Information Systems 71 (2017) 211–227

Contents lists available at ScienceDirect

Information Systems

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

Merging event logs: Combining granularity levels for process flow analysis

Lihi Raichelson[a,*], Pnina Soffer[a], Eric Verbeek[b]

[a] Department of Information Systems, University of Haifa, Haifa 31905, Israel
[b] Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands

ARTICLE INFO

Article history:
Received 13 October 2016
Revised 10 August 2017
Accepted 27 August 2017
Available online 30 August 2017

Keywords:
Process mining
Multiple instances
Merging logs
End-to-end process flow
Abstraction level

ABSTRACT

Process mining techniques enable the discovery and analysis of business processes and the identification of opportunities for improvement. Processes often comprise separately managed procedures documented in separate log files, which are impossible to mine in an integrative manner as the complete end-to-end process flow is obscure. These procedures can have simple (one-to-one) or complex (many-to-one or many-to-many) relationships among them. When complex relationships exist, typically different granularity levels are involved. In this paper, we present a merging algorithm that results in a comprehensive merged log that can handle all kinds of relationships between the procedures. Addressing differences in the granularity levels, it offers two views of the end-to-end process: a case view and an instance view. This enables the identification of process flow problems that could not be detected by previous techniques.

The unified log can be used by process mining techniques to identify flow problems, particularly at the point of integration between the processes under consideration. The procedure proposed in this paper has been implemented and evaluated using both synthetic and real-life logs.

1. Introduction

Process mining has emerged in the past few years as a means for improving overall business performance through discovery and analysis of the actual business process [2]. Process mining allows business processes to be visually mapped from inception through execution to termination, thereby facilitating the identification of the process problems and their causes [1]. This is done based on an event log, depicting all the events recorded during process execution; such logs are largely available in information systems that support business processes. The work described in this paper has emerged from the need to identify flow and performance problems that can only be detected when a full, end-to-end business process is viewed through its different abstraction levels (i.e., main and sub- processes). For process mining analysis, a log that captures the full process is needed. However, often in organizational settings different parts of processes are managed by different information systems, each generating its own log. Our primary concern, then, is the need to connect logs recorded in different systems that are related to a single, full, end-to-end business process in a manner that would reflect different abstraction (or granularity) levels. As our work progressed it soon became evident that three additional issues required our attention: (a) the complexity of relationships between the different process parts; (b) the possible lack of equivalence in identifying log items that pertain to a single occurrence of the full, end-to-end business process; and (c) the burden that may be caused by irrelevant data in log entries. This paper thus delineates a solution for merging log files of processes distributed over different systems in an automated manner. The proposed solution reflects different granularity levels and is applicable to all possible types of inter-log relationships, matching log items that are not necessarily equivalent in their identification method, and which may be confusing due to additional data that is irrelevant for the merging (typically as free text). Challenges presented by the existence of disparate logs must be overcome and a single, unified log produced from which it is possible to trace the entire end-to-end business process at its different abstraction levels before performance analysis be conducted and flow anomalies be detected. Achieving these processes was the secondary goal of our work.

In the next section, we describe the different granularity levels which we delineate in detail in an example. In the third section, we explain the premises for our merging approach, addressing both our major and secondary concerns while in the fourth section we evaluate the effectiveness of this approach. The fifth section is devoted to related works found in the literature while

* Corresponding author.
E-mail addresses: lihiraos@gmail.com (L. Raichelson), Spnina@is.haifa.ac.il (P. Soffer), h.m.w.verbeek@tue.nl (E. Verbeek).

http://dx.doi.org/10.1016/j.is.2017.08.010
0306-4379/

the last two sections are devoted to a brief discussion and conclusion.

2. Combining different granularity levels

An obstacle to the effectiveness of process mining is that operations are carried out through different systems and recorded in disparate log files so that they cannot be mined in an integrative manner. To date, existing approaches [11] propose automated techniques that facilitate the reconstruction of the full, end-to-end process based on a single (merged) log file representing events recorded in both the instigating (main) process and the responding (sub) processes. However, these techniques are limited in scope to only one granularity level view whereas at least two granularity levels are relevant in such complex processes: a high level representing the view of the main process and a low level representing the view of the subprocesses. Certain problems in the business process can only be detected in a combined view of the two granularity levels. In this section, we explore which problems require the combined view of the different granularity levels. We focus on two issues that are commonly addressed when investigating business processes: business performance and flow analysis. In order to discuss these, we first explain granularity levels, show an example to which we refer throughout this work, and discuss relationships between the process and the recorded events.

