Eindhoven University of Technology

MASTER

Clustering low-level events into activities
a time-based approach

Dündar, S.

Award date:
2018

Link to publication

Disclaimer
This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 19. May. 2025

Technische Universiteit
Eindhoven
University of Technology

Department of Mathematics and Computer Science
Architecture of Information Systems Research Group

Clustering Low-Level Events into Activities: A Time-Based Approach

Master Thesis

S. Dündar, BSc

Supervisors:
dr. M. de Leoni
dr. ir. H. Eshuis
dr. V. Menkovski

Final Version

Eindhoven, October 2018

Abstract

Process mining techniques analyze processes based on event data. A crucial assumption for process analysis is that events correspond to occurrences of meaningful activities. Often, low-level events recorded by information systems do not directly correspond to these. Abstraction methods, which provide a mapping from the recorded events to activities recognizable by process workers, are needed. Existing supervised abstraction methods require process stakeholders to provide a full model of the entire process as input. This Master thesis project has investigated techniques that do not rely on this strong assumption. Data-mining techniques are combined with process mining to cluster sequences of low-level events to generate high-level activities. The centroids of the clusters provide meaningful information for a process stakeholder to identify the high-level activity/ies that correspond to the cluster. To support stakeholder in this identification, visualization techniques are explored, based on heat maps. To ensure that the techniques are feasible and provide benefits in real-life scenarios, they are tested and evaluated with real-life event data, such as with the event data referring to the usage of werk.nl, where unemployed people can apply for available jobs, as well as they can upload and upgrade their curriculum vitae.

Acknowledgments

This Master's thesis is the result of my graduation project which completes my Business Information Systems study at Eindhoven University of Technology. The project was performed internally at the Architecture of Information Systems (AIS) group of the Mathematics and Computer Science department of Eindhoven University of Technology.

First of all, I would like to thank my main supervisor Massimiliano de Leoni for guiding me throughout this Master's project. His enthusiasm and criticism made the project challenging, but definitely joyful. Thanks for the feedback and advice. Your help has been invaluable and has contributed greatly to the work presented in this thesis. Additionally, I would like to thank Rik Eshuis and Vlado Menkovski for joining the assessment committee.

A special thanks to Marcus Dees for his valuable input with regard to the information about UWV. Your input has helped me greatly to get more insight in the dataset used in this thesis and helped be greately with my project.

I thank all of my friends and colleagues for the great and unforgettable time during my study in Eindhoven. And of course, I would like to thank my mother, father, my sister and two brothers for their support throughout the years. Without them I would not have come this far.

Safa Dündar
September 2018

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This Master's thesis is the result of a graduation project for the *Business Information Systems* Master at Eindhoven University of Technology (TU/e). The project is carried out within the *Architecture of Information Systems* (AIS) group of the Mathematics and Computer Science department of TU/e. The AIS research group focuses on process-aware information systems, process modeling and process mining.

In Section 1.1 the thesis context and the problem is discussed. Section 1.2 states the project goal. Lastly, in Section 1.3, the outline of the thesis project is discussed to reach our project goal.

1.1 Thesis Context

In today's business, information systems are an indispensable aspect of the organization. Nearly every event occurred in a company is recorded in these systems. The most known examples of information systems are the Workforce Management (WFM), Customer Relationship Management (CRM), Supply Chain Management (SCM), Enterprise resource Planning (ERP), and Product Data Management (PDM) systems.

Events are recorded in event logs to support business processes. With these event logs, company stakeholders can gain more insight in their business processes. Process mining provides a powerful way to analyze operational processes based on purely raw event data recorded by information systems. Process mining is truly process-oriented, unlike classical data-driven approaches. The purpose is discovering, monitoring and improving real processes and it can be considered as a relatively young research discipline [1].

Often, the events recorded by information systems do not correspond to meaningful business activities, for the reason that they are low-level. In addition, real life processes may have several variants of the same process within an organization. This results in too much variability, yielding a "Spaghetti-like" process model. Examples are web-site navigation, processes in hospitals and healthcare, and online or physical shops. As a more concrete example, consider the browsing behavior on the website of www.werk.nl. Browsing to a particular website can be accomplished by many different ways. If one applies existing discovery techniques to the log data of browsing, one gets such a "Spaghetti-like" model as in Figure 1.1. The consequence is the increased complexity for process stakeholders to gain more insight in their processes.

We start from the believe that events need to be grouped into high-level activities through some abstraction method, which provide a mapping from the recorded events to activities recognizable by process workers. Section 3.4 details the existing research approaches, which are fully supervised and/or do not consider the time dimension. In this thesis, an abstraction based on time is proposed. This approach is beneficial when the behavior can be grouped in true sessions and each session can be abstracted as one single event.

Therefore the problem tackled in this thesis can be summarized as follows:

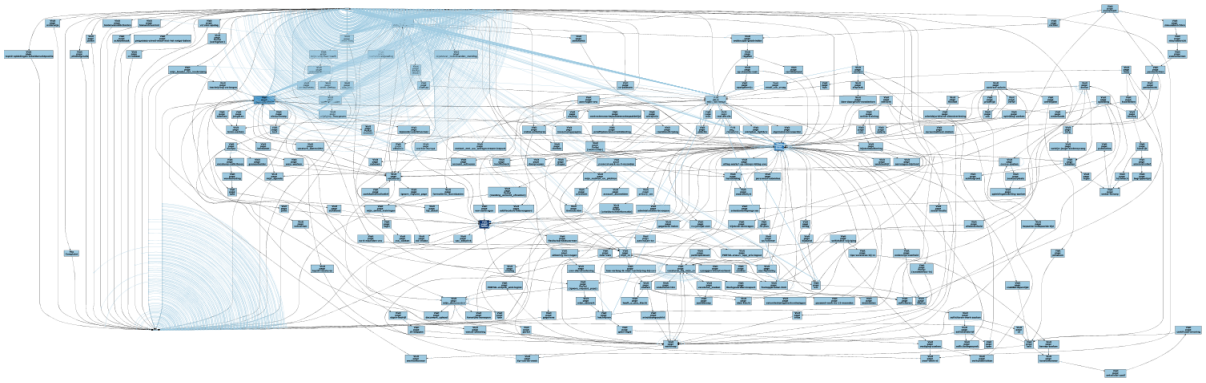

Figure 1.1: An example of a "Spaghetti-like" process model.

Problem Statement: *An abstraction technique from low-level events to high-level activities need to be investigated that 1) do not require an extensive amount of domain knowledge as input (e.g. a set of local process models), 2) is time-based, and 3) is beneficial when time sessions can be identified.*

1.2 Project Goal

The goal of the project can be summarized as follows:

Project Goal: *Develop a technique, whereby data-mining and process mining techniques are combined to cluster sequences of low-level events into high-level activities based on time-based abstraction.*

To achieve this goal, this technique is implemented as a plug-in in ProM. ProM is a process mining framework developed by the Architecture of Information Systems research group at Eindhoven University of Technology. The ProM framework has more than 600 plug-ins, each contributing to research and analysis of the process mining field [2, 7, 25, 28].

Independent of the clustering algorithm, we provide an implementation on K-Means and DBSCAN clustering algorithms for clustering the low-level events intho high-level activities.

The abstraction method is tested on two different datasets. The main focus is on the the real-life event data of UWV. This event data refers to the usage of www.werk.nl, where unemployed people can apply for available jobs, as well as they can upload and upgrade their curriculum vitae. The second dataset is an online retail dataset provided by the UCI Machine Learning Repository.

1.3 Research Method and Outline

To reach the goal of this Master's project, the following steps are taken:

- **Preliminaries**

  In Chapter 2 first, the concept of Process Mining is discussed in detail. Second, the generation of high-level activities from low-level events is discussed. Lastly, some clustering techniques will be discussed, whereby we elaborate on K-Means and the DBSCAN clustering algorithm.

- **Time -and Frequency-Based Abstraction**

  Chapter 3 describes in detail the techniques used to create the high-level event log. In addition, some related works about these techniques is discussed as well.

- **Implementation**

  Chapter 4 elaborates about the implementation of the above mentioned abstraction techniques.

- **Evaluation**

  In Chapter 5, we evaluate our proposed technique. This is done by comparing logs of with and without abstraction. In addition, the comparison between the K-Means and DBSCAN clustering algorithm is discussed in detail.

- **Conclusions**

  In Chapter 6, we conclude the project.

Chapter 2

Preliminaries

This chapter introduces preliminary concepts used throughout this thesis. In Section 2.1 the concept of Process Mining is explained. Section 2.2 concepts of event abstraction is introduced. In Section 2.3 several clustering techniques is considered where the focus is on the K-Means and DBSCAN clustering algorithms, which is needed for event abstraction.

