Decision Support Systems 59 (2014) 93–107

Contents lists available at ScienceDirect

Decision Support Systems

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

Improving business process decision making based on past experience

Johny Ghattas, Pnina Soffer *, Mor Peleg

University of Haifa, Carmel Mountain, 31905 Haifa, Israel

ARTICLE INFO

Article history:
Received 5 August 2012
Received in revised form 9 August 2013
Accepted 26 October 2013
Available online 13 November 2013

Keywords:
Business process
Decision making
Context
Goal
Decision tree

ABSTRACT

Business processes entail a large number of decisions that affect their business performance. The criteria used in these decisions are not always formally specified and optimized. The paper develops a semi-automated approach that improves the business performance of processes by deriving decision criteria from the experience gained through past process executions. The premise that drives the approach is that it is possible to identify a process path that would yield best performance at a given context. The approach uses data mining techniques to identify the relationships between context, path decisions, and process outcomes, and derives decision rules from these relationships. It is evaluated using a simulation of a manufacturing process, whose results demonstrate the potential of improving the business performance through the rules generated by the approach.

1. Introduction

Organizations conduct their operations through business processes, designed to achieve their business goals. Business processes entail a variety of decisions, such as the selection of a path from several available ones, deciding on quantities, or resource assignment. These decisions affect the outcome of the process and the success of achieving its goal.

Attempting to maintain and improve their business performance, organizations employ various mechanisms to guide decision making in business processes. These include process models, procedures and regulations, and knowledge management systems. Still, many times decisions are based on application of personal knowledge, gained through experience. When no formal decision criteria are available, humans rely on their own sense-making and experience-based knowledge for decision making. Doing so, they typically relate to the specific situation (case properties like patient's age) and select the option they find most suitable for the situation and most likely to maximize the expected results of the process.

The results or outcomes of a process can be assessed in two main dimensions. First, a binary result indicating whether the process has achieved its 'hard' goal, namely, a state the process intends to achieve (e.g., ordered goods are supplied to the customer). Second, a result which can be evaluated on a scale indicating the extent to which business objectives have been achieved (e.g., time to delivery, quality level, costs). This dimension is sometimes referred to as 'soft' goals [28] or Key Performance Indicators (KPI). Both dimensions can be addressed as the business performance of the process.

Improving the business performance of processes has long been addressed. Process redesign initiatives [17] have been proposed mainly for increasing the efficiency, and to a lesser extent also for providing clear and effective decision criteria. However, such redesign typically relies on human creativity, using data analysis as indication of improvement opportunities.

This paper aims at developing a semi-automated approach that improves the business performance of processes by learning and deriving decision criteria formulated as decision rules from the experience gained through past process executions. These executions are specific instances of a defined process, hence they are termed process instances. Our premise is that to learn and improve process performance over time, three process elements need to be tied together. First, what has been done in past process instances, namely, the actual paths that have been followed and decisions made within the activities. Second, we need to take into account the situations in which these executions have taken place. We generally address these situations as the context of each process instance [22]. Third, evaluate the outcomes or business performance achieved in these executions, considering the goals of the business process. Tying these three elements together should enable us to identify decisions that lead to a high performance at a given context, imitating the way a human learns from experience. Decision rules derived accordingly are expected to improve this performance.

We note that approaches that support automated or semi-automated learning from past experience have been proposed in various areas. In the area of control systems, a closed loop model [21] provides feedback about errors for the system to be adapted accordingly. However, our aim is to improve performance in general, not focusing on errors. Case-Based Reasoning (CBR) approaches (e.g., [4]) refer to past cases that bear similarity to a current case, so applied courses of action can be reused. In contrast to the approach taken here, CBR emphasizes case

* Corresponding author. Tel.: +972 48288506; fax: +972 48288522.
E-mail addresses: john@smart-path.com (J. Ghattas), spnina@is.haifa.ac.il (P. Soffer), morpeleg@is.haifa.ac.il (M. Peleg).

0167-9236/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.dss.2013.10.009

similarity as a retrieval criterion, rather than the achieved outcome. In addition, CBR retrieves specific relevant cases, while we aim to aggregate knowledge from all past cases in the form of decision rules. Decision techniques, such as Bayesian Networks [7], offer methods for extracting knowledge from data. They are capable of addressing incomplete data, learning causal relationships, and combining the use of a-priori domain knowledge. However, to become applicable for business process learning, all these approaches need to be operationalized in this specific context. To the best of our knowledge, an approach that specifically targets improvement of business process decisions, considering the decisions as well as their context and outcomes, is still missing.

Our approach is grounded as follows. Since the basic intention is to discover knowledge from data, we operate in the general area of data mining, long used for knowledge discovery [10]. Specifically dealing with processes, we turn to process mining. One of the challenges identified for process mining in [3] is to make predictions and recommendations for running process instances based on historical data. This challenge is addressed in this paper.

