $0 \leq x_i \leq 1$. The LP solution might turn out to have all variables taking integer values at the LP optimal solution. In this case we obtain an optimal integer solution. If we have variables taking fractional values at the LP optimal solution, then we can round these to the nearest integer value. However, in many cases the rounded LP relaxation solution either violates a constraint or yields a non-optimal solution, i.e., LP relaxation is fast (polynomial time) but is may be inaccurate to some degree. Nevertheless, IP problems are typically easier to solve than methods requiring the construction of the full state space.

2.4 Related work

The starting point of this work is the literature on process mining [4–6, 8, 18, 20, 26, 32, 35]. The idea of applying process mining in the context of workflow management was first introduced in [6]. Since then several researchers have been working on this topic and we refer to [4] for a survey on process mining. ProM [14] is an example of a tool for process mining. An example of a commercial tool is the ARIS Process Performance Manager (PPM) [20]. Some of the ideas

developed in the context on the ProM tool have been adopted in tools like PPM (e.g., the OrgAnalyzer in version 4).

Although not explicitly addressed in this paper, our work is related to reference modeling [7, 30]. One of the most comprehensive models is the SAP reference model [9, 24]. Its data model includes more than 4000 entity types and the reference process models cover more than 1000 business processes and inter-organizational business scenarios. Most of the other dominant ERP vendors have similar or alternative approaches towards reference models. We have developed a new reference modeling language: Configurable EPCs [31], i.e., an extension of the EPC language [23] used by SAP and ARIS. Using classical process mining techniques we have developed an approach to discover the configuration [21].

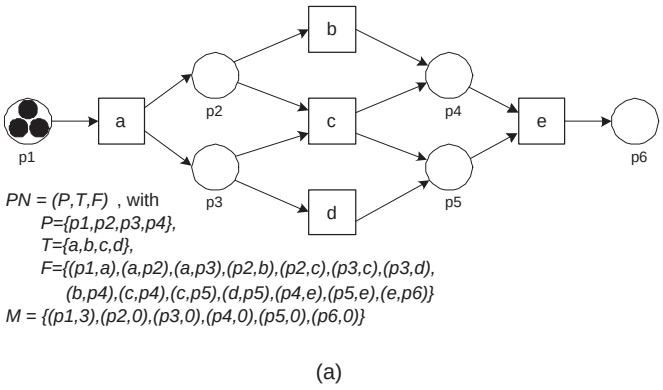
In a technical sense, the work presented is most related to the "Marking Equation" known from Petri net theory [27, 11, 34] and this paper builds on some of these results. However, the approach presented differs in at least two ways. First of all, the marking equation considers the initial *and* resulting marking while we only consider the initial marking. Second, we allow for transition frequencies that are unknown, i.e., the frequency profile may be incomplete. Moreover, the approach allows for the extensions described in Section 5 while the marking equation does not. Clearly there are also relations with the classical results on place and transition invariants [12, 34, 28]. However, these are less direct.

3 Matching a marked Petri net and a frequency profile

As indicated in the introduction, we use Petri nets to model processes. However, other types of models, e.g., the EPCs used by the SAP reference model, can be mapped onto Petri nets.[4] Petri nets may be used to model a wide variety of processes. A Petri net can model what we think the process is (i.e., a *descriptive* model) but it can also model what the process should be (i.e., a *prescriptive* model). In both cases, the real process may deviate from what is modeled in the Petri net. In this section, we investigate whether the modeled behavior (i.e., Petri net) and the observed behavior match. Since in reality we often cannot inspect the state and just observe events, it is realistic to assume that we can only monitor the firing of transitions.

[4] Note that the mapping of semi-formal models such as EPCs is a not a trivial task. It may be necessary to remove ambiguities before mapping the model onto a Petri net [2, 10, 25].

Moreover, we assume that we cannot link transition occurrences to specific tokens or exploit their ordering in time, i.e., we only know the *frequency profile*.



p	fp(p)
a	3
b	2
c	2
d	2
e	3

(b)

p	fp(p)
a	3
b	2
c	?
d	2
e	3

(c)

Fig. 3. A process model with two frequency profiles.

To illustrate the problem, we again show the Petri net used in the introduction. Figure 3, shows a Petri net and two frequency profiles. Both the graphical and textual representation of the marked Petri net are given in Figure 3(a). For a Petri net with transitions T, the frequency profile refers to a subset of T, i.e., frequency profile $fp \in T \not\to \mathbb{N}$ is a partial function. For $t \in dom(fp)$, $fp(t)$ is the number of times t occurred/fired. For $t \notin dom(fp)$ this is unknown. If $dom(fp) = T$, the frequency profile is *complete*. Figure 3(b) shows a complete frequency profile. The frequency profile shown in Figure 3(c) is incomplete because $fp(c)$ is not given (i.e., $c \notin dom(fp)$).

The marked Petri net shown in Figure 3(a) and frequency profile given in Figure 3(b) do not "match", because there is no firing sequence starting from the initial marking resulting in the fp shown ($fp(b) + fp(c)$ cannot exceed $fp(a)$ since b and c depend on the tokens produced by a, but it does). However, a match with the frequency profile given in Figure 3(c) is possible. The firing sequence $(a, b, d, e, a, b, d, e, a, c, e)$ fires a and e three times, b and d two times, and c once, i.e., it is consistent with the fp shown in Figure 3(c).

Both for complete and incomplete frequency profiles we define the predicate $match(PN, M, fp)$ to formalize the notions just introduced.

Definition 2 (Match). *Let (PN, M) be a marked Petri net with $PN = (P, T, F)$ and $fp \in T \not\to \mathbb{N}$ a frequency profile. (PN, M) and fp match if there exists a firing sequence σ enabled in M (i.e., $M \xrightarrow{\sigma}$) such that for all $t \in dom(fp)$: $fp(t) = \pi_\sigma(t)$. (Notation: $match(PN, M, fp)$.)*

Clearly, $match(PN, M, fp) = false$ for Figure 3(b) and $match(PN, M, fp) = true$ for Figure 3(b). Note that for any marked Petri net there is a trivial matching profile fp with $dom(fp) = \emptyset$.