2.1 Process Mining

Data is an indispensable aspect of today's business. Data have to be used in a smart way by companies to get hidden insights and improve existing processes in order to survive. The famous metaphor "Data is the new oil" even emphasizes the importance of data [20]. To lead a successful company, only looking at the data and data mining techniques is not enough. Data mining techniques are typically not process-centric and needs to be related to process analysis. This is the place where *Process Mining* comes into play. Process mining is a research discipline that closes the gap between process modeling on the hand and machine learning and data mining on the other hand (Figure 2.1) [1].

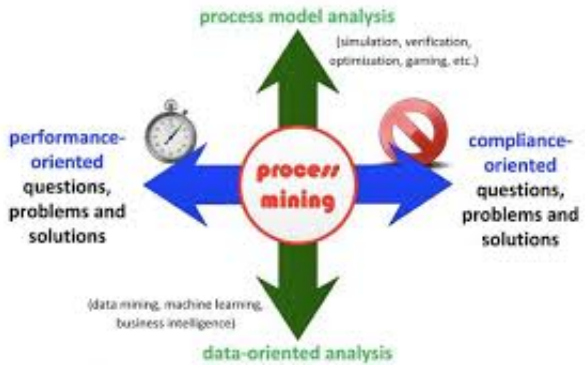

Figure 2.1: The positioning of Process Mining

Real executions of processes are recorded by information systems in event logs. Process mining is the field where *valuable, non-trivial process-related information* is extracted from these event logs [1]. The main idea is to discover, monitor and improve real processes. Process *discovery, conformance* checking and *enhancement* are common applications to gain more insight in process data. Figure 2.2 shows an overview of process mining.

Process Discovery

Discovery consists of analyzing process-centric data to describe the behavior of business processes. This means that, using an event log, a process model is produced by a discovery technique without using a-priori information.

There are four perspectives in process discovery [1]:

- **Control-flow** perspective focuses on explaining the behavior of event logs. An example of a process discovery is a Petri net, produced by the Alpha-algorithm [22], in which the behavior in the event log is explained.
- **Organizational** perspective focuses on information about the involvement and relation of resources hidden in the event log. An example is a social network analysis of resources in the process [21].
- **Case** perspective focuses on properties of cases/instances in an event log.
- **Time** perspective is concerned with the timing and frequency of events. Hereby it is possible, for example, to monitor the processing time of a certain case and do some bottleneck analysis.

Conformance Checking

Conformance concerns with checking if reality, events recorded in event logs, conforms to the process model produced by a discovery technique and vice versa [1]. This is achieved by comparing a process model with an event log of the same process. Thus, conformance checking may be used to detect, locate and explain deviations in the actual process.

The performance of the models can be decomposed into four measures[6]:

- **Fitness** is a measure to quantify the extent to which the discovered model can accurately reproduce the cases recorded in the event log.
- **Precision** is a measure to quantify the fraction of behavior allowed by the model which is not seen in the log.
- **Generalization** is a measure that assesses to which extend the resulting model will be able to reproduce future behavior of the process.
- **Simplicity** is a measure that assesses to which extend the resulting model is easy to understand.

The values for the *fitness, precision* and *generalization* is between 0 and 1. The higher these values, the better the resulting model. For *simplicity* it depends on the calculation of the method.

Enhancement

Enhancement consists of augmenting a process model with extra information about the actual process recorded in an event log. So, it is *not* about conformance but improving an existing process model [1].

To conclude, only looking at the data is not enough. Process mining and data mining need to be combined to answer more advanced questions. Before moving to the next Section, two discovery techniques will be discussed. In this thesis the focus is on the *Visual Miner* and the *Heuristics Miner*.

2.1.1 Inductive Miner

The Inductive Miner is a discovery technique that constructs a *process tree* from an event log [14]. A process tree represents a sound block-structured workflow net. The leaves in the rooted tree are labeled with activities and all other nodes are labeled with operators. There are four standard

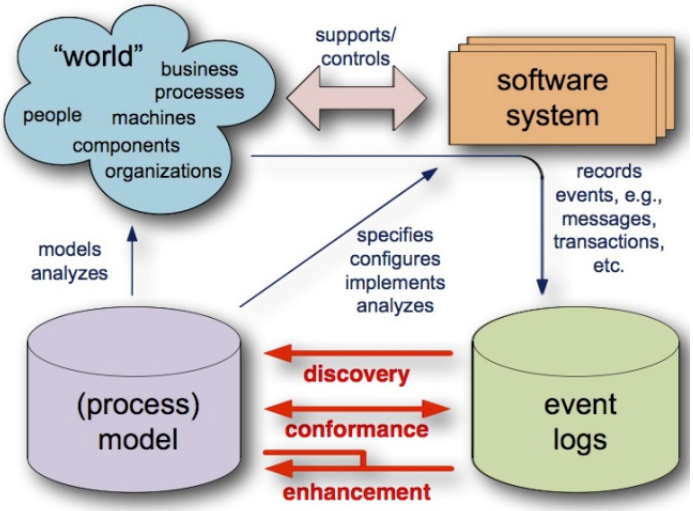

Figure 2.2: Positioning the three main types of process mining: *discovery*, *conformance*, and *enhancement*

operators: operator $\times$ means the exclusive choice, $\rightarrow$ the sequential execution, $\circlearrowleft$ the structured loop and $\wedge$ means a parallel (interleaved) execution.

The Inductive Miner works recursively in four steps. It first selects a *cut*: a division of activities in an event log and an operator. Based on this cut, the log is then split into smaller sub-logs. Recursively, the first two steps are applied on each sub-log until the base case is reached. This is a log containing only a single activity. Lastly, if no cut can be selected, the Inductive Miner returns a *flower model*, allowing for all sequences of activities.

The discovered models by the Inductive Miner are always *sound* (an end state is always reachable and all the process steps can be executed), and the models always fits, i.e., the model can reproduce the events recorded in the event log. In addition, the models are discovered in polynomial time by the Inductive Miner [14].

Extensions of the Inductive Miner exists as well. The first one is the *Inductive Miner - infrequent* (IMi), which takes takes into account the infrequent behavior in a given event log [13]. The second one is the *Inductive Miner - life cycle* (IMlc), which takes the life cycles of activities into account [15].

2.1.2 Heuristics Miner

The Heuristics Miner discovers a process model that describes the control flow perspective that was captured in the given event log, showing the relationship between the activities. The Heuristics Miner allows for finding complex relationships between activities that other basic algorithms like the Alpha-miner are unable to discover. The models that are discovered by the Heuristics Miner, gives a general overview of the process in question [27].
The starting point of the Heuristics Miner starts with the construction of the so-called *dependency graph* (DC). It first constructs a frequency matrix based on the directly follows relation. A directly follows relation is where an activity is directly followed by another activity. By this it can be indicated how certain the strength of the directly follows relation is between two activities A and B. From the frequency matrix, the dependency matrix can be constructed based on a given metric. If the dependency value between two activities in the dependency matrix is above a certain threshold, the dependency graph is constructed for these activities.

The same procedure holds for other relationships, e.g. the loop relation, where an activity is followed by itself. Instead of the directly follows relation, we construct a frequency matrix based

on the other relationships. Then the dependency matrix is constructed based on a metric specific for these relations.

The basic relationships between the activities are [27]:

- Directly follows relation E.g., activity A is directly followed by activity B ($< A, B >$).
- Long distance relation E.g., activity A is indirectly followed by B with other activities occurring in between ($< A, .., B >$).
- Length-one loop E.g., Activity A is followed by itself ($< A, A >, < A, A, A >$).
- Length-two loop E.g., Activity A followed by B is repeated ($< A, B, A >$).

The Heuristics Miner is robust to noise and can deal with low frequent behavior. The downside of this algorithm, is that the Petri Nets created from this algorithm are not sound [27].

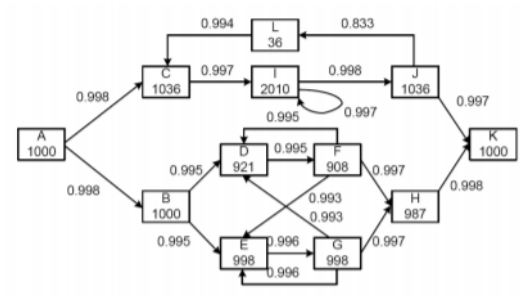

Figure 2.3: Example of a Dependency Graph (DG), adopted from [26].

2.2 From Low-Level Events to High-Level Activities

In this section the concept of an *event, trace* and *event log* is explained as well as the concept of *clustering.*

As mentioned in Section 2.1, real executions of processes are recorded by information systems in event logs. An event log is a collection of sequentially recorded events such that each event refers to an *activity* (i.e., a well defined step in the process) and is related to a particular *case* (i.e., a process instance). A sequence of events belonging to a single case is called a *trace.* Additional information could be stored in an event log such as the *resource* executing the event, the *timestamp* of the event, or other elements recorded in the event (e.g., the quantity of the order).

