A Framework to Evaluate and Compare Decision-Mining Techniques

Toon Jouck[1](✉), Massimiliano de Leoni[2], and Benoît Depaire[1]

[1] UHasselt - Hasselt University, Hasselt, Belgium
{toon.jouck,benoit.depaire}@uhasselt.be

[2] Eindhoven University of Technology, Eindhoven, The Netherlands
m.d.leoni@tue.nl

Abstract. During the last decade several decision mining techniques have been developed to discover the decision perspective of a process from an event log. The increasing number of decision mining techniques raises the importance of evaluating the quality of the discovered decision models and/or decision logic. Currently, the evaluations are limited because of the small amount of available event logs with decision information. To alleviate this limitation, this paper introduces the 'DataExtend' technique that allows evaluating and comparing decision-mining techniques with each other, using a sufficient number of event logs and process models to generate evaluation results that are statistically significant. This paper also reports on an initial evaluation using 'DataExtend' that involves two techniques to discover decisions, whose results illustrate that the approach can serve the purpose.

Keywords: Decision mining · Evaluation · Log generation

1 Introduction

Automated process discovery from event logs has mainly focused on the control-flow perspective of processes. The control-flow perspective can be considered as the process backbone; however, many other perspectives should also be considered to ensure that the model is sufficiently accurate. The decision perspective (a.k.a. the data or case perspective) focuses on how the routing of process-instance executions is affected by the characteristics of the specific process instance, such as the amount requested for a loan, and by the outcomes of previous execution steps, e.g. the verification result. The representation of this decision perspective on a process in an integrated model or as separate tables is nowadays gaining momentum due to the introduction of the Decision Model and Notation (DMN) standard [1]. Decision mining focuses on discovering the decision perspective of a process from an event log. Some techniques (e.g. [2,3]) augment the routing decisions of a control-flow model with the decision logic induced from the data attributes in the event log. Other techniques (e.g. [4,5]) have focused on discovering a decision model in the form of a Decision Requirements Diagram.

The increasing number of decision mining techniques raises the importance of evaluating the quality of the discovered decision models and/or decision logic. Currently, no standard evaluation framework has been proposed in literature. The techniques presented in [2,4,5] have been evaluated informally: by showing it can rediscover some example routing decision logic [2], an example decision model [4] or by demonstrating it on a real-life data set [5]. In contrast, the techniques of [3,6] have been formally evaluated, yet they have applied their techniques on a small number of data sets. Due to their small scale, no statistical tests can be applied to determine the significance of the results.

Current decision mining evaluations have used only 5 of the publicly available repository of event logs[1]. Moreover, the repository does not contain a reference process and decision model and thus the characteristics of the real underlying process are unknown. As a consequence, the evaluations cannot generalize the results to conclude that technique A on average outperforms technique B when confronted with certain process and decision characteristics. However, this is necessary to gain insights in the strengths and weaknesses of the existing decision mining techniques.

This paper introduces a framework to evaluate and compare decision mining techniques with some guarantee that the results are statistically significant, and, hence, generally valid. The framework, which is presented in Sect. 2, extends the existing method in [7] to generate random artificial control-flow models with a decision dimension and simulates those into event logs. It allows users to control for the process and decision characteristics of the generated event logs as needed during evaluation. An initial validation was conducted on two techniques in Sect. 3. Related work is discussed in Sects. 4 and 5 summarizes the paper with conclusions and future work.

2 Evaluation Framework

Decision mining algorithms discover the decision perspective of a process based on an event log that contains both control-flow and decision perspectives. Artificially generating such event logs is challenging. It requires to generate an artificial sound process model and a decision model with decision rules (i.e. data dependencies) first and then simulate the process model and rules into a multiperspective event log. Soundness is a necessary condition for the generated models with rules as otherwise runtime errors can occur during simulation [8]:

- Deadlock: a case gets stuck in the middle of the process where it is not possible to execute any activities.
- Dead activity: an activity part of the process can never be executed for any case.

[1] https://data.4tu.nl/repository/collection:event_logs_real.

