Decision Support Systems xxx (2013) xxx–xxx

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 xxxx

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 ($G \cap EX = \varnothing$). 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,

Please cite this article as: J. Ghattas, et al., Improving business process decision making based on past experience, Decis. Support Syst. (2013), http://dx.doi.org/10.1016/j.dss.2013.10.009

it cannot be changed by actions of the organization, but it is not in the goal set since such situations are outside the scope of the process model.

Definition 3. Termination state.

The final state t of a process instance is termed its termination state, t ∈G∪EX.

To improve the business performance of the process, we should seek to avoid exceptional situations or to minimize their occurrence over time.

In addition to the hard goal, organizations usually have business objectives that are associated with their business processes and supported to different extents by different process instances. These are not measured on a binary scale hence we refer to them as soft goals.

Definition 4. Soft goal.

A soft goal is a preference order relation of states in the goal set.

Soft goals are typically operationalized as performance indicators, used for assessing success level in meeting the business objectives. For example, in the above mentioned sales process such performance indicator could be time to delivery, so supplying goods in two days is higher ranked than in two weeks (although both are goal states). There might be several, sometimes conflicting, soft goals associated with a given process. When attempting to improve process performance, it is essential to understand what should be improved and how it can be measured. If several performance indicators exist, a clear objective needs to be defined for that endeavor, selecting from or combining the different indicators.

2.3. Process context

Context usually refers to aspects of a situation that affect the execution of a process. We need a practical definition of context to support predicting what path would yield best outcome for a given context. GPM makes a clear distinction between the process domain and its environment. Based on this distinction, context refers to all the environmental effects on a process instance. In particular, these include specific case properties, which are known when the process initiates (at a state in I), and external events which occur in the environment during the process and affect the state of the domain through mutual state variables.

Definition 5. Context.

The context C of a process instance is a tuple $C = <s_0, X>$, where $s_0 \in I$ is the initial state of the process and X is a set of external events that occur during execution.

Note that the events in X are represented in terms of the change they cause to the domain state, namely, to the values of state variables. These events can be included in E, but not necessarily, since E only holds expected external events. As experience is gained through process executions, E can be extended with event types that have occurred. However, we can never assume that all the possible external events are known in advance and are included in E. Also note that s_0 captures all the case properties that are known when the process initiates (e.g., the age of a patient, the type of the customer, etc.).

Having defined all the necessary elements, we can now define a process instance, which is a specific execution of the process.

Definition 6. Process instance.

An instance of a process whose model is <I, G, L, E> is a tuple <C, P, t>, where C is the context of the process instance, P is its path, and t its termination state.

Note that this definition relates to an already completed process instance, while a currently running instance has a partial path and no termination state.

The main premise taken in this paper is that for different contexts, different paths would lead to desirable performance outcomes. However, context information can be very rich, while the actual effect on the process is only caused by a subset of the state variables involved. To be able to analyze processes in their relevant context, we need to distinguish the relevant context properties from the irrelevant ones, and this is a hard task. Techniques for establishing such distinction have been proposed [8,11,12]. Still, in practice, this distinction usually relies on domain knowledge of the process designer or operator. Process designers can embed some context-related decisions into the process model (e.g., assign a certain path for business customers), but these will only address specific and most obviously influencing context variables. Process operators make decisions at runtime, based on the specific situation (or context) and their knowledge-based estimation of which action would increase the chances of successful process termination. The knowledge applied is usually tacit knowledge, building on their experience in a non systematic way, and not necessarily shared with other operators of the process. Furthermore, there is no evidence that process operators are even aware of the entire set of context variables and understand their effect on the process. It can hence be assumed that our knowledge of the process context and its effects is partial. Having a full knowledge of the context variables that affect the process, we could classify each process instance based on its context in a way which is meaningful for predicting its outcome. We term the criteria for such classification *ideal context categories*.

Definition 7. Ideal context category.

An ideal context category CC is a predicate over context variables such that for process instances where the predicate evaluates to TRUE, following similar paths implies reaching similar termination states. Let i,j be process instances, $CC(i) = CC(j) = True$, then if P_i is similar to P_j, $\Rightarrow$ t_i is similar to t_j.

Not having the full knowledge required for establishing the ideal context categories, we settle for approximate context categories, which rely on the partial context knowledge that is available. Like ideal context categories, approximate context categories are predicates over context state variables. However, similar paths taken by process instances of the same context categories do not imply similar termination states with certainty. Rather, they increase the likelihood of reaching similar termination states and the accuracy in which the termination state can be predicted. Since, as discussed, we only have partial context knowledge, herein we use the term context categories referring to *approximate* ones.

In summary, the GPM view provides a basis for learning and improving process decisions based on past experience. Using this view, the process instance definition sets requirements for the data needed for learning, namely, context data, path data including state variable values at each process step, and a termination state, denoting the performance outcomes of the process instance (in terms of hard and soft goals). Furthermore, the above discussion indicates that learning should use some knowledge about the context in the form of approximate context categories. The next section describes a learning procedure that builds on this model.

3. The learning procedure

This section presents the proposed approach for learning and improving decision making in processes, based on accumulated experience.

3.1. Overview of the learning approach

The proposed approach uses machine learning techniques for systematically deducing from accumulated experience what paths are more preferable for process instances of given context categories, increasing the likelihood of achieving high performance executions. To this end, a prerequisite is the existence of an experience base, namely,

Please cite this article as: J. Ghattas, et al., Improving business process decision making based on past experience, Decis. Support Syst. (2013), http://dx.doi.org/10.1016/j.dss.2013.10.009

recorded data of past executed process instances. Following the discussion in Section 2, this data should include for each process instance values of context state variables (initial state and external events), the specific sequence of states traversed through the actual path, and its termination state. We also assume that approximate context categories have been established, either based on domain knowledge or through the application of an automated context identification procedure (e.g., [11,12]). The proposed learning procedure is illustrated in Fig. 1.

Step 1 in the procedure is to define the objective to be achieved by learning. This objective relates to the soft goals associated with the process. However, there might be several soft goals, even contradicting (e.g., minimizing cost vs. maximizing product quality). The learning objective should be well defined and measureable for a given process instance. It may relate to a single selected soft goal (if one soft goal is identified as dominating other ones) or to a weighted combination of several soft goals. Since this is more common, we hereafter relate to a weighted soft goal combination as representing the learning objective.

In Step 2 of the procedure, the knowledge base is screened so process instances, for which the defined objective is irrelevant, are filtered out. As an example, consider a learning objective of time to delivery. This is not applicable for process instances where delivery has not been made, so these instances are filtered out and not used for further analysis. Note that this screening has the drawback that exceptional terminations, which could provide valuable information for learning, are not taken into account. On the other hand, it provides a common basis for evaluation of the process instances that are considered, and enables a fine-grained analysis based on the defined weighted soft-goal combination. As discussed below, special attention is given to avoid an increase in the occurrence of exceptional terminations.

Based on the premise that different context categories should entail different paths for achieving higher performance, the next three steps of analysis are performed separately for each approximate context category. We assume that although these context categories are approximate, they exhibit some degree of uniformity. Hence, analyzing their behavior separately can reveal fine-grained differences in the relationship between process decisions and the achieved performance. Should the analysis relate to all process instances without separating the context categories, the range and variety of behaviors would be too broad for revealing these fine-grained differences.

In Step 3, for a given context category, the process instances that are in the category are evaluated for their weighted soft goal achievement. The evaluation is done separately for each context category since performance assessment can be context-dependent. For example, quality standards for medical products are higher than for chemicals. The output of this step is a performance score associated with each process instance. The performance scores are defined on a discrete scale to facilitate the next analysis steps.

Step 4, path mining, aims to establish the relationship between context, path and performance. It uses a decision tree growing algorithm where path and context of a process instance are the independent variables, while the achieved weighted soft goal scores are the dependent ones. We further elaborate on this step later on. The result is a set of decision paths, which, for the given context category and based on additional context information, provide predictions of process instance performance. Based on this, the procedure identifies the best performing path for the context category and formulates decision rules for that category accordingly. This is done in Step 5. Finally, in Step 6 the suggested decision rules are evaluated by a domain expert and approved before they can be embedded into the process model.

3.2. Establishing relations between context, path decisions, and outcomes

This section describes the details of Step 4 — path mining, which trains a decision tree and is a main step of the LPM procedure. The input to this step is the data of past process instances of an approximate context category, represented according to the GPM view discussed in Section 2. Below we discuss how the data representation should enable and support learning.

