Feature Aggregation in Process Mining Requirements Specification

Kai Liehr, Inês Cardo, Thomas Hsu

November 2, 2021

Contents

1 Introduction

1.1 Purpose

In this document we will be describing the software requirements for the Feature Aggregation in Process Mining tool and giving an in-depth insight of the the project.
The purpose of this project is to deliver a publicly available Python tool that allows different process-centered analysis methods to be applied more efficiently and with less need for highly qualified process analysts to program their own tools for each occasion.

1.2 Intended Audience and Reading Suggestions

This projects is under the guidance of the course Process Discovery Using Python coordinators therefore this document is useful for the development team, users and the coordinators.

1.3 Product Scope

Given an event log either as XES or CSV file, the tool allows users to:

- Add features to the event log using already existing features
- Filter the event log and perform use case analysis
- Cluster the events
- Visualize the resulting process model

1.4 References

The article introducing this approach implemented in the ProM software:

- M. de Leoni, et al., A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs, Information Systems (2015)

IEEE Recommended Practice for Software Requirements Specifications:

- "IEEE Recommended Practice for Software Requirements Specifications," in IEEE Std 830-1998 , vol., no., pp.1-40, 20 Oct. 1998, doi: 10.1109/IEEESTD.1998.88286.

PM4Py is a process mining package for Python that implements methods of process mining:

- https://pm4py.fit.fraunhofer.de/docs

Django is a Python web framework that aids in the creation of database-driven websites:

- https://docs.djangoproject.com/en/3.2/

2 Overall Description

2.1 Product Perspective

Feature Aggregation is a process discovering tool to provide a more efficient process analysis by adding additional attributes to the event log. The event log can then be filtered and clustered based on the use case analysis. This tool is an alternative to the Java-based *FeaturePrediction* package implemented in the desktop application ProM. In contrast to the *FeaturePrediction* package the tool is Python-based and independent from ProM.

2.2 Product Functions

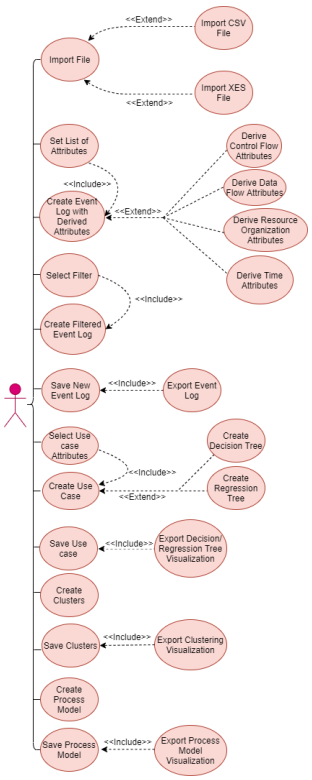

Figure 1: Use Case Diagram

2.2.1 Enrich the event log

The tool should derive attributes based on existing attributes in the event log. These attributes can be separated into four categories:

- Time related attributes
- Workload related attributes
- Control flow
- Data attributes

2.2.2 Use case analysis and cluster events

After the enrichment of the event log the tool should be able to create decision/regression tree based on a use case analysis and cluster the events based on the tree's leaves. The obtained sub-event logs can then be used for further analysis.

2.2.3 Discover and visualize the process model

Lastly the tool should discover and visualize the obtained process model so that correlations between the different sub-event logs can be easier observed.

2.3 User Classes and Characteristics

Overall almost everyone working in the field of process mining can benefit from using the tool because attribute derivation, use case analysis, clustering and discovering and visualizing process models are basic task in this field. The user class that benefits the most are those people analyzing different process characteristics for example the cause of bottlenecks or frequent deviations. They can use every feature of the tool to created and compare process models of different sub-event logs.

2.4 Operating Environment

The tool will be operating in a docker container environment and will therefore be independent on the host operating system. It uses the following software components:

- Python (v3.10.0)
  - Python is a high-level programming language. Due to its overall popularity, especially in the field of Data Science and Process Mining, python profits from many open source libraries.
- PM4Py (v2.2.15)
  - PM4Py is an open source process mining library for python and includes many implementations for process discovery, conformance checking and many other process mining related tasks.
- Django (v3.2.8)
  - Django is a python-based open source web framework.
- Docker (v20.10.7)
  - Docker is a software that allows the user to separate an application from the hosts infrastructure into an isolated environment called container. These containers can have their own specific software, libraries and configuration. Multiple containers can run on the same operating system kernel.