Definition 2 refers to the existence of one firing sequence σ. This firing sequence may refer to multiple process instances (called "cases" in workflow jargon) as shown in the example. $(a, b, d, e, a, b, d, e, a, c, e)$ symbolizes the complete processing of the three cases in place $p1$. In Figure 3(a) the initial marking determines the number of cases. However, it is also possible to add source transitions (i.e., transitions without any input places) and sink transitions (i.e., transitions without any output places). In the example of Figure 3 we could have started with an empty initial marking (no tokens) and a source transition t_{start} with $\bullet t_{start} = \emptyset$ and $t_{start}\bullet = \{p1\}$. In this case, Figure 3(b) still does not match while Figure 3(c) does. This example shows that $match(PN, M, fp)$ can be applied to "open nets" (source and sink transitions and no initial tokens), "closed nets" (no source and sink transitions and initially some places are marked), and mixtures of the latter two.

Even for moderate examples, the number of firing sequences may be too large to check $match(PN, M, fp)$. Therefore, in the spirit of [11, 27], we can try to formulate a linear algebraic representation. Given the discrete nature of firing transitions, we propose an Integer Programming (IP) problem rather than an Linear Programming (LP) problem [33, 36]. In other words, we consider the function $match(PN, M, fp)$ and try to formulate it in terms of an IP problem.

Definition 3 (Integer programming problem). *Let (PN, M) be a marked Petri net with $PN = (P, T, F)$ and $fp \in T \not\to \mathbb{N}$ a frequency profile. $IP(PN, M, fp)$ is the corresponding*

Integer Programming (IP) problem:

$$
\begin{array}{ll}
\min \sum_{t \in T} f_t & \\
\text{s.t. } f_t = \mathit{fp}(t) & \text{for all } t \in \mathit{dom}(\mathit{fp}) \\
\quad f_{(t,p)} = f_t & \text{for all } (t,p) \in F \cap (T \times P) \\
\quad f_{(p,t)} = f_t & \text{for all } (p,t) \in F \cap (P \times T) \\
\quad M(p) + \sum_{t \in {}^\bullet p} f_{(t,p)} - \sum_{t \in p^\bullet} f_{(p,t)} \geq 0 & \text{for all } p \in P \\
\quad f_t \geq 0 & \text{for all } t \in T \\
\quad f_t & \text{integer for all } t \in T \\
\quad f_{(x,y)} & \text{integer for all } (x,y) \in F
\end{array}
$$

There are two types of positive integer variables: f_t for transition frequencies and $f_{(x,y)}$ for arc frequencies. The first constraint specifies that the transition frequencies should match the frequency profile. Note that for some transitions there may not be a frequency in the frequency profile. The second and third constraint refer to the fact that transition frequencies and arc frequencies need to be aligned. The fourth type of constraint is the most interesting one. For each place, there should be a balance between the inflow of tokens and the outflow of tokens, i.e., it is not possible to consume more tokens than the initial ones plus the produced ones. The objective function minimizes the number of firings. Given the nature of the problem this is of less importance and alternative objective functions can be defined, e.g., an objective function maximizing or minimizing the number of tokens in the net.

Before we discuss the relation between $\mathit{match}(PN, M, \mathit{fp})$ and $\mathit{IP}(PN, M, \mathit{fp})$, let us return to the Petri net shown in Figure 3(a). Assuming some initial marking M and some frequency

profile fp, $IP(PN, M, fp)$ is formulated as follows.

$$
\begin{aligned}
\min\ & f_a + f_b + f_c + f_d + f_e \\
\text{s.t. } & f_a = fp(a) \\
& \ldots \\
& f_{(a,p2)} = f_a \\
& \ldots \\
& f_{(p1,a)} = f_a \\
& \ldots \\
& M(p1) - f_{(p1,a)} \geq 0 \\
& M(p2) + f_{(a,p2)} - f_{(p2,b)} - f_{(p2,c)} \geq 0 \\
& M(p3) + f_{(a,p3)} - f_{(p3,c)} - f_{(p3,d)} \geq 0 \\
& M(p4) + f_{(b,p4)} + f_{(c,p4)} - f_{(p4,e)} \geq 0 \\
& M(p5) + f_{(c,p5)} + f_{(d,p5)} - f_{(p5,e)} \geq 0 \\
& M(p6) + f_{(e,p6)} \geq 0 \\
& f_a \geq 0 \\
& \ldots \\
& f_a \quad \text{integer} \\
& \ldots \\
& f_{(p1,a)} \quad \text{integer} \\
& \ldots
\end{aligned}
$$

Applying this to the initial marking shown in Figure 3(a) and the frequency profile $fp(a) = 3$, $fp(b) = 2$, $fp(c) = 2$, $fp(d) = 2$, and $fp(e) = 3$ indeed results in an IP problem without a solution. While applying it to the second frequency profile $fp(a) = 3$, $fp(b) = 2$, $fp(d) = 2$, and $fp(e) = 3$ yields the solution where $f_c = 1$. In the latter case the value of the objective function is 11.

In the remainder of this section we investigate the relation between $match(PN,\ M, fp)$ and $IP(PN, M, fp)$, i.e., "Can the IP problem be used to determine whether the modeled and observed behavior match?". It is important to establish this relation because, $IP(PN, M, fp)$ an be solved more efficiently than determining $match(PN,\ M, fp)$ on the basis of constructing and traversing the coverability graph [12, 27, 29].

The following theorem shows that, as expected, the IP problem indeed provides necessary requirements.

Theorem 1. *Let (PN, M) be a marked Petri net with $PN = (P, T, F)$ and $fp \in T \not\to \mathbb{N}$ a frequency profile. If $match(PN, M, fp)$, then $IP(PN, M, fp)$ has a solution.*

Proof. If $match(PN, M, fp)$, then there exists a firing sequence σ enabled in M (i.e., $M \xrightarrow{\sigma}$) such that for all $t \in T$: $fp(t) = \pi_\sigma(t)$. Let M' be the resulting marking. Now consider the IP

problem. The only constraint that could be violated is $M(p) + \sum_{t \in \bullet p} f_{(t,p)} - \sum_{t \in p\bullet} f_{(p,t)} \geq 0$ for some $p \in P$. However, this constraint follows directly from the firing rule. In fact, $M(p) + \sum_{t \in \bullet p} f_{(t,p)} - \sum_{t \in p\bullet} f_{(p,t)} = M'(p)$. □

The theorem shows that, if $IP(PN, M, fp)$ does not have a solution, $match(PN, M, fp)$ does not hold. This allows for the quick detection of mismatches between the model and the observed behavior.