In what follows, Section 2 presents conceptual foundations, formalizing the notions required for addressing the three process elements discussed above. According to [3], log extraction should be driven by formal questions to support formal analysis. The formal conceptual foundation provides a basis for the data analysis performed later. Next, in Section 3, we develop a learning procedure that uses mining techniques to derive decision rules from past process instances. The procedure is evaluated by applying it to simulated data in Section 4. Simulation enables the generation of data representing a baseline set of process instances, as well as a manipulated set, where decision rules can be evaluated. In Section 5, we discuss the findings and their implications and then review related work in Section 6. Finally, conclusions and future research directions are provided in Section 7.

2. Conceptual foundations

This section discusses the three process elements required for learning from experience, namely, path, goal, and context, and provides a conceptual basis on which learning can build. We start by presenting the process view that underlies our analysis.

2.1. Processes and process paths

As a basic process view we use the Generic Process Model (GPM) [28,29], which is a formal framework based on Bunge's ontology [6], designed for process analysis. We use this view for several reasons. First, as opposed to many commonly used process modeling languages which are activity-based, GPM is state-based. Hence, it is capable of capturing a wider range of information about process execution than purely activity-based models. Second, GPM provides well-defined means for capturing the context of a process. Third, GPM addresses goals as an integral part of a process model, thus it facilitates the assessment of how successful a process instance is.

Consider, for example, a bottle production process, where a mixture of new and recycled raw material is prepared. An activity-based process model would present this as an activity; a state/transition model (e.g., Petri net) would present this as a transition after which the raw material mixture is ready. In both, the decision regarding the % of new and recycled material to be used would not be explicitly represented if this point of control has not a priori occurred as such. GPM presents the state that follows the material preparation, specifying the % of recycled material, thus it makes the related decision explicit, and enables process mining to treat the different % ranges as different pathways. In addition to making the decision explicit, GPM supports representation of context properties (e.g., bottle size) and goals (maximize quality, minimize cost). It is hence possible to mine past process instances and, for example, indicate that for bottles smaller than 200 cc, a mixture of over 30% recycled material yields severe quality problems, while for larger bottles up to 60% would be acceptable. Since using lower shares of recycled material in the mixture increases the quality in general, but also increases the total cost, such findings can promote both quality and cost-related goals.

The focus of attention in GPM is the *domain* where the process takes place. The process domain is represented by a set of *state variables*, whose values at a moment in time denote the *state* of the domain. A state can be *unstable*, in which case it will transform according to the *transition law* of the domain, or *stable*, namely, it will not change unless invoked by an event in the environment (*external event*). GPM views an enacted process as a set of state transitions in the process domain. Transitions occur either within the domain (due to its transition law), or by actions of the environment on the domain. A process ends when the domain reaches a desired (*goal*) state, which is stable and where no more changes occur.

A process model is an abstract representation of the process, defined as follows.

Definition 1. GPM process model.

A process model in a given domain is a tuple <I, G, L, E>, where:

I: the set of possible initial states — a subset of unstable states of the domain.
G: the goal set — a subset of the stable states reflecting stakeholders' objectives.
L: the transition law defined on the domain — specifies possible state transitions as mappings between sets of states.
E: a set of relevant external events that can or need to occur during the process.

Note that sets of states are usually specified as a partial assignment of values or as conditions that should hold on the values of part of the domain state variables.

While Definition 1 relates to a process model, our intention in this paper is to learn from past process instances. For this, we need to address the actual paths followed in these instances. While the law specifies lawful state transitions between sets of states, a path is a sequence of specific states, each denoted by the values assumed by all the state variables of the domain.

Definition 2. Path.

A process path is a sequence of states from an initial unstable state (in I) to a stable state where the process ends.

Addressing a path as a sequence of states (as opposed to the commonly used sequence of activities) enables capturing all the decisions that are taken, some relating to activity selection and some to decisions made within activities (e.g., what quantity to order). The latter cannot be captured by an activity-based process view, which only captures decisions that are associated with splits in the process model, namely, activity selection and ordering.

2.2. Process goals

For the purpose of process improvement and learning, it is vital to have defined goals. As evident from Definition 1, the goal is an integral part of GPM's process model. The goal, which is a set of stable states the process intends to achieve, is a hard goal, measured on a binary scale. In a given instance of the process, the final state on which the execution terminates is either in the goal set or not. When a process instance terminates on a stable state which is not in the goal set, we term this an exception state, and the set of all exception states is termed EX ($G \cap EX = \emptyset$). For example, in a sales process the customer might have received goods, paid with an invalid credit card, and cannot be located any more. In this case, the state of the process domain is stable, namely,