2.5 Software development tools

The organizational aspect of the tool development is supported by several application. The different tasks in the development is managed by *Trello*, a web-based task-board. The source code management is handled by *GitHub*, which offers distributed version control. Do to the small group size of three people and the length of the projects development being only 3 sprints of roughly 2 weeks each, we decided to forego the usage of a requirements analysis tool. Tools like JIRA or ReQtest offer useful features such as requirements traceability and documentation all the way up to and including the testing phase, but the overhead was just not worth any organizational gains.

2.6 Design and Implementation Constraints

Their are several design and implementation constrains that are listed as followed:

- The imported event log should in XES or CSV format.
- The tool should be Python based. Therefore functions and algorithms the tool uses should be implemented in Python using the PM4Py library.
- The implemented algorithms and functions should be efficient to prevent unnecessary computation time and be comprehensible for the user. The tool should also be adaptable to the user so additional attributes can be derive based on specific needs.

2.7 User Documentation

The tool should be easy to use for everyone working in the field of process mining. For that an user manual will be provided on how to access the different functions of the tool.

2.8 Assumptions and Dependencies

It is assumed that the tool are based on many algorithms and functions that are already included in PM4Py. In addition the imported event log should be in XES or CSV format with valid syntax. The tool is dependent on the PM4Py and Django library

3 External Interface Requirements

In this chapter we summarize how the tool will interface with other components.

3.1 User Interfaces

The user interface shall be implemented using Django, a python-based free and open-source web framework.

To better explain the user interface we developed a mock UI, which is a graphic representation of what the final web page will look like.

Home Page
When the user access the web page he will first see the Home page. This page offers a user guide with an introduction about the tool and explanations on the functionalities of the tool.

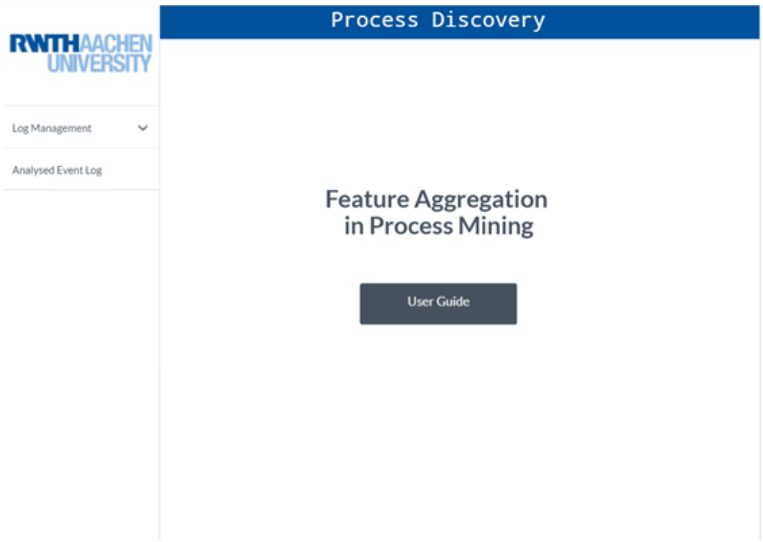

Figure 2: User Interface: Home Page

Import Page
The Import page allows the user to import his event log in the form of XES or CSV file and returns an error message, in case no file was added or is not an XES/CSV file, and a success message otherwise.