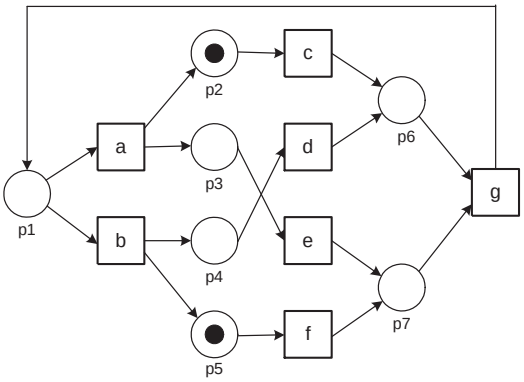


Fig. 4. Counter example.

Unfortunately, the result does not hold in the opposite direction, as can be shown by an example taken from [12]. Figure 4 shows a marked Petri net. Let $fp(t) = 1$ for all transitions t except for $t = g$ which occurs twice (i.e., $fp(g) = 2$). It is easy to verify that $IP(PN, M, fp)$ has a solution. However, the marked Petri net and the frequency profile do not match because there is no firing sequence (starting in the initial marking shown in Figure 4) that fires g twice and all other transitions once. (Note that it is impossible to return to the initial marking.) Fortunately, for certain subclasses the result does hold in the opposite direction. In the remainder we will explore some of these subclasses for which $match(PN, M, fp)$ if and only if $IP(PN, M, fp)$ has a solution. The following theorem, shows that this is the case for *all* acyclic processes.

Theorem 2. *Let (PN, M) be an acyclic marked Petri net with $PN = (P, T, F)$ and $fp \in T \not\to \mathbb{N}$ a frequency profile such that $IP(PN, M, fp)$ has a solution. There exists a firing sequence σ enabled in M such that for all $t \in dom(fp)$: $fp(t) = \pi_\sigma(t)$, i.e., $match(PN, M, fp)$.*

Proof. In the solution of $IP(PN, M, fp)$ each transition $t \in T$ fires f_t times. Let $n = \sum_{t \in T} f_t$. If $n = 0$, the empty sequence is enabled and the theorem holds. If $n > 0$, remove all transitions t for which $f_t = 0$. Moreover, remove all places and arcs not connected to a transition t for which $f_t > 0$. Let PN' be the resulting net and M' the resulting marking. Clearly, PN' is acyclic. At least one transition is enabled in (PN', M'). (If not, the fact that PN' is acyclic would imply that there is an empty source place p with some output transition t'. However, $M(p) + \sum_{t \in \bullet p} f_{(t,p)} - \sum_{t \in p\bullet} f_{(p,t)} = M'(p) + 0 - f_{(p,t')} - \ldots = 0 + 0 - f_{t'} - \ldots \geq 0$. Clearly, this leads to a contradiction.) Fire this enabled transition t^* and let M^* be the resulting marking and fp^* such that $fp^*(t^*) = fp(t^*) - 1$ and for all other $t \in dom(fp)$: $fp^*(t) = fp(t)$. Clearly, $IP(PN, M^*, fp^*)$ has a solution. Repeat the above process until $n = 0$. In each step, a transition t^* is fired thus forming a sequence σ enabled in M. □

Note that the proof of this theorem is similar to Theorem 16 in [27]. Consider Figure 4 with the arc from g to $p1$ removed and a new place $p8$ added as an output place of g. Now for any marking M and any frequency profile fp such that $IP(PN, M, fp)$ has a solution, there exists a corresponding firing sequence, i.e., $match(PN, M, fp)$. For example, given the marking shown in Figure 4 and the acyclic variant of the net, the IP problem has a solution for the following frequency profile fp: $fp(a) = fp(b) = fp(d) = fp(e) = 0$, $fp(c) = fp(f) = fp(g) = 1$. Indeed, as suggested by Theorem 2, there is a firing sequence firing c, f and g (e.g., cfg).

The counter example shown in Figure 4 is free-choice [12]. Therefore, one could consider to proving Theorem 2 for subclasses of free-choice nets (i.e., replace the requirement that the net is acyclic with some other structural requirement). Two well-known subclasses are the class of *marked graphs* and the class of *state machines* [12, 27, 29].

A *marked graph* is a Petri net with for each place $p \in P$: $|\bullet p| = |p \bullet| = 1$ (i.e., places cannot have multiple input or output transitions). A *circuit* is a circular path in the Petri net such that no element (i.e., place or transition) occurs more than once. It is easy to see that in a marked graph the number of tokens in a circuit is constant. Therefore, a circuit remains (un)marked if it is (un)marked in the initial marking. Using existing results it is easy to prove that Theorem 2 applies to (cyclic) marked graphs where each circuit is marked.

Theorem 3. *Let (PN, M) be an marked graph with $PN = (P, T, F)$ and $fp \in T \not\to \mathbb{N}$ a frequency profile. If each circuit is initially marked, then $IP(PN, M, fp)$ has a solution if and only if $match(PN, M, fp)$.*

Proof. As shown in Theorem 1, $match(PN, M, fp)$ implies that $IP(PN, M, fp)$ has a solution. Remains to prove that $IP(PN, M, fp)$ has a solution also implies $match(PN, M, fp)$. Consider a solution assigning values to each f_t and $f_{(x,y)}$. Let M' be a marking defined as follows: $M(p) + \sum_{t \in \bullet p} f_{(t,p)} - \sum_{t \in p\bullet} f_{(p,t)} = M'(p)$ for all $p \in P$. Note that M' is indeed a marking, i.e., for each $p \in P$, $M'(p)$ is a non-negative integer. This implies that the marking equation $M + N.X = M'$ has a solution. (N is the incidence matrix and X is a vector.) This solution is given by the values assigned to f_t. Because there is a solution, M and M' agree on all place invariants. For live marked graphs a marking M' is reachable from M if and only if both agree on all place invariants (cf. Theorem 3.21 in [12]). A marked graph where each circuit is initially marked is live (cf. Theorem 3.15 in [12]). Therefore, M' is reachable from M and $match(PN, M, fp)$. □

Figure 5 shows a marked graph. For any initial marking M, the IP problem has a solution if and only if $match(PN, M, fp)$ (provided that every circuit is initially marked).

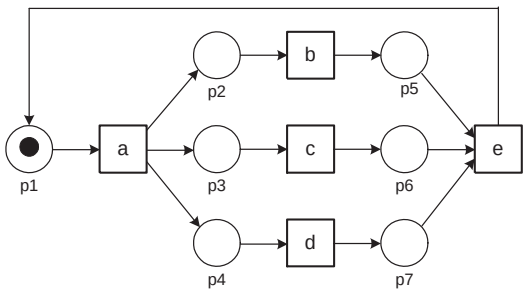


Fig. 5. Marked graph.

