Concept Drift Log Generator Tool (CDLG)

A Tool for the Generation of Event Logs with Concept Drifts

TUTORIAL

This tutorial provides a step-by-step demonstration of how to use the CDLG tool. The tool itself can be accessed through our repository: https://gitlab.uni-mannheim.de/processanalytics/cdlg_tool. Before using the tool, clone the project and follow the provided installation instructions.

Reference: "*CDLG: A Tool for the Generation of Event Logs with Concept Drifts"* by Justus Grimm, Alexander Kraus, and Han van der Aa, submitted to the demo track of BPM 2022.

Execution modes

The CDLG tool can be used in three different execution modes:

1. Terminal mode
2. Parameter-file mode
3. Python package

1. Terminal mode

The terminal mode is an advanced mode that provides the highest degree of customization to a user. This mode starts by allowing users to define the process models that should be used for a drift scenario. The process models can be imported, randomly generated, or generated according to desired characteristics. Then, the tool collects general information about the drift scenario. For any drift type, users can use imported or generated process trees to define the process models before and after each drift. In addition, users can use the controlled evolution of any process tree, which allows to have full control over the drift changes using the tool. The controlled process tree evolution is done in a simple interactive dialog with users providing visualized interim process trees. Finally, users can decide to insert noise and, if needed, add another drift to the log. Note that the terminal mode provides default settings for all options, which means that users have considerable freedom when establishing drift scenarios but can skip customization of certain aspects if desired.

To demonstrate the terminal mode, we provide a step-by-step demonstration of how to generate an event log with a sudden drift.

1. To start the terminal mode, run the python file "start_generator_terminal.py":

```
cdlg_tool>python start_generator_terminal.py
```

2. Specify if there are models that should be imported or if the tool should generate random models:

```
Import models or use randomly generated models [import, random]: random
```

The user can randomly generate or import the required number of process trees that are needed to generate a target drift scenario. The tool can also import block-structured BPMN models and Petri nets that can be transformed into a process tree.

3. Decide if the model after a drift should be randomly generated or if it should be an evolution of the initial model

```
Evolution of one randomly generated model or use of two randomly generated models [one_model, two_models]: one_model
```

The option “one model” allows to control the evolution of the initial process model. Using this option, users can create drifts with the desired changes. The option “two models” means that the process trees before and after the drift will be randomly generated, or it will be provided by a user.

4. Choose if the generation of the initial random model should be controlled or not:

```
Do you want to adjust the various settings/parameters for the process tree, which will be used to generate the model randomly [yes, no]?
```

If the initial model is not provided by a user, the tool allows controlling the generation of a random process tree using the PTandLogGenerator tool. A detailed description of all parameters can be found in the respective paper[1]:

```
Do you want to adjust the various settings/parameters for the process tree, which will be used to generate the model randomly [yes, no]? yes
MODE: most frequent number of visible activities (default 20): 20
MIN: minimum number of visible activities (default 10): 10
MAX: maximum number of visible activities (default 30): 30
SEQUENCE: probability to add a sequence operator to tree (default 0.25): 0.25
CHOICE: probability to add a choice operator to tree (default 0.25): 0.25
PARALLEL: probability to add a parallel operator to tree (default 0.25): 0.25
LOOP: probability to add a loop operator to tree (default 0.25): 0.25
OR: probability to add an or operator to tree (default 0): 0
SILENT: probability to add silent activity to a choice or loop operator (default 0.25): 0.25
DUPLICATE: probability to duplicate an activity label (default 0): 0
LT_DEPENDENCY: probability to add a random dependency to the tree (default 0): 0
INFREQUENT: probability to make a choice have infrequent paths (default 0.25): 0.25
NO_MODELS: number of trees to generate from model population (default 10): 10
UNFOLD: whether or not to unfold loops in order to include choices underneath in dependencies: 0=False, 1=True
        if lt_dependency <= 0: this should always be 0 (False)
        if lt_dependency > 0: this can be 1 or 0 (True or False) (default 10): 0
MAX_REPEAT: maximum number of repetitions of a loop (only used when unfolding is True) (default 10): 10
```

The option “no” allows to skip these configurations and to use a default setting that leads to a simple, middle, or complex process tree:

```
Do you want to adjust the various settings/parameters for the process tree, which will be used to generate the model randomly [yes, no]? no
Complexity of the process tree to be generated [simple, middle, complex]: middle
```

5. Provide input for a target drift scenario by specifying the drift type, the total number of traces, and the drift moment:

```
--- INPUT DRIFT ---
Type of concept drift [sudden, gradual, recurring, incremental]: sudden
Number of traces in the event log (x >= 100): 20000
Starting point of the drift (0 < x < 1): 0.5
```

Depending on the drift type, different parameters need to be provided.