Definition 1 (Event, Trace, Log)**.** Let $\mathcal{E}$ be a set of events. A ***trace*** $\sigma \in \mathcal{E}^*$ is a sequence of events. An ***event log*** $\mathcal{L}$ consists of a set of traces, i.e. $\mathcal{L} \in \mathcal{E}^*$.

Given an event $e \in \mathcal{E}$, $\lambda_A(e)$ and $\lambda_T(e)$ return the activity associated with the event and the timestamp when the event occurred, respectively. In the remainder, given a (sub-)trace $\sigma' = \langle e_1, \ldots, e_n \rangle$, the first and last events are denoted as follows: $\textsc{First}(\sigma') = e_1$ and $\textsc{Last}(\sigma') = e_n$. Furthermore, given a second (sub-) trace $\sigma'' = \langle f_1, \ldots, f_m \rangle$, $\oplus$ indicates the concatenation of (sub-)trace, i.e. $\sigma' \oplus \sigma'' = \langle e_1, \ldots, e_n, f_1 \ldots, f_m \rangle$.

Definition 2 (Clustering)**.** Let $\mathfrak{S} = (D_1 \times \ldots \times D_N)$ be the set of all data points defined over the cartesian product of domains $D_1, \ldots, D_n$. Given a multiset $S \in \mathbb{B}(D_1 \times \ldots \times D_N)$[1] of data points, a clustering algorithm can be abstracted as function $\textsc{Cluster} \in \mathfrak{S} \rightarrow 2^{\mathbb{B}(D_1 \times \ldots \times D_N)}$, which, returns a clustering of the multiset S, namely namely $\textsc{Cluster}(S) = \{S_1, \ldots, S_n\}$ such that, for all S_i and S_j, $S_i \cap S_j \neq \{\} \Rightarrow S_i = S_j$, and $S_i \cup \ldots \cup S_n \subseteq S$.

[1] Given a set X, $\mathbb{B}(X)$ is the set of all multisets that can be built with elements of X.

2.3 Clustering Techniques

Clustering is an unsupervised learning method where data points are grouped into disjoint clusters. The intracluster similarity is high and intercluster similarity is low. This section will elaborate more on the clustering techniques used for this thesis. In this thesis the focus is on the K-Means and the DBSCAN clustering algorithms. In Subsection 2.3.1 and 2.3.2 The K-Means and DBSCAN clustering algorithms are discussed respectively.

2.3.1 K-Means Clustering Algorithm

The K-Means algorithm, proposed by MacQueen et al. in 1967, is one of the most popular partition method of clustering [16]. K-Means is an unsupervised, non-deterministic, numerical, and iterative method of clustering. K-Means aims at classifying the given data points into k different clusters.

The algorithm consists of two steps. In the first step, k centers (or centroids) are selected randomly, where the value of k is set by the user a-priori. The next step is to associate each data point to the nearest centroid by using a distance measurement, like the *Euclidean distance*.

For each of the cluster, a new mean value is then computed. This process is iterated until the criterion of the sum of squared errors is minimized. The sum of squared errors is defined as:

$$E = \sum_{i=1}^{k} \sum_{x \in C_i} |x - x_i|^2 \tag{2.1}$$

Whereby k is the number of clusters, x the data point and x_i the mean (centroid) of cluster C_i.

The process of the K-Means algorithm is as follows:

Input: Number of clusters k, and dataset $D = \{d_1, d_2, \ldots, d_n\}$ containing n data points.
Output: A set of k clusters.
Steps:

1. Select k data points from dataset D randomly as initial centroids.
2. Compute distance between each data point and the cluster centroid, assign each data point to closest cluster.
3. Recompute each centroid by taking the mean of each data point in that cluster.
4. Repeat steps 2 and 3 until there is no change in the clusters.

Figure 2.4 illustrates the concept of the K-Means clustering algorithm.

There are several drawbacks of the K-Means algorithm. First of all, this algorithm is sensitive to the selection of the initial cluster centers. Wrong clusters can be formed if the cluster centers are not chosen in a proper way.

Next, there is no rule for the decision of value of k. There will be different results for different initial values of k. This problem can be tackled by running K-Means more than one time for each cluster number. The elbow method is a method to find the optimal number of clusters in a dataset. The sum of squared errors (SSE) describes the percentage of variance explained in a cluster. The SSE is 0 when k is equal to the number of data points in the dataset. The reason for this is that each data point is then its own cluster, and there is no error between the center of its cluster. One strives to find the 'elbow' in this graph as in Figure 2.5, where increasing the number k does not has an effect on the SSE anymore.

Lastly, there is no notion of noise and this algorithm can be easily affected by abnormal data points in the dataset. [19, 29].

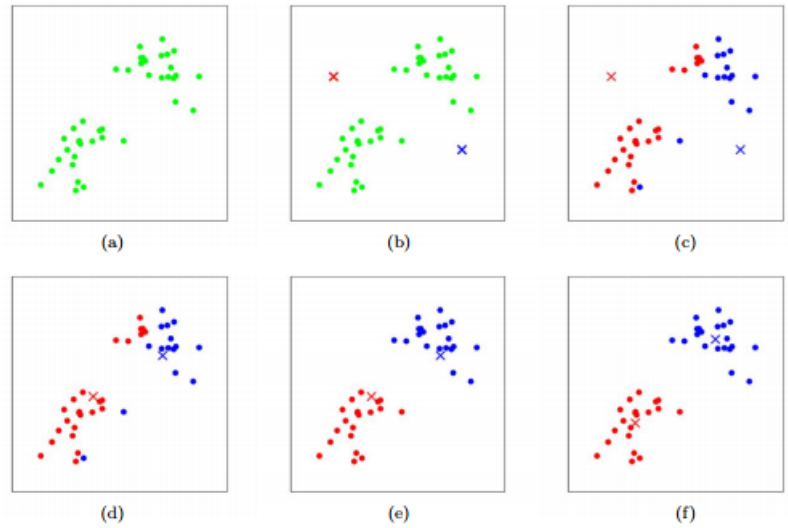

Figure 2.4: Illustration of the K-Means clustering algorithm.

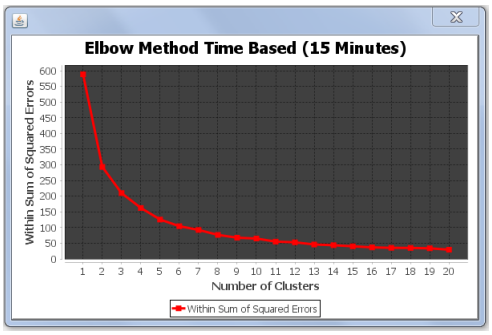

Figure 2.5: Visualization of the elbow method for the K-Means Clustering Algorithm

2.3.2 DBSCAN Clustering Algorithm

The *Density-Based Spatial Clustering of Applications with Noise* clustering algorithm, also known as DBSCAN, is one of the most used clustering algorithms proposed by Ester et al. in 1996 [8]. DBSCAN is density-based, points that are closely packed together are grouped together, points that lie alone are marked as outliers.

The DBSCAN clustering algorithm only requires two parameters:

- $Eps(\epsilon)$, the minimum distance between two data points or the radius of a data point.
- $MinPts$, the minimum number of points to form a dense region.

The algorithm starts the first phase by assigning each data point as core, border or noise point as follows:

- A point p is a core point if it has at least (including itself) $minPts$ in its specified radius ϵ. These are the *directly density-reachable* points from p.
- A point q is directly density-reachable from p if p is a core point and q is within radius ϵ of p.
- A point q is *density-reachable* from p if there is a path $p_1, \ldots, p_n$ with $p_1 = p$ and $p_n = q$, where each p_{i+1} is directly-density reachable from p_i (all the points on the path from p to q must be core points).

- A border point has fewer $minPts$ within its specified radius ϵ, however it is density-reachable from core point p (it is in the radius (ϵ) of p).
- A noise point is neither a core nor a border point (points not reachable from any other points).

In the second phase each core and border point is assigned to a cluster. An unassigned point, x, in the dataset is randomly selected and a depth-first search is performed. All the points in x's radius ϵ is added to the same cluster as x. In this radius ϵ, each search is applied recursively on an unexplored core point. This results in a single cluster of points. The algorithm finishes until all core points are assigned to a cluster where every border point is assigned to one of the clusters of its associated core points.

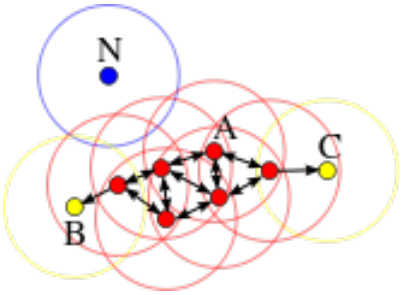

Figure 2.6: Illustration of the DBSCAN clustering algorithm

Figure 2.6 illustrates the DBSCAN clustering algorithm. In this visualization, $minPts = 4$. Point A and the other red points are core points, for the reason that within the radius ϵ for each of the red points, there are at least 4 points (including the point itself). Since these red points are directly density-reachable from one another, they will form a single cluster. Note that points B and C are marked as yellow, because they are not core points. Because of the density-reachability from A, points B and C belong the the cluster as well. Point N is neither a core point nor directly density-reachable and is thus classified as noise.

