SAMPLE: A semantic approach for multi-perspective event log generation - Research Paper

Joscha Grüger[1,2], Tobias Geyer[2], David Jilg[2], and Ralph Bergmann[1,2]

[1] Business Information Systems II, University of Trier, Trier, Germany
[2] German Research Center for Artificial Intelligence (DFKI), Trier, Germany

Abstract. Data and process mining techniques can be applied in many areas to gain valuable insights. For many reasons, accessibility to real-world business and medical data is severely limited. However, research, but especially the development of new methods, depends on a sufficient basis of realistic data. Due to the lack of data, this progress is hindered. This applies in particular to domains that use personal data, such as healthcare. With adequate quality, synthetic data can be a solution to this problem. In the procedural field, some approaches have already been presented that generate synthetic data based on a process model. However, only a few have included the data perspective so far. Data semantics, which is crucial for the quality of the generated data, has not yet been considered. Therefore, in this paper we present the multi-perspective event log generation approach SAMPLE that considers the data perspective and, in particular, its semantics. The evaluation of the approach is based on a process model for the treatment of malignant melanoma. As a result, we were able to integrate the semantic of data into the log generation process and identify new challenges.

Keywords: process mining · event log generation · synthetic data · data petri nets

1 Introduction

In many data-rich domains, the application of data analysis methods of data and process mining opens up great potentials. For example, with the right analytical approaches, operational processes can be designed more effective and efficient, new insights can be gained, and predictions can be made. However, there is a lack of data, especially in the context of research, as access is often difficult. This is particularly the case in areas where the data contains personal information (e.g., in medicine) or business secrets (e.g., in industry and business). This fact hinders progress in the development of new approaches and solutions.

One way to address this problem is to work on high-quality synthetic data. For instance, procedural data can be generated based on process models using techniques such as token-based simulation, finite state automata simulation, abduction, constraint satisfactory problem, or Boolean satisfiability problem.

However, all approaches either focus only on the control-flow perspective or, if they are able to generate variable values for the data perspective, they do so based solely on the defined conditions of the process model. In this case, the values are generated without considering the semantics and the focus of the data generation is only on the fulfillment of the conditions. This leads to unrealistic values for the variables and consequently to synthetic data with low quality.

Therefore, we present the SAMPLE approach, a multi-perspective event log generator that considers the data perspective and its semantics in particular. In the approach, variables are described by a meta-model and a triple of semantic information (values, dependencies, distributions). Using this, combined with a play-out algorithm to generate the control-flow perspective, leads to the creation of synthetic, correct data with semantically meaningful variable values. By generating synthetic data, this approach presents a method for preserving patient privacy and security that addresses challenge C7 of the characteristics and challenges for process mining by Munoz-Gama et al. [13].

The remainder of the paper is organized as follows. Section 2 provides information on related work and Section 3 on the components of our approach. Section 4 describes the methodological approach for the multi-perspective synthetic event log generator. In Section 5 the implementation is presented, Section 6 presents the evaluation process, and Section 7 concludes the paper.

2 Related Work

In token-based simulation, tokens are propagated through a process model and executed transitions are recorded to generate event logs. When using Petri nets, the propagation is achieved by firing enabled transitions until all transitions are disabled or a final state is reached. Different strategies can be used to determine which transition to fire, such as random selection. The order of the transitions fired is recorded to generate the traces. The transitions are then referenced with activities to obtain a valid trace, which can be added to the event log. The process is repeated until the desired number of traces is reached.

Token-based event log generation has evolved from work in modeling simulation, such as reference nets [8] or Colored Petri nets (CPNs) [6]. In [8], Kummer et al. developed the application RENEW which is a Java-based high-level Petri net simulator. It provides a flexible modeling approach based on reference nets as well as the feature to dynamically create an arbitrary number of net instances during a simulation. Alves de Medeiros and Günther [11] state the need for correct logs (i.e., without noise and incompleteness) for the development and tuning of process mining algorithms, since imperfections in the log hinder these activities. Their approach is an extension of Colored Petri nets to generate XML event logs with the simulation feature of the CPN Tools [16]. Nakatumba et al. [14] present an approach that incorporates workload-dependent processing speeds in a simulation model and how it can be learned from event logs. Moreover, they show how event logs with workload-dependent behavior can be generated by simulation using CPN Tools [14].