6. Select the model evolution option:

```
Controlled or random evolution of the process tree version [controlled, random]: controlled
```

If needed for the concept drift scenario, the terminal mode provides a full control over the evolution of a process model using the four main transformation steps: change activity, change operator, change tree fragment, delete silent activities. Using these steps, the user can modify any generate process or an imported process tree. This is done using a visualization of a process tree and an interactive dialog with the tool through the terminal.

[1] Jouck, Toon, and Benoît Depaire. "PTandLogGenerator: A Generator for Artificial Event Data." *BPM (Demos)* 1789 (2016): 23-27.

For example, the terminal commands below transform the process tree in Figure 1 to the process tree in Figure 2 by replacing the “xor loop” operator with an “and“ operator.

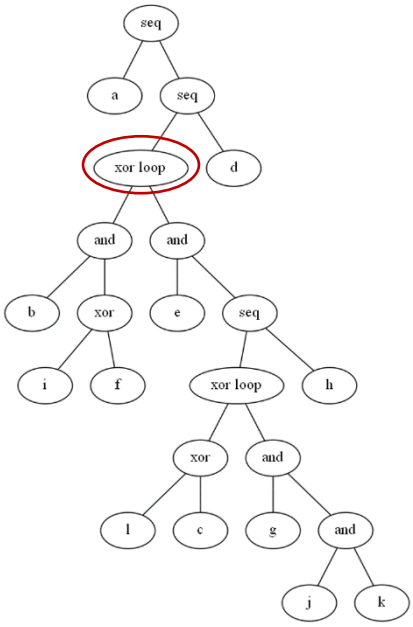

Figure 1: Initial process tree

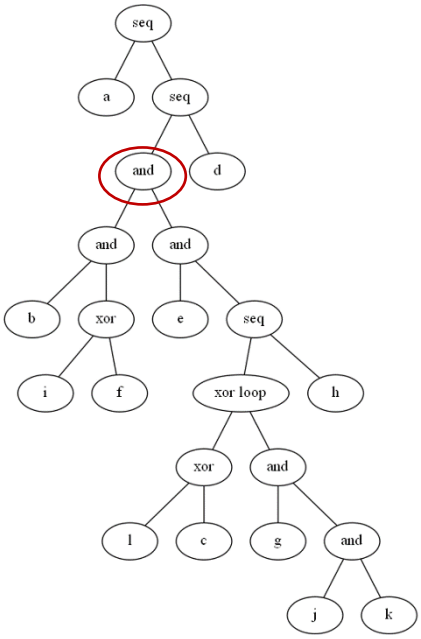

Figure 2: Controlled evolution of the initial process tree

The replacement of the operator is done through the selection of the change type:

```
Operator for the 1. intermediate evolution of the process tree version [change_activity, change_operator, change_tree_fragment, delete_silent_ac]: change operator
```

After the change type is set, the tool locates the change in the process tree via a dialog:

```
Procedure for changing operator/s in the process tree version [replace_op, swap_ops]: replace_op
Existing operator to be replaced [xor, seq, or, xor loop, and]: xor loop
Depth of the node with the operator to be changed (int): 2
New operator [xor, seq, or, xor loop, and]: and
Do you want to continue with the evolution of the version [yes, no]:
```

Finally, the user can skip the interactive process tree evolution and choose a random process evolution by specifying the proportion of activities that should be affected by a random evolution:

```
Controlled or random evolution of the process tree version [controlled, random]: random
Proportion of the activities in the process tree to be affected by random evolution (0 < x < 1): 0.2
```

7. Define whether another drift should be added to the log or not

```
Do you want to add an additional drift to the event log [yes, no]?
```

If “yes” is selected, then the tool moves to step 5.

8. The final configuration allows to add noise to the event log

```
Do you want to add noise to the event log [yes, no]? no
```

If option “yes” is selected, then users can insert noise to the event log. The noise can be created from an existing model or from a completely new randomly generated model. Then, users can select any time moment when the noise should be inserted, e.g., during a drift period, directly after or before, or it can be inserted without any inference with the drift.

```
Do you want to add noise to the event log [yes, no]? yes

--- INFORMATION FOR THE INTRODUCTION OF NOISE ---
The noise will be randomly distributed in the sector to be determined.
The proportion of noise for this sector can also be predefined.
This gives the possibility to either disguise the drift by placing the noise around/inside the drift or to fake another drift by placing the noise away from the drift.
The following two types of noise exist:
        changed_model: the initial model version is changed randomly, whereby the proportion of changes in the version can be specified.
        random_model: a completely new model is used, which means that there is little or no similarity to the other model versions.
The created model, which will appear as an image, will be used to generate traces introduced in the event log as noise.

Type of model for generating the noise [changed_model, random_model]: random_model
Start point of noise in the event log (0 <= x <= 1): 0.7
End point of noise in the event log (0 <= x <= 1): 0.8
Proportion of noise in the set sector of the event log (0 < x < 0.5): 0.1
```

Finally, the created event log is stored in the folder "data/generated_logs/" with a unique name:

```
Resulting event log stored as data/generated_logs\terminal_log_1653481470.xes
```

All relevant drift information is stored as a log attribute in the generated xes-file.