The advantages of DBSCAN are that DBSCAN does not require one to specify the number of clusters in the dataset a priori, as opposed to K-Means. Next to that, DBSCAN can find arbitrarily shaped clusters, whereas this is convex for K-Means (Figure 2.7). Lastly, DBSCAN has a notion of noise. It can detect and ignore outliers from the dataset.

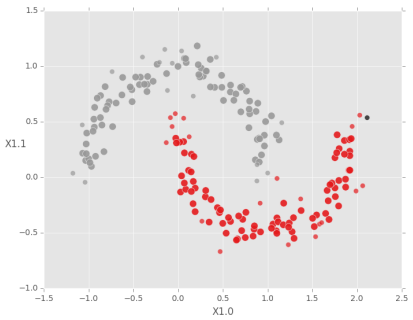

Figure 2.7: DBSCAN can find arbitrarily shaped clusters. This example cannot adequately be clustered by the K-Means algorithm.

The disadvantages of DBSCAN are that it is sensitive to parameter choice. If ϵ is chosen too small, sparse clusters are labeled as noise. If ϵ is too high, dense clusters are merged together

[12]. In addition, DBSCAN is not deterministic, as opposed to K-Means. Border points that are density-reachable from more than cluster can be part of either cluster depending on the order of the data is processed [8].

Chapter 3

Time -and Frequency-Based Abstraction

In this chapter, we introduce the methodology of clustering low-level events into high-level activities. This procedure consists of four main steps, as visualized in Figure 3.1. The starting point is to first define the concept of sessions. A session is a sequence of events such that the duration of consecutive events is within a given time frame. In Section 3.1, this concept is more discussed in detail.

First, each trace in the event log is split into sessions whereby the time frame of a session is defined by the user a priori. Second, the clustering phase starts. K-Means or DBSCAN with different clustering parameters are used to cluster the points (sessions) in the dataset. Third, the clusters are visualized in a heat map to determine an appropriate name for each cluster, whereby the cluster centroids give useful information for a stakeholder. Finally, the abstracted event log is created. The sessions are replaced by the cluster and their respective name which this session is mapped to, resulting in an abstracted event log.

This chapter is structured as follows. In Section 3.1 we introduce preliminary concepts about the creation of sessions. In Section 3.2 we cluster the sessions. Lastly, in Section 3.3 the clusters are visualized in heat maps and the abstracted log is created.

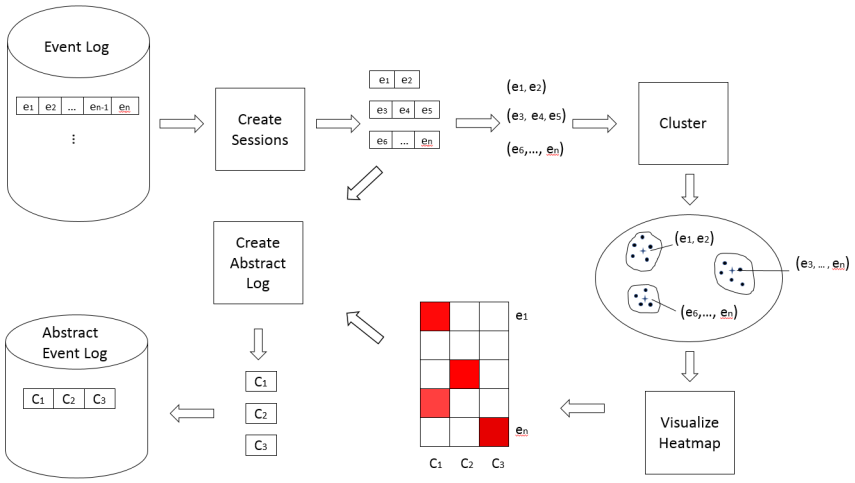

Figure 3.1: Steps involved to cluster low-level events into high-level activities to obtain an abstracted event log.

3.1 Creation of Sessions

Here, we introduce the concepts of the sessions in which an event log trace can be broken down.

Definition 3 (Sessions of a Trace)**.** Let $\sigma = \langle e_1, \ldots, e_n \rangle \in \mathcal{E}^*$ be a log trace. Given a sequence σ, $\textsc{First}(\sigma) = e_1$ and $\textsc{Last}(\sigma) = e_n$. Let Δ be a time interval. The sessions of σ is a sequence of (sub-)traces $Ⓢ_\Delta(\sigma) = \langle s_1, \ldots, s_m \rangle$ such that *(1)* $\sigma = s_1 \oplus \ldots \oplus s_n$ and *(2)* for all $1 \leq i < m$, $\lambda_T(\textsc{First}(s_{i+1})) - \lambda_T(\textsc{Last}(s_i)) > \Delta$.

For each trace $\sigma = \langle e_1, \ldots, e_n \rangle$ in the event log, we go trough all the events. Given a time interval Δ, the time difference between the current event e_i and the next event e_{i+1} is calculated $(\lambda_T(e_{i+1}) - \lambda_T(e_i))$. If the time difference is higher or greater than the given time interval Δ, then this means it is the end of a session and a new session is created.

As stated in Definition 3, this means that the sessions of σ is a sequence of (sub-)traces such that *(1)* the concatenation of the sessions should result in σ and *(2)* the time difference between the last event of the current session and the first event of the next session, should result in a time difference greater than Δ.

This concept is best illustrated by an example. Suppose the time interval $\Delta = 5$ (minutes) and we have the following trace with events:

$$\sigma = \langle e_1, e_2, e_3, e_4, e_5 \rangle$$

And suppose we have the following timestamps (in minutes) for these events:

$$\lambda_T(e_1) = 1, \lambda_T(e_2) = 3, \lambda_T(e_3) = 4, \lambda_T(e_4) = 10, \lambda_T(e_5) = 13$$

From this trace we can see that the time difference between e_3 and e_4 is greater than the given time interval Δ $(\lambda_T(e_4) - \lambda_T(e_3) > \Delta)$. This will result in two sessions:

$$Ⓢ_\Delta(\sigma) = \langle s_1, s_2 \rangle$$

Whereby $s_1 = \langle e_1, e_2, e_3 \rangle$ and $s_2 = \langle e_4, e_5 \rangle$. Note that the concatenation results in σ, $(\sigma = s_1 \oplus s_2)$. In addition, $\lambda_T(\textsc{First}(s_{i+1})) - \lambda_T(\textsc{Last}(s_i)) = \lambda_T(e_4) - \lambda_T(e_3) > \Delta$.

3.2 Clustering of Sessions

Using Definition 3, given a value for Δ and a clustering algorithm Cluster, we introduce a function $\textsc{Encode}(s)$ that returns a tuple for each sequence s of events. Thus, we repeat the following steps for each trace σ and for each session $s \in Ⓢ_\Delta(\sigma)$:

1. Encode the session s as a data point $\textsc{Encode}(s) : \mathcal{E}^* \to D_1 \times \ldots \times D_N$.
2. Add $\textsc{Encode}(s)$ to a data point set S.

After that, we invoke a clustering algorithm $\textsc{Cluster}(D)$ to cluster sessions.

The above procedure is independent of the clustering algorithm. In the actual operationalization, we leverage on the K-Means and DBSCAN algorithm. For the K-Means algorithm, one requires to decide the number of clusters a priori. We use the elbow method to estimate the best number of clusters. For the DBSCAN algorithm, one requires to decide the $Eps(\epsilon)$ and $minPts$ parameters.

In the following subsections we will describe two possible implementations of the Encode function. In subsection 3.2.1 we will encode the sessions based on *time*. In subsection 3.2.2 we will encode the sessions based on *frequency*.

3.2.1 Time-Based Encoding Of a Session

After creating the sessions using Definition 3, the sessions need to be encoded to be used for the clustering step. Before diving into the algorithm we have to introduce some concepts. Let $Int : A \to \mathbb{N}$ be the value of the time duration for each activity name in the activities set. Initially this value is set to zero. The values are calculated by taking the time difference between the next event and the current event in the session. Note that if the current event is the last event of the session, the time difference for that specific activity cannot be calculated, due to the fact that there is no next event. For this reason, we introduce a function $avg(a, \mathcal{L})$ that calculates the average of that specific activity:

$$avg(a, \mathcal{L}) = \frac{\sum_{\sigma \in \mathcal{L}} \sum_{e_i \in \sigma : e_i \notin \textsc{Last}(\sigma) \cap \lambda_A(e_i) = a} \lambda_T(e_{i+1}) - \lambda_T(e_i)}{\sum_{\sigma \in \mathcal{L}} |\{e_i \in \sigma : e_i \notin \textsc{Last}(\sigma) \cap \lambda_A(e_i) = a\}|}$$

This function returns the sum of the time difference for that specific activity name in the entire log and divides it by the number of occurrences of that specific activity in the entire log. The encoding of a session s is presented in Algorithm 1.