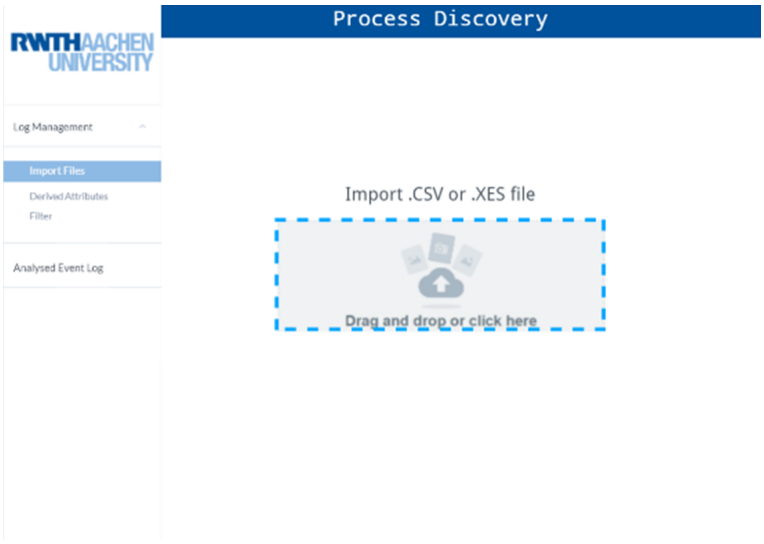

Figure 3: User Interface: Import Page

Derived Attributes Page
The Derived Attributes page allows the user to choose a list of attributes to be derived from the already existing attributes, to generate a new event log with the additional attributes by selecting the button 'Create Event Log' and, finally, it allows the user to download the new event log, with the additionally derived attributes, by selecting the button 'Download Event log'.

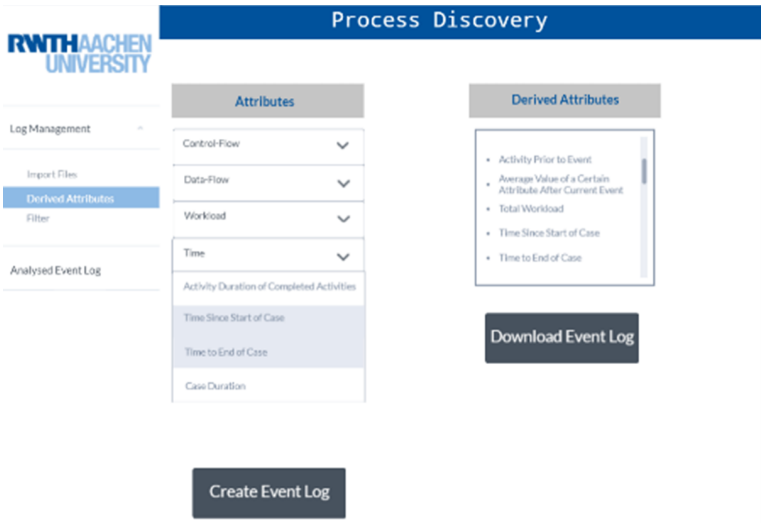

Figure 4: User Interface: Derived Attributes Page

Filtering Page
The Filter page allows the user to filter the event log by choosing a filter from a list of possible filters and selecting the 'Filter Event Log' button, the user can also download the filtered event log by selecting the button 'Download Filtered Event log'.

Figure 5: User Interface: Filtering Page

Analysed Event Log Page
In the Analysed Event Log page the user can define an analysis use case. The user will be able to choose a dependent attribute and any number of independent attributes from a list of the attributes from the event log, including any previously derived ones. Additionally the user can select the 'Use Case Analysis' button and be redirected to the Decision/Regression Tree page, the 'Clustering' button and be redirected to the Clustering page or the 'Process Model' button and be redirected to the Process Model page.