Many approaches address the lack of data for appropriate (process)mining algorithm testing and evaluation [11,17,1,12,15]. The approach of Shugurov and Mitsyuk [17] allows the generation of event logs and sets of event logs to support large scale automated testing. Furthermore, noise can be added to event logs to simulate more realistic data. Vanden Broucke et al. [1] present a ProM [3] plugin enabling the rapid generation of event logs based on a user-supplied Petri net. The approach offers features such as the configuration of simulation options, activities, activity and trace timings. Mitsyuk et al. [12] present an approach to generate event logs from BPMN models to provide a synthetic data base for testing BPMN process mining approaches.They propose a formal token-based executable BPMN semantic that considers BPMN 2.0 with its expressive constructs [12]. The approach simulates hierarchical process models, models with data flows and pools, and models interacting through message flows [12]. Pertsukhov and Mitsyuk [15] present an approach that generates event logs for Petri nets with inhibitor and reset arcs. These arc types improve the expressiveness of nets and are useful when ordinary place/transition-nets are not sufficient [15].

One rather unique use of token-based simulation is proposed by Kataeva and Kalenkova [7]. Their approach generates graph-based process models by applying graph grammar production rules for model generation. A production rule replaces one part of a graph by another [7]. The approach uses a simulation consisting of applying production rules that propagate tokens through the graph to generate event logs. Another approach is presented by Esgin and Karagoz [4], addressing the problem of unlabeled event logs, i.e., the lack of mapping of case identifiers to process instances in real event logs. Instead of fixing the log, the approach simulates a synthetic log from scratch using the process profile defining the activity vocabulary and the Petri net in tabular form as input [4]. In [2], the generation of random processes is extended by the complete support for multi-perspective models and logs, i.e., the integration of time and data. Furthermore, online settings, i.e., the generation of multi-perspective event streams and concept drifts, are supported [2].

Although important problems are addressed, a drawback of most related work in this field is the limitation to the pure control-flow. Besides exceptions such as [2], the data perspective is not considered. Overall, the semantics of data and its impact on the reality of event logs is not explicitly in addressed.

3 Fundamentals

In the following, we introduce the notions required for our approach.

3.1 Basic Notations

Definition 1 (universes, general function). *We define the following universes and functions to be used:*

- *$\mathcal{C}$ is the universe of all possible case identifiers*

- *$\mathcal{E}$ is the universe of all possible event identifiers*
- *$\mathcal{A}$ is the universe of all possible activity identifiers*
- *$\mathcal{AN}$ is the universe of all possible attribute identifiers*
- *$dom(f)$ denotes the domain of some function f.*

Definition 2 (attributes, classifier [18]). *Attributes can be used to characterize events and cases, e.g. an event can be assigned to a resource or have a timestamp. For any event $e \in \mathcal{E}$, any case $c \in \mathcal{C}$ and name $n \in \mathcal{AN}$, $\#_n(e)$ is the value of attribute n for event e and $\#_n(c)$ is the value of attribute n for case c. $\#_n(e) = \perp$ if event e has no attribute n and $\#_n(c) = \perp$ if case c has no attribute n. We assume the classifier $\underline{e} = \#_{activity}(e)$ as the default classifier.*

Definition 3 (trace, case [18]). *Each case $c \in \mathcal{C}$ has a mandatory attribute trace, with $\hat{c} = \#_{trace}(c) \in \mathcal{E}^* \setminus \{\langle\rangle\}$. A trace is a finite sequence of events $\sigma \in \mathcal{E}^*$ where each event occurs only once, i.e. $1 \leq i < j \leq |\sigma| : \sigma(i) \neq \sigma(j)$. By $\sigma \oplus e = \sigma$ we denote the addition of an e event to a trace σ.*

Definition 4 (event log [18]). *An event log is a set of cases $\mathcal{L} \subseteq \mathcal{C}$, in the form that each event is contained only once in the event log. If an event log contains timestamps, these should be ordered in each trace.*

Definition 5 (multiset [18]). *Let X be its set. A multiset is a tuple $M = (X, m)$ with $m : X \to \mathbb{N}$. We use $x \in M$ to express that x is contained in the multiset M, therefore $x \in X$ and $m(x) \geq 1$. We denote by $\mathcal{B}(X)$ the set of all multisets over X.*

3.2 Petri nets, marked Petri net

Petri nets are process models that describe the control-flow perspective of a process while ignoring all other perspectives. [10].