Simply adding decision rules to random process models generated by existing control-flow based techniques, e.g. [7], would not guarantee soundness. On the other hand, the only existing method for generating multiperspective models and logs [9] is not tailored for adding decision information to the control-flow model (see Sect. 4). Therefore, this paper introduces the 'DataExtend' approach for generating artificial event logs with both control-flow and decision perspectives. In the next part of this Sect. 2.1 we will present the idea behind the 'DataExtend' using an example. The final part of this Sect. 2.2 formally describes the steps of 'DataExtend'.

2.1 Illustration of Generating Multiperspective Logs

The make-to-order process as illustrated in Fig. 1 will be used as an example throughout the rest of the paper. The process handles the production of a customer order: it starts with issuing the customer order, then materials are prepared, the products are produced, possibly followed by an inspection, then products are packaged, and finally, the products are delivered or the order is canceled when something went wrong. It contains three XOR-splits (indicated in the figure as 'choice x') where choices between multiple activities need to be made:

– the first choice is whether to use new materials or mixed (recycled and new) materials,
– the second choice is about the inspection of the produced products: no inspection, a normal inspection or a thorough inspection,
– the third choice is whether the products will be delivered or canceled.

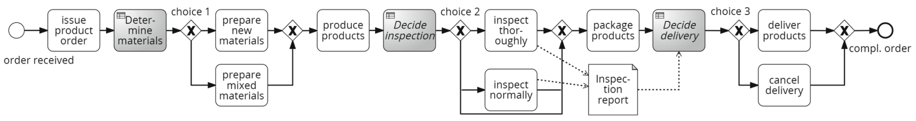

Fig. 1. Make-to-order process

The process model in Fig. 1, without the tasks in grey and the data object, presents the control-flow perspective of the process, i.e. it does not contain information on the decision perspective of the process. In this paper we assume that the decision perspective consists of a decision model and rules that explain the choices in the process. From DMN [1] we adopt the DRD as a decision model that visualizes the dependencies between decisions and the inputs (here case attributes). This paper assumes that each exclusive choice (XOR-split) in the process is preceded by a decision that is modelled in Fig. 1 using a business rule task: 'Determine materials' (choice 1), 'Decide inspection' (choice 2), and

'Decide delivery' (choice 3). Each decision in the DRD corresponds to one business rule task in the process model. For each of the decisions in the DRD we can specify the logic as rules in a decision table.

In the example, the 'Determine materials' (see Fig. 1) decision cannot depend on an earlier made decisions in the process as it is the first decision in the process. It can depend on case attributes, but it is not necessary. Suppose that in this process the decision between new and mixed materials relies upon some contextual information not embedded in the underlying information system. In that case we do not generate decision rules but rather represent the decision stochastically by assigning a probability of choosing each decision output: on average for 50% of the orders the activity 'prepare new materials' is executed and for 50% of the orders the activity 'prepare mixed materials' is performed.

The 'Decide inspection' decision (see Fig. 1) can depend on the decision about the used materials and case attributes. In this example the inspection decision depends on the outcome of the first decision and a case attribute 'premium' which is related to the type of the customer placing the order. This results in the DRD in Fig. 2 where the 'Determine matrials' decision and the customer type are inputs of the inspection decision. The policy is that products produced with mixed materials always need to be inspected thoroughly regardless of what the customer type is. Products consisting of new materials are only inspected for premium customers, otherwise the inspection is skipped to save costs. Such decision logic can be represented as rules as illustrated in Table 1.

Table 1. Decision tables for 'Decide inspection' (left) and 'Decide delivery' (right)

Rule	Materials	Premium?	Inspection
1	new material	True	inspect normally
2	new material	False	(skip inspection)
3	mixed material	True	inspect thorougly
4	mixed material	False	inspect thorougly

Rule	OK quality?	Delivery
1	True	deliver
2	False	cancel

Finally, the 'Decide delivery' decision could depend on the outcome of the first and second decision and some case attribute(s). Suppose that the inspection results in an inspection report (the data object in Fig. 1) based on which the quality of the products is labeled as acceptable or non-acceptable. If an inspection was skipped, acceptable quality of the products is assumed. A delivery will only be executed if the quality of the products are acceptable, otherwise the order is cancelled. These decision dependencies can be illustrated in the DRD in Fig. 2 and the decision logic as shown in Table 1.