3.2.1. Context representation

Context representation: as noted, the analysis is performed separately for each approximate context category. However, since these categories are approximate, context variables which are not accounted for in the context category definition might still affect the process. For example, for a bottle manufacturing process assume an approximate context category of cosmetic products (defined by the variable of product market type). The path outcome performance relation might also be affected by other variables, such as the product size. Note that if the initial

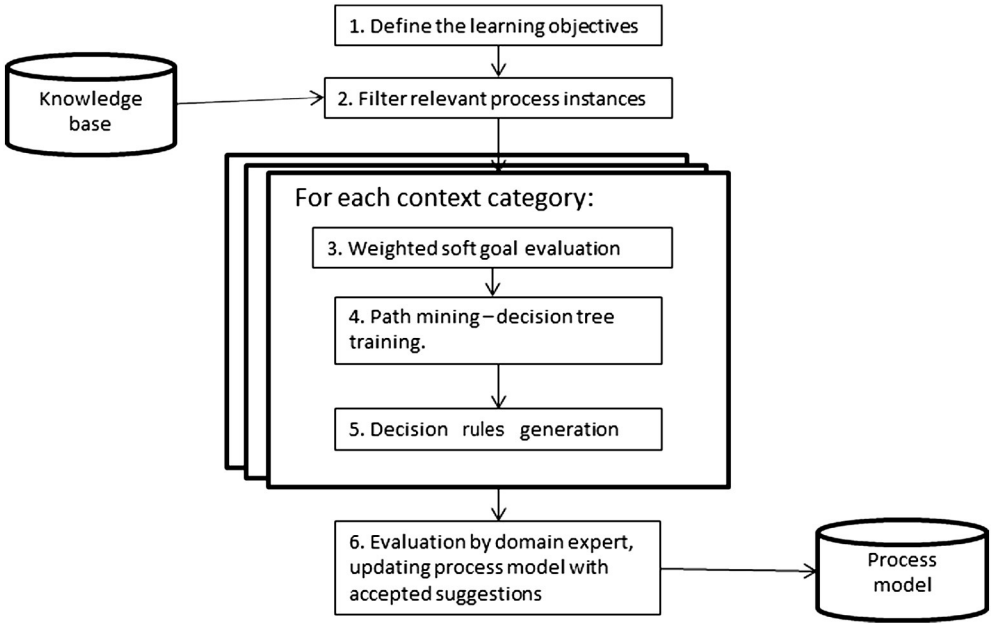


Fig. 1. Learning procedure architecture.

Please cite this article as: J. Ghattas, et al., Improving business process decision making based on past experience, Decis. Support Syst. (2013), http://dx.doi.org/10.1016/j.dss.2013.10.009

approximation of context categories is very accurate, namely, the approximate context categories are actually the ideal ones, then all the affecting variables are accounted for and the decision path that leads to best outcome performance for that context category can easily be identified. However, as discussed, our knowledge of the affecting context variables is partial. To overcome this, we let the decision tree growing algorithm consider all context variables (not only those that are part of the context category definition) and evaluate the ones that correlate with the outcome performance.

3.2.2. Path representation

Path representation: GPM addresses a path as a sequence of states whose changes are results of activities and external events. We would like to capture differences in the decisions entailed in process instances and abstract from minor activity ordering deviations. These decisions would be reflected in values of state variables that form the output of activities.

Still, not all the decisions are equally important, as some decisions might affect the process performance while others may not. For example, assume all the workers are equally skilled and motivated. The decision of worker assignment is hence not expected to have any effect on the outcome performance of the process. We cannot, however, tell in advance which decision affects the outcome and how, since this knowledge is usually not available. We expect the decision tree to discover the most important decisions and their criteria. To this end, the path representation of each process instance should include all the state information throughout the process, structured so that similar decisions in different process instances can be identified.

3.2.3. Outcome performance representation

Outcome performance representation: the outcome performance relates to the termination state of the process instance. For our purpose, outcome performance should be represented to reflect the extent to which the process instance meets the business objective. In other words, the outcome is represented as a measure indicating hard and soft-goal achievement, quantified through an objective function (e.g., weighted soft-goal value).

Following this high-level discussion, we now turn to a more technical level of data representation and specific methods. The input used for the decision tree training consists of process instances, each represented as a vector which includes all state variable values along the process as independent variables, and the outcome performance achieved as a dependent variable. The input vector includes the initial state followed by the state (output variable values) following each of the process activities and external events, as illustrated in Fig. 2.

Using the context and path vector as independent variables, a decision tree is grown, with performance in terms of the weighted soft goal value as dependent variable. In general, decision tree growing algorithms use a top-down induction, identifying at each step a splitting criterion of the population to determine which variable is best to split that portion of the population represented by a node. The splitting process ends when no criterion is found for further splitting the node population, or when just a few instances remain in a node. Specifically, we use a modified Chi-square Automatic Interaction Detection (CHAID) growing decision tree algorithm to construct the decision tree that represents the path groups and their relationships [19]. CHAID is a commonly used algorithm for decision tree training. Applied to our data, it tries to split the process instance data of context and path into nodes that contain instances whose dependent variable value (the soft-goal weighted score) are the same. The tree is used in our approach as a basis for proposing improved decision rules. We now discuss and formalize the structure of the tree using Definitions 8–10, and then present the rule generation algorithm that builds on this formalization.

The tree includes nodes and edges (see example in Fig. 3). The root node $n_0$represents all the process instances included in the data, and each node n_k represents a certain population of instances. A node n_k is associated with a distribution of its instance population.

Definition 8. Node distribution.

Let n_k be a node in the tree. Its distribution $d(n_k)$ is a vector $< x_1,x_2,\ldots x_r>$, where r is the number of weighted soft goal possible values[1] *and x_i is the percentage of the node instances whose weighted soft goal score is ranked i (1 is the highest score, r is the lowest).* $\sum_{i=1}^{r} x_i = 1$.

The distribution of each node in Fig. 3 is represented by a bar chart in that node. A parent node is connected by edges to its child nodes, splitting its instance population. Leaf nodes have no children. The edges have conditions by which the instances of the parent node are split into the child nodes.

Definition 9. Edge condition.

Let $e = (n_k, n_j)$ be an edge connecting a parent node n_k to a child n_j. The edge condition of e,c(e),is a predicate over one state variable, such that the process instances of node n_j are a subset of the process instances of n_k, satisfying c(e).

In Fig. 3, the state variables used for splitting the population are x_1, x_2, x_3, x_4, and the edge conditions are specified for each edge (e.g., $C2(x_1)$). Note that the splitting state variable can relate to the context of the process (e.g., x_2) or to a decision made (e.g., x_1). Accordingly, the edge condition can be a contextual condition (e.g., referring to a patient's state) or a decision-related one (e.g., selecting an activity). At each node, the algorithm uses a chi-square test to determine the state variable that can best explain the performance variance of the instances of this node, and sets the edge conditions leading to its descendants accordingly. When all the instances of a node have the same soft goal score or when no variable that can explain the variance is found, no children are generated and the node becomes a leaf node. The edge conditions form the basis for obtaining the semantic definition of a node in the tree.

Definition 10. Node semantics

Let n_j be connected to the root node by a path, whose edges are $e_1,\ldots e_k$. The semantic definition of n_j is a predicate $smt(n_j) \equiv \bigwedge_{i=1}^{k} c(e_i)$; all the process instances of n_j satisfy $smt(n_j)$.

As an example, in Fig. 3 the semantic definition of node N_2 is $smt(N_2) = C2(x_1)$ and of node N_4 is $smt(N_4) = C3(x_1)$ AND $C4(x_2)$.

Although CHAID aims to split the root node into uniform nodes, each containing process instances of the same performance score, not all the formed nodes are uniform. For example, in Fig. 3, nodes 6, 7, and 10 are uniform, as can be seen by the single column in the bar graphs. The semantic definition of such nodes is hence very useful for predicting performance for the respective combinations of context and decisions made in the process. In contrast, node 8 is not uniform, as it contains process instances with different levels performance. This is a leaf node, meaning that the algorithm was unable to further split it in a meaningful way to achieve a more uniform grouping of these process instances. It is hence hard to predict the performance level of process instances that entail the context-path combination it stands for.

3.3. Formulating decision rules

In Step 5, the decision tree serves as a basis for identifying improved decision criteria, formulated as decision rules, semi-automatically. Candidate rules are proposed automatically, and then they are evaluated and consolidated by humans (in Step 6). Furthermore, inapplicable rules can drive human-initiated exploration of the data, yielding additional rules. Eventually, the rules should be presented to domain experts for evaluation and approval.

[1] As noted, weighted soft goal scores are given on a discrete scale, thus have a finite number of possible values.

Please cite this article as: J. Ghattas, et al., Improving business process decision making based on past experience, Decis. Support Syst. (2013), http://dx.doi.org/10.1016/j.dss.2013.10.009

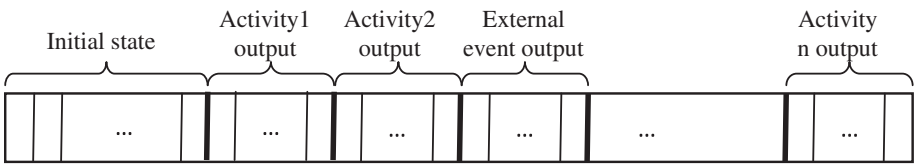


Fig. 2. Input context and path vector structure. Note that the vector includes all the activity executions and the values they assign to their output state variables along the process (including repetitions of activities due to loops).

3.3.1. Candidate rule generation

Two preparation steps are performed prior to the generation of rules. First, nodes that include a low number of instances are pruned. A threshold of instance percentage should be determined (e.g., with respect to the overall standard deviation of the tree), so branches whose number of instances is below this threshold are pruned.

Second, an evaluation function of the performance of the population in each leaf node needs to be defined. The evaluation function should enable a relative ranking of nodes that share a common parent node. We do not prescribe this function as it should be based on domain expert considerations. It would be easy to rank leaf nodes whose instance population is uniform based on their relative performance. However, when the nodes have a mixed instance population, the evaluation function can use different strategies, which could relate to the average soft goal score of the node's population, the most probable score in a node, down-scoring nodes that include instances whose score is below a certain threshold, etc.