Definition 6 (Petri net [18]). *A Petri net is a triple $N = (P, T, F)$ where P is a finite set of places, T is a finite set of transitions, and $F \subseteq (P \times T) \cup (T \times P)$ is a set of flow relations that describe a bipartite graph between places and transitions. $^\bullet t$ denotes the input places of a transition t.*

3.3 Data Petri Nets

A data Petri net is a Petri net extended by the data perspective.

Definition 7 (data Petri net [9]). *A data Petri net (DPN) $N = (P, T, F, V, U, R, W, G)$ consists of:*

- *a Petri net (P, T, F);*
- *a set V of variables;*
- *a function U that defines the values admissible for each variable $v \in V$, i.e. if $U(v) = D_v$, D_v is the domain of variable v;*

- *a read function $R \in T \to 2^V$ labeling each transition with the set of variables that it must read;*
- *a write function $W \in T \to 2^V$ labeling each transition with the set of variables that it must write;*
- *a guard function $G \in T \to G_V$ associating a guard with each transition.*

For the naming of transitions, we introduce labeled data Petri nets. Invisible transition are enabled and fired, but do not refer to a process activity.

Definition 8 (labeled data Petri net [10]). *Let $N = (P, T, F, V, U, R, W, G)$ be a Data Petri net. Then the triple $LN = (N, \lambda, \nu)$ is a labeled Petri net, with:*

- *$\lambda : T \to (\mathcal{E} \cup \{\tau\})$ is an activity labeling function, mapping transitions on an activity label and invisible transitions on τ.*
- *$\nu : V \to \mathcal{AN}$ a labeling function, mapping variables to attribute names.*

Similar to Petri nets, data Petri nets always have a certain state described by the current markings and variable values. This is defined as follows.

Definition 9 (state of a DPN [9]). *Let $N = (P, T, F, V, U, R, W, G)$ be a Data Petri net with $D = \cup_{v \in V} U(v)$, then tuple (M, A) is the state of N with*

- *$M \in \mathbb{B}(P)$ is the marking of the Petri net (P, T, F)*
- *$A : V \to D \cup \{\bot\}$. For $v \in V$ it holds, if no value is given for v, $A(v) = \bot$*

With (M_0, A_0) the initial state and $M_0(p_0) = 1, \forall p \in P \setminus \{p_0\} : M_0(p) = 0$ and $\forall v \in V : A_0(v) = \bot$ and (M_F, A_F) the final marking.

Petri nets change their state by firing transitions. A valid firing of a transition is defined as follows.

Definition 10 (valid firing [10]). *Let $N = (P, T, F, V, U, R, W, G)$ be a Data Petri net. A firing of a transition is a double (t, w) with $t \in T$ and the variables that are written with the respective values. Let (M, A) be a state of N, then (t, w) is a valid transition firing, if*

- *$\forall p \in {}^{\bullet}t : M(p) \geq 1$, i.e. each place in t's preset contains at least one token.*
- *$dom(w) = W(t)$, the transition writes the prescribed variables*
- *$\forall v \in dom(w) : w(v) \in U(v)$, i.e. the assigned values for variables are valid*
- *Guard $G(t)$ evaluates true with A*

4 Synthetic multi-perspective Log Generation

To generate a realistic event log $\mathcal{L}$ for a given data Petri net N, SAMPLE divides the procedure into the control-flow perspective (Section 4.1) and data perspective (Section 4.2). While the generation of the control-flow perspective of the log is based on the token-based simulation approach, the generation of the data is built on a rule-based approach. Figure 1 depicts the log generation process. In the following, we first describe the generation of the control-flow perspective and extend the approach to the data perspective in the next step.