Algorithm 1: Encoding of a Session s Based on Time

Input: Session $s = \langle e_1, \ldots, e_n \rangle$
Result: A tuple

$tuple \leftarrow \{\}$
foreach $e_i \in s$ ***s.t.*** $i < n$ **do**
  $Int(\lambda_A(e_i)) \leftarrow Int(\lambda_A(e_i)) + (\lambda_T(e_{i+1}) - \lambda_T(e_i))$
  $Int(\lambda_A(\textsc{Last}(s))) \leftarrow avg(\textsc{Last}(s), \mathcal{L})$
end
$tuple \leftarrow (t_{a_1}, \ldots, t_{a_n})$ **s.t.**
  $A = \{a_1, \ldots, a_n\}$
  $t_{a_i} = |\{e \in s : \lambda_A(e) = a_i\}|$
$tuple_{norm} \leftarrow \textsc{Normalize}(tuple)$
return $tuple_{norm}$

After calculating the values for each of the activities that occur in a session, these values are then put in a *tuple*. Sometimes, it can be the case that some activities have very high values (time duration) compared to the other activities in session. It could be the case that there is e.g. a bottleneck in the process. This arises the problem of a skewed dataset. To solve this problem we introduce the function Normalize. For each value in the tuple, divide it by the sum of the tuple.

So far, we introduced the concepts and the algorithm to encode a session. For a better understanding, let us consider the following example log with 3 traces $\sigma_1, \sigma_2, \sigma_3$ with their respective events shown as activity names with the timestamps in minutes:

$$\mathcal{L} = [\quad \sigma_1 = \langle a_1, c_3, d_5, f_6 \rangle, \\ \sigma_2 = \langle a_0, c_2, a_3, c_5, f_{10}, d_{12} \rangle, \\ \sigma_3 = \langle a_0, b_1, b_2, c_7, d_9, e_{10}, c_{11}, d_{14}, f_{15} \rangle \quad]$$

Consider a time interval of $\Delta = 4$, using Definition 3 this will result in the following sessions:

$$Ⓢ_\Delta(\sigma_1) = \langle s_1 \rangle,\ Ⓢ_\Delta(\sigma_2) = \langle s_2, s_3 \rangle,\ Ⓢ_\Delta(\sigma_3) = \langle s_4, s_5 \rangle$$

Whereby $s_1 = \langle a_1, c_3, d_5, f_6 \rangle$, $s_2 = \langle a_0, c_2, a_3, c_5 \rangle$, $s_3 = \langle f_{10}, d_{12} \rangle$, $s_4 = \langle a_0, b_1, b_2 \rangle$ and $s_5 = \langle c_7, d_9, e_{10}, c_{11}, d_{14}, f_{15} \rangle$.

As a small illustration, let us apply Algorithm 1 on the first session of the first trace, $Ⓢ_{\Delta}(\sigma_1) = \langle s1 \rangle$:

$$a = c_3 - a_1 = 2$$
$$c = d_5 - c_3 = 2$$
$$d = f_6 - d_5 = 1$$

Because activity f is the last activity of the session, we have to use the function $avg(f, \mathcal{L})$ to get the value. Looking at log $\mathcal{L}$, f occurs three times. In session s_3 of the second trace σ_2, the time difference can be calculated: $d_{12} - f_{10} = 2$. Therefore the value for f would be: $avg(f, \mathcal{L}) = \frac{2}{3} = 0.667$. The result of applying Algorithm 1 would be:

$$(a = 2, c = 2, d = 1, f = 0.667)$$

However, we are putting the encoded values in a tuple. All the activities that occur in the log will be put in an alphabetical order in the tuple. So, the final representation would be as follows:

$$(2, 0, 2, 1, 0, 0.667)$$

Whereby the first value is the value of a, the second the value for b, the third the value for c, and so on.

Finally, applying Algorithm 1 to all of the sessions, will result in the following intermediate dataset D^*:

$$\begin{aligned} D^* = [\quad & (2,\ 0,\ 2,\ 1,\ 0,\ 0.667), \\ & (4,\ 0,\ 2.60,\ 0,\ 0,\ 0), \\ & (0,\ 0,\ 0,\ 1,\ 0,\ 2), \\ & (1,\ 1.50,\ 0,\ 0,\ 0,\ 0), \\ & (0,\ 0,\ 5,\ 1,\ 1,\ 0.667) \quad] \end{aligned}$$

As it can be seen, the values for the events that are not the last of a session are the summation of the time differences between the next and the current events with the same activity name. As for the events that are the last of a session, we sum all the time differences of that event with the same activity name and divide it by the number of occurrences in the entire log. For example, looking at activity c in the second *tuple* of dataset D^*, the value is calculated as follows:

$$\begin{aligned} & s_1 : c = 2 \\ & s_2 : c = 1 \\ & s_5 : c = 2 + 3 = 5 \\ & \Rightarrow c = \frac{2 + 1 + 2 + 3}{5} = 1.60 \end{aligned}$$

However, the value of c in the second session s_2 can be calculated normally $(a_3 - c_2)$. Therefore, we add this value as well, resulting in $c = 1.60 + 1 = 2.60$.

The next step is to apply the NORMALIZEfunction of Algorithm 1 to avoid the problem of a skewed dataset. This function divides each value in a tuple by the sum of the values in that same tuple. So, applying the NORMALIZEfunction to the intermediate dataset D^* will result in the final dataset D:

$$\begin{aligned} D = [\quad & (0.3529,\ 0,\ 0.3529,\ 0.1765,\ 0,\ 0.1177), \\ & (0.6060,\ 0,\ 0.3939,\ 0,\ 0,\ 0), \\ & (0,\ 0,\ 0,\ 0.333,\ 0,\ 0.667), \\ & (0.4,\ 0.6,\ 0,\ 0,\ 0,\ 0), \\ & (0,\ 0,\ 0.6521,\ 0.1304,\ 0.1304,\ 0.87) \quad] \end{aligned}$$

3.2.2 Frequency-Based Encoding Of a Session

Again, let $Int : A \to \mathbb{N}$ be the value of the frequency for each activity name in the activities set. Let $COUNT(\lambda_A(e_i))$ be the occurrence of the event in the session. The values are the number of occurrences of each activity in a specific session. The encoding of a session s is presented in Algorithm 2.

Algorithm 2: Encoding of a Session s Based on Frequency

Input: Session $s = \langle e_1, \ldots, e_n \rangle$
Result: A tuple

$tuple \leftarrow \{\}$
foreach $e_i ins$ ***s.t.*** $i < n$ **do**
| $Int(\lambda_A(e_i)) \leftarrow Int(\lambda_A(e_i)) + COUNT(\lambda_A(e_i))$
end

$tuple \leftarrow (t_{a_1}, \ldots, t_{a_n})$ **s.t.**
$A = \{a_1, \ldots, a_n\}$
$t_{a_i} = |\{e \in s : \lambda_A(e) = a_i\}|$
$tuple_{norm} \leftarrow$ NORMALIZE$(tuple)$

return $tuple_{norm}$

These values are then put in a *tuple*. The function NORMALIZE is applied as well in this algorithm to prevent the problem of a skewed dataset.

As a small illustration, let us apply Algorithm 2 on the first trace, $\text{Ⓢ}_\Delta(\sigma_1) = \langle s1 \rangle$ as in the previous subsection. Hereby, we only count the occurence of each activity in the session:

$$COUNT(a) = 1$$
$$COUNT(c) = 1$$
$$COUNT(d) = 1$$
$$COUNT(f) = 1$$

After, the values are put in a tuple. The final representation would be as follows:

$$(1, 0, 1, 1, 0, 1)$$

Whereby the first value is the value of a, the second the value for b, the third the value for c, and so on.

Finally, applying Algorithm 2 to all of the sessions, will result in the following intermediate dataset D^*:

$$\begin{aligned} D^* = [\quad & (1, 0, 1, 1, 0, 1), \\ & (2, 0, 2, 0, 0, 0), \\ & (0, 0, 0, 1, 0, 1), \\ & (1, 2, 0, 0, 0, 0), \\ & (0, 0, 2, 2, 2, 1) \quad] \end{aligned}$$

As it can be seen, the values are the number of occurrences, *frequency*, of each event with the same activity name.
The final step is to apply the NORMALIZEfunction of Algorithm 2 to avoid the problem of a skewed dataset. This function divides each value in a tuple by the sum of the values in that same tuple. So, applying the NORMALIZEfunction to the intermediate dataset D^* will result in the final dataset D:

$$D = [\quad (0.25, 0, 0.25, 0.25, 0, 0.25), \\ (0.5, 0, 0.5, 0, 0, 0), \\ (0, 0, 0, 0.5, 0, 0.5), \\ (0.333, 0.667, 0, 0, 0, 0), \\ (0, 0, 0.2857, 0.2857, 0.2857, 0.1429) \quad]$$

3.3 Visualization of Heat Maps and Creation of Abstract Event Logs

After encoding the sessions as a set of data points D and using a clustering algorithm $\textsc{Cluster}(D)$ we obtain a set of clusters $\{S_1, \ldots, S_n\}$. This phase starts from this set of clusters $\{S_1, \ldots, S_n\}$.