The rule generation algorithm generates two types of candidate rules: a *positive* rule, supporting the selection of a certain node (by selecting a respective value of a decision variable), and a *negative* rule, for avoiding a certain node. Considering a parent node n_p, related to a child node n_c by an edge whose condition addresses a decision variable x_i, the rules are specified as follows:

Positive(n_p,n_c):Ifsmt(n_p), then select a value of x_i that satisfies $c(n_p,n_c)$.
Negative(n_p,n_c):Ifsmt(n_p), then avoid a value of x_j that satisfies $c(n_p,n_c)$.

The algorithm makes a distinction between split criteria that relate to decision variables and those relating to context variables. The values of decision variables are controlled and determined by the process decisions. Hence, split conditions related to these variables form a basis for decision rules. In contrast, context variables are not controlled by the process. At run time, they can be known and serve as input for decisions (e.g., the age of a patient). Hence, splits that address context variables will not yield decision rules, but will be used by decisions at lower levels of the tree. Accordingly, each parent node in the tree is marked either as a decision node or as a context node, depending on the split variable of its outgoing edges.

Based on this distinction, for sub-trees that include only decision-related splits, decision rules can be formulated at every split, favoring the node whose score according to the evaluation function is highest. Hence, once the leaves are evaluated using the evaluation function, a parent's evaluation is determined as the maximal score achieved by its descendants. In the example of Fig. 3, for the sub-tree whose root node is N_1, a candidate rule *Positive(1,6)* will be generated, and the score of N_1 will be equal to that of N_6 (its highest-score child).

In contrast, splits related to context variables do not yield rules, since the context of each specific process instance will only become known at run time. If decision-related splits appear above (i.e., closer to the root than) context-related splits in the tree, candidate rules will be generated only for a child node which is clearly superior (positive rule) or inferior (negative rules) to any possible outcome that can be reached(depending on context) down the tree.

In the example of Fig. 3, the split at the root node relates to x_1 (a decision variable). It leads to N_1 (that can be scored according to its descendants), N_2 (that can be scored as a leaf), and N_3 (that will not be scored since it is a context node). N_2 has a high probability

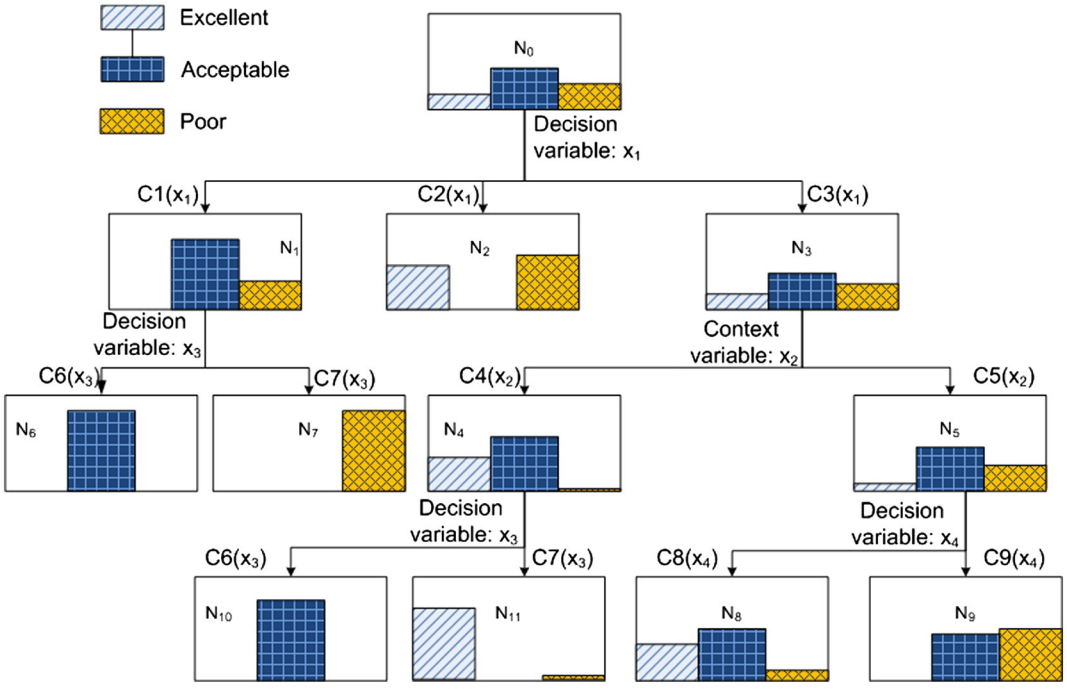


Fig. 3. Decision tree example. The bar chart in each node contains 3 bars, one per each possible value of soft goal score on a discrete scale of *Excellent*, *Acceptable* and *Poor* performance.

Please cite this article as: J. Ghattas, et al., Improving business process decision making based on past experience, Decis. Support Syst. (2013), http://dx.doi.org/10.1016/j.dss.2013.10.009

of poor performance, which is clearly worse than every context-dependent descendant of N_3 (N_4 and N_5). Hence, the algorithm will formulate a *Negative(0,2)* rule.

The algorithm applies a Depth-first (DFS) recursive function (Scoring) to the root node of the tree. The function, specified in pseudo code in Listing 1, calculates the nodes scores and generates candidate rules as discussed above.

As an example, the following rules will be generated for the tree in Fig. 3, using an evaluation function that prefers the node whose probability of excellent performance is higher, and ranks nodes whose percentage of unacceptable performance instances is above 10% as unacceptable.

Positive(n_1,n_6):If C1(x_1), then select a value of x_3 that satisfies C6(x_3)
Positive(n_4,n_{11}):If C3(x_1) AND C4(x_2), then select a value of x_3 that satisfies C7(x_3).
Positive(n_5,n_8):If C3(x_1) AND C5(x_2), then select a value of x_4 that satisfies C8(x_4).
Negative(n_0, n_2):Always avoid a value of x_1 that satisfies C2(x_1).

Note that not all the automatically generated rules are necessarily applicable, thus they require further human consideration through rule consolidation or data exploration.

3.3.2. Rule consolidation

Consolidation is a manual task. It examines the candidate rules that have been suggested and considers several factors. First, it looks for several rules that relate to the same decision variable or for a decision variable addressed by several different levels in the tree. If such are found, it might be possible to formulate a single rule addressing different possible values of the decision variable at different situations. Second, the tree growing algorithm relates only to the significance of the correlation between state variables and the achieved performance, not considering the decisions' order according to the process model.[2] Hence, some candidate rules might imply dependence of a decision that should be made at an early step of the process on one that is taken later. Such rules are not directly applicable for decision making. Consolidation of several rules together can sometimes overcome this. Finally, some phrasing adjustment of the rules would make them easier to use by humans during the process.

3.3.3. Human-initiated exploration

Rules that remain inapplicable after consolidation can serve as a driver for human-initiated exploration, intended to reveal correlations which have not been evident from the tree (they might be less significant than the ones identified already). For example, assume that according to the process model the value of the decision variable $x_4$should be determined earlier in the process than the variable $x_1$which appears at a higher level in the decision tree. Then *Positive(n_5,n_8)* is not applicable. The user can generate separate decision trees, each for a different value of x_4. When each decision tree has a constant value of x_4, its general impact on the performance can be assessed. This would enable evaluating the impact of this decision in situations that do not appear in the original tree and deriving applicable rules accordingly.

4. Evaluation

4.1. Evaluation strategy

We have evaluated the proposed approach by applying it to the data of simulated process instances of a plastic bottle manufacturing process. Simulating this process, we have obtained a set of process instances, each having complete data of its context variable values, execution path, and termination state. It was then possible to apply our learning procedure based on this data and obtain decision rules for improving the process outcome performance. Furthermore, in real life, improvement can only be evaluated after the recommendations are implemented and data of a sufficient number of process instances that follow these recommendations becomes available. In contrast, simulation makes it possible to generate a new set of process instances where the proposed decision rules are applied. Then improvement can be evaluated by comparing the outcome performance achieved by the new instances to that of the first set.

We start by presenting the simulated process, which, for convenience, is illustrated in the BPMN (Business Process Modeling Notation) model depicted in Fig. 4, and then describe the simulation and evaluation.

The (hard) goal of the process is to reach a state where customers' acceptance of delivery is achieved. Other states in which the process might terminate (exception termination states) are states where delivery is canceled due to quality problems or where the customer rejects delivery (due to quality problems or unacceptable delivery times). Various soft goals can be specified for the process, including overall cost, product quality, time to delivery, and others.

The process includes several decision points. Some of the decisions appear as XOR splits in the BPMN model, like the decision about the raw material to be used (new or mixed), and the decision about the quality inspection level. Other decisions are not shown as splits in the model, and are implied by the nature of specific activities. Examples include determining the rate of overproduction, made during production and intended to compensate for quality problems. Another implied decision is the assignment of a specific worker for the order. The context of the process includes external events, e.g., the appearance of some problem during production, and state variables standing for properties known at process initiation. These include properties of the specific product such as market type (medical product, cosmetics, food, etc.), product size (from 0.05 to 10 l, roughly classified as small, medium, and large), and others; order properties such as time to due date, maximum acceptable defect rate, and production plant properties such as maintenance state of the machines.

Recall, we do not know which exact subset of these state variables affects the relationship of path and termination. Still, there are some properties which are known to be relevant, so approximate context categories can be established based on domain knowledge.