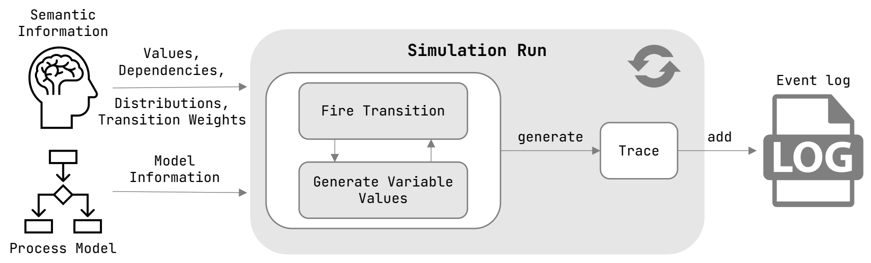


Fig. 1. The figure outlines the event log generation process of the approach. The model information and the semantic data serve as input for the simulation runs, in which traces (i.e., sequences of events with data) are generated and added to the log.

4.1 Generation of the Control-Flow Perspective

The control-flow perspective is realized through random trace generation, i.e., an approach that randomly traverses the Petri net to generate sequences of activities. Each single trace is generated by a simulation run. A simulation run starts with the initial marking of the model and starts firing random transitions until a deadlock or a final marking is reached.

Transitions can be weighted to generate particularly realistic logs. These weightings affect the probability of the transitions firing and could be learned, e.g., based on the distribution in real logs or manually defined by domain experts. For this purpose, we introduce a mapping of transitions to weights:

Definition 11 (transition weights). *Let $N = (P, T, F, V, U, R, W, G)$ be a data Petri net. Then $\omega : T \to \mathbb{R}$ is the mapping of transitions to weights. For all $t \in T$, t_ω is the short form for $\omega(t)$.*

Definition 12 (transition selection). *Let $N = (P, T, F, V, U, R, W, G)$ be a data Petri net and (M, A) the current state of the DPN. Then transition t is selected randomly by $rt(N, (M, A))$ by considering the weights $\omega(t)$. R_ω denotes*

$$\{t \in T | \forall p \in {}^\bullet t : M(p) \geq 1\} \xleftarrow{R_\omega} t$$

a weighted random selection. If no transition is enabled, $\perp$ is returned.

4.2 Generation of Data Perspective

The data perspective is addressed in data Petri nets by variables V and their values U. These are read and written by transitions and evaluated in guards. For the generation of realistic variable values for transitions, extensive knowledge about the variables is necessary. This can be partially learned from the process model or the event log, or must be specified by domain experts. Partially, this knowledge could be modeled in DPNs, but would grow the process

Algorithm 1 Log Generation

```
    Input labeled DPN $LN = ((P, T, F, V, U, R, W, G), \lambda, \nu)$, traces to generate $n \in \mathbb{N}$
    Output Log $\mathcal{L}$
 1: procedure Generate Log(N,n)
 2:     $\mathcal{L} \leftarrow \{\}$                                        ▷ empty log $v$
 3:     while $|\mathcal{L}| < n$ do
 4:         $\sigma = GenerateTrace(N, (M_0, A_9), \{\})$
 5:         if $\sigma \neq \emptyset$ then
 6:             $\mathcal{L} = \mathcal{L} \cup \{\sigma\}$
 7: procedure Generate Trace($LN, (M, A), \sigma$)
 8:     $t \leftarrow rt(N, (M, A))$                          ▷ random transition selection
 9:     if $t = \perp$ then                                              ▷ Deadlock
10:         Return $\emptyset$
11:     else
12:         $(t, w) = GenerateVariable(t, N, (M, A))$            ▷ Generate Variables
13:         $e \leftarrow$ new event
14:         $\underline{e} \leftarrow \lambda(t)$                                     ▷ Set event name
15:         for all $v \in dom(w)$ do
16:             if $is\_tv(v)$ then
17:                 $\#_{\nu(v)}(\sigma) \leftarrow w(v)$          ▷ add trace attributes (generated before)
18:             else
19:                 $\#_{\nu(v)}(e) \leftarrow w(v)$           ▷ add event attributes (generated before)
20:         $\sigma \leftarrow \sigma \oplus e$                                  ▷ Add event to trace
21:         $(M, A) \rightarrow fireTransition(t, w)$     ▷ fire fransition and change Model State
22:         if $(M, A) = (M_F, A_F)$ then
23:             Return $\sigma$
24:         else
25:             Return Generate Trace($LN, (M, A), \sigma$)
```

models and reduce their maintainability. In the following, the knowledge and the procedure for the generation of realistic variable values are specified. To describe the variables, we introduce a variable meta-model.