A heat map is shown with the centroids of $S_1, \ldots, S_n$. Each cluster S_i is given a name $\textsc{Name}(S_i)$ by a user. Then, Algorithm 3 is applied.

Algorithm 3: Abstraction of the Event Log

Input: Event Log $\mathcal{L} \in \mathbb{B}(\mathcal{E}^*)$, a set $S_1, \ldots, S_n$ of clusters, a set $\textsc{Name}(S_1), \ldots, \textsc{Name}(S_n)$ of cluster names

Result: Abstracted Event Log

$\mathcal{L}' \leftarrow \{\}$
foreach $\sigma \in \mathcal{L}$ **do**
  $\sigma' \leftarrow \langle\rangle$
  foreach $\langle e_1, \ldots, e_m \rangle \in \text{Ⓢ}(\sigma)$ **do**
    $c \leftarrow \textsc{Encode}(\langle e_1, \ldots, e_m \rangle)$
    $\sigma' \leftarrow \sigma' \oplus \langle n^{start} \rangle \oplus \langle n^{complete} \rangle$ **s.t.**
      $\lambda_A(n^{start}) = \lambda_A(n^{complete}) = \textsc{Name}(S_i)$ **s.t.** $c \in S_i$
      $\lambda_T(n^{start}) = \lambda_T(e_1)$
      $\lambda_T(n^{complete}) = \lambda_T(e_m)$
  end
  $\mathcal{L}' \leftarrow \mathcal{L}' \cup \{\sigma'\}$
end
return $(\mathcal{L}')$

In this algorithm we go through each trace σ in the event log $\mathcal{L}$. Each session is a (sub-)trace of σ and is mapped to a certain cluster. These sessions are replaced by two abstracted events with the activity name provided by the user.

The first abstract event is the event with lifecyle START. The value for this event is the timestamp of the first event of this session. The second abstract event is the event with lifecyle COMPLETE. The value for this event is the timestamp of the last event of this session.

Next, the abstracted events are concatenated to a abstracted trace σ'. The abstracted trace σ' is then put in the abstracted event log $\mathcal{L}'$. This procedure is applied to each trace σ in $\mathcal{L}$.

Let us again consider the example of the event log and the normalized dataset D of Subsection 3.2.1:

$$\mathcal{L} = [\quad \langle a_1, c_3, d_5, f_6 \rangle, \\ \langle a_0, c_2, a_3, c_5, f_{10}, d_{12} \rangle, \\ \langle a_0, b_1, b_2, c_7, d_9, e_{10}, c_{11}, d_{14}, f_{15} \rangle \quad]$$

$$
\begin{aligned}
D = [\quad & (0.3529,\ 0,\ 0.3529,\ 0.1765,\ 0,\ 0.1177),\\
& (0.6060,\ 0,\ 0.3939,\ 0,\ 0,\ 0),\\
& (0,\ 0,\ 0,\ 0.333,\ 0,\ 0.667),\\
& (0.4,\ 0.6,\ 0,\ 0,\ 0,\ 0),\\
& (0,\ 0,\ 0.6521,\ 0.1304,\ 0.1304,\ 0.87) \qquad]
\end{aligned}
$$

Consider that a clustering algorithm has created 3 clusters out of this dataset D and a user has given the following names for each cluster: cluster1 = A, cluster2 = B and cluster3 = C.

Suppose that *tuple*1 and *tuple*5 is mapped to cluster1, *tuple*2 and *tuple*4 is mapped to cluster2, and *tuple*3 is mapped to cluster3.

The result of the abstract event log $\mathcal{L}'$ will be:

$$
\begin{aligned}
\mathcal{L}' = [\quad & \langle A_1^{start}, A_6^{complete}\rangle,\\
& \langle B_0^{start}, B_5^{complete}, C_{10}^{start}, C_{12}^{complete}\rangle,\\
& \langle B_0^{start}, B_2^{complete}, C_{11}^{start}, C_{15}^{complete}\rangle \quad]
\end{aligned}
$$

3.4 Related Work

Earlier research has been conducted on log abstraction. The corresponding papers can be grouped in two category: supervised and unsupervised abstraction. The main difference between these two categories is that supervised approaches require domain knowledge or process models as inputs. Unsupervised methods do not rely on this assumption.

3.4.1 Supervised Abstraction Methods

Numerous research has been conducted on log abstraction. Baier et al. provided several matching approaches that maps events in an automated manner by using existing process documentation, by transforming the mapping problem into an optimization of constraint satisfaction problem using log-reply techniques or to match events from an event log to activities in a given process model [3, 4, 5]. Ferreira et al. introduced a hierarchical Markov model whereby both high-level behavior of activities and low-level behavior of events are captured [10]. Given a description of a process behaviors in terms of high-level activities and the uncertainty between the mapping between the events and activities, Fazzinga et al., provided a unified probabilistic framework to find all the interpretations of each trace in a given event log [9]. Lastly, based on behavioral activity patterns, Mannhardt et al. proposed a supervised abstraction method that captures domain knowledge on the relation between activities and events [18].

The drawback of these methods are that they require existing process documentation, a full process model or domain knowledge by as input by process stakeholders. Existing process documentation or domain knowledge might not be available or is limited, and when the process contains many activities it becomes a tedious task to model each of them manually.

3.4.2 Unsupervised Abstraction Methods

As can be observed, these abstraction methods are supervised. Existing process documentation, a full process model or domain knowledge are required by process stakeholders as input. Other, unsupervised, techniques has to be investigated that do not rely on this strong assumption. In the field of unsupervised abstraction, numerous amount of research has been conducted as well. Mannhardt and Tax proposed a method to discover *Local Process Models* (LPM) and use those models to lift the event log to a higher level of abstraction. Hence, LPMs are used to replace domain knowledge in supervised abstraction [17]. In another work by Günther et al., trace segmentation is used to group low-level events into clusters, which represents the higher level activities in the process models. This method is accomplished by taking into account the correlation of global event classes in an event log [11]. Next, based on similarity between activities, van Dongen and

Adriansyah derive a *Simple Precedence Diagram* (SPD) by clustering events using a fuzzy k-medoids algorithm to obtain a high-level view on the actual process [23]. Lastly, van Eck et al. automatically discover human behaviour from sensor data. A transformation approach to bridge the gap between continuous sensor data and event logs that is used as input is introduced to automatically derive an event log [24].

Hence, several research on unsupervised abstraction has been conducted as well to tackle the problem of using supervised abstraction methods that rely on existing process models as input. For the unsupervised abstraction, however, it is observable that the notion of *time* is not taken into account as a metric for creating the higher-level activities.

To summarize, existing supervised abstraction methods require process stakeholders to provide a full model of the entire process as input. In the field of unsupervised abstraction, there is no metric that takes the notion of *time* into account.

Chapter 4

Implementation

This chapter discusses how the abstraction of an event log based on the notion of *time* has been implemented as a plugin in ProM. In Section 4.1, the implementation of the plugin is discussed and in Section 4.2 the plugin implemented in the ProM framwork is discussed.

4.1 Implementing the Time -and Frequency-Based Abstraction Plugin

This thesis has been implemented in the ProM framework. The implementation consists of two parts. First the mining of the sessions has been implemented in ProM from scratch. Second, a graphical user interface has been designed whereby the user can select the session encoding procedure for creating the sessions (3.2.1) and a clustering algorithm with the respective parameter(s). The clustering is visualized in a heat map through which the user is able to define appropriate names for each cluster. Lastly, the plugin generates an abstracted event log for further analysis in ProM.

Within the implementations described above, three third-party libraries has been used to create clusters and visualize heat maps:

- **WEKA**
  Waikato Environment for Knowledge Analysis

  This library is used for the implementation of the K-Means clustering algorithm.
  https://www.cs.waikato.ac.nz/ml/weka/index.html
- **ELKI**
  Environment for Developing KDD-Applications Supported by Index-Structures

  This library is used for the implementation of the DBSCAN clustering algorithm.
  https://elki-project.github.io/
- **JHeatChart**
  This library is used for the visualization of the heat map.
  http://www.javaheatmap.com/

For DBSCAN ELKI has been chosen over WEKA, for the reason that it is much faster than WEKA. Tested on the UWV dataset, the visualization of the heat map took approximately 11 hours with the WEKA library, as for the ELKI library it took approximately 1 to 2 hours.