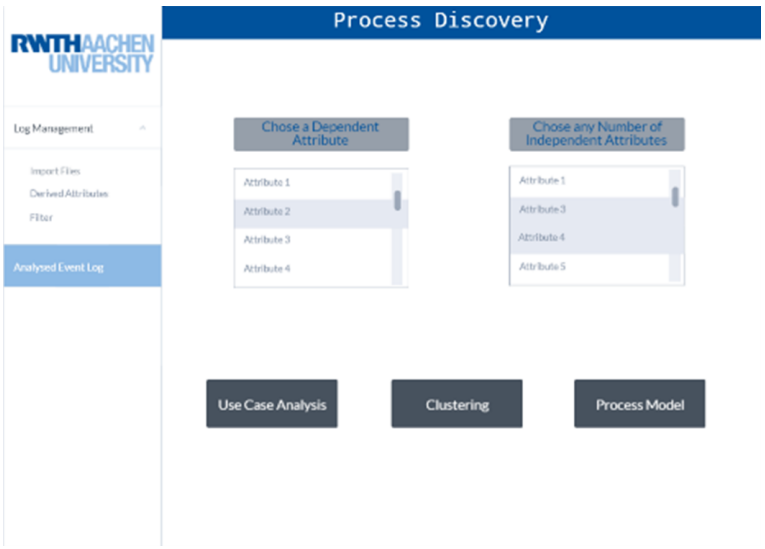

Figure 6: User Interface: Analysed Event Log Page

Decision/Regression Tree Page
In the Decision/Regression Tree page the user can visualize and download, by selecting the 'Export Image' button, the decision or regression tree created based on the use case defined on the previous page. The user can also go back to the Analysed Event Log page to explore the two other available visualizations, the clusters and the process model.

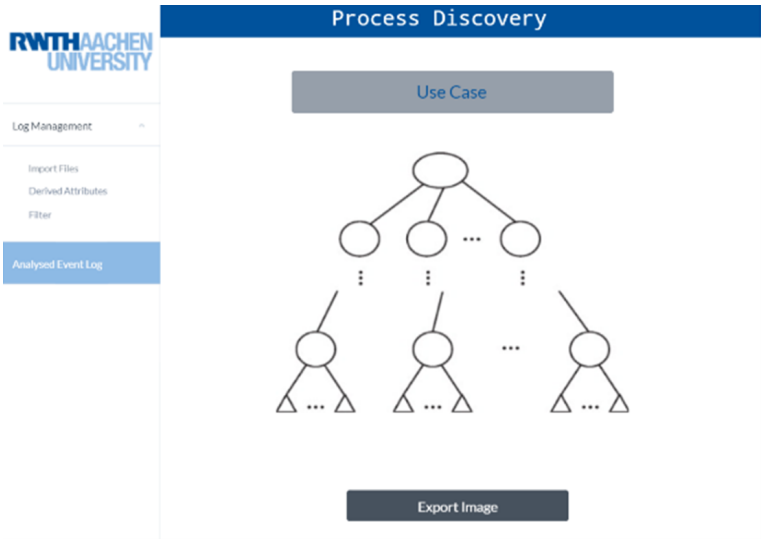

Figure 7: User Interface: Decision/Regression Tree Page

Clustering Page
In the Clustering page the user can visualize and download, by selecting the 'Export Image' button, the clustering based on the decision or regression tree. The user can also go back to the Analysed Event Log page to explore the two other available visualizations, the decision or regression tree and the process model.

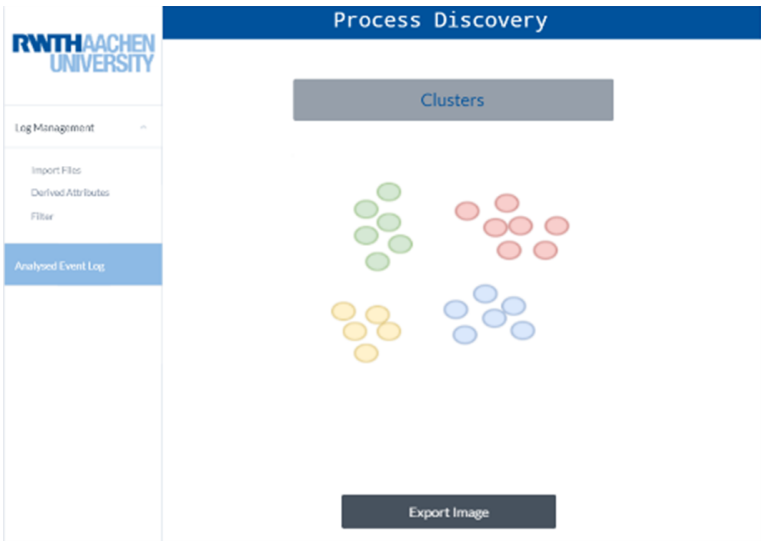

Figure 8: User Interface: Clustering Page

Process Model Page
In the Process Model page the user can visualize and download, by selecting the 'Export Image' button, the discovered process models.