The control-flow model together with the above decision model and rules can then be simulated into an event log. The simulator evaluates the rules tied to each decision in the process in order to decide which outcome, i.e. choice branch, to activate. An example case is shown in Table 2. Notice that we only displayed the inputs of each decision as case attributes to save space.

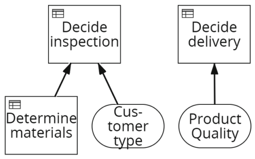

Fig. 2. Decision requirements diagram of example process

Table 2. Example case of the produce order process

Event ID	Activity	Materials	OK quality?	Premium?
1	issue			True
2	prep. mixed mater.	mixed material		True
3	produce	mixed material		True
4	inspect thoroughly	mixed material	True	True
5	package	mixed material	True	True
6	deliver	mixed material	True	True

2.2 Formal Method for Generating Multiperspective Logs

This subsection formalizes the steps of the 'DataExtend' method we propose to generate an event log containing both control-flow and decision perspectives. 'DataExtend' involves the following five steps:

1. Generate a random control-flow model from a population of models
2. Randomly build a decision model
3. Randomly generate decision logic for decisions in the decision model
4. Simulate control-flow model with decision model and logic into event log

Step 1: Generating Random Control-Flow Models. For this step we rely on the existing method in [7] to generate a random control-flow model from a process model population. This method always generates block-structured models that are guaranteed to be sound. A block-structured model can be decomposed in properly nested subprocesses such that each subprocess has a single entry and singly exit point, e.g. the model in Fig. 1. The population definition describes the behavioral characteristics of the models: the size and the control-flow patterns of the models. The user defines a population of models by setting the following parameters:

- Model Size Parameters = {mode, min, max}.
- Control-flow Characteristic Probabilities = {sequence (WCP-1), parallelism (WCP-2/3), exclusive choice (WCP-4/5), multi-choice loop (WCP-6/7), loop (WCP-21), silent activity, reoccurring activity, infrequent path}.

The model size parameters define a triangular distribution that will be used to determine the number of activities that will be present in the model. The WCP-x between brackets refers to the standard control-flow workflow patterns as defined by Russell [10]. A silent activity is used for modelling a skip (e.g. the third branch of the second choice in Fig. 1). Reoccurring activities allow the same activity to appear in different parts of the process. Infrequent paths make some outgoing branches of an exclusive choice more likely to occur than others. The control-flow characteristic probabilities influence the probability for each characteristic to be included in the resulting model. For example, if the probability of an exclusive choice is 0.2, then on average 20% of the nested subprocesses will be exclusive choices.

Step 2: Randomly Build Decision Model. This step will initiate the decision perspective that is added on top of the generated control-flow model. This paper assumes that the routing of the cases throughout the process depends on decision outcomes. This corresponds to adding the business rule tasks before each exclusive choice in the model of the example in Fig. 1, e.g., the outcome for 'Determine materials' influences whether activity 'prepare new materials' or 'prepare mixed materials' is executed for a particular case. Adding these decisions initiates the Decision Requirement Diagram that will contain a decision for each exclusive choice in the model. As such, we will not add decisions before multi-choice and loop constructs in this paper. In a next step, we randomly determine the inputs of the decisions in the DRD. Here we assume that a decision can depend on a case attribute or a previously made decision.

'DataExtend' generates a decision $d_i \in D$ for each exclusive choice in the generated control-flow model. Then, it assigns zero or more attributes as inputs, either a case attribute or a previous decision, to each decision. A decision without attributes means that it is based on some information not embedded in the information system (e.g. the 'Determine materials' decision in the above example).

Definition 1 (Assign). *Given a set D of decisions and a set V of case attributes (including previous decisions), Assign: $D \mapsto \mathbb{P}(V)$ is a function that labels each decision d_i with a set $V' \subseteq V$ of attributes which d_i is based upon.*

The attribues Assign(d_i) used in decision d_i can take on values on the basis of decisions that 'precede' d_i:

Definition 2 (Precedence). *Precedence: $D \mapsto \mathbb{P}(D)$ is a function that labels each decision with a set of preceding decisions. The precedence is based on the control-flow semantics of the model.*

Consider again the example in Sect. 2.1. The 'Decide inspection' decision (d_2) is preceded by the 'Determine materials' decision (d_1): Precedence(d_2)$\mapsto \{d_1\}$. Then, in the example, the 'Decide inspection' decision is assigned the 'Determine materials' decision and 'premium' as attributes: Assign(d_2) $\mapsto \{d_1, premium\}$. The assigned attributes of each decision are visualized in the DRD as shown in Fig. 1 where 'Determine materials' and 'Customer type' are inputs of the 'Decide inspection' decision.

