Computers in Industry 70 (2015) 194–207

Contents lists available at ScienceDirect

Computers in Industry

journal homepage: www.elsevier.com/locate/compind

Using logical decision trees to discover the cause of process delays from event logs

Diogo R. Ferreira [a,*], Evgeniy Vasilyev [b]

[a] Instituto Superior Técnico, University of Lisbon, Portugal
[b] GREE Inc., Tokyo, Japan

ARTICLE INFO

Article history:
Received 26 August 2014
Received in revised form 16 February 2015
Accepted 24 February 2015
Available online 21 March 2015

Keywords:
Process mining
Performance analysis
Logical decision trees
Regression trees
Root cause analysis

ABSTRACT

In real-world business processes it is often difficult to explain why some process instances take longer than usual to complete. With process mining techniques, it is possible to do an *a posteriori* analysis of a large number of process instances and detect the occurrence of delays, but discovering the actual cause of such delays is a different problem. For example, it may be the case that when a certain activity is performed or a certain user (or combination of users) participates in the process, the process suffers a delay. In this work, we show that it is possible to retrieve possible causes of delay based on the information recorded in an event log. The approach consists in translating the event log into a logical representation, and then applying decision tree induction to classify process instances according to duration. Besides splitting those instances into several subsets, each path in the tree yields a rule that explains why a given subset has an average duration that is higher or lower than other subsets of instances. The approach is applied in two case studies involving real-world event logs, where it succeeds in discovering meaningful causes of delay, some of which having been pointed out by domain experts.

1. Introduction

Why does a business process become delayed? There may be many possible reasons, but here we focus on causes that can be inferred from run-time data about the past executions of the process. Such data is usually recorded in the form of an event log [1,2], which can be analyzed from a number of different perspectives, namely the control-flow perspective, the organizational perspective, and the performance perspective:

- In the control-flow perspective, the goal is to extract a process model from the sequence of tasks recorded in the event log (examples of techniques are the α-algorithm [1], the heuristics miner [3], and the genetic miner [4]).
- In the organizational perspective, there are techniques to analyze the interaction between process participants and to extract a social network from the event log [5,6].
- In the performance perspective, there are ways to calculate performance indicators and to detect bottlenecks based on the timestamp of events [7,8].

A practical case study that includes these three different perspectives can be found in [9]. Here, we focus mainly on the performance perspective, but we also seek causes of delay that are possibly related to the control-flow and to the organizational perspectives. Since an event log contains information about the tasks that have been performed and the users who have performed them [10], it should be possible to identify causes of delay such as:

- when a certain activity is executed;
- when a certain user participates in the process;
- when a certain user performs a certain activity;
- when a certain activity follows another activity;
- when a specific group of users participate in the process, etc.

The causes that can be considered are only limited by the type of data recorded in the event log. If the event log includes information about the data perspective (as in e.g. [11]), it may be possible to find causes based on data properties as well.

As a baseline, we assume that the event log has some minimal information about each event, namely: a case id (i.e. the process

* Corresponding author at: Instituto Superior Técnico, Campus do Taguspark, Avenida Prof. Dr. Cavaco Silva, 2744-016 Porto Salvo, Portugal.
Tel.: +351 21 423 35 52.
E-mail addresses: diogo.ferreira@tecnico.ulisboa.pt (D.R. Ferreira), vasiliev.evgeni@gmail.com (E. Vasilyev).

http://dx.doi.org/10.1016/j.compind.2015.02.009
0166-3615/

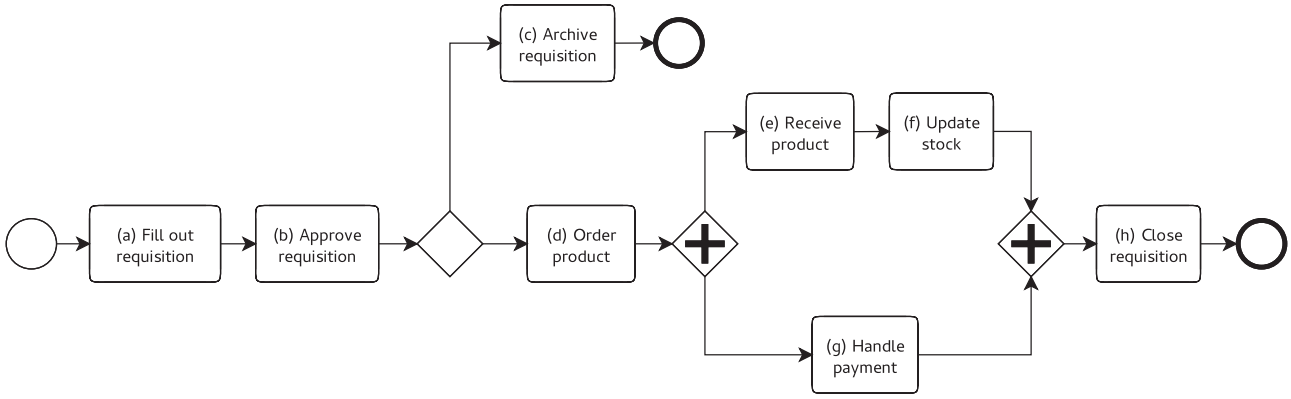

Fig. 1. BPMN model of a simple purchase process.