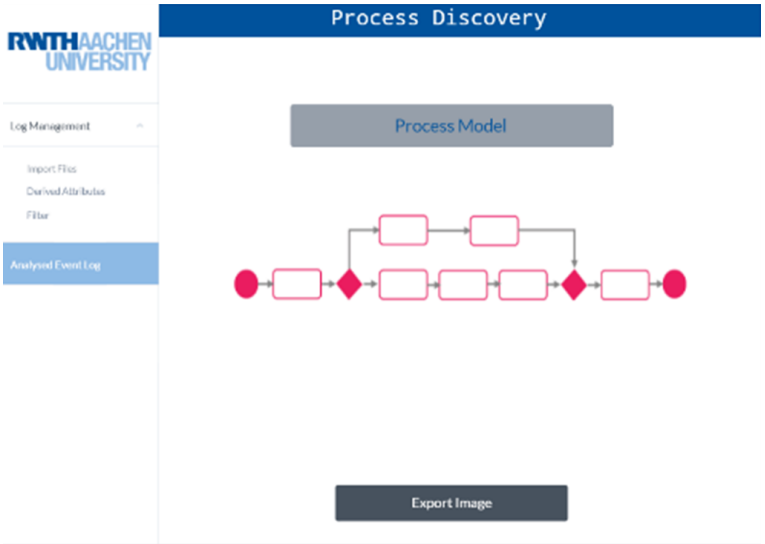

Figure 9: User Interface: Process Model Page

3.2 Hardware Interfaces

Since the web application does not need any direct hardware interfaces it only needs either Mac, Windows or linux operating systems default web browsers.

3.3 Software Interfaces

To implement the project we have chosen python language for its more interactive support and simplified syntax.
We will use PM4Py which is a python library that supports the latest and most useful process mining algorithms in python for the the main functionalities of this tool.
For web development we will use Django which is a high-level python web framework.
The tool can be used directly in python or via Docker which is a software platform that enables developers to package applications into containers that contain everything needed to run the application.

4 System Features

In this chapter we go over the functional requirements ordered by system feature.

4.1 Import XES or CSV

4.1.1 Description

The user can import an event log as XES or CSV into the tool.

4.1.2 Stimulus/Response Sequences

The user may click a button to import an XES or CSV file from memory space or another to delete currently imported ones. To proceed the tool requires the user to have currently imported at least one event log.

4.1.3 Functional Requirements

- Reads into program memory XES or CSV event log
- User interface with buttons for importing/deleting event logs, list of currently imported ones and a button to proceed to the next step described in 4.2 "Adding derived attributes"
- Error handling if user tries to proceed without any currently imported event logs or tries to import files that are neither in XES or CSV format

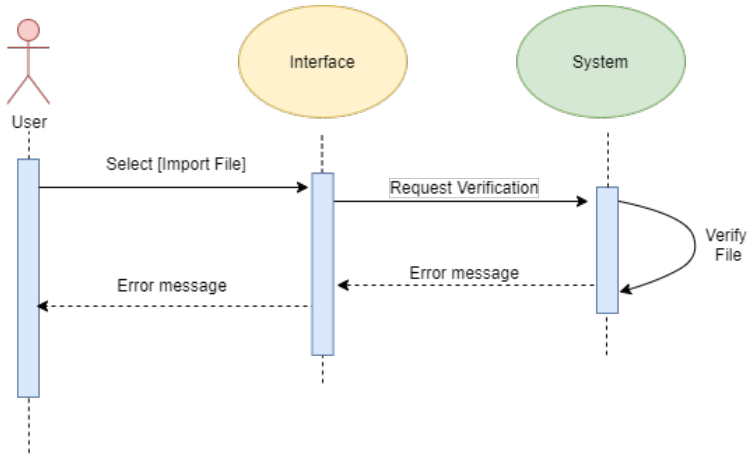

Figure 10: Sequence Diagram for the Import XES or CSV Feature

4.2 Adding derived attributes

4.2.1 Description

For a given event log add attributes to each event that can be derived from the event log. This is the core functionality of the tool and as such high priority.

4.2.2 Stimulus/Response Sequences

The interaction between user and tool starts with the user giving an event log as XES or CSV file, the system responds by asking whether derived attributes should be added. Once the user confirms this, a list of possible attributes to derive is given, from which the user may choose any number of iteratively. These will then be derived and subsequently added into the event log. If the user declines to add attributes, go straight to the second system feature 4.3 "Filtering of event log ".

