Available online at www.sciencedirect.com
ScienceDirect
Decision Support Systems 42 (2006) 1843–1859
Decision Support Systems
www.elsevier.com/locate/dss

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

Received 7 January 2005; received in revised form 23 March 2006; accepted 26 March 2006
Available online 15 May 2006

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 labeled 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 models are *descriptive* or *prescriptive*, i.e., they are used to describe a process or they are used to control or guide the system. Typical examples are the so-called reference models in the context of Enterprise Resource Planning (ERP) systems like SAP [15]. The SAP reference models are expressed in terms of so-called Event-driven Process Chains (EPCs) [14] describing how people should/could use the SAP R/3 system. Similar models are used in the workflow domain [25], and also in many other domains ranging from flexible manufacturing and telecommunication to operating systems and software components [17]. 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 [28] to SAP R/3 we were 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*

E-mail address: w.m.p.v.d.aalst@tm.tue.nl.

0167-9236/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.dss.2006.03.013

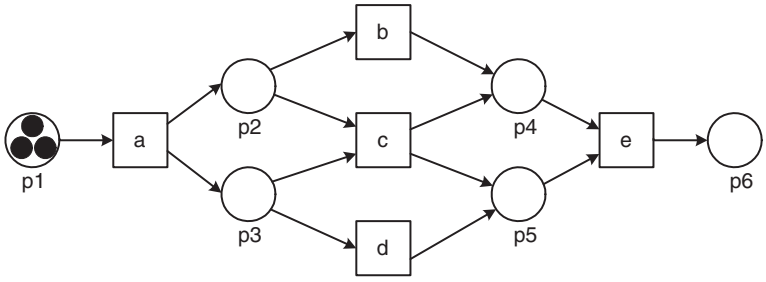

Fig. 1. A Petri net.

(i.e., the EPC) and the observed behavior (i.e., the transaction frequencies) match?

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 [18,19] and a *frequency profile* which is a partial function indicating how many times certain transitions fired. Consider for example the marked Petri net shown Fig. 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) [23,35]. 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 a 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 [28–30,34]. The goal of process mining is to extract information about processes from transaction logs [28]. 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 perspective and (3) the case perspective. The *process perspective* focuses on the control-

[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 these from this paper. However, in our process mining tools event types were taken into account.