A Petri net is a *state machine* iff transitions cannot have more than one input or output place, i.e., for each transition $t \in T$: $|\bullet t| = |t \bullet| = 1$. It is easy to prove that Theorem 3 also holds for state machines as long as the the net is strongly connected (i.e., there is a directed path from any node to any other node in the net) and initially there is at least one token.

Theorem 4. *Let (PN, M) be a strongly-connected state machine with $PN = (P, T, F)$ and a non-empty initial marking M and $fp \in T \not\to \mathbb{N}$ a frequency profile. $IP(PN, M, fp)$ has a solution if and only if $match(PN, M, fp)$.*

Proof. As shown in Theorem 1, $match(PN, M, fp)$ implies that $IP(PN, M, fp)$ has a solution. Remains to prove that the reverse also holds. Consider a solution assigning values to each f_t and $f_{(x,y)}$. Let M' be a marking defined as follows: $M(p) + \sum_{t \in {}^\bullet p} f_{(t,p)} - \sum_{t \in p^\bullet} f_{(p,t)} = M'(p)$ for all $p \in P$. Note that M' is indeed a marking, i.e., for each $p \in P$, $M'(p)$ is a non-negative integer. The number of tokens in M equals the number of tokens in M', in fact M and M' agree on all place invariants. Moreover, the marked state machine is live because PN is a strongly-connected state machine and M is non-empty (cf. Theorem 3.3 in [12]). Using the second reachability theorem (cf. Theorem 3.8 in [12]), it follows that M' is reachable from M and $match(PN, M, fp)$. □

Figure 6 shows a strongly connected state machine. For any non-empty initial marking M $IP(PN, M, fp)$ has a solution if and only if $match(PN, M, fp)$.

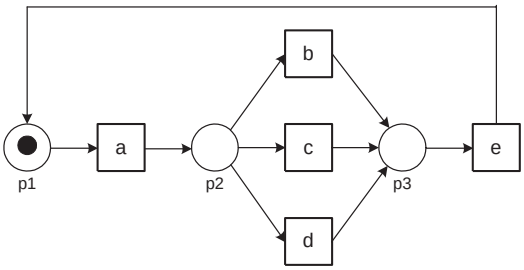


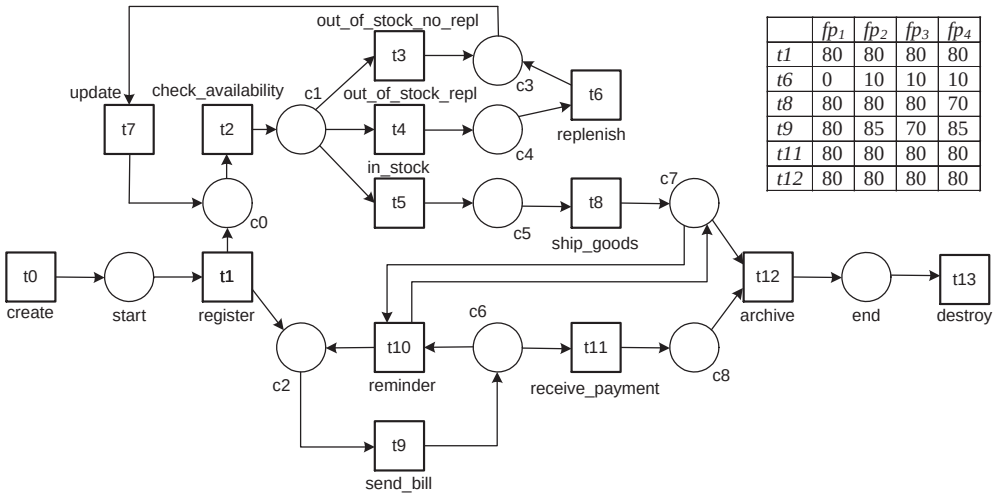
Fig. 6. State machine.

In this section, we explored the relation between $match(PN, M, fp)$ (i.e., the predicate indicating that a process model and observed transition frequencies fit together) and $IP(PN, M, fp)$ (i.e., an integer programming problem). In the remainder, we consider a larger example, possible extensions, and the application of the results in the SAP context.

4 Example

After showing a number of abstract examples, we now use the more realistic example shown in Figure 7. It describes the workflow [1] of handling orders. The upper half models the logistical

subprocess while the lower half models the financial subprocess. Most of the workflow should be self explanatory except perhaps for the construct involving $c7$ and $t10$ (*reminder*): A reminder can only be sent if the goods have been shipped.



	fp_1	fp_2	fp_3	fp_4
t1	80	80	80	80
t6	0	10	10	10
t8	80	80	80	70
t9	80	85	70	85
t11	80	80	80	80
t12	80	80	80	80

Fig. 7. A Petri net modeling the processing of customer orders and four frequency profiles.

Unlike the other two Petri nets, the initial marking is empty. Instead a source and a sink transition have been added. Transition $t0$ (*create*) creates the order while $t13$ (*destroy*) marks the end of the order. This pattern is often used to model an unknown number of cases.

Suppose that only the steps $t1$ (*register*), $t6$ (*replenish*), $t8$ (*ship_goods*), $t9$ (*send_bill*), $t11$ (*receive_payment*), and $t12$ (*archive*) are recorded. Figure 7 shows four frequency profiles (fp_1, fp_2, fp_3, and fp_4). The IP problems corresponding to the first two profiles (fp_1 and fp_2), both have a solution. It is also easy to see that fp_1 and fp_2 both indeed match with the Petri net. Note that in the first profile there are no replenishment orders and no reminders, i.e., $t4$, $t6$ and $t10$ do not fire. It is also interesting to note that the number of times $t3$ and $t7$ fire is not constrained by fp_1, however, by the objective function their frequencies are set to 0. In the second profile there are 10 replenishment orders and 5 reminders. The IP problems corresponding to the last two profiles (fp_3 and fp_4), both do not have a solution and, indeed, fp_3 and fp_4 do not match with the Petri net. In fp_3 there are not enough bills (70) to justify the number of payments (80). In fp_4 there are not enough shipments.

5 Extensions

A Linear Programming (LP) problem can be solved in polynomial time while an IP problem is NP complete [33, 36]. Therefore, it may be interesting to consider the LP relaxation of $IP(PN, M, fp)$. We expect that in some cases this will provide good results. Note that often the rounded LP relaxation provides a feasible but non-optimal solution (but not always, cf. the example net shown on page 269 in [11]). Since the objective function is of less interest, this is not a problem. Also note that if the IP problem has a solution the LP problem will also have a solution. Therefore, Theorem 1 also holds for the LP relaxation. As a result the LP problem can be used to quickly point out discrepancies between the process model and the frequency profile.