4.2.3 Functional Requirements

- ATTR-1: User interface for choosing which attributes to add if any
- ATTR-2: Error handling if no file is given but user tries to proceed
- ATTR-3: Functions to compute the various attributes listed in the table 1

ID	Description	Type
C1	Number of times a certain activity is executed before an event	Control-Flow
C2	Next activity after event constrained by an activity set	Control-Flow
C3	Activity prior to event	Control-Flow
D1	Last assigned value of a certain attribute prior to event	Data-Flow
D2	Last assigned value of a certain attribute after current event	Data-Flow
D3	Average value of a certain attribute after current event	Data-Flow
D4	Max value of a certain attribute after current event	Data-Flow
D5	Min value of a certain attribute after current event	Data-Flow
D6	Sum value of a certain attribute after current event	Data-Flow
R1	Workload of event resource at time of event	Workload
R2	Total Workload	Workload
T1	Activity duration of completed activities	Time
T2	Time since start of case	Time
T3	Time to end of case	Time
T4	Case duration	Time

Table 1: Table of derivable attributes.

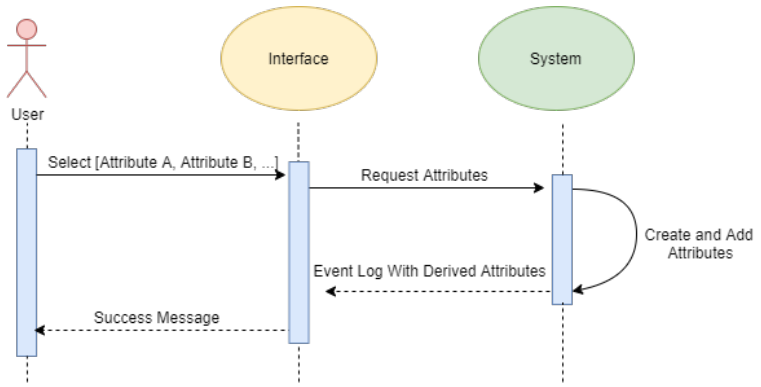

Figure 11: Sequence Diagram for the Adding Derived Attributes Feature

4.3 Filtering of the event log

4.3.1 Description

For the purpose of performing an analysis for the analysis use case, the tool allows the log to be filtered first.

4.3.2 Stimulus/Response Sequences

The user is presented with a display of filter options such as selecting only last events or only those with a particular activity. He then chooses a filter and upon confirming the tool proceeds to 4.4 "Defining an Analysis Use Case".

4.3.3 Functional Requirements

- FIL-1: User interface for choosing filters
- FIL-1: Error handling
- FIL-3: Functions to filter an event log as seen in table 2

F1	Keep all events (effectively no filter)
F2	Only keep first event of a case
F3	Only keep last event of a case
F4	Only keep "Complete" events
F5	Only keep "Start" events
F6	Only keep events with a certain activity
F7	Only keep events with a certain resource

Table 2: Table of possible filters for the analysis use case.

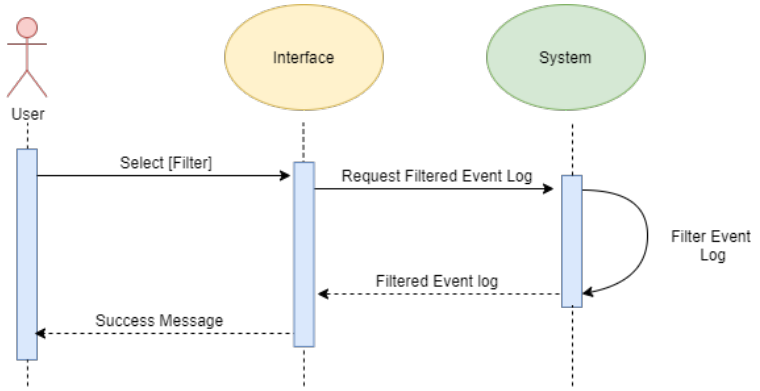

Figure 12: Sequence Diagram for the Filtering of the Event Log Feature

4.4 Defining an Analysis Use Case

4.4.1 Description

Allows the definition of a dependent and a number of independent attributes to figure out correlations between them.