The evaluation included the following steps:

1. Pre-configuration of the simulation parameters. While the simulation model exhibits partial consistency of the relationship between context, path decisions, and outcomes, the actual variables affecting this relationship and their actual effect were not known a-priori to the researchers. Rather, they were determined by a pre-configuration step, randomly setting parameter values based on defined distribution functions. This was intended to ensure a lack of bias that could stem from a-priori knowledge when applying the procedure to the simulated data and proposing decision rules.
2. Simulation execution, generating 50,000 process instances. The performance achieved in these instances was measured and recorded.
3. Applying the procedure to instances of one approximate context category to generate decision rules for this context category.
4. Embedding the rules in the simulation logic.
5. Repeating the simulation using the updated model to generate 50,000 new instances.
6. Performance measurement of the instances in the relevant context category and comparison with performance achieved prior to the application of the generated rules.

[2] Note that CHAID can be applied in a controlled manner, following the feasible decision order. However, this constraint might cause the algorithm to disregard some significant correlations and affecting variables. We hence decided to use the algorithm in an unconstrained manner.

Please cite this article as: J. Ghattas, et al., Improving business process decision making based on past experience, Decis. Support Syst. (2013), http://dx.doi.org/10.1016/j.dss.2013.10.009

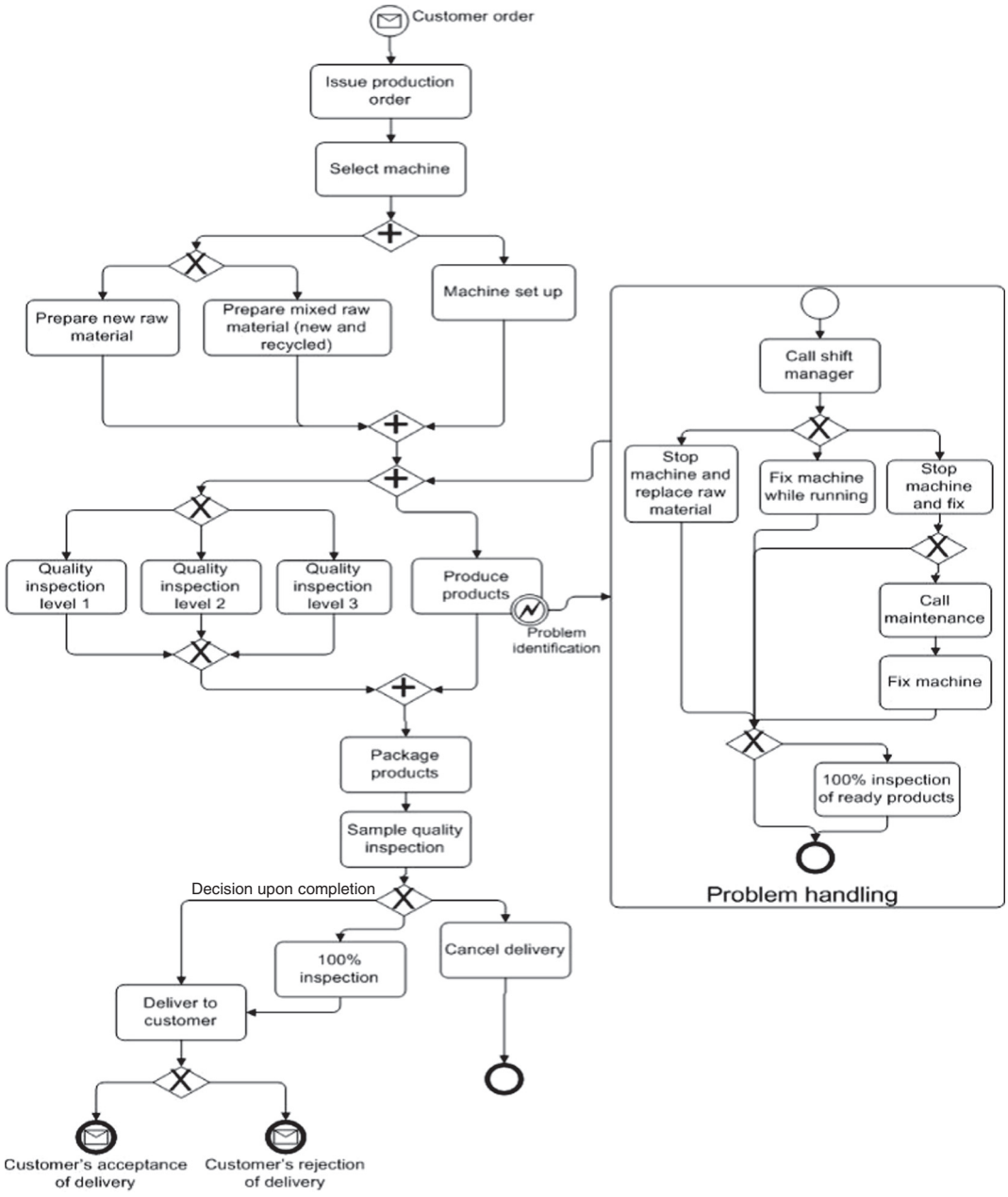


Fig. 4. A BPMN representation of the simulated process.

The main idea is to simulate a real life situation, where improvement opportunities exist (stemming from the hidden relationship context-path decisions-outcome). After applying the learning procedure, the generated rules are 'implemented' – embedded in the simulation logic – and then it is possible to test whether performance improvements have been achieved. Note that the rules that are formulated through the procedure do not encompass all of the decision variables. Rather, they are a subset identified by the tree. It is hence of interest to check the extent to which performance can be improved when implementing these rules.

4.1.1. Simulation settings

The simulation model of the bottle manufacturing process included the process flow as depicted in Fig. 4, operation time distributions, resources, and some decision rules that could be considered obvious (e.g., to prefer assigning the highest-skill worker available). Other decisions, as well as the behavior of performance variables, were represented as parameterized distribution functions. These distribution functions were defined based on the actual behavior as obtained in interviews and observations at the bottle manufacturing firm.[3] The pre configuration step randomly assigned values to the parameters of these functions, so the actual behavior logic would be unknown to the researchers who performed the simulation. The model, not including cases of machine faults, entailed sixteen context variables and seventeen path variables. Hence we had a relations matrix of 16*17 (context * path variable combinations). The simulation pre-configuration phase randomly selected: (a) the subset of context variables to be involved in the values selection of path variables; (b) the subset of path variables that are related to the selected context variables. (c) the logic relating the randomly selected path variables to the randomly selected context variables; (d) the subset of context and path variables affecting

[3] Note, the actual data from the bottle company was not of sufficient quality in terms of completeness and reliability to be used directly for learning. Hence it was decided to use the observed behavior as a basis for simulated process instances whose quality would be satisfactory.

Please cite this article as: J. Ghattas, et al., Improving business process decision making based on past experience, Decis. Support Syst. (2013), http://dx.doi.org/10.1016/j.dss.2013.10.009

the termination state; and finally (e) the logic which related the randomly selected context and path variables to each category of termination state.

Three termination sets of states were distinguished for the process: (0) order delivered but rejected by the customer, (1) order delivered and accepted by the customer, and (2) delivery canceled. Among these, only termination set 1 is desirable.

Using the simulated data of 50,000 process instances, we have performed the learning procedure steps (see Fig. 1).

Step 1: establishing the learning objective. The learning objective for the process considered three soft goals: delivery time (aiming at its reduction), total cost of production (aiming at its reduction), and product quality (aiming at its increase). We considered delivery time as a leading performance dimension, so it would be weighted accordingly (see Step 3).

Step 2: Filtering irrelevant process instances. Since the learning objective addressed delivery time, we filtered out instances whose termination state was in sets 0 and 2 (where delivery is cancelled or the goods are rejected by the customer), and used only those whose termination was in set 1 (order delivered and accepted by the customer). Nevertheless, at the final evaluation of the results achieved by the improved model, we have also checked that the rate of instances in sets 0 and 2 has not increased due to the new decision rules.

Step 3: Evaluating weighted soft goal scores. This step (as well as the following ones) was performed to instances belonging to one approximate context category, defined by a domain expert, with instances of products designated for the medical market where no machine fault has occurred. The computation of the weighted score was done in two steps. First, threshold criteria were set for each soft goal to establish a discrete measurement scale (see Table 1).

Second, the scores assigned to each instance were weighted to reflect the relative importance of the soft-goals (as obtained in interviews with the operations manager):

$$\boldsymbol{SGw} = (0.8 * \boldsymbol{SG_1}(\text{del.time}) + 0.15 * \boldsymbol{SG_2}(\text{Tot.cost of production}) + 0.05 * \boldsymbol{SG_3}(\text{Productquality}))$$

Finally, the resulting score was rounded to 1 (excellent), 2 (acceptable), or 4 (unacceptable). Note that we have decided to mark unacceptable results by 4 to make a clear distinction between what is and what is not acceptable. This decision reflects specific managerial preferences, and can be altered in different situations. This measurement was applied to the simulated process instances in the selected context category. The result is given in Table 2.

Step 4: Path mining. This step used the data of the simulated process instances of the selected context category and generated the decision tree depicted in Fig. 5. The details of the leaf nodes of the generated tree are given in Table 3.

The classification success rate of the tree, calculated as the % of instances that are either in the uniform leaf nodes or in the largest category in the mixed leaf nodes, is above 96% with a 0.3% standard deviation of the classification success rate. Since the smallest leaf node (12) includes 0.8% of the instances, we used the entire tree for rule generation without pruning.