The LP relaxation is also interesting if the frequency profile is not exact or if we want to abstract from exceptions, i.e., if we consider *noise* we are not interested in the exact number of firings but in an approximate number. Suppose we want to allow a margin of 10 percent. To specify this we replace the first constraint in Definition 3 ($f_t = fp(t)$) by two weaker constraints: $f_t \geq 0.9fp(t)$ and $f_t \leq 1.1fp(t)$. Such approximations are also needed if we collect data for a *limited period* with an *unknown* number of tokens in the *initial marking*.

Definition 4. *Let (PN, M) be a marked Petri net with $PN = (P, T, F)$, $fp \in T \not\to \mathbb{N}$ a frequency profile, and α the noise level ($0 \leq \alpha \leq 1$) . The corresponding LP (IP) problem allowing for α noise:*

$$
\begin{array}{ll}
\min & \sum_{t \in T} f_t \\
\text{s.t.} & f_t \geq (1-\alpha) fp(t) \quad \text{for all } t \in dom(fp) \\
& f_t \leq (1+\alpha) fp(t) \quad \text{for all } t \in dom(fp) \\
& f_{(t,p)} = f_t \quad \text{for all } (t,p) \in F \cap (T \times P) \\
& f_{(p,t)} = f_t \quad \text{for all } (p,t) \in F \cap (P \times T) \\
& M(p) + \sum_{t \in {}^\bullet p} f_{(t,p)} - \sum_{t \in p^\bullet} f_{(p,t)} \geq 0 \quad \text{for all } p \in P \\
& f_t \geq 0 \quad \text{for all } t \in T \\
& f_t \quad \text{(integer) for all } t \in T \\
& f_{(x,y)} \quad \text{(integer) for all } (x,y) \in F
\end{array}
$$

Note that Definition 4 defines both an LP and and IP problem. The only difference is that for the LP problem the variables do not need to be integers.

Definition 4 allows for the application of our approach in the context of noise. Moreover, it can also resolve issues such as partial or inaccurate knowledge of the initial marking. However,

we would also like to point at the fact that the addition of source and sink transitions can be used to make the whole approach more robust (cf. beginning of Section 3).

Another extension is the situation where multiple transitions refer to the same event, e.g., in SAP multiple functions in the EPC may generate the same transaction. This corresponds to a labeled Petri net with multiple transitions having the same label. Again this is easy to incorporate in the IP problem. The frequency profile is no longer a mapping from transitions to frequencies but from transition labels to frequencies and the first constraint should be replaced as indicated below.

Definition 5. *Let (PN, M) be a marked Petri net with $PN = (P, T, F)$, L a set of labels, $lab \in T \not\to L$ a labeling function, and $fp \in L \not\to \mathbb{N}$ a frequency profile. The corresponding IP problem is:*

$$
\begin{array}{ll}
\min & \sum_{t \in T} f_t \\
\text{s.t.} & \sum_{t \in dom(lab) \mid lab(t)=l} f_t = fp(l) \quad \text{for all } l \in L \\
& f_{(t,p)} = f_t \quad \text{for all } (t,p) \in F \cap (T \times P) \\
& f_{(p,t)} = f_t \quad \text{for all } (p,t) \in F \cap (P \times T) \\
& M(p) + \sum_{t \in {}^\bullet p} f_{(t,p)} - \sum_{t \in p^\bullet} f_{(p,t)} \geq 0 \quad \text{for all } p \in P \\
& f_t \geq 0 \quad \text{for all } t \in T \\
& f_t \quad \text{integer for all } t \in T \\
& f_{(x,y)} \quad \text{integer for all } (x,y) \in F
\end{array}
$$

All results given in Section 3 can be extended to labeled Petri nets.

Note that definitions 4 and 5 can be combined. These extensions show that the formulation in terms of an LP/IP problem is easy to refine or extend.

6 Application in the context of SAP

The problem addressed in this paper applies to a wide variety of systems. However, the first time we were confronted with this phenomenon was when we started to apply process mining in the context of SAP R/3 [19, 24]. Given the widespread use of SAP, this has been the main motivation for the research reported in this paper. Based on a detailed analysis of the various SAP logs we discovered that there is no event log that allows for the type of log as shown in Table 1 [16]. There are two reasons why we have been unable to obtain references to case identifiers in SAP R/3. First of all, most logs only cover a small part of the SAP system, e.g., just the workflow module. Second, the logging facilities in SAP R/3 at a system-wide scope can be linked to transaction codes but not to individual cases.

This section will show that the approach described in this paper can be applied in the context of SAP R/3. We will show this in two steps. First, we show that the SAP logs allow for the discovery of a frequency profile *fp*. Second, we show that it is possible to obtain predefined process models (i.e., models of a descriptive or prescriptive nature) and map them onto Petri nets.

6.1 Obtaining a frequency profile in SAP

If we look at a logging facility in SAP R/3 with a system-wide scope, then the so-called *transaction monitor*[5] is the most obvious candidate to start. Every transaction that is executed is stored in the transaction monitor together with some basic information as is shown in Figure 8. Transaction codes can be linked to concrete activities and also information such a timestamp, originator, etc. are supplied. As indicated, there is no way to link transactions in the transaction monitor to cases. Therefore, classical process mining techniques do not apply. Fortunately, it is possible to obtain a frequency profile as shown in Figure 8. The first column on the right gives the transaction code (*Tcode*) and the second column gives the frequency (*Dialog steps*). As shown it is possible to refine the frequency into a frequency for every user (see smaller window).

Instead of directly using the ST03 transaction monitor, one can also use the Reverse Business Engineer (RBE). RBE is a tool for analyzing run-time SAP R/3 data. RBE is based on transaction frequencies and provides a more convenient way to obtain the information needed.

We also tried to use a completely different approach using the so-called *document flows*. SAP R/3 contains thousands of tables and an activity in some process often generates a record in a specific table. The problem is that these tables are linked and a-priori knowledge about the relations between these tables is needed to link the addition of a record to a concrete case. For example, when a purchase requisition is entered into SAP R/3 (via transaction code ME51), a new record is added to the purchase requisition table EBAN. The purchase requisition is uniquely identified by the purchase requisition number (BANFN). However, if for the same case a purchase order is created (via transaction code ME21), this purchase order results in the addition of a record in the purchasing table EKKO without a link to the purchase requisition

[5] The transaction monitor can be accessed via transaction code ST03.