4.4.2 Stimulus/Response Sequences

The user is asked whether or not to define a use case. In case of a yes, the user chooses from a list of attributes of the event log, notably including any previously derived ones, one attribute as the dependent one and any number but at least one as the independent ones. The tool now performs the analysis and creates a decision or regression tree and shows them to the user. The user may now either close the tool or proceed with the results to a new round of analysis starting this interaction from the beginning again or do clustering based on the tree, with each leaf leading to a cluster (see 4.5 "Clustering"). If the user declines to define an analysis use case, he may still proceed to clustering or leave the tool.

4.4.3 Functional Requirements

- USE-1: User interface for defining analysis use case
- USE-2: Error handling for invalid use cases, namely more than or no dependent attribute chosen or dependent attribute in set of independent ones or no independent attributes chosen at all
- USE-3: algorithm for performing use case analysis
- USE-4: Visualization of decision and regression trees in UI

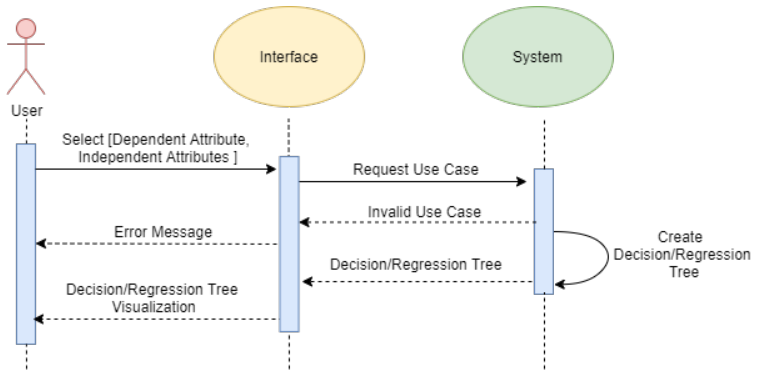

Figure 13: Sequence Diagram for the Defining an Analysis Use Case Feature

4.5 Clustering

4.5.1 Description

The previously performed use case analysis created either a regression or decision tree, whose leaves are now the basis of splitting the event log into sublogs. Then process models can be constructed for each sublog.

4.5.2 Stimulus/Response Sequences

The user chooses to proceed with clustering after having done a use case analysis. The tool then creates on the basis of the tree and event log a cluster for every leaf of the tree. To make homogeneous process models more likely some outlier traces may be pruned. Finally, the tool outputs the discovered process models to the user.

4.5.3 Functional Requirements

- CLU-1: Use prediction tree to split all traces of event log into sublogs
- CLU-2: Filter outlier traces
- CLU-3: Discover process models for each sublog

- CLU-4: Visualize process models

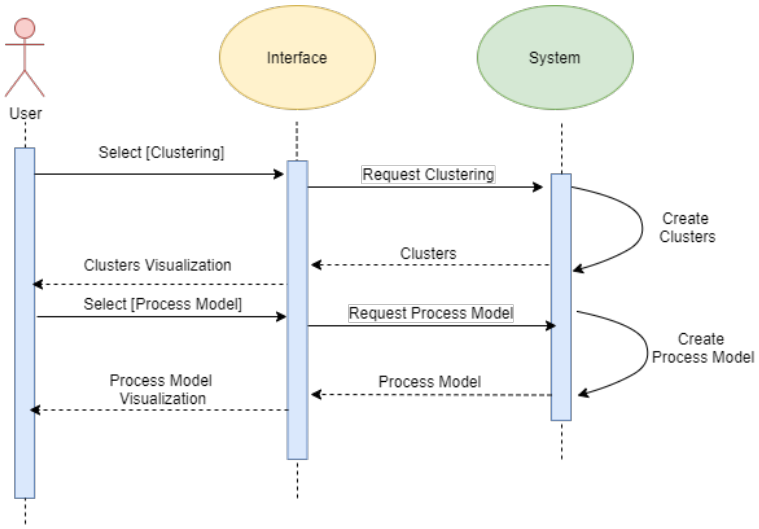

Figure 14: Sequence Diagram for the Clustering Feature

5 Appendix

The class diagram in Figure 15 describes the structure of the system by showing the different system classes. The *UI Handler* class manage the interaction between the program and the user. It take the users specific needs, such as the filter option or the wanted additional attributes, as input and calls the corresponding classes. Base on the input given by the user the *Enrichment* class derives the attributes while the *Filtering* class handle the filtering of the event log. The *Decision Tree* class computes the decision tree that is used by the *Clustering* class to cluster the event log and create a process model. The *Files* class contains different in- and export functions.