Definition 13 (variable meta-model). *Let $N = (P, T, F, V, U, R, W, G)$ be a data Petri net, then the meta-model for a variable $v \in V$ is described by:*

- *$U(v)$ describing the domain of valid variable values.*
- *$is\_tv : V \rightarrow \{0, 1\}$ defining v is a trace variable (1) or an event variable (0)*
- *optional semantic information*

The semantic information allows the specification of values that the variable can take. The values can comprise the complete domain of the variable or only a section. In addition, frequency distributions can be specified via value weights.

Definition 14 (semantic information: values). *Let $N = (P, T, F, V, U, R, W, G)$ be a data Petri net with $v \in V$ and $D_v \subseteq U(v)$, the defined possible values for v. Then VW_v is a multiset with $VW_v = (D_v, m)$ with $m : D_v \rightarrow [0, 1]$, the weight function.*

Furthermore, the variables may be interdependent. This must be included to generate realistic values. Each dependency is described by a logical expression and a resulting constraint. If the logical expression is true, the constraint has implications on the possible values for a given variable.

Definition 15 (semantic information: dependencies). *Let $N = (P, T, F, V, U, R, W, G)$ be a data Petri net with $v \in V$, $EXPR$ the Universe of all possible logical expressions (including disjunction and conjunction). Let $C = (EXPR \times (O_C \times U))^*$ be the set of all possible dependencies, with $O_C = \{'==', '!=', '<', '<=', '>', '>='\}$ the set of constraint operators. Then for all $v \in V$, the function $dep : V \to C$ defines the set of all dependencies holding for v. The following shorthands are defined for the dependency sets:*

- *$dep_{int} : V \to (EXPR \times ((O_C \setminus \{'==', '!='\}) \times U))^*$, interval related dependencies*
- *$dep_{eq} : V \to (EXPR \times (\{'=='\} \times U))^*$, dependencies setting fixed values*
- *$dep_{ne} : V \to (EXPR \times (\{'!='\} \times U))^*$, dependencies excluding values*
- *Let (M, A) be a DPN state and $c \in C$ a constraint, then $eval(c, (M, A))$ evaluates the logical expression building on the current DPN state. Let $C' \subseteq C$ be a set of dependency constraints, then $eval(C', (M, A))$ returns a set of all $c' \in C'$ with $eval(c', (M, A)) = 1$.*

This can be used, e.g., to specify dependencies excluding "prostate cancer" as a value for a variable for persons of female gender. It can also be used to define ranges of values or intervals by setting the logical expression to true.

```
'gender == female' => (!=,'prostate cancer')

'true' => (<,5)
```

The third type of semantic information, distribution, refers to the weighting of the values. Instead of concrete weights, however, distribution functions can be specified here. While values are more suitable for discrete, categorical values, the possibility to define distribution functions is directed towards numerical values.

Definition 16 (semantic information: distribution). *Let $N = (P, T, F, V, U, R, W, G)$ be a data Petri net, then for $v \in V$ a distribution function $distr_v : \mathbb{R}^* \to \mathbb{R} \cup \{\perp\}$ provides values, considering the defined deviation (uniform, normal, ...). If $distr_v = \perp$, no distribution is defined for v.*

5 Implementation

For evaluation, SAMPLE was implemented using Python[3]. The tool allows the generation of multi-perspective event logs using semantic information about the