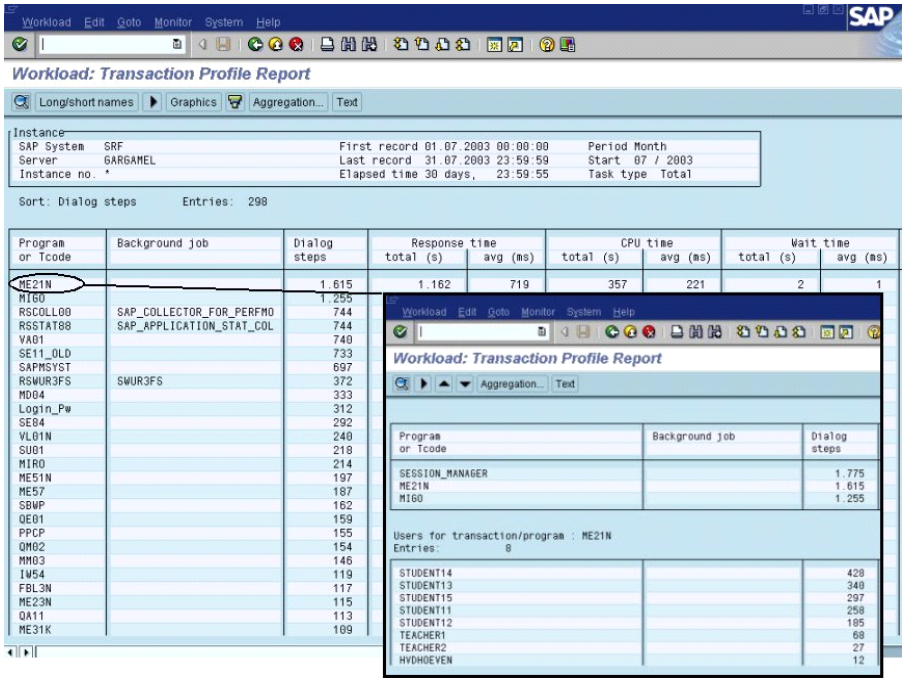


Fig. 8. A screenshot of the SAP R/3 transaction monitor (ST03).

number (BANFN). However, the record in the EBAN will get a pointer to the corresponding record in the EKKO table. An approach based on document flows requires knowledge of the underlying database. Therefore, it can only be supported for specific processes [16]. In fact, the ARIS PPM tool [20] of IDS Scheer provides a kind of process mining for some of the (hard-coded) SAP processes.

To summarize: it is possible to derive the transaction frequencies for *fp* but there is no way to link transactions to cases in a generic manner.

6.2 Obtaining a process model in SAP

The approach presented in this paper not only requires a frequency profile *fp*, it also needs an explicit process model *PN* expressed in terms of a Petri net. Fortunately, SAP has a comprehensive reference model including more than 4000 entity types and more than 1000 business processes and inter-organizational business scenarios [9, 24]. These models describe the functionality of SAP and can be used to understand and/or configure the system. Given the nature of this paper, we focus on the reference models expressed in the so-called Event-driven

Process Chains (EPCs) [23, 24]. Figure 9 shows a screenshot of ARIS showing a fragment of a reference model.

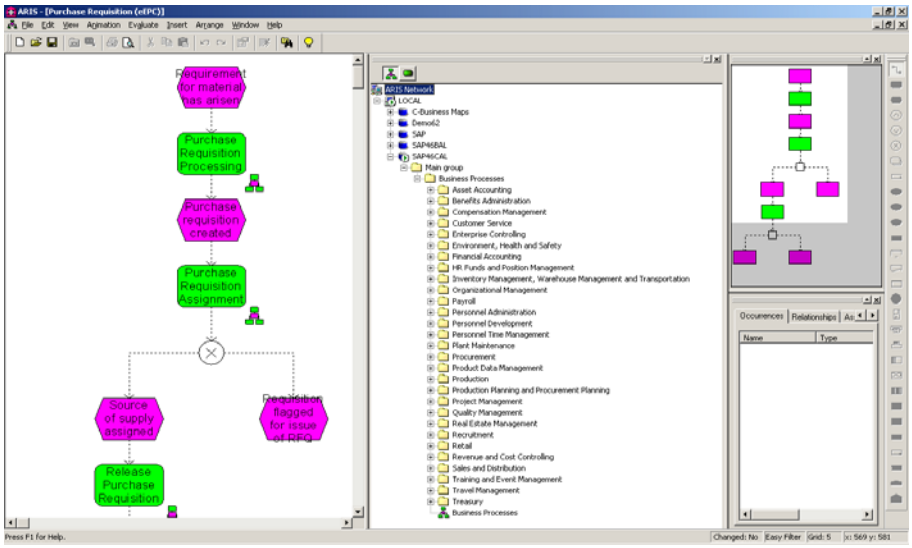


Fig. 9. A screenshot of an SAP reference model in ARIS for mySAP. The purchase requisition EPC is shown on the left and right half is used to navigate this EPC and other SAP reference models.

An EPC consists of three main elements. Combined, these elements define the flow of a business process as a chain of events. The elements used are:

- *Functions*, which are the basic building blocks. A function corresponds to an activity (task, process step) which needs to be executed. A function is drawn as a box with rounded corners.
- *Events*, which describe the situation before and/or after a function is executed. Functions are linked by events. An event may correspond to the position of one function and act as a precondition of another function. Events are drawn as hexagons.
- *Connectors*, which can be used to connect functions and events. This way, the flow of control is specified. There are three types of connectors: $\wedge$ (and), $\times$ (xor) and $\vee$ (or). Connectors are drawn as circles, showing the type in the center of the circle.

Functions, events and connectors can be connected with edges in such a way that (i) events have at most one incoming edge and at most one outgoing edge, but at least one incident edge (i.e. an incoming or an outgoing edge), (ii) functions have precisely one incoming edge

and precisely one outgoing edge, (iii) connectors have either one incoming edge and multiple outgoing edges, or multiple incoming edges and one outgoing edge, and (iv) in every path, functions and events alternate.

Figure 9 shows part of a bigger EPC. The left window shows four events, three functions, and one connector. The connector is an xor-split (denoted by the $\times$ symbol). The three functions are non-atomic, i.e., they can be further decomposed. There are several approaches to map an EPC onto a Petri net. In this paper we will not elaborate on this, because this is far from trivial and, depending on the EPC, this can only be partly automated. Instead we refer to only a few of the many papers on this topic [2, 10, 15, 25]. Moreover, we would like to emphasize that in the context of the ProM framework there is a plug-in to translate an EPC into a Petri net [15].

