### Concept Drift Log Generator Tool (CDLG)

A Tool for the Generation of Event Logs with Concept Drifts

### TUTORIAL

This project comes as an extension of the initial project CDLG tool that is described in the paper:\_CDLG: A tool for the Generation of Event Logs with concept Drifts\_by Justus Grimm, Alexander Kraus, and Han van der Aa. This tutorial provides a step-by-step demonstration of how to use the CDLG tool. 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.](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.

# The Tool’s Purpose

Overall the tool includes two fundamental features:

1. Generation of a collection of logs with noise and multiple drifts
2. The evaluation of drifts in actual logs by comparing it to tool generated drifts

# Generation of a Collection of Logs

Unlike the previous version, here we opted for a parameter-file mode to generate a collection of logs or single event logs. This method allows to generate a collection of logs with each containing a single or multiple drifts. A predefined set of files are already specified to generate collections with different drift scenarios. The parameter files can be found in “***src\input\_parameters”.***

**(What file are we going to keep here in the final tool version ? Add a screenshot of the files that will be used )**

A screenshot of a computer code

Description automatically generated

# As we can see in this sample parameter file, a new important feature compared to the previous version is the inclusion of multiple drifts per log as depicted by the *Drift\_types* parameter.

# To execute the parameter-file mode user should:

# Specify the parameters in the corresponding text files placed in “Data/parameters”, if needed.

# In the file “***src\configurations.py”***, uncomment the name of the parameter-file that will be used for the collection of logs generation.

# A screenshot of a computer program Description automatically generated

# In this example the parameter file **default\_JK** will be used if we generate a collection of logs.

# Execute the file **“generate\_collection\_of\_logs.py”**.

# The collection of logs generated can be found in **“output”** folder. The folder contains a CSV file named **“drift\_info.CSV”** which is a data frame that summarizes all the logs and drifts generated.

# A screenshot of a computer Description automatically generated

# The folder also contains multiple XES files each one of them stores the information (drift, attributes, traces…) of a single log generated by the tool.

# Evaluation of actual logs

# The evaluation of actual logs refers to the process through which the user compares a collection of logs that he owns with a tool generated one. Overall, the evaluation process can be divided into three steps. The first step consists of generating a collection of logs using the tool (this step can be executed by referring to section 1). Second step consists of “reformatting” the collection of logs owned by the user to a structure that is recognized by the tool. The last step consists of running the evaluation which will compare the two collection of logs and return evaluation metrics. It is important to note that the tool can only evaluate two collection logs if they are in the format of a class **“Collection”** that is generated by the tool. When loading logs used for the automated evaluation a step that converts the data uploaded by the user to a class **“Collection”** is needed as we will see in the more detailed explanation below.

# Reformatting the actual collection of logs

# The user can proceed in two ways to convert his actual collection of logs to a structure that is recognized by the tool.

# First the user can either provide a CSV file with the same structure as the one generated by the tool.

# A screenshot of a computer Description automatically generated

# In this case the user proceeds as described below. First, generate two empty collections of logs. Use the first one to import the user collection from a csv file using the function **“import\_drift\_and\_noise\_info\_from\_flat\_file\_csv(path\_of\_the\_file)”**. This function is a method of the collection class which can be found in **“src/data\_classes/class\_collection”.**

# A screen shot of a computer Description automatically generated

# Next the user should provide a second collection of logs by again either providing a CSV file or a set of XES files, each containing log information.

# Second the user can provide a folder containing a set of XES files. In this case to run the automated evaluation the user proceeds as described in the following figure.

# A screen shot of a computer program Description automatically generated

# The user uses the **Collection** class method **Extract\_collection\_of\_drifts** which extracts log data from a set of XES files to a Collection class.

# Evaluation

# To run the evaluation the user can use the function **Automated\_evaluation**, which needs an actual collection class and a detected collection class, as well as a lag parameter. As a result of this function, an evaluation report will be generated which after a matching of the two collections will generate the level of accuracy, recall, precision and F1 score in a XLS file.

# 

# 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