instance identifier), a task, a user, and a timestamp. The analysis is based on the total time that each process instance takes to complete. This will be referred to as *duration* and it is measured from the timestamp of the first event to the timestamp of the last event recorded for a given process instance.[1] An instance is said to be delayed if it takes longer than usual to complete, meaning that it takes longer than the average duration of all instances recorded in the event log.

We transform the event log into a logic representation, so that it becomes possible to use inductive reasoning to find the cause of delays. In particular, we capture the information in the event log as a set of first-order logical predicates, and we feed this knowledge base to a decision tree learner which induces a logical decision tree [12,13]. This decision tree splits the process instances into a set of classes according to their duration. Each class is defined by a specific rule, which is expressed as a conjunction of predicates (e.g. "the instances where task a is performed and user u_1 participates have an average duration of 252.4 h"). This rule can be interpreted as the reason why that group of instances has such duration. For those classes of instances which take longer to complete, we take the rule as an indication of a possible reason of delay.

The main challenge in this approach is not in inducing the logical decision tree (an algorithm for that purpose, known as TILDE [12], already exists), but in defining the predicates that will appear in such tree. In essence, those are the predicates that will be used to express the cause of delays. The approach we present here is based on two layers of predicates:

- The first layer comprises a small set of *base predicates* that are used to create a logical representation of events as they have been recorded in the event log.
- The second layer consists in a set of *rules* that define new predicates in terms of the base predicates. These new predicates are meant to represent high-level concepts such as the flow between tasks or the handover of work between users, and they can be automatically inferred from the logical representation of the event log.

In real-world applications, it is possible for the analyst to use custom predicates to focus on domain-specific issues or to analyze specific causes of delay. Therefore, the main goal of this work is two-fold: on one hand, we aim to show how effective the use of logical decision trees can be in discovering the causes of delay and, on the other hand, we intend to illustrate how the analyst may use of custom predicates to analyze specific causes of delay. These goals are better achieved by resorting to concrete examples, so we present two case-study applications involving real-world event logs. The approach itself is also presented by means of a simple example.

The structure of the paper is as follows: Section 2 develops the base predicates and rules that lay the foundation for a logical representation of the event log. Section 3 discusses the induction of logical decision trees and the use of regression to support continuous variables, as is the case with duration. A first example of a logical decision tree which captures a cause of delay is also introduced in Section 3. Then Section 4 presents two case studies using real-world event logs. The second case study illustrates the use of custom predicates. Finally, the paper ends with an overview of some related works.

2. Event log representation

Consider a simple purchase process which can be described as follows:

> An employee fills out a requisition form and sends it to a manager for approval. If the requisition is not approved, it is archived and the process ends. Otherwise, the requisition is approved, and the requested product is ordered from a supplier. Then two things will happen in parallel: the warehouse receives the product and updates the stock, and the accounting department takes care of payment to the supplier. When these tasks are complete, the requisition is closed, and the process ends.

A model for this process, using the BPMN language,[2] is shown in Fig. 1. Each task in this process is assigned to some user (e.g. u_1, u_2, etc.). There may be several users who are able to perform the same task, but only one user will be selected to perform the task for a given process instance. On the other hand, each user may be assigned multiple tasks, either from the same process instance or from different process instances. The result is that each user has its own list of work assignments (a.k.a. "task list", "work list", or "to-do list"), and it is assumed that users carry out their assignments one at a time.

Table 1 shows an excerpt of a sample event log that can be generated from such process. This is similar to the event logs that are typically used for process mining (see e.g. [14,15]). In the excerpt of Table 1 there are only three process instances, but it is possible to recognize several features of the process. For example, case 1 has a trace in the form *abdefgh*, while case 2 has a trace in the form *abdgefh*, which comes as a result of the parallelism between *g* and the branch *ef*. Also, in cases 1 and 2 the requisition is approved,

[1] The concept of duration will be discussed in more detail in Section 2.2.

[2] http://www.omg.org/spec/BPMN/2.0/.