Functions in the SAP reference model can be linked to the SAP transaction codes. For example, ARIS for mySAP shows the transaction codes of functions that can be directly linked to SAP. This mapping is partial, but our approach does not require a full mapping. (Note that $fp \in T \not\rightarrow \mathbb{N}$ is a partial function.)

Using the SAP reference model and the transaction monitor (or RBE) we can deduce in a number of steps the frequency profile fp and process model PN. However, we cannot deduce the initial marking without more knowledge of the SAP system. Fortunately, as shown in Section 5, there are ways to work around the problem. By observing the process over a longer period of time and allowing for a noise level, the initial marking becomes of less importance.

6.3 SAP example

Let us consider the fragment of the invoice verification process to illustrate the overall approach in SAP. Figure 10 shows a fragment of the process in terms of an EPC. We focus on the four functions in this EPC fragment. For convenience these functions have been renamed to a, b, c, and d. Using the transaction monitor (ST03) or RBE we can obtain the frequencies of the corresponding transactions. The upper half of the diagram refers to the information obtained from SAP and ARIS for mySAP. The lower half shows the translation into the notations used in this paper, i.e., the frequency profile fp and process model PN. Both can be translated into an IP problem using Definition 4, i.e., $fp(a) = 56$, $fp(b) = 876$, $fp(c) = 323$, $fp(d) = 1278$,

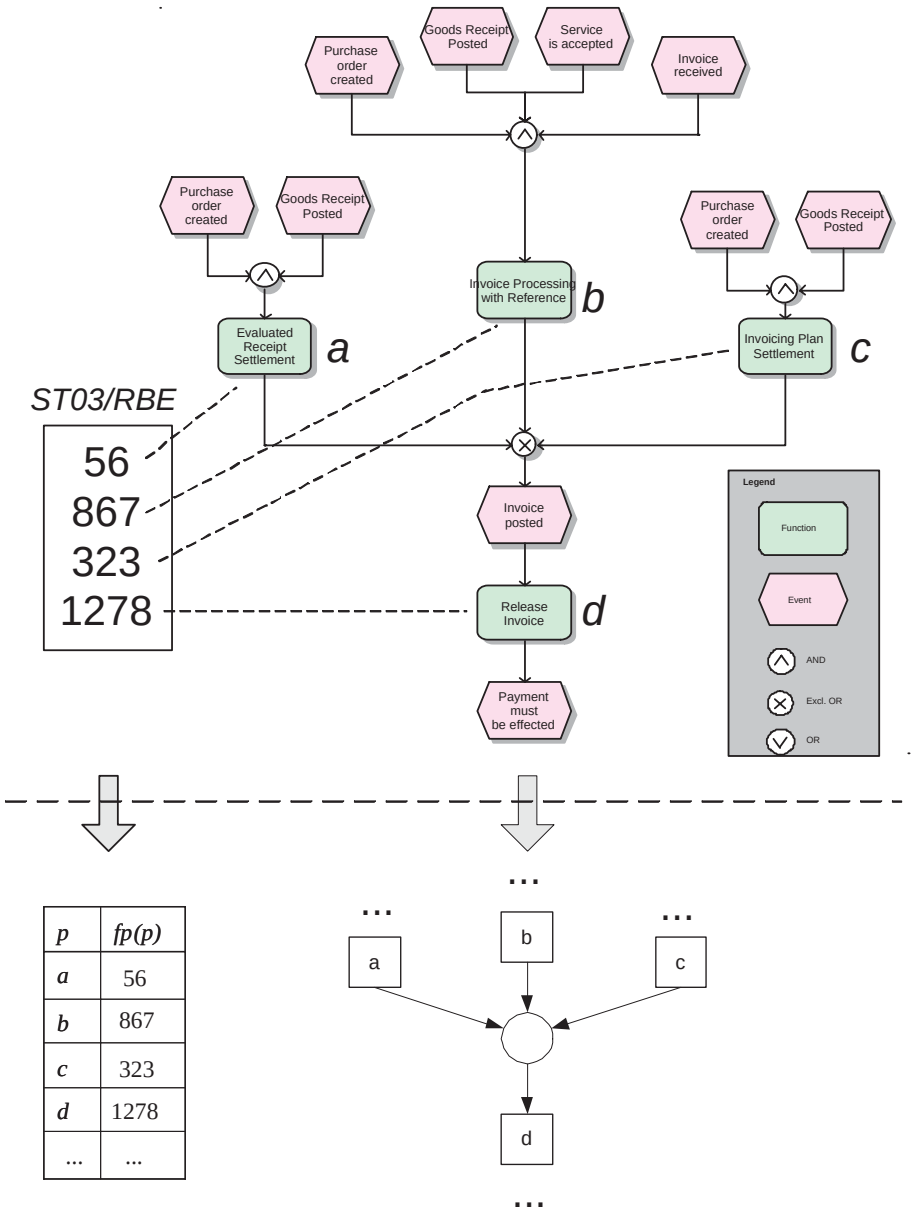


Fig. 10. The application of the approach in the context of SAP.

and PN as as shown in Figure 10. The initial marking of the place connecting a, b, c, and d can be assumed to be zero (of some better guess). If $\alpha = 0.05$, then $IP(PN, M, fp)$ has a solution because $fp(a) + fp(b) + fp(c) = 1246 \geq (1 - 0.05)fp(d) = 1214.1$. This suggests that the reference model and the frequency profile match. However, if $fp(d)$ would have been substantially larger, e.g., 1500, the IP problem would not have had a solution thus indicating that both do not match.

7 Conclusion

Inspired by a problem encountered when applying process mining techniques to SAP transaction logs, the paper tackled the problem of checking whether a Petri net and a frequency profile match. An IP problem was proposed to efficiently implement a necessary but not sufficient condition. The approach allows for extensions not possible in the traditional linear algebraic approaches [27, 11, 34]. Clearly, the application is not limited to SAP transaction logs but is applicable in any situation where processes are only monitored at an aggregate level, i.e., frequency profiles rather than event traces.

Future research is aiming at a better characterization of the class of nets for which $IP(PN, M, fp)$ has a solution if and only if $match(PN, M, fp)$. In this paper, it was shown that for acyclic nets, marked graphs, and state machines this is the case. It seems that the characterizations given in [17] and the class of ST-nets (nets obtained by composing marked graphs and state machines) are a good starting point for a beter understanding when solutions of the IP problem are actually realizable.