Step 5: Decision rule generation. The decision tree relates mostly to decision variables, with only one context variable (machine maintenance state). The evaluation function we used considered the probability of excellent performance in a node, with a minimal score to a node whose probability of unacceptable performance exceeded 10% (e.g., nodes 6 and 12).

Table 1
Threshold criteria for soft goal discrete values.

Soft goal	Measured by	Soft goal score		
		1 (Excellent)	2 (Acceptable)	4 (Unacceptable)
SG_1 delivery time)	Days	<3	[3,6]	>6
SG_2 (Total cost of production)	1000 USD	<5.72	[5.72,10.46]	>10.46
SG_3 (Product quality)	% defects	<1	[1,3.4]	>3.4

Table 2
Performance of process instances in the selected context category.

Soft goal score	Termination state category			% of termination state 1 instances	% of total instances
	0	1	2		
1-Excellent	NA	2,398	NA	20.4%	18.9%
2-Acceptable	NA	8,837	NA	75.0%	69.6%
4-Unacceptable	NA	548	NA	4.7%	4.3%
Total	806	11,783	104		
% of total instances	6.3%	92.8%	0.8%		

Applying the algorithm, seven candidate rules have been generated: Positive(4,7), Positive(8,11), Positive(3,5), Positive(5,9), Positive(11,17), Positive(10,15), and Positive(9,13). When consolidating the rules, we took into account that the already defined simulation logic of assigning the highest-skill worker available has made the candidate rules Positive(4,7), Positive(5,9), Positive(11,17), and Positive(10,15) redundant. Hence, they were not taken into account in the candidate rule set. After consolidation, the following three decision rules, relating to two decision points in the process were formulated.

Rule 1. Corresponding to the split from node 3 to nodes 5 and 6.

If (Quality control level = Medium) and (Machine Maintenance state ≤ Medium)
Then: Select DecisionUponCompletion = Deliver.

Rule 2. Corresponding to the splits at nodes 3 and 8.

If (Quality control level = Medium) and (Worker skill level = Novice | MediumLow)
Then: Select DecisionUponCompletion = Deliver.

Rule 3. Corresponding to the split from node 9 to nodes 13 and 14.

If (Quality control level = Medium) and (Machine Maintenance state ≤ Medium) and (Worker skill level = Expert)
Then: Select % Over Production due to QA to ≤4.0%

Rules 1 and 2 relate to *DecisionUponCompletion*; Rule 1 is Positive(3,5), which has not been consolidated with any other rule. Rule 2 consolidates Positive(8,11) with Positive(3,5). It applies to cases where the worker skill level is not one of the most preferred ones, and indicates that in this case, regardless of the maintenance state of the machine, the better decision upon completion would be to deliver the order. This recommendation is understandable considering the low weight of the quality soft goal and the high weight of on-time delivery in the defined objective function. Finally, Rule 3 relates to the decision on over-production percentage and builds on Positive(9,13).

4.2. Results

The three rules were embedded into the simulation logic, which was then used for generating 50,000 new process instances. As in the first simulation run, we isolated the instances belonging to the medical market context category and evaluated the performance. The results, presented in Table 4, show that the suggested decision

Please cite this article as: J. Ghattas, et al., Improving business process decision making based on past experience, Decis. Support Syst. (2013), http://dx.doi.org/10.1016/j.dss.2013.10.009

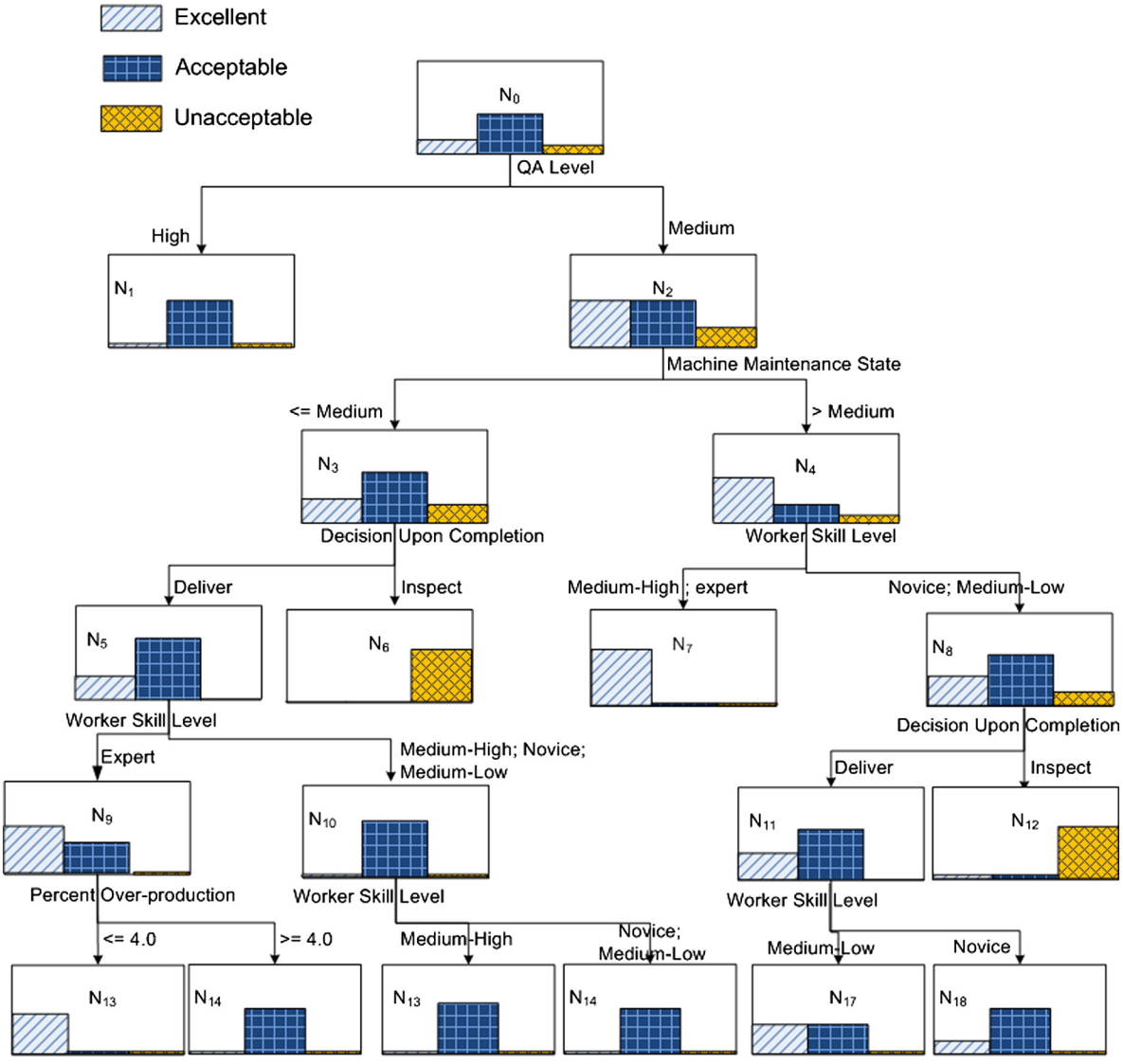


Fig. 5. The decision tree generated for the context category of medical products in the plastic bottle manufacturing process. Outcome performance assessed as 1 (excellent), 2 (acceptable), and 4 (unacceptable).

rules lead to an increase of 16.2% in the number of instances whose performance was excellent and a reduction of 74.5% in the number of process instances with unacceptable performance, as compared to the results of the initial simulation. These are measured with respect to the process instances whose termination state category was 1 (the customer accepted the delivered goods). Also note, that although the rules were suggested only based on analyzing the instances of termination state category 1, there was no significant change in the share of the other two termination state categories (instances of termination state 1 decreased only by 0.3%). Termination states 0 (customer rejects) and 2 (delivery cancelled by the organization) stand for process instances that did not achieve the hard goal of the process. Hence, while improving the soft goal-related performance, we kept the hard goal success rate almost constant. The instances of termination state 1 (delivery approved by the customer) form the majority of the process instances (92.5%), thus every improvement in this category is of importance.

4.3. Additional insights

In addition to the rule-based improvement, the decision tree can provide insights for the human decision maker, who can raise questions and creatively explore further improvement possibilities. We provide two examples for such possibilities.