4.2 Time -and Frequency-Based Abstraction Plugin

The plugin requires an XES event log as input and directs the user to the graphical user interface. Figure 4.1 shows a screenshot of the plugin.

The first thing the user will see in the graphical user interface are the options for encoding the sessions and choosing a clustering algorithm as well as the parameters related to the clustering algorithm as shown in Figure 4.2. The heat map is visualized after choosing an encoding for the sessions and setting the parameter(s) for a specific clustering algorithm. Figure 4.3 shows the final visualization of the plugin. In this figure, the parts are labeled with numbers to go step-by-step through each of the parts of the visualization.

1. In this part of the plugin, the options and parameters are visualized. The plugin can be customized by using the following options:

   - **Session Encoding:** The creation and encoding of the sessions. The first one is *Create Sessions Based On Time* (Algorithm 1, page) and the second one is *Create Sessions Based On Frequency* (Algorithm 2, page).
   - **Clustering Algorithm:** The algorithm to be used for clustering.

   In addition, the user is able to set different parameters, which partly depends on whether DBSCAN or K-Means is chosen.

   - **Time Interval:** The time interval can be set for creating the sessions. The user is able to choose between *minutes, hours, days, weeks* and *months* as time units.
   - **Number of Clusters:** For the K-Means clustering algorithm, the user is able to select the number of clusters with the aid of the elbow graph as in Figure 2.5.
   - $Eps(\epsilon)$**:** The radius to be set as a parameter for the DBSCAN clustering algorithm.
   - $MinPts$**:** The the minimum number of points to be set as a second parameter for the DBSCAN clustering algorithm.
   - **Assigning Noisy Sessions to Clusters:** Option for the DBSCAN clustering algorithm to filter out noisy sessions or assigning them to clusters.

2. This part of the plugins visualizes the heat map as a result of clustering. In order to experiment with different options and parameters, the heat map is visualized in a tabbed pane. The user is then able to switch back and forth between different results and select the one he prefers.

3. In the bottom part of the plugin, appropriate names can be given for each of the clusters. Initially, the names are automatically generated. The user is still able to change the text boxes.

   Clicking on the button *Generate Abstracted Log* generates the abstracted event log into the ProM framework for further analysis.

4. In the right-top of the plugin, the clustering information is visualized to summarize the selected encoding session, clustering algorithm, number of clusters and the time interval.

   Besides, the look of the heat map can be adjusted. The user is able to select between the *logarithmic, linear* and *exponential* colouring scale. In addition, a threshold value can be set with the slider. Activities with a value below this threshold will not be displayed in the heat map.

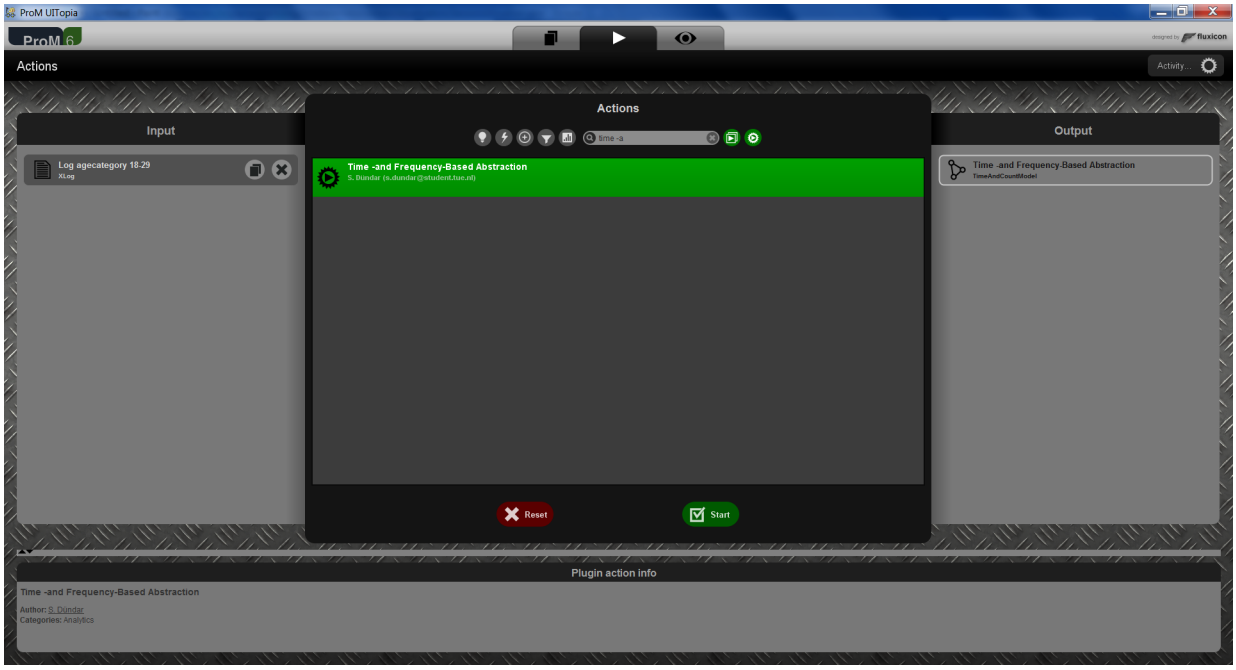

Figure 4.1: A screenshot of the Time -and Frequency-Based Abstraction plugin implemented from the ProM framework.

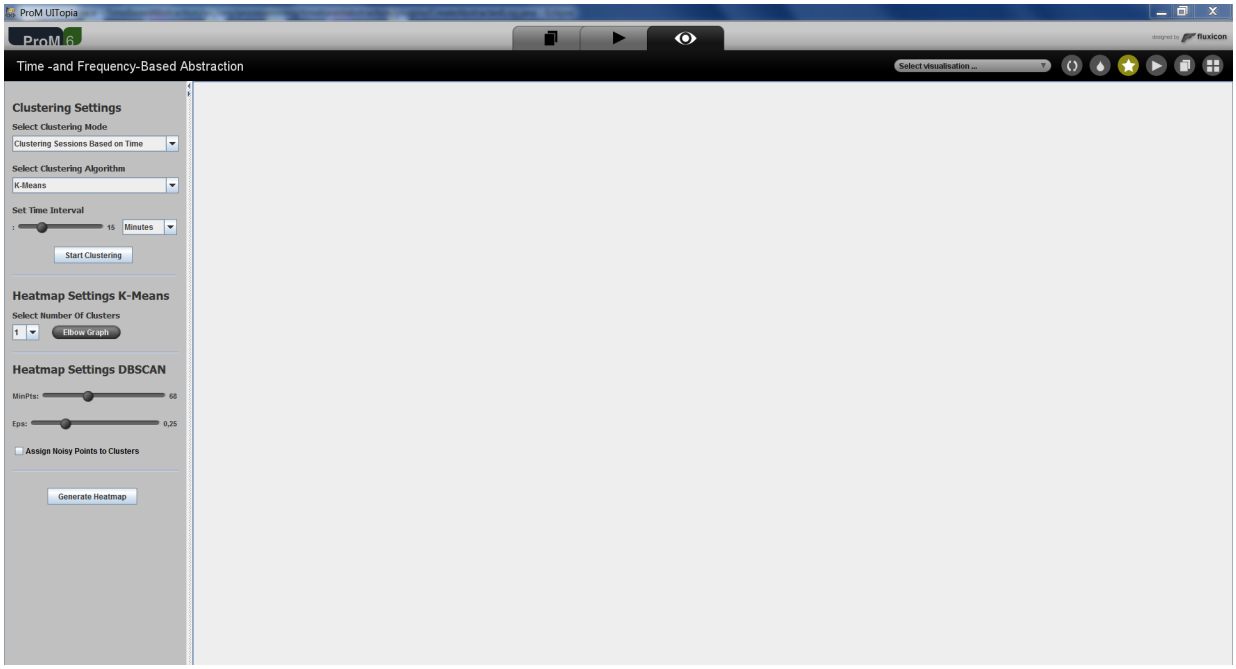

Figure 4.2: A screenshot of the visualization of the Time -and Frequency-Based Abstraction plugin.