Step 3: Randomly Generate Decision Logic. This step will specify the decision logic for each decision in the decision model generated in the previous step. More specifically, each decision influences a choice between multiple alternative branches in the process model. The values of the assigned attributes of each decision restrict the possible branches that can be activated. These restrictions, also called decision dependencies, can be expressed as decision rules. A decision rule is defined as a mapping:

Definition 3 (Decision Rule). A decision rule is a mapping

$$V_1 \bowtie q_1, \ldots, V_w \bowtie q_w \mapsto \times_{jk}$$

where $V_i \in V$ is the set of attributes, $\bowtie$ is a relational operator $\in \{<, \leq, >, \geq, =, \neq\}$, $q_1, \ldots, q_w$ are constants, $\times_{jk}$ denotes outgoing branch k of choice $\times_j$ after decision j.

The set of all decision rules related to a routing decision can be represented as a decision table such as Table 1, where rule 1 of the left table expresses the decision rule: materials = 'new materials', premium? = 'True' $\mapsto$ 'inspect normally' (i.e. the second outgoing branch of the second decision 'Decide inspection').

'DataExtend' initially generates all possible decision rules, i.e. each possible combination of attribute values can lead to any of the outgoing branches.[2] For example, consider Table 3 that shows the initial set of decision rules for decision 'Decide inspection' in the make-to-order example process. When a case has the following attribute values: materials = 'new materials' and premium? = 'True', then the three outgoing choice branches containing activities 'inspect thoroughly', 'inspect normally', and skip inspection are all possible according to rules 1, 2 and 3.

Table 3. Example complete decision table for the 'Decide inspection' decision in the make-to-order example process.

Rule	Materials	Premium?	Inspection
1	new material	True	inspect thorougly
2	new material	True	(skip inspection)
3	new material	True	inspect normally
4	new material	False	inspect thorougly
5	new material	False	(skip inspection)
6	new material	False	inspect normally
7	mixed material	True	inspect thorougly
8	mixed material	True	(skip inspection)
9	mixed material	True	inspect normally
10	mixed material	False	inspect thorougly
11	mixed material	False	(skip inspection)
12	mixed material	False	inspect normally

Randomly removing rules from the initial set of decision rules restricts the decision outcomes, i.e. the possible outgoing branches at the choice impacted by each decision. In this way, 'DataExtend' creates decision dependencies. However, it cannot restrict the behavior too much as this could create unsound behavior in the form of deadlocks and dead parts which break the simulator in the next step. Therefore, the following soundness constraints are imposed on the rule removal step:

– each decision table has at least one rule for each possible outgoing choice branch to prevent *dead activities*
– each decision table has at least one rule for each value combination of the attributes values to prevent *deadlocks*.

Additionally, the user can set a stopping criterion for the removal of random decision rules. Without such a stopping criterion, 'DataExtend' will remove rules until no removal can happen without violating the soundness constraints. This results in fully deterministic decisions, i.e. for any combination of attribute values there is only one outgoing branch possible. However, business rules are often non-deterministic and this 'cannot be solved until the business rule is instantiated in a particular situation' [11]. This ambiguity can occur due to conflicting

[2] Impossible combinations happen when a decision depends on two other decisions that are mutually exclusive. Such combinations are removed from the decision table.

rules or missing contextual information. Therefore, the approach allows users to set a determinism level as stopping criterion. The determinism level is defined as the number of decision rules removed relative to the maximum amount of decision rules that could possibly be removed (without violating the soundness constraints). The maximum determinism level of 1 results in a fully deterministic decisions. The minimum value of 0 denotes the initial state, i.e. any combination of attribute values can lead to all possible decision outcomes. The user specifies the target determinism level, which is the average determinism level over all decisions *with* input attributes after the removal of rules. We explicitly leave out decisions without assigned attributes, e.g. 'Determine materials' decision in the make-to-order example, because these decisions always have a determinism level of 0, i.e. no rules can be removed, which makes it impossible to reach an average determinism level of 1.