2. Parameter-file mode

The parameter-file mode allows to generate an event log with a concept drift using a single text file that specifies all relevant parameters. In contrast to the terminal mode, users do not have to go through all possible configurations via a dialog in a terminal. This mode provides an option to generate a single event log with a concept drift of interest or a collection of event logs with different drift scenarios.

2.1 Single event log

User can generate a single event log with a specific concept drift by using a text file with relevant parameters. Table 1 shows the controllable parameter for each drift type. The parameter-files are stored in the folder "*data/parameters*".

Table 1: Parameter-files for each drift type

Sudden drift	Gradual drift	Incremental drift	Recurring drift
Complexity_Random_Tree: middle Timestamp_First_Trace: 20/8/3 8:0:0 Min_Time_Activity_Sec: 120 Max_Time_Activity_Sec: 1200 Number_Traces: 1000 Change_Point_Sudden_Drift: 0.4 Proportion_Random_Evolution: 0.3 Start_Sector_Noise: None End_Sector_Noise: 0.5 Proportion_Noise_In_Sector: 0.05 Type_Noise: random_model	Complexity_Random_Tree: middle Timestamp_First_Trace: 20/8/3 10:0:0 Min_Time_Activity_Sec: 120 Max_Time_Activity_Sec: 1200 Number_Traces: 1000 Start_Point_Gradual_Drift: 0.3 End_Point_Gradual_Drift: 0.6 Change_Type: linear Proportion_Random_Evolution: 0.2 Start_Sector_Noise: 0.1 End_Sector_Noise: 0.8 Proportion_Noise_In_Sector: 0.05 Type_Noise: random_model	Complexity_Random_Tree: middle Timestamp_First_Trace: 20/8/3 8:0:0 Min_Time_Activity_Sec: 120 Max_Time_Activity_Sec: 1200 Number_Traces_Initial_Model: 300 Number_Traces_Drift_Models: 100 Number_Traces_Evolved_Model: 300 Number_Intermediate_Models: 3 Proportion_Random_Evolution: 0.2 Start_Sector_Noise: 0 End_Sector_Noise: 1 Proportion_Noise_In_Sector: 0.05 Type_Noise: random_model	Complexity_Random_Tree: middle Timestamp_First_Trace: 20/8/3 8:0:0 Min_Time_Activity_Sec: 120 Max_Time_Activity_Sec: 1200 Number_Traces: 1000 Start_Point_Recurring_Drift: 0.2 End_Point_Recurring_Drift: 0.8 Seasonal_Changes: 3 Proportion_Initial_Model_During_Drift: 0.5 Proportion_Random_Evolution: 0.2 Start_Sector_Noise: 0.1 End_Sector_Noise: 0.9 Proportion_Noise_In_Sector: 0.05 Type_Noise: random_model

To execute the parameter-file mode, user should:

1. Specify the parameters in the corresponding text files placed in "Data/parameters", if needed.
2. Run one of the following commands:
    a. `python generate_log_sudden_drift_from_doc.py`
    b. `python generate_log_gradual_drift_from_doc.py`
    c. `python generate_log_incremental_drift_from_doc.py`
    d. `python generate_log_recurring_drift_from_doc.py`

Note: own models can be imported by specifying their file path after the execution file:
`python filename [path_to_own_model_1] [path_to_own_model_2]`
Each model specifies the process before and after the drift. For an incremental drift, users can only provide the initial model, the resulting model is given by the evolution process.

2.2 Collection of event logs

Finally, the parameter-file mode also provides an option to generate an entire collection of user-defined number of event logs with one concept drift scenarios per log with a single run. This method provides slightly less customization options compared to parameters in Table 2, but can provide users with a large number of event logs suitable for the automated assessment of the concept drift detection accuracy of an approach.

Table 2: Parameter-file for generation of a collection of event logs

```
Complexity_Random_Tree: middle
Number_Event_logs: 20
Number_Traces_Per_Event_Log: 1000
Drifts: sudden;gradual;recurring;incremental
Drift_Area: 0.2-0.8
Proportion_Random_Evolution_Sector: 0.1-0.4
Proportion_Noise_Sector: 0.0-0.1
Timestamp_First_Trace: 20/8/3 8:0:0
Min_Time_Activity_Sec: 120
Max_Time_Activity_Sec: 1200
```

To generate a collection of event logs using the parameter-file mode, user should:

1. Specify the parameters in the text file “parameters_logs” stored in “data/parameters” if needed.
2. Run the following command: `python generate_collection_of_logs.py`