[3] https://github.com/DavidJilg/DALG, GNU GLP 3 license

Algorithm 2 Generation of realistic variable values

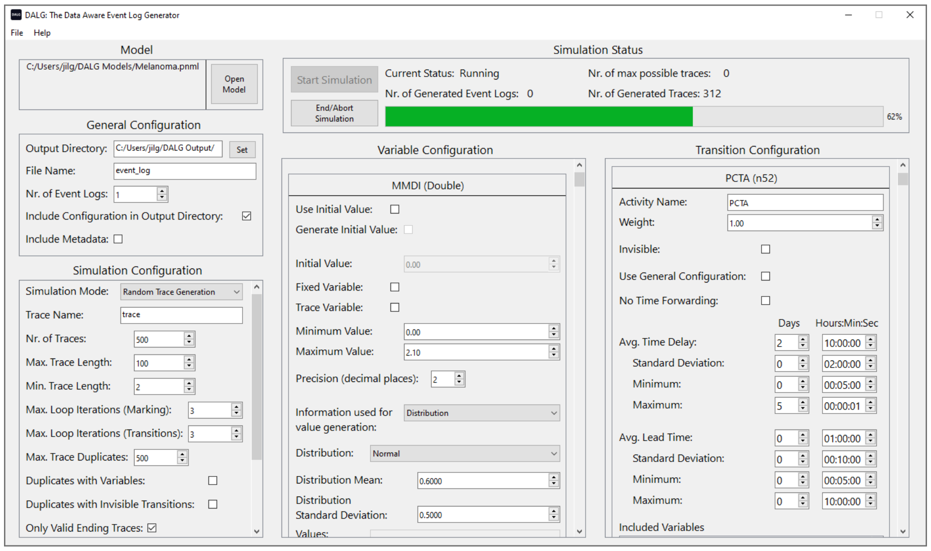
```
    Input Transition t, DPN N = (P, T, F, V, U, R, W, G), DPN State (M,A)
    Output Firing (t,w)
 1: procedure GENERATE VARIABLES(t,N,(M,A))
 2:     for all $v \in W(t)$ do
 3:         $PV \leftarrow U(v)$                                    ▷ Set of possible values for $v$
 4:         if $|dep_{int}(v)| > 0$ then                            ▷ intervall dependencies
 5:             $PV \leftarrow \{value\ \forall value \in PV | value\ in\ dep_{int}(v)\}$
 6:         if $|dep_{eq}(v)| > 0$ then                  ▷ Dependencies setting fixed values
 7:             $PV \leftarrow \{value | \forall (expr,(op,value)) \in eval(dep_{eq}(v),(M,A))\}$
            ▷ Dependencies excluding values
 8:         $PV \leftarrow PV \setminus \{value | \forall (expr,(op,value)) \in eval(dep_{ne}(v),(M,A))\}$
            ▷ value restriction defined
 9:         if $VW_v \neq \emptyset$ then
10:             set $w[v] = randByWeight(\{(val,wei) | (val,wei) \in VW_v : val \in PV\})$
11:         else if $distr_v \neq \perp$ then                       ▷ distribution function defined
12:             set $w[v] = distr_v() | with\ distr_v() \in PV$
13:         else
14:             set $w[v] = randomValue(PV)$
15:     Return (t,w)
```

variables. In addition to the random trace generation approach, the implementation allows to fully explore Petri nets for event log generation.

The implementation facilitates the configuration of semantic information about the variables (distribution, dependencies, etc.). Due to the large amount of semantic information that needs to be provided to generate realistic event logs, the tool offers a function that analyzes the model and tries to suggest semantic information based on the model. For example, guards are analyzed to identify data types, upper and lower bounds, or possible values. Furthermore, the described approach is extended by the time perspective and allows the generation of timestamps. For ease of use, the implementation provides a graphical user interface based on the QT framework, which allows the configuration of the simulation and semantic data and thus the event log generation (see Figure 2).

6 Evaluation

The implementation described in Section 5 was used to evaluate the SAMPLE approach. First, the correctness of the approach was checked by the conformance of generated traces. For this purpose, process models were used including all combinations of semantic information types. All generated traces were conforming to the models and considered the given semantic information entirely. Subsequently, the realism of generated event logs was investigated, which is the main focus of the research. For preliminary investigations, the *activities of daily*



Model

C:/Users/jilg/DALG Models/Melanoma.pnml

General Configuration

Output Directory: C:/Users/jilg/DALG Output/

File Name: event_log

Nr. of Event Logs: 1

Include Configuration in Output Directory: ☑

Include Metadata: ☐

Simulation Configuration

Simulation Mode: Random Trace Generation

Trace Name: trace

Nr. of Traces: 500

Max. Trace Length: 100

Min. Trace Length: 2

Max. Loop Iterations (Marking): 3

Max. Loop Iterations (Transitions): 3

Max. Trace Duplicates: 500

Duplicates with Variables: ☐

Duplicates with Invisible Transitions: ☐

Only Valid Ending Traces: ☑

Simulation Status

Current Status: Running

Nr. of max possible traces: 0

Nr. of Generated Event Logs: 0

Nr. of Generated Traces: 312

62%

Variable Configuration

MMDI (Double)