Definition 4 (Determinism level). *Let d_i be a decision and $\#rule(d_i)$ be the number of rules in d_i. Let $\bar{d_i}$ be a decision obtained from d_i after removing a number of rules, then:*

$$DeterminismLevel(d_i, \bar{d_i}) = \frac{\#rule(d_i) - \#rule(\bar{d_i})}{\#rule(d_i) - \#minimum(d_i)}$$

where $\#minimum(d_i)$ is the minimum number of rules to ensure soundness and is determined by taking the maximum of the number of possible decision outcomes for d_i and the number of attribute value combinations of Assign(d_i).

In the make-to-order example (see Sect. 2.1) the desired determinism level is set to 1. This means that as much rules as possible have to be removed from the decision table of the 'Decide inspection' and 'Decide delivery' decisions. The initial decision table for 'Decide inspection' (see Table 3) contains 12 rules. The soundness constraints imply that at least one rule for each of the three possible decision outcomes should remain to avoid dead activities. Additionally, the soundness constraints require that the decision table should contain at least one rule for unique combination of case attribute values: {'prepare new', 'prepare mix'} × {'True', 'False'} = 4 thus #minimum = $max(3, 4) = 4$. Removing rules 1, 2, 4, 6, 8, 9, 11 and 12 from Table 3 results in the decision Table 1 with a maximum determinism level: $\frac{12-4}{12-4} = 1$. Similarly, decision rules are removed for 'Decide quality' which ends in the decision Table 1 with determinism level 1. This makes the average determinism level equal to 1 as all routing decisions with assigned attributes are fully deterministic.

Algorithm 1 summarizes the steps 2 and 3 of 'DataExtend'.

Step 4: Simulate Control-Flow Model with Decision Perspective into Event Log. 'DataExtend' will simulate the models with decision rules into an event log. It takes a user specified number of cases to be generated, the process model, and the set of decision rules as input. Then, each attribute, except the ones that correspond to a previous decision[3], are initialized with a random value.

[3] These attributes are initialized with the decision outcome when it is executed.

Algorithm 1. Extend process model with decision perspective

```
1: Input:
2:         $M$ : process model
3:         $dl$ : target determinism level
4: Output:
5:         $M$ : process model
6:         $\mathcal{R}$ : set of decision rules
7: Start ExtendModel($M, dl$)
8: for each exclusive choice in $M$ do
9:     $Assign(d_i) \mapsto V_{random}$
10:    $R_{d_i} \leftarrow$ initial decision table $d_i$
11:    $\mathcal{R} \leftarrow \mathcal{R} \cup R_{d_i}$
12: end for
13: while $AverageDeterminismLevel(PT) < dl$ do
14:    Remove random rule from $\mathcal{R}$ without violating soundness constraints
15: end while
16: return $\mathcal{R}$
```

For example, in the make-to-order process the ‘premium’ case attribute gets value ‘True’.

The simulation algorithm will execute each activity in the model according to the control-flow semantics and include this in the resulting event log. When it encounters a decision d_i (business rule task) it will execute the decision using the generated decision rules $R_{d_i} \in \mathcal{R}$. Therefore, ‘DataExtend’ will collect the values of each of the assigned attributes $\{V_1, \ldots, V_w\}$ to make a state of the current case. Then it will iterate over all the decision rules to collect the possible decision outcomes. A decision outcome is possible if a rule condition matches with the state. Finally, the decision outcome leads to a particular outgoing choice branch to be executed after the decision.

The simulation of a process model with the decision perspective yields an event log with both control-flow and case information as needed for decision mining evaluation.

3 Demonstration

This section presents an empirical analysis of two decision mining algorithms to validate ‘DataExtend’. ‘DataExtend’ has been implemented in the ProM framework as part of the ‘PTandLogGenerator’ package.[4] It enables the evaluation of decision mining techniques that discover a decision model from an event log (e.g. [4,5]) or techniques that discover the decision logic from an event log (e.g. [3,6]). Due to a lack of space, the experiments will focus on the latter type of techniques.