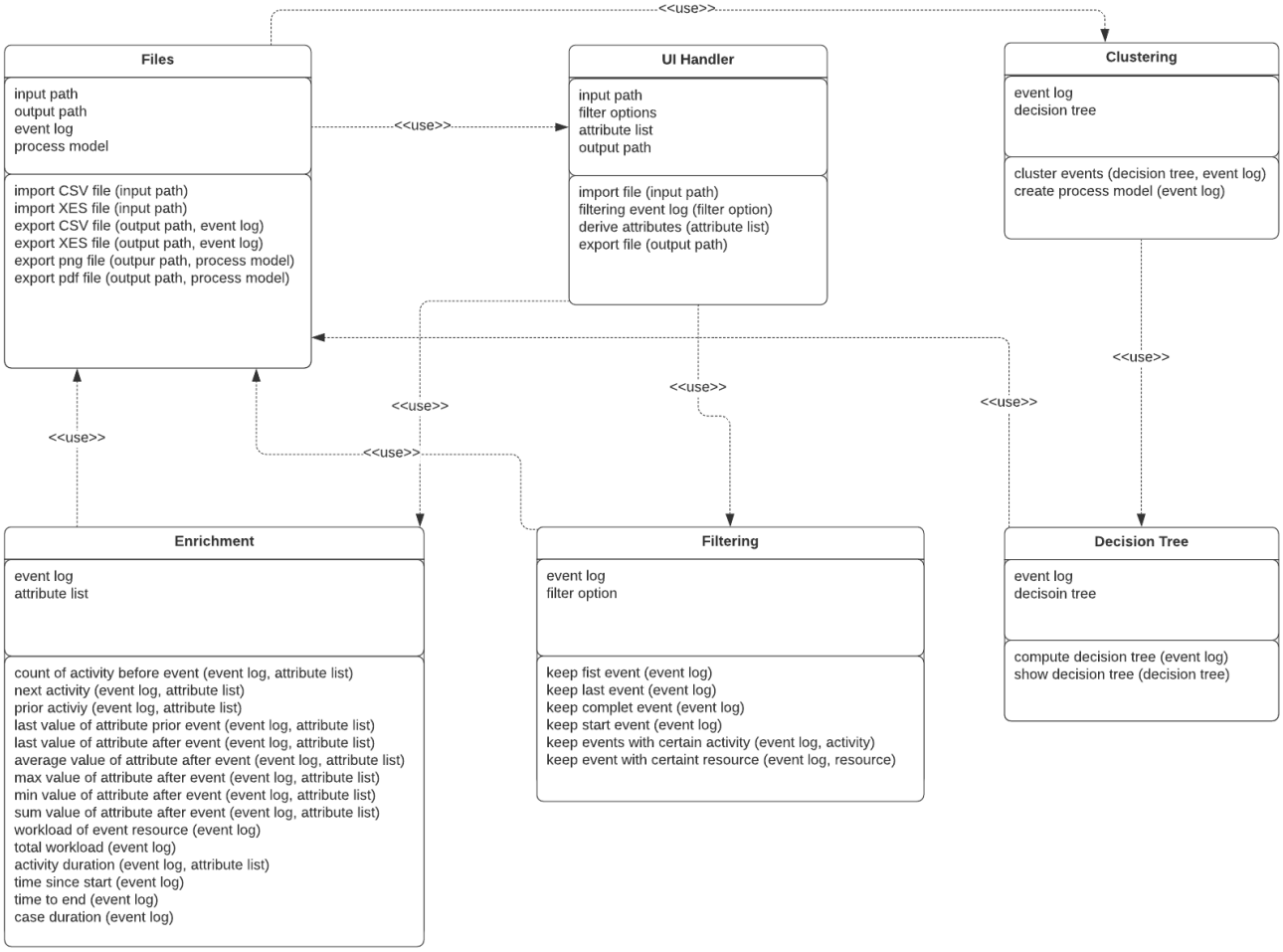

Figure 15: Class diagram for this project