The terminal output:

```
(venv) C:\                          \cdlg_tool>python generate_collection_of_logs.py
Generating 20 logs
exporting log, completed traces :: 100%|████████████████████████████████| 1000/1000 [00:00<00:00, 5794.88it/s]
exporting log, completed traces :: 100%|████████████████████████████████| 1000/1000 [00:00<00:00, 5648.29it/s]
exporting log, completed traces :: 100%|████████████████████████████████| 1000/1000 [00:00<00:00, 5467.71it/s]
exporting log, completed traces :: 100%|████████████████████████████████| 1000/1000 [00:00<00:00, 5731.49it/s]
exporting log, completed traces :: 100%|████████████████████████████████| 1000/1000 [00:00<00:00, 6748.05it/s]
exporting log, completed traces :: 100%|████████████████████████████████| 1000/1000 [00:00<00:00, 5168.43it/s]
exporting log, completed traces :: 100%|████████████████████████████████| 1000/1000 [00:00<00:00, 5149.09it/s]
exporting log, completed traces :: 100%|████████████████████████████████| 1000/1000 [00:00<00:00, 6114.29it/s]
exporting log, completed traces :: 100%|████████████████████████████████| 1000/1000 [00:00<00:00, 5564.49it/s]
exporting log, completed traces :: 100%|████████████████████████████████| 1000/1000 [00:00<00:00, 5501.35it/s]
exporting log, completed traces :: 100%|████████████████████████████████| 1000/1000 [00:00<00:00, 5998.62it/s]
exporting log, completed traces :: 100%|████████████████████████████████| 1000/1000 [00:00<00:00, 5834.03it/s]
exporting log, completed traces :: 100%|████████████████████████████████| 1000/1000 [00:00<00:00, 4327.30it/s]
exporting log, completed traces :: 100%|████████████████████████████████| 1000/1000 [00:00<00:00, 5724.93it/s]
exporting log, completed traces :: 100%|████████████████████████████████| 1000/1000 [00:00<00:00, 6563.61it/s]
exporting log, completed traces :: 100%|████████████████████████████████| 1000/1000 [00:00<00:00, 6114.94it/s]
exporting log, completed traces :: 100%|████████████████████████████████| 1000/1000 [00:00<00:00, 6699.78it/s]
exporting log, completed traces :: 100%|████████████████████████████████| 1000/1000 [00:00<00:00, 5730.73it/s]
exporting log, completed traces :: 100%|████████████████████████████████| 1000/1000 [00:00<00:00, 5646.07it/s]
exporting log, completed traces :: 100%|████████████████████████████████| 1000/1000 [00:00<00:00, 4686.11it/s]
Finished generating collection of 20 logs in data/generated_collections/1653756327
```

In addition to the generated collection of event logs, the tool provides for each event log a text file with used parameters and drift information:

```
--- USED PARAMETERS ---
number of traces: 1000; drift: gradual; start point: 0.49; end point: 0.76; distribution: linear; random evolution: 0.37

--- DRIFT INFORMATION ---
event log: event_log_0; perspective: control-flow; type: gradual; specific_information: linear distribution; drift_start
```

Also, the tool generates a file that stores drift information for each event log in a csv file. This file can be used in the evolution as a gold standard.

Event Log	Drift Perspective	Drift Type	Drift Specific Information	Drift Start Timestamp	Drift End Timestamp	Noise Proportion	Activities Added	Activities Deleted	Activities Moved
event_log_0	control-flow	gradual	linear distribution	3/30/2020 21:34	2020-04-11 03:40:23 (0.76)	0.0785	[Random activity 1]	[c]	[b, a]
event_log_1	control-flow	incremental	3 versions of the process model	4/5/2020 7:02	2020-04-09 17:54:19 (0.76)	0.0476	[]	[f, g]	[]
event_log_2	control-flow	gradual	exponential distribution	3/25/2020 23:58	2020-04-13 10:22:45 (0.79)	0.0287	[Random activity 1, Random activity 2]	[c]	[h, k]
event_log_3	control-flow	sudden	N/A	4/5/2020 15:34	N/A	0.0757	[]	[]	[j, a, g]
event_log_4	control-flow	recurring	5 seasonal changes	3/27/2020 2:56	2020-04-06 09:21:03 (0.77)	0.0272	[]	[a]	[h, k]
event_log_5	control-flow	incremental	6 versions of the process model	3/18/2020 15:16	2020-04-03 10:08:52 (0.68)	0.0012	[Random activity 1, Random activity 2, Random activity 1]	[Random activity 1, f]	[]

3. Python package mode

Finally, the tool has a python package version. The package contains all essential functionality presented in this tutorial. The package can be accessed through our repository:

https://gitlab.uni-mannheim.de/processanalytics/cdlg-package

The repository provides detailed installation instructions and explains all functions and their parameters.