[4] See https://svn.win.tue.nl/repos/prom/Packages/PTAndLogGenerator/.

Experiment Setup. The experiment will evaluate the *mutually-exclusive* technique [3] that discovers fully-deterministic decision rules based on case attributes in the input event log, and the *overlapping* technique [6] that allows to discover non-deterministic decision rules. The goal is to determine the effect of different determinism levels of the generated decision rules by 'DataExtend' on the quality of the discovered decision rules. We have generated a random sample of 129 process models for each miner from six model populations shown in Table 4, i.e. one population for each value combination for the determinism level {0.5, 0.75, 1} and infrequent paths {0 (False), 1 (True)} parameters. The other process characteristics are fixed for each model population. The probability of the sequence, exclusive choice and parallelism patterns is fixed at values 46%, 35% and 19%, respectively based on the analysis of a large collection of models by Kunze et al. [12]. The size of the models varies between 6 and 10 activities, with a mode of 8 activities. Furthermore, we have specified that the case attributes introduced by 'DataExtend' are of three different types: boolean, string and numerical. Each numerical attribute is discretized to a random number of intervals, between 1 and 4, following a uniform distribution to make a finite number of decision rules for each decision. Secondly, the number of case attributes that are assigned to a decision varies between 0 and 3 following a discrete uniform distribution. Each model with rules is simulated into an event log containing between 200 and 1000 cases.

Table 4. Model population parameters for the experiments, where X and Y are assigned all 6 combinations of values in {0, 5.75, 1} and {0 (False), 1 (True)} respectively.

Parameter	MP_{data}
No visible activities	(6,8,10)
Sequence ($\Pi^{\rightarrow}$)	0.46
Parallel ($\Pi^{\wedge}$)	0.19
Choice ($\Pi^{\times}$)	0.35
Infrequent paths (Π^{In})	Y
Sample size (# models)	129
Logs per model	1
Number of cases	[200,1000]
Determinism level	X
Attribute type	∈ {bool, string, numerical}
# intervals	$\sim uniform(1, 4)$
# assigned attributes	$\sim uniform(0, 3)$

For each generated control-flow model, 'DataExtend' will first generate decision rules and an event log. Each generated log is split into a training log (90% of the cases) and a test log (10% of the cases). The generated control-flow model and the training log are used as input of the decision mining techniques to discover decision rules. Then, we evaluate the discovered rules using a classification approach. We first alter the attributes of half of the cases in the test log such that they do not comply with the generated decision rules. Next, the discovered rules are used to classify the cases in the test log as fitting or non-fitting (violating the discovered rules). This enables us to quantify the quality of the discovered rules using the quality metrics recall (how much fitting cases are classified as fitting), precision (how much cases classified as fitting are actually fitting) and their harmonic average the F_1 score.

Analysis of Results. The graph in Fig. 3 illustrates the average F_1 scores for the two decision mining techniques over different determinism levels. The bars indicate the 95% confidence interval for the averages. The graph indicates a positive trend, i.e. increasing the determinism level has a positive effect on F_1 scores. The Kruskall-Wallis test [13] shows that the differences in F_1 scores between fully-deterministic (determinism of 1) and non-deterministic decision rules (determinism of 0.5 or 0.75) are statistically significant for the *mutually-exclusive* technique. For the *overlapping* technique only the differences in F_1 score between the largest and the smallest determinism levels are statistically significant. Therefore we can conclude that the determinism level has an effect on the quality of the two decision mining techniques. This effect is smaller for the *overlapping* technique than the *mutually-exclusive* technique. This result is not surprising given the fact that the *overlapping* technique specifically focuses on discovering non-deterministic decision rules as included in the experiments.

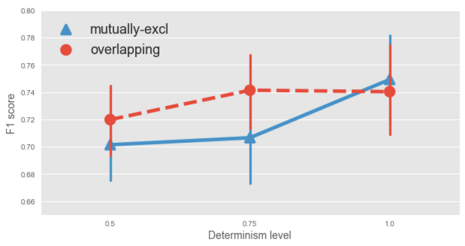

Fig. 3. F_1 scores for decision mining techniques for different levels of determinism

4 Related Work

PLG2 [9] allows for extending control-flow models with data attributes, but in a more general sense. It can add case attributes to activities such that an activity can either generate a case attribute or require a case attribute. The latter is implemented by automatically generating the required case attribute before the execution of that activity. The user cannot control that this case attribute requirement happens to activities after a decision in the process.[5] Nevertheless, this is necessary for the evaluation of decision mining techniques as they focus on discovering the decision model and rules.