Example 1: The decision tree indicates a major effect of the maintenance state of the machine on the process outcome. Table 5 presents the decision tree data (see Fig. 5) organized by machine maintenance state and worker skill level. It can be seen that for machines of good maintenance state, excellent outcome instances amount to 73% (17.02% out of 23.3%), while for machines whose maintenance state is medium or less only 16.2% (3.96% out of 24.4%) of the instances have excellent performance. Similarly, unacceptable performance was reported for 3.4% of the instances where machine state was good, as compared to 15.1% for a maintenance state of medium or less. Based on these results, it can be considered to increase the frequency of maintenance operations routinely performed to the machines. Even without such actions, Table 5 indicates that the effect of worker skill level depends on the maintenance state of the machine. While for well-maintained machines novice and Med-low skill workers achieved excellent outcome in 33.2% of the process

Please cite this article as: J. Ghattas, et al., Improving business process decision making based on past experience, Decis. Support Syst. (2013), http://dx.doi.org/10.1016/j.dss.2013.10.009

```
score Function Scoring(node)
If node.type="leaf" then ScoreValue :=score(node)
Else
{
 ScoreValue:=0
 For all unvisited child nodes of node
      value:= Scoring(child)
      If node.type="context" then ScoreValue:=MaxScore
                  //MaxScore is some maximal score, marking context-related nodes
      Else //decision node
          {
               if value=ScoreValue then node.Positive:=Null
                //if several child nodes share the highest score, no positive rule is generated
              Else
              {
                 if value> ScoreValue then
                  {
                    ScoreValue:=value
                     If ScoreValue=MaxValue then Node.Positive:=Null
                     Else node.Positive:=Positive(node,child)
                        //Score is determined as the highest score, and if it is not MaxValue then a positive rule is
                        generated
                  }
                 Else // value < ScoreValue
                    if node.Positive=Null And Not(ScoreValue=MaxValue)
                      then node.Negative=Negative(node,child)
                      //Set Negative rule if child is scored lower than others
               }
          }
  Endfor
  If ScoreValue=MaxValue and node.type="decision" then
        //For decision nodes that are above context nodes – set Positive / Negative rules for sub-trees that are absolutely
        superior / inferior to best / worst possible context-dependent outcome
    {
     MinScore:=0
     MaxScore:=Maxvalue
     For all child nodes, visiting context nodes first:
      If child.ScoreValue=MaxValue then
       {
         If MinScore<Min(descendant_score)then MinScore:=Min(descendant_score)
         If MaxScore>Max(descendant_score)then MaxScore:=Max(descendant_score)
}
        Else
       {
       If child.ScoreValue<MinScore then Node.Negative:=Negative(node,child)
       If child.ScoreValue>MaxScore then Node.Positive:=Positive(node,child)
       }
    Endfor
  }
}
return ScoreValue
End
```

Listing 1. Recursive scoring and rule generation function, using the following global node data: Node.type: can be 'context', 'decision', or 'leaf'; Node.positive: holds *Positive* rule for the node; Node.negative: holds *Negative* rule for the node; Node.scorevalue: holds the score of the node.

instances they were assigned to, no excellent outcome was achieved by them for machine maintenance state of medium or less. This might imply that instead of assigning lower-skilled workers randomly when no higher skilled workers are available (the current rule), it should be preferred to assign these workers to better maintained machines, and thus increase their chances of achieving higher outcomes.

Example 2: Rules 1 and 2, preferring a decision upon completion of delivery (avoiding 100% inspection), might raise doubts since they practically eliminate 100% inspection when facing severe quality problems (for the specific context category). They are a result of the relatively high weight of time to delivery as compared to the quality soft goal in our defined objective. Different weights would likely lead to different performance assessment and to different recommendations. Facing the given recommendations, the decision maker might consider changing the soft-goal weights. A closer look at the detailed soft-goal scores of nodes 6 and 12 in the tree reveals that the unacceptable performance reflects not only delays in time to delivery, but also high costs due to the inspection work and wasted material. The second simulation, where 100% inspection was avoided, yielded a small increase in the instances rejected by the customer, probably cases where the quality was low due to applying the rules. However, this increase is only by 0.2% of the total instances, while the decrease in the unacceptable outcome is by 3.2% of the total instances. Hence, it seems the rules are justified.

Please cite this article as: J. Ghattas, et al., Improving business process decision making based on past experience, Decis. Support Syst. (2013), http://dx.doi.org/10.1016/j.dss.2013.10.009

Table 3
Leaf nodes in the decision tree (Fig. 5).

Leaf node	Parent node	Leaf node semantics (smt)	% of total instances	Score distribution		
				Excellent	Accept.	Unacceptable
1	0	QALevel = High	52.3%		100%	
6	3	(QALevel = Medium) and (Machine maintenance state ≤ Medium) and (DecisionUponCompletion = Inspect)	3.7%			100%
7	4	(QALevel = Medium) and (Machine maintenance state > Medium) and (Worker skill level = MedHigh or Expert)	13.9%	100%		
12	8	(QALevel = Medium) and (Machine maintenance state > Medium) and (Worker skill level = MedLow or Novice) and (DecisionUponCompletion = Inspect)	0.8%			100%
13	9	(QALevel = Medium) and (Machine maintenance state ≤ Medium) and (DecisionUponCompletion = Deliver) and (Worker skill level = Expert) and (% Over production due to QA ≤ 4)	3.2%	100%		
14	9	(QALevel = Medium) and (Machine maintenance state ≤ Medium) and (DecisionUponCompletion = Deliver) and (Worker skill level = Expert) and (% Over production due to QA > 4)	3.3%		100%	
15	10	(QALevel = Medium) and (Machine maintenance state ≤ Medium) and (DecisionUponCompletion = Deliver) and (Worker skill level = MedHigh)	5.7%	13.3%	86.7%	
16	10	(QALevel = Medium) and (Machine maintenance state ≤ Medium) and (DecisionUponCompletion = Deliver) and (Worker skill level = Novice or MedLow)	8.4%		100%	
17	11	(QALevel = Medium) and (Machine maintenance state > Medium) and (Worker skill level = MedLow) and (DecisionUponCompletion = Deliver)	5.3%	47.1%	52.9%	
18	11	(QALevel = Medium) and (Machine maintenance state > Medium) and (Worker skill level = Novice) and (DecisionUponCompletion = Deliver)	3.3%	18.8%	81.2%	

5. Discussion

The simulation results demonstrate the potential value of the proposed approach and its capability of achieving improvement in business processes. The simulation logic was intended to resemble a real life situation, where some decision rules are explicit (e.g., assigning a worker based on skill level), while others might have some implicit logic and are applied not in a fully consistent manner or not at all. This was imitated by the distribution functions assigned to process decisions. The proposed procedure provides explicit decision criteria to the decisions that most likely affect the performance level achieved by the process, thus they promote an overall improvement in the performance. This effect is not necessarily known in advance, especially since it might be different for different context categories. It is discovered by the decision tree analysis for each context category separately.

Applying the procedure to the simulated process instances, we considered a specific objective function (manifested as the weighted soft goal score), which presumably captures the goals and priorities of a specific enterprise. The following analysis and recommended rules are a result of this defined objective. The recommended rules should be approved by the process owner before they are implemented, possibly by simulating the modified process as was done here. In case the generated rules seem to contradict their view of desirable performance, the procedure can be reapplied with a different objective function, or different weighing schemes can be tested until the recommendations are in line with the process owner's or management view. This can serve as a means for reaching an accurate and agreed upon objective.

The proposed procedure can be used periodically and promote continuous process improvement in two main ways. First, when the decision tree reveals contextual effects such as the machine maintenance state, it might be possible to refine the definitions of the approximate context categories. When the procedure is then reapplied, the analysis would be performed to more accurate context categories, e.g., separately addressing medical products whose machine state is good, medium, and bad. This can result in finer-grained decision criteria and lead to further improvements. Second, when new decision criteria emerge from using the procedure, this is expected to lead to a more uniform decision making. Performing the analysis after a time period would enable the identification of other affecting variables, whose effect was 'hidden' before, as they were less dominant than the ones addressed by the newly applied decision criteria. This, again, is expected to drive further improvement.

The main limitation of the proposed approach is that it relies on statistically established data, hence is not suitable for learning from exceptional situations or lines of action. Rather, it requires sufficient repetition of a line of action (or decision path) to draw conclusions from. This has two main consequences. First, the approach is only applicable to an already established process, where sufficient data of past executions is available. Second, if some ad-hoc action is taken differently than the normal procedures, or when a new path is introduced to the process, it cannot be immediately assessed. Only after accumulating sufficient experience, will our procedure be capable of addressing it and drawing experience-based rules for taking this path. Similarly, if the performance achieved by some newly introduced path is extremely low, this would better be evaluated by a human so the path is avoided before sufficient experience is accumulated for the procedure to recommend it. If mixed results are obtained, then the procedure might assist in recommending when a path should be used and when it should not (based on context).

Table 4
Results of the second simulation run.

Soft goal score	Termination state category			% of termination state 1 instances		% of total instances		Improvement	
	0 (Reject delivery)	1 (Approve delivery)	2 (Cancel delivery)	Before rules	After rules	Before rules	After rules	Termin. state 1	Total instances
1-Excellent	NA	2,746	NA	20.4%	23.7%	18.9%	22%	16.2%	16.4%
2-Acceptable	NA	8,687	NA	75.0%	75.1%	69.6%	69.4%	NA	NA
4-Unaccept.	NA	136	NA	4.7%	1.2%	4.3%	1.1%	74.5%	74.4%
Total	825	11,569	116						
% of total instances	6.5%	92.5%	0.9%						

Please cite this article as: J. Ghattas, et al., Improving business process decision making based on past experience, Decis. Support Syst. (2013), http://dx.doi.org/10.1016/j.dss.2013.10.009

Table 5
Decision tree analysis by machine maintenance level and worker skill.