2.1. Understanding granularity levels

Processes can be viewed at a variety of granularity levels, specifically essential in processes that handle composite objects, so they can be traced at the composite level or at the level of its components. This need is amplified where several parties are involved in different parts of a single business process. For example, several departments order equipment from the company's warehouse, and there is some overlap between ordered equipment among the different departments. At the warehouse, items are picked collectively and then distributed individually to respective departments. This process can be viewed from two different perspectives: one concerning the order (composite object), placed by each department with all of its individual items, and the other relating to the flow of each single item (component) ordered by a department. We refer to the first perspective as the high level *case view* and to the second as the low level *instance view*.

It is possible to model the business process based on process mining from three perspectives: that of the case view (high level, emphasizing the main process), that of the instance view (low level, emphasizing the subprocess), and that of a combined view showing relationships between the two. When considering a full, end-to-end process involving multiple occurrences of a subprocess, a coarse-granularity view of the end-to-end flow at the case level provides limited analysis capabilities of problems that occur at the instance level and vice versa. In addition, some problems can only be detected by considering both views simultaneously, as we show below. To the best of our knowledge, to date, there are no studies that consider both high and low granularity levels of the end-to-end process flow together.

2.2. Motivating example

To showcase the full extent of our approach, throughout this paper we use a running example based on a plausible scenario. The running example concerns a purchase ordering process, where different departments directly order office supplies through an ordering system. The warehouse receives the deliveries from the suppliers with aggregated amounts of goods ordered by all departments, registers them, and then distributes them to the original ordering departments. We consider the ordering process as the main process and the delivery handling as a subprocess. Table 1 shows a simplified log of the main process while Table 2 shows a simplified log of the warehouse subprocess. The double line in each table serves to separate entries pertaining to a single "case" (referring to orders in Table 1 and deliveries in Table 2).

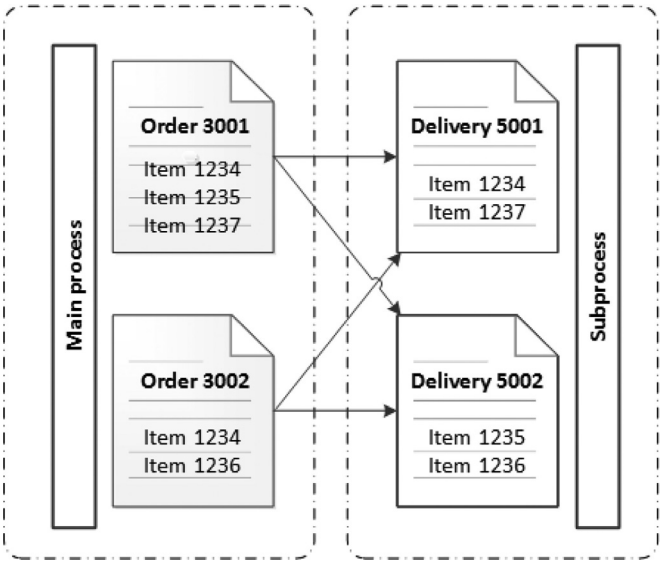

Fig. 1. Illustration of many-to-many relationships between a main process and a subprocess of the running example.

As illustrated in Fig. 1, both processes employ multiple instance procedures, where the lower-level instances in both processes refer to ordered items. However, the grouping of items to cases is different for the main and the subprocess. In the main process, the grouping is by order (serving as case-ID in the log), and in the subprocess the grouping is by delivery (case-ID), where items ordered by different departments are supplied together. Furthermore, there is no explicit connection between the Case IDs of the two logs. As a result, multiple cases of the main process may correspond to multiple cases of the subprocess. To enable mining of the end-to-end process, the two logs must be merged in a manner that will overcome this difference in case-IDs, thus a choice must be made between two possible entities that would serve as the unique identifier for events in the unified log.

Consider two possible views of the process: one concerning the order (including all its items) and the other relating to the flow of each single item ordered by a department. We refer to the first one as the case view and to the second as the instance view. Should we support the case perspective when merging the logs, we would be finding for each case in the log of the main process (main log) its related cases in the log of the subprocess (*sub* log). In contrast, for a merged log supporting the instance perspective, each case of the sub log would be related to corresponding cases of the main log. It would be natural to assume that there is no substantial difference between the views. Yet, following each view solely might result in loss of information. In our example, item "1237" is not included in any of the orders. Following the end-to-end process with the case view would not reveal this problem. Hence, we claim that both views are imperative when analyzing the full, end-to-end process flow.

As an illustration, we manually generated an end-to-end case view and instance view of our running example, to enable the identification of different process flow problems. To demonstrate this need, Fig. 2 presents three mined models of the running example, using Fluxicon Disco (https://fluxicon.com/disco/) based on the manually merged logs.