5 Conclusion and Future Work

This paper has introduced the 'DataExtend' framework that allows evaluating and comparing decision-mining techniques using a sufficient amount of event logs and process models to detect statistically significant quality differences. For the generation of event logs enriched with data attributes we developed a novel approach, because the only technique to generate such event logs, namely PLG2 [9] does not allow to control the generation of attributes values to influence the decision perspective. A demonstration of 'DataExtend' involved two decision

[5] This is for the random model generator. PLG2 allows users to add the requirements also manually, however, that would not lead to random samples and thus obstruct the generalization of evaluation results.

mining techniques and its results illustrated that the novel approach can serve the evaluation purpose. Future work needs to provide a more extensive evaluation that includes more techniques, such as [4,5]. Furthermore, we want to extend the framework so as to incorporate the loop and multi-choice patterns that involve decisions. The initial evaluation is also based on an implementation that requires a lot of tedious and manual repetition of the application of the techniques for each of the generated event logs. As future work, we aim to integrate it in a scientific-workflow tool, which would automate the currently-tedious work.

References

1. Object Management Group: Decision Model And Notation 1.1, June 2016
2. Rozinat, A., van der Aalst, W.M.P.: Decision mining in ProM. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 420–425. Springer, Heidelberg (2006). https://doi.org/10.1007/11841760_33
3. de Leoni, M., van der Aalst, W.M.P.: Data-aware process mining: discovering decisions in processes using alignments. In: Proceedings of the 28th Annual ACM Symposium on Applied Computing, pp. 1454–1461. ACM (2013)
4. Bazhenova, E., Buelow, S., Weske, M.: Discovering decision models from event logs. In: Abramowicz, W., Alt, R., Franczyk, B. (eds.) BIS 2016. LNBIP, vol. 255, pp. 237–251. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39426-8_19
5. De Smedt, J., Hasić, F., vanden Broucke, S.K.L.M., Vanthienen, J.: Towards a holistic discovery of decisions in process-aware information systems. In: Carmona, J., Engels, G., Kumar, A. (eds.) BPM 2017. LNCS, vol. 10445, pp. 183–199. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-65000-5_11
6. Mannhardt, F., de Leoni, M., Reijers, H.A., van der Aalst, W.M.P.: Decision mining revisited - discovering overlapping rules. In: Nurcan, S., Soffer, P., Bajec, M., Eder, J. (eds.) CAiSE 2016. LNCS, vol. 9694, pp. 377–392. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39696-5_23
7. Jouck, T., Depaire, B.: Generating artificial data for empirical analysis of control-flow discovery algorithms: a process tree and log generator. Bus. Inf. Syst. Eng. 18 (2018)
8. Van Der Aalst, W.M.P., Ter Hofstede, A.H.: Verification of workflow task structures: a petri-net-baset approach. Inf. Syst. **25**(1), 43–69 (2000)
9. Burattin, A.: PLG2: multiperspective process randomization with online and offline simulations. In: BPM (Demos), pp. 1–6 (2016)
10. Russell, N., ter Hofstede, A.H.M., van der Aalst, W.M.P., Mulyar, N.: Workflow controlflow patterns: a revised view. Technical report 06-22 (2006)
11. Rosca, D., Wild, C.: Towards a flexible deployment of business rules. Expert Syst. Appl. **23**(4), 385–394 (2002)
12. Kunze, M., Luebbe, A., Weidlich, M., Weske, M.: Towards understanding process modeling – the case of the BPM academic initiative. In: Dijkman, R., Hofstetter, J., Koehler, J. (eds.) BPMN 2011. LNBIP, vol. 95, pp. 44–58. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25160-3_4
13. Siegel, S., Castellan Jr., N.J.: Nonparametric Statistics for the Behavioral Sciences, 2nd edn. Mcgraw-Hill Book Company, New York (1988)