Use Initial Value: ☐

Generate Initial Value: ☐

Initial Value: 0.00

Fixed Variable: ☐

Trace Variable: ☐

Minimum Value: 0.00

Maximum Value: 2.10

Precision (decimal places): 2

Information used for value generation: Distribution

Distribution: Normal

Distribution Mean: 0.6000

Distribution Standard Deviation: 0.5000

Values:

Transition Configuration

PCTA (n52)

Activity Name: PCTA

Weight: 1.00

Invisible: ☐

Use General Configuration: ☐

No Time Forwarding: ☐

	Days	Hours:Min:Sec
Avg. Time Delay:	2	10:00:00
Standard Deviation:	0	02:00:00
Minimum:	0	00:05:00
Maximum:	5	00:00:01
Avg. Lead Time:	0	01:00:00
Standard Deviation:	0	00:10:00
Minimum:	0	00:05:00
Maximum:	0	10:00:00

Included Variables

Fig. 2. Graphical user interface of the DALG-tool (Data Aware Event Log Generator)

living of several individuals[4] dataset was chosen to enable initial evaluations with respect to realism without expert knowledge. Afterwards, the approach was evaluated by domain experts for skin cancer treatment. Therefore, a model was used that represents the diagnosis and treatment of malignant melanoma [5]. It consists of 51 places, 26 variables, 76 transitions, and 52 guards. Using the model, event logs were generated and a representative set of traces was used for evaluation. Therefore, a visual representation of the data was presented to the domain experts. The task of the evaluation was to investigate whether a physician with the same information would have chosen the same treatment options as in the synthetic traces. Additionally, it should be evaluated whether the generated variables are realistic for the given patient.

Compared to other existing approaches, the results show that the inclusion of semantic information enables the generation of data that is more meaningful and correct from a semantic point of view. This means that the sequence of activities shown in the trace makes sense in terms of the data available at a point in time. Consequently, the correct activities are performed based on the variable values. For example, an additional excision will only be performed on a patient if the variable indicating that there is still tumor residue in the skin is set to `true`. Additionally, the value ranges of the variables prevent the generation of unrealistic values. Moreover, specifying dependencies between variables leads to traces, where the variables' values in the trace also make sense when considered as a whole. For example, the variable representing the patient's cancer stage in

[4] `https://doi.org/10.4121/uuid:01eaba9f-d3ed-4e04-9945-b8b302764176`

the used model is dependent on several other variables, such as the presence of metastases or tumor thickness. The transition weights improved the realism of the event log data by influencing which transitions are triggered more frequently during random trace generation and thus which activities occur more frequently.

However, it was found that the semantic information provided for implementation was insufficient to some extent. While many dependencies were correctly integrated, the domain experts noted that some dependencies between the variables were not or not correctly integrated. This is due to the large number of variables in the model and the resulting large amount of time and expert knowledge required to model the complex dependencies. Besides, the current implementation of the approach has some limitations in terms of semantic information that can be modeled. Currently, for example, only the lead time of activities and the delay between them can be specified, resulting in some events with unrealistic timestamps. For instance, medical procedures are usually performed by day, except emergencies. Currently, it is not yet possible to define time periods during which an activity may take place. Besides functional extension, the configuration for log generation and the state of the model are also crucial for the quality of the log. Configuration requires a high level of domain knowledge and a lot of time to properly address all dependencies. The more thoroughly this step is performed, the better the resulting log will be. The process model is essential as a basis for generating synthetic data. Therefore, it must be examined in detail together with domain experts and checked whether all circumstances are represented correctly and all dependencies are taken into account.

In summary, generating synthetic event logs using the SAMPLE approach produces semantically more realistic data. Nevertheless, there are challenges in the definition of the dependencies, the integration of all necessary semantic data into the implementation, as well as in the effort of modeling the dependencies.