Machine maintenance state > medium (23.3% of the instances)					
Worker skill Outcome	Expert	Med-high	Med-low	Novice	Total
Excellent	13.90%		2.50%	0.62%	17.02%
Acceptable	0.00%		2.80%	2.68%	5.48%
Unacceptable	0.00%		0.80%		0.80%
Machine maintenance state ≤ medium (24.4% of the instances)					
	Expert	Med–high	Med-low	Novice	total
Excellent	3.20%	0.76%	0.00%	0.00%	3.96%
Acceptable	3.30%	4.94%	8.400%		16.64%
Unacceptable	3.70%				3.70%

Another related limitation is that the procedure filters out process instances for which the objective function cannot be evaluated. These are usually process instances where exceptions have occurred. As a result, these exceptions are not considered and learnt from, even if their number is sufficient for statistical analysis. Furthermore, any causality that might exist between some context or path variable and the occurrence of exceptions is overlooked. The motivation for the decision to filter out exceptions was to provide a firm basis for comparison and evaluation of the process instance's performance. It turns out that the proposed procedure currently enables improvement of what is considered 'normal' behavior. Yet, it can easily be extended to also address the exceptions whose number is significant enough for statistical analysis, by assigning them some below unacceptable performance score.

The evaluation of the approach, as reported in this paper, is based on simulated data of one specific process. This can be perceived as a limitation of the evaluation. Still, applying the procedure to the simulated process provides a firm indication that improvement can be achieved following our approach. Further investigations of the extent of improvement and the kinds of processes it is applicable to are still needed. Moreover, while the analysis of separate context categories intends to increase the accuracy of the results, we might obtain rules which are hard to implement when all context categories are considered together. In particular, this applies to rules related to the allocation of scarce resources. Considering each context category separately cannot indicate how to prioritize the use of such resources. This should also be considered at a larger scale realistic experimentation with the proposed procedure.

6. Related work

Process improvement has traditionally been considered a manual task, requiring human creativity and expertise. Support offered for this task has mainly been methodological (e.g., [17]), also aided by measurement and monitoring applications such as Business Activity Monitoring (BAM) [13]. Such applications support identifying improvement opportunities (e.g., identifying bottlenecks in the process), while the improvements themselves are designed by humans (e.g., introducing concurrency of certain tasks).

Similarly, process mining analysis methods have been proposed [1], identifying performance bottlenecks that can or should be addressed by improvement initiatives. The performance indicators addressed are mainly time, and to some extent resource usage. Considering time, process mining-based performance analysis [1] can identify parts of the process where tasks are delayed, performance time statistics, and more. Considering resources, it is possible to identify the number of employees involved in a process or in parts of it [27], and the extent of communication between them. Based on this analysis, process designers can suggest improvement directions, and measure their results once they are implemented.

As opposed to traditional, human-based improvement approaches, some automated or semi-automated approaches for improving or predicting process performance have been proposed. Reviewing these works, we shall address the three process aspects emphasized by our approach, to enable comparison with our work. First, we discuss the process decisions that are guided and supported by each approach. Second, the extent to which context is taken into account. Third, the way in which performance is addressed.

6.1. Guiding process decisions

The decision rules generated by our approach relate to all kinds of decision variables, including control-flow (path selection) decisions as well as decisions made within activities, not explicitly viewed in the process model (e.g., determining over-production %). In contrast, most of the reviewed works support only control-flow related decisions. In particular, this is common to runtime operational support approaches, providing recommendations for activity selection [2,15,16,18,24,26,31] or performance predictions [1,32] based on a partial trace of the execution up to a certain moment. Other approaches [5,9,14], similarly to our approach, propose an off-line periodic analysis, aimed at generating rules for future process improvement. However, these rules relate to activity and path selection decisions only. A different kind of decision support is proposed in [25], using a process mining-based simulation model that addresses multiple process instances running in parallel. The simulation starts from the current state as depicted in the workflow system, and creates logs of simulated process instances, which can be analyzed for evaluating alternative plans, concerning resource allocation among the different process instances. This aspect is not addressed by our approach, which relates to decisions taken within a single process instance.

6.2. The effect of context of decision-making

The effect of context on process decisions and resulting performance is largely overlooked by works in this area. Many approaches [1,9,25,26,32] relate to the state where a decision is made as the set of activities performed up to that moment, disregarding other environmental effects. Context is implicitly acknowledged in [20,24,31]. These works propose a business process life-cycle that relies on ADEPT [23], which is an adaptive process management system, allowing ad-hoc changes to the process model at runtime. These changes are mined later on and serve for process improvement [15]. The selection of change operation at a given situation is supported by a case-based reasoning (CBR) mechanism which retrieves previous change operations that were made at similar situations (cases) [24,31]. The situation is characterized by a textual problem description and a set of question–answer pairs. These can capture contextual information as well as any other information. Hence, while context is not explicitly addressed by this approach, it can still be taken into account as part of the change considerations. Context is also implicitly addressed by [14,18,30,32], who use state information as a basis for decision support and for predictions. However, no distinction is made between the state of the process (internal variables) and the context (environmental effects). One approach[5] bears much resemblance to ours, addressing context explicitly, and building on a combination of context and path to achieve improved performance. Similarly to [20] and [24], it allows run time modifications and then mines these modifications to suggest process variants. These variants are context-based, and are expected to improve process performance. However, it differs from our approach in

Please cite this article as: J. Ghattas, et al., Improving business process decision making based on past experience, Decis. Support Syst. (2013), http://dx.doi.org/10.1016/j.dss.2013.10.009

addressing only activity selection (control flow) decisions, and not decisions that are taken within the activities.

6.3. Process performance

Performance assessment is addressed in various ways by the various approaches reviewed here. Some of the approaches are limited to a single performance metric, such as time [1,26] or efficiency [9]. Some enable relying on past experience, but without explicitly considering the performance achieved [15,24], or with some informal user ratings [18,31]. Addressing defined performance indicators or objectives, in a similar manner to our approach, is enabled by [5,14,30,32]. However, the performance indicators of [24] and [32] differ from our objective being defined on a binary scale (success/failure). Both approaches attempt to predict performance indicators based on intermediary metrics that reflect runtime state.

Finally, the existing approaches can also be characterized by the mechanism used for supporting process decisions or for performance prediction. Process mining is a basis for a number of approaches. It is used for performance prediction in [1]; for activity recommendation when using a declarative process model in [25,26]; for generating a simulation model in [19]; and for mining ad-hoc process changes in [15,5]. Case based reasoning (CBR) is used in [20,24,31] for retrieving similar cases to a given situation. The advantage of CBR is the ability to learn from ad-hoc changes and exceptional situations without having to wait for sufficient experience to be accumulated. This is not possible when following our approach. Furthermore, the adaptive process management system [18] over which CBR approaches operate, allows an increased variability gained through the ad-hoc changes to which the decisions apply. This kind of extended variability is also supported by [18]. In contrast, the variability supported by our approach includes the different paths in the process model, as well as combinations of parts of previous paths, that can be different than all existing path traces. However, the CBR-based approach does not provide for a systematic assessment of the performance associated with a case. Additionally, the cases in the case base can be considered as anecdotal evidence rather than a statistically established one. As opposed to our approach, CBR relies on specific cases at a time, unlike our approach which aggregates knowledge from a corpus of cases. Hence, CBR approaches are not certain to improve process performance.

Various clustering and machine learning approaches are also used [9,14,18]. In particular, decision trees are used by [14,30,32]. Grigori et al. [14] propose a comprehensive framework for analyzing, mining, improving processes, and predicting their runtime performance. Similarly to our approach, they use decision trees to identify influential factors on performance. Unlike in our procedure, the decision trees are applied to all available process instances without separating them based on their context, and this might reduce the accuracy of the results, since large variations in the behavior, stemming from contextual differences, might hamper the ability of the tree to reveal the outcomes of decisions taken at a given context. For prediction, a set of decision trees are used at different phases of the process, each considering the state variables whose value can be known at that phase. Decision trees are used by[30] and [32], attempting to predict key performance indicators on a binary scale from past process instances data. They use a decision tree analysis to discover dependencies between process metrics, not necessarily related to process decisions.

In summary, supporting process decisions based on past experience is currently an emerging area. Compared to existing approaches, our approach has two main advantages. It supports both control flow decisions and decisions embedded in process activities, and it builds on a clear distinction of context categories. While some of the reviewed approaches address one of these issues, none of them addresses both. In addition, although most works include some experimental evaluation, either using real data sets or simulated ones, the performance assessment is usually not comparable to ours, either being restricted to specific performance metrics or to informal ratings, and when more generic metrics are used, their range of values is limited to binary ones.

7. Conclusions

The importance of supporting decision making in business processes for improving business performance has long been acknowledged. The approach presented in this paper emphasizes three aspects in this respect. First, relating to all process decisions, including activity selection as well as decisions embedded within activities. Second, taking the context in which decisions are made into account. Third, assessing business performance in terms of clearly specified goals. Analyzing these three elements in data of past process instances, the approach identifies partial consistencies in the behavior, which may reflect some experience acquired by individual workers over the years. We thus believe that our approach could facilitate organizational learning and a transformation of tacit knowledge, possessed by individuals, into explicit knowledge shared by all, in the form of clear decision rules.

The proposed approach builds on sound conceptual foundations that enable precise definitions of the concepts involved. In particular, since a GPM process operates over a defined domain, a clear distinction of context is made, unlike in some of related works. In addition, the state-based view supports the approach in addressing all process decisions, as opposed to activity-based process views, where the focus tends to be on control-flow decisions.

A main limitation of the approach is its inability to relate to exceptional situations or new lines of action, before they acquire sufficient experience data. A possible research direction that follows is to extend our approach by CBR-based methods that should apply to unique or new situations. Allowing the approach to operate over an adaptive process management system would allow for more variations that follow ad-hoc changes. Then adaptation to less-frequent variations would be necessary. Further evaluation of the approach by applying it to real data of different domains should enable better understanding of its applicability.