Acknowledgments. The author would like to thank Eric Verbeek of proof-reading an early version of the paper and Monique Jansen-Vullers and Michael Rosemann for their joint work on mining SAP and configurable process models which uncovered the problem addressed in this paper. Moreover, Martijn van Giessel contributed with his Master thesis on mining SAP logs.

References

[1] W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.

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

[3] W.M.P. van der Aalst and M. Song. Mining Social Networks: Uncovering Interaction Patterns in Business Processes. In J. Desel, B. Pernici, and M. Weske, editors, *International Conference on Business Process Management (BPM 2004)*, volume 3080 of *Lecture Notes in Computer Science*, pages 244–260. Springer-Verlag, Berlin, 2004.

[4] 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.

[5] 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*, 16(9):1128–1142, 2004.

[6] 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.

[7] J. Becker, M. Kugeler, and M. Rosemann, editors. *Process Management: A Guide for the Design of Business Processes*. Springer-Verlag, Berlin, 2003.

[8] J.E. Cook and A.L. Wolf. Discovering Models of Software Processes from Event-Based Data. *ACM Transactions on Software Engineering and Methodology*, 7(3):215–249, 1998.

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

[10] J. Dehnert and W.M.P. van der Aalst. Bridging the Gap Between Business Models and Workflow Specifications. *International Journal of Cooperative Information Systems*, 13(3):289–332, 2004.

[11] J. Desel. Basic Linear Algebraic Techniques of Place/Transition Nets. In W. Reisig and G. Rozenberg, editors, *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*, pages 257–308. Springer-Verlag, Berlin, 1998.

[12] J. Desel and J. Esparza. *Free Choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, UK, 1995.

[13] J. Desel, W. Reisig, and G. Rozenberg, editors. *Lectures on Concurrency and Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2004.

[14] B. van Dongen, A.K. Alves de Medeiros, H.M.W. Verbeek, A.J.M.M. Weijters, and W.M.P. van der Aalst. The ProM framework: A New Era in Process Mining Tool Support. In G. Ciardo and P. Darondeau,

editors, *Application and Theory of Petri Nets 2005*, volume 3536 of *Lecture Notes in Computer Science*, pages 444–454. Springer-Verlag, Berlin, 2005.

[15] B.F. van Dongen, W.M.P. van der Aalst, and H.M.W. Verbeek. Verification of EPCs: Using Reduction Rules and Petri Nets. In O. Pastor and J. Falcao e Cunha, editors, *Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAiSE'05)*, volume 3520 of *Lecture Notes in Computer Science*, pages 372–386. Springer-Verlag, Berlin, 2005.

[16] M. van Giessel. Process Mining in SAP R/3. Master's thesis, Eindhoven University of Technology, Eindhoven, 2004.

[17] K. van Hee, N. Sidorova, and M. Voorhoeve. Soundness and Separability of Workflow Nets in the Stepwise Refinement Approach. In W.M.P. van der Aalst and E. Best, editors, *Application and Theory of Petri Nets 2003*, volume 2679 of *Lecture Notes in Computer Science*, pages 335–354. Springer-Verlag, Berlin, 2003.

[18] J. Herbst. A Machine Learning Approach to Workflow Management. In *Proceedings 11th European Conference on Machine Learning*, volume 1810 of *Lecture Notes in Computer Science*, pages 183–194. Springer-Verlag, Berlin, 2000.

[19] J. Hernandez. The SAP R/3 Handbook, 1997.

[20] IDS Scheer. ARIS Process Performance Manager (ARIS PPM): Measure, Analyze and Optimize Your Business Process Performance (whitepaper). IDS Scheer, Saarbruecken, Gemany, http://www.ids-scheer.com, 2002.

[21] M.H. Jansen-Vullers, W.M.P. van der Aalst, and M. Rosemann. Mining Configurable Enterprise Information Systems. BPM Center Report BPM-05-09, BPMcenter.org, 2005. (Accepted for publication in *Data and Knowledge Engineering.*).

[22] N. Karmarkar. A New Polynomial-Time Algorithm for Linear Programming. *Combinatorica*, 4(4):373–396, 1984.

[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] G. Keller and T. Teufel. *SAP R/3 Process Oriented Implementation.* Addison-Wesley, Reading MA, 1998.

[25] E. Kindler. On the Semantics of EPCs: A Framework for Resolving the Vicious Circle. In J. Desel, B. Pernici, and M. Weske, editors, *International Conference on Business Process Management (BPM 2004)*, volume 3080 of *Lecture Notes in Computer Science*, pages 82–97. Springer-Verlag, Berlin, 2004.

[26] M. zur Mühlen and M. Rosemann. Workflow-based Process Monitoring and Controlling - Technical and Organizational Issues. In R. Sprague, editor, *Proceedings of the 33rd Hawaii International Conference on System Science (HICSS-33)*, pages 1–10. IEEE Computer Society Press, Los Alamitos, California, 2000.

[27] T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

[28] W. Reisig. *Petri Nets: An Introduction*, volume 4 of *EATCS Monographs in Theoretical Computer Science*. Springer-Verlag, Berlin, 1985.

[29] W. Reisig and G. Rozenberg, editors. *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1998.

[30] M. Rosemann. Application Reference Models and Building Blocks for Management and Control (ERP Systems). In P. Bernus, L. Nemes, and G. Schmidt, editors, *Handbook on Enterprise Architecture*, pages 596–616. Springer-Verlag, Berlin, 2003.

[31] 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. (Accepted for publication in *Information Systems*.).

[32] M. Sayal, F. Casati, U. Dayal, and M.C. Shan. Business Process Cockpit. In *Proceedings of 28th International Conference on Very Large Data Bases (VLDB'02)*, pages 880–883. Morgan Kaufmann, 2002.

[33] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, New York, 1998.

[34] M. Silva, E. Teruel, and J.M. Colom. Linear Algebraic and Linear Programming Techniques for the Analysis of Place/Transition Net Systems. In W. Reisig and G. Rozenberg, editors, *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*, pages 309–373. Springer-Verlag, Berlin, 1998.

[35] A.J.M.M. Weijters and W.M.P. van der Aalst. Rediscovering Workflow Models from Event-Based Data using Little Thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162, 2003.

[36] L.A. Wolsey. *Integer Programming*. John Wiley & Sons, New York, 1998.