7 Conclusion

The presented SAMPLE approach of multi-perspective log generation sets itself apart from previous approaches by considering data semantics. The evaluation showed that the approach leads to medical event logs with higher quality, since the data perspective is implemented more realistically. Nevertheless, challenges were identified in the analysis of the results that need to be addressed.

In the future, we want to simplify the generation of logs and reduce the effort required. Therefore, we investigate possibilities to further automate the process and to optimize the configuration process. Furthermore, the semantic information identified as missing will be integrated into the approach. We also want to extend the approach to declarative process models. Future enhancements of the approach should enable the generation of more realistic logs. The evaluation of the domain experts is essential and guides the improvement of the approach.

References

1. vanden Broucke, S., Vanthienen, J., Baesens, B.: Straightforward petri net-based event log generation in prom. SSRN Electronic Journal (01 2014)
2. Burattin, A.: Plg2: Multiperspective processes randomization and simulation for online and offline settings. ArXiv **abs/1506.08415** (2015)
3. van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H.M.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: The prom framework: A new era in process mining tool support. In: Applications and Theory of Petri Nets 2005. pp. 444–454 (2005)
4. Esgin, E., KARAGOZ, P.: Process profiling based synthetic event log generation. In: International Conference on Knowledge Discovery and Information Retrieval. pp. 516–524 (01 2019)
5. Grüger, J., Geyer, T., Kuhn, M., Braun, S., Bergmann, R.: Verifying guideline compliance in clinical treatment using multi-perspective conformance checking: A case study. In: Munoz-Gama, J., Lu, X. (eds.) Process Mining Workshops. pp. 301–313. Springer International Publishing, Cham (2022)
6. Jensen, K., Kristensen, L.M., Wells, L.: Coloured petri nets and cpn tools for modelling and validation of concurrent systems. International Journal on Software Tools for Technology Transfer **9**(3), 213–254 (2007)
7. Kataeva, V., Kalenkova, A.: Applying graph grammars for the generation of process models and their logs. In: Proceedings of the Spring/Summer Young Researchers' Colloquium on Software Engineering proceeding (01 2014)
8. Kummer, O., et al.: An extensible editor and simulation engine for petri nets: Renew. In: Applications and Theory of Petri Nets 2004. pp. 484–493. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
9. de Leoni, M., van der Aalst, W.M.P.: Data-aware process mining. In: Shin, S.Y., Maldonado, J.C. (eds.) Proceedings of the 28th Annual ACM Symposium on Applied Computing. p. 1454. ACM Digital Library, ACM, New York, NY (2013)
10. Mannhardt, F.: Multi-perspective process mining. Ph.D. thesis, Mathematics and Computer Science (Feb 2018)
11. Medeiros, A., Günther, C.: Process mining: Using cpn tools to create test logs for mining algorithms. In: Proceedings of the Sixth Workshop on the Practical Use of Coloured Petri Nets and CPN Tools (CPN 2005) (01 2004)
12. Mitsyuk, A.A., Shugurov, I.S., Kalenkova, A.A., van der Aalst, W.M.: Generating event logs for high-level process models. Simulation Modelling Practice and Theory **74**, 1–16 (2017)
13. Munoz-Gama, J., et al.: Process mining for healthcare: Characteristics and challenges. Journal of Biomedical Informatics **127**, 103994 (2022)
14. Nakatumba, J., Westergaard, M., van der Aalst, W.M.: Generating event logs with workload-dependent speeds from simulation models. In: International Conference on Advanced Information Systems Engineering. pp. 383–397. Springer (2012)
15. Pertsukhov, P., Mitsyuk, A.: Simulating petri nets with inhibitor and reset arcs. Proceedings of ISP RAS **31**, 151–162 (10 2019)
16. Ratzer, A.V., et al.: Cpn tools for editing, simulating, and analysing coloured petri nets. In: International conference on application and theory of petri nets. pp. 450–462. Springer (2003)
17. Shugurov, I., Mitsyuk, A.: Generation of a set of event logs with noise. Institute for System Programming of the Russian Academy of Sciences (2014)
18. van der Aalst, W.: Data science in action. In: van der Aalst, W. (ed.) Process Mining, pp. 3–23. Springer, Berlin, Heidelberg (2018)