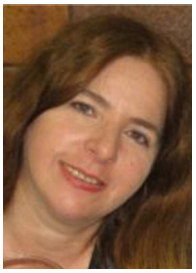
References

[1] W.M.P. van der Aalst, B.F. van Dongen, Discovering workflow performance models from timed logs, in: Y. Han, et al., (Eds.), Proc Int. Conf. on Engineering and Deployment of Cooperative Information Systems, LNCS, 2480, Springer, 2002, pp. 45–63.
[2] W.M.P. van der Aalst, M. Pesic, M.S. Song, Beyond process mining: from the past to present and future, Proc. of CAiSE, LNCS, 6051, Springer, 2010, pp. 38–52.
[3] W.M.P. van der Aalst, et al., Process mining manifesto, BPM Workshops, LNBIP, vol 99, Springer, 2012, pp. 169–194.
[4] A. Aamodt, E. Plaza, Case based reasoning: foundational issues, methodological variations and system approaches, AI Communications (1994) 39–59.
[5] A. Bucchiarone, A. Marconi, M. Pistore, A. Sirbu, A context-aware framework for business processes evolution, 15th IEEE EDOCW, IEEE Com. Soc., 2011, pp. 146–154.
[6] M. Bunge, Treatise on basic philosophy, Ontology I: The Furniture of the World, vol. 3, Reidel, Boston, 1977.
[7] E. Castillo, J.M. Gutiérrez, A.S. Hadi, Learning Bayesian Networks, Monographs in computer scienceSpringer-Verlag, New York, ISBN: 0-387-94858-9, 1997, pp. 481–528.
[8] J.L. De La Vara, R. Ali, F. Dalpiaz, J. Sánchez, P. Giorgini, Business processes contextualisation via context analysis, Proceedings of the ER 2010, Springer-Verlag, Berlin, 2010, pp. 1–6.
[9] A. Dohmen, J. Moormann, Identifying drivers of inefficiency in business processes: a DEA and data mining perspective, in: I. Bider (Ed.), Enterprise, Business-Process and Information Systems Modeling, LNBIP, 50, Springer-Verlag, Berlin, 2010, pp. 120–132.
[10] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, From data mining to knowledge discovery in databases, AI Magazine 17 (3) (1996).

Please cite this article as: J. Ghattas, et al., Improving business process decision making based on past experience, Decis. Support Syst. (2013), http://dx.doi.org/10.1016/j.dss.2013.10.009

[11] J. Ghattas, P. Soffer, M. Peleg, A formal model for process context learning, Proc. of 5th Int. Workshop on Business Process Intelligence(BPI 2009), Ulm, Germany, 2009.
[12] J. Ghattas, M. Peleg, P. Soffer, Y. Denekamp, Learning the context of a clinical process, Proc. Workshop on health-care processes (PROHealth 2009), Ulm, Germany, 2009.
[13] M. Golfarelli, S. Rizzi, I. Cella, beyond data warehousing: what's next in business intelligence? DOLAP'04, ACM Press, 2004. 1–6.
[14] D. Grigori, F. Casati, M. Castellanos, U. Dayal, M. Sayal, M.C. Shan, Business process intelligence, Computers in Industry 53 (2004) 321–343.
[15] C. Guenther, S. Rinderle, M. Reichert, W. van der Aalst, Change mining inadaptive process management systems, Proc. CoopIS'06, 2006, pp. 309–326.
[16] C. Haisjackl, B. Weber, User assistance during process execution — an experimental evaluation of recommendation strategies, Proc. of BPI Workshop, LNBIP, vol. 66, Springer, 2011, pp. 135–145.
[17] M. Hammer, J. Champy, Reengineering the corporation — a manifesto for business revolution, Nicholas Brealey Publishing, London, 1994.
[18] Z. Huang, X. Lu, H. Duan, Using recommendation to support adaptive clinical pathways, Journal of Medical Systems 36 (2012) 1849–1860.
[19] G.V. Kass, An exploratory technique for investigating large quantities of categorical data, Journal of Applied Statistics 29 (1980) 119–127.
[20] C. Li, M. Reichert, A. Wombacher, The MinAdept clustering approach for discoveringreference process models out of process variants, International Journal of Cooperative Information Systems 19 (2010) 159–203.
[21] A.V. Oppenheim, A.S. Willsky, Signals & systems, 2nd edition Prentice Hall, 1997.
[22] K. Ploesser, M. Peleg, P. Soffer, M. Rosemann, J. Recker, Learning from context to improve business processes, BP Trends 6 (2009) 1–7.
[23] M. Reichert, S. Rinderle, P. Dadam, DEPT workflow management system: flexible support for enterprise-wide business processes, Proc. Intl. Conf. on Business Process Management (BPM), Springer, 2003, pp. 370–379.
[24] S. Rinderle, B. Weber, M. Reichert, W. Wild, Integrating process learning andprocess evolution — a semantics based approach, Proc. BPM'05, Springer, 2005, pp. 252–267.
[25] A. Rozinat, M.T. Wynn, W.M.P. van der Aalst, A.H.M. terHofstede, C. Fidge, Workflow simulation for operational decision support, Data and Knowledge Engineering 68 (2009) 834–850.
[26] M.H. Schonenberg, B. Weber, B.F. van Dongen, W.M.P. van der Aalst, Supporting flexible processes through recommendations based on history, Proc.of BPM, LNCS, 5240, Springer, 2008, pp. 51–66.
[27] M. Song, W.M.P. van der Aalst, Towards comprehensive support for organizational mining, Decision Support Systems 46 (2008) 300–317.
[28] P. Soffer, Y. Wand, On the notion of soft goals in business process modeling, Business Process Management Journal 11 (2005) 663–679.
[29] P. Soffer, Y. Wand, Goal-driven multi-process analysis, Journal of the AIS 8 (2007) 175–203.
[30] B. Wetzstein, P. Leitner, F. Rosenberg, I. Brandic, S. Dustdar, F. Leymann, Monitoring and analyzing influential factors of business process performance, EDOC'09: Proc. 13th IEEE Int. Enterprise Distributed Object Computing Conf, 2009, pp. 141–150.
[31] B. Weber, S. Rinderle, W. Wild, M. Reichert, CCBR-driven business process evolution, Proc. Int. Conf. on Case-Based Reasoning (ICCBR'05), Chicago, 2005.
[32] L. Zeng, C. Lingenfelder, H. Lei, H. Chang, Event-driven quality of service prediction, ICSOC 2008, LNCS, 5364, Springer-Verlag, Berlin Heidelberg, 2008, pp. 147–161.

Johny Ghattas, MSc. Telecommunication Engineering, MBA, Phd Student (2006–April 2012) at University of Haifa. His fields of interest both academic and professional are business process learning automation through machine learning, requirement engineering, DSS, artificial intelligence and complex systems. He specializes in the Telecom and IT domains, being a certified ITIL Expert and providing Business consulting services to Big IT and Telecom companies in Israel, Western Europe and Latin America, through his company Smart Path ltd (, www.smart-path.com). He has an extensive knowledge of IT, business process and enterprise architecture related standards including ITIL v3, ISO-20000, eTOM (NGOSS/SID) telecom framework and COBIT.



Dr. Pnina Soffer is a senior lecturer in the Information Systems Department at the University of Haifa. She received her BSc (1991) and MSc (1993) in Industrial Engineering from the Technion, Ph.D in Information Systems Engineering from the Technion (2002). Her research deals with business process modeling and management, requirements engineering, and conceptual modeling, addressing issues such as goal orientation, flexibility, interoperability, and context-aware adaptation. Her research has appeared in journals such as Journal of the AIS, European J of IS, Requirements Engineering, Information Systems, and others. She has served as a guest editor of a number of special issues related to various business process topics such as business process flexibility, coordinated development of business processes and information systems, and business process design. Pnina has served in program committees of numerous conferences, including CAiSE, BPM, ER, and others. She is a member of the CAiSE steering committee and an organizer of the BPMDS working conference. She is a member of the editorial board of the Journal of the AIS.

Prof. Mor Peleg is Assoc. Prof at the Dept. of Information Systems, University of Haifa, Israel, since 2003, and Department Head since 2009. Her BSc and MSc in Biology and PhD in Information Systems are from the Technion, Israel. She spent 6 years at Stanford BioMedical Research during her postdoctoral studies and Sabbatical. She was awarded the New Investigator Award by the American Medical Informatics Association (AMIA). Her research concerns knowledge representation, decision support systems, and process-aware information systems in healthcare, and appeared in journals such as JAMIA, International Journal of Medical Informatics, Journal of Biomedical Informatics, IEEE Transactions on Software Eng, TKDE, Bioinformatics. She is the coordinator of the FP7-ICT large-scale project MogiGuide. She has edited a number of special issues related to process support in healthcare and artificial intelligence in medicine. Mor has served in program committees of numerous conferences, including, among others, AI in Medicine (Where she chaired the scientific PC in 2011), Medinfo, ER. She has been co-chair of the BPM ProHealth Workshop five times and an organizing committee member of Knowledge Representation for Healthcare Workshop four times. She is a member of the editorial board of Journal of BioMedical Informatics and Methods of Information in Medicine.

Please cite this article as: J. Ghattas, et al., Improving business process decision making based on past experience, Decis. Support Syst. (2013), http://dx.doi.org/10.1016/j.dss.2013.10.009