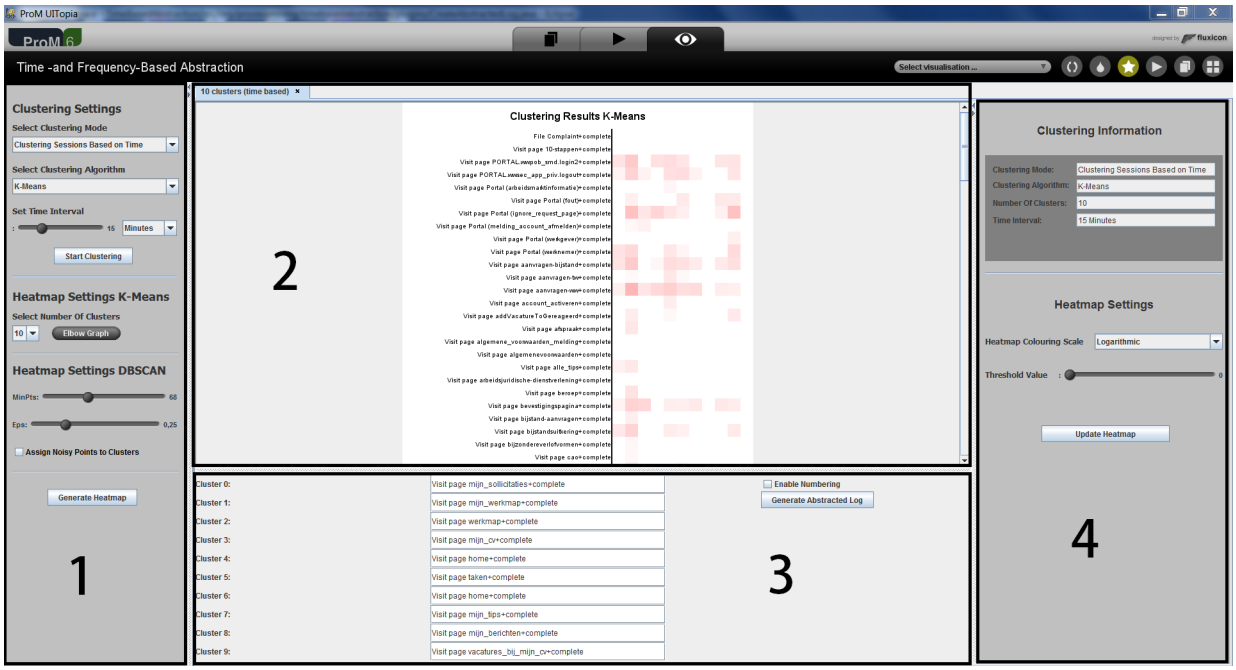

Figure 4.3: A screenshot of the visualization of the Time- and Frequency-Based Abstraction plugin after the sessions are encoded and clustered.

Chapter 5

Evaluation

In order to evaluate the abstraction technique introduced in Chapter 3, some process data is required. Two datasets has been used to evaluate the proposed abstraction technique. The first one is a click data of the website www.werk.nl. It refers to the browsing behavior of unemployed people where they can apply for jobs. The second dataset is a retail dataset provided by UCI Machine Learning repository.

In this Chapter, a comparative analysis is performed between event logs before abstraction and event logs after abstraction. The *Inductive Miner* algorithm was used for analyzing the models of the abstracted event logs of both datasets. However, the model produced by this algorithm for the UWV dataset was not good. This algorithm produced a so-called flower model where every behavior in the model is possible. Therefore, the *Heuristics Miner* algorithm is used for the analysis of the UWV dataset.

In the following sections, first a comparative analysis of event logs before and event logs after abstraction is performed. In addition, the performance between K-Means and the DBSCAN clustering algorithm is discussed.

5.1 UWV

UWV stands for Employee Insurance Agency. It is commissioned by the Ministry of Social Affairs and Employment to implement employee insurances and provide labour market and data services.

The main goal of UWV is to work with partners to make a difference for people by promoting work. This accomplished by providing its tasks in the area of employment, reintegration, temporary income and data management.

The dataset provided by UWV comes from the website www.werk.nl. It refers to the browsing behabior of unemployed people where they can apply for available jobs and perform taks to reintegrate into the social work-life as soon as possible. This data is from July 1st 2015 until February 29th 2016 and it consists of 2,624 cases and 335,655 events.

The first naive application of the Heuristics Miner algorithm on the original UWV dataset, resulted in a very unstructured process model as seen in Figure 5.1. These so-called "Spaghetti-Models" are unusable and, thus, abstraction needs to be applied to get more insight in the data. Based on discussions with one process stakeholder, it became clear that people are browsing on the website of UWV in sessions of 15 minutes. Therefore, the time interval for the creation of the sessions is set to 15 minutes.

Note that it is assumed that people start browsing from the homepage. Therefore, we have only used cases where a person start from the homepage. In addition, the same parameters are used for the Heuristics Miner algorithm to get the models before abstraction and after abstraction.

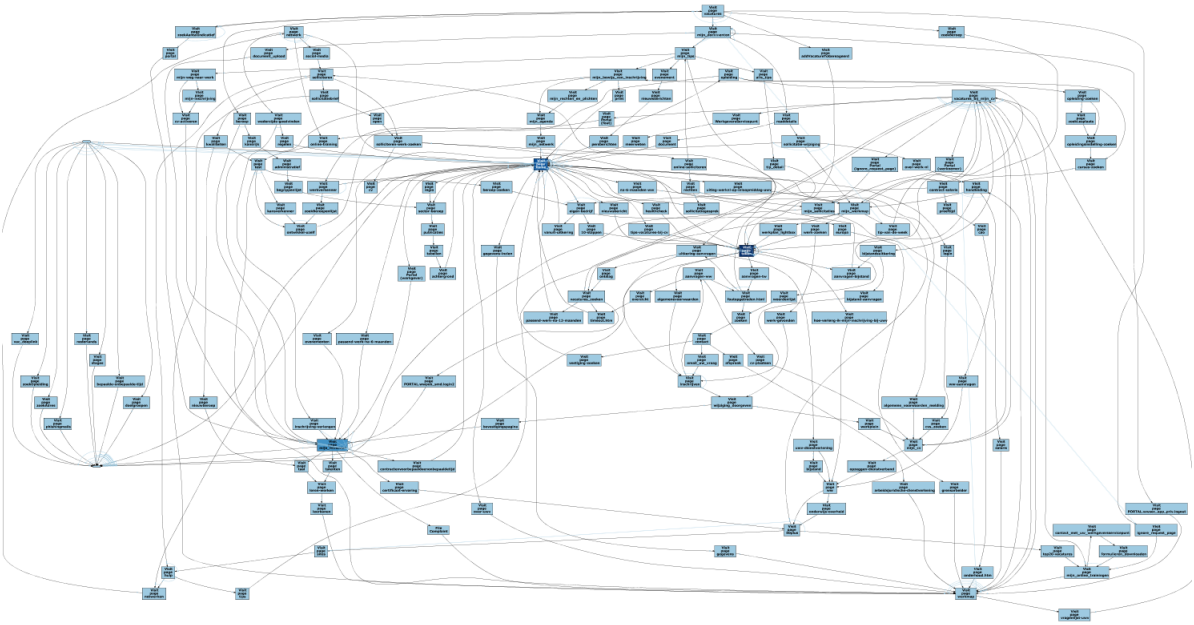

Figure 5.1: The unstructured process model obtained from the initial process discovery step for the UWV dataset.

5.1.1 K-Means

The first step of the K-Means algorithm is to determine the number of clusters beforehand. Using the elbow graph in Figure 5.2, we came to the conclusion $k = 10$ is the right number of clusters. The clustering of the sessions is presented in a heat map (Figure 5.3).

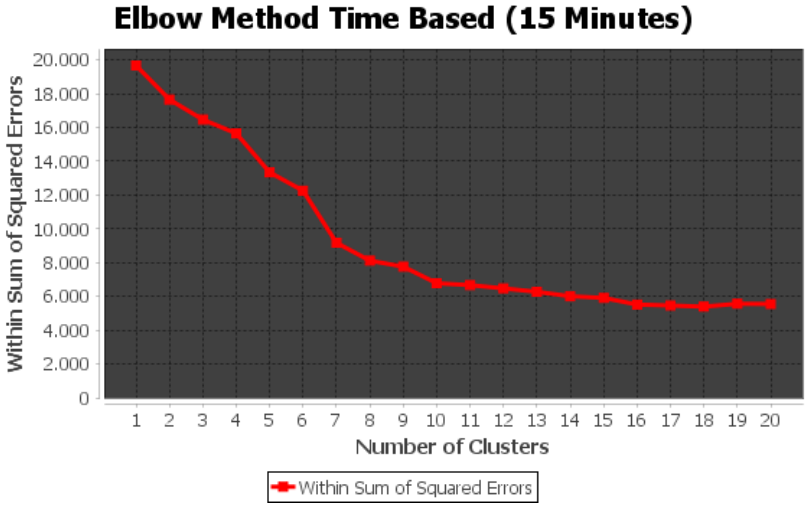

Figure 5.2: The elbow graph for determining the optimal number of clusters for the K-Means algorithm.

Based on this heat map and discussion with the stakeholder, the following names are given for each of the clusters, as presented in Table 5.1.

The number of events of the abstracted event log is reduced to 164,824 containing 2624 cases. As stated before, only cases has been used where a person starts from the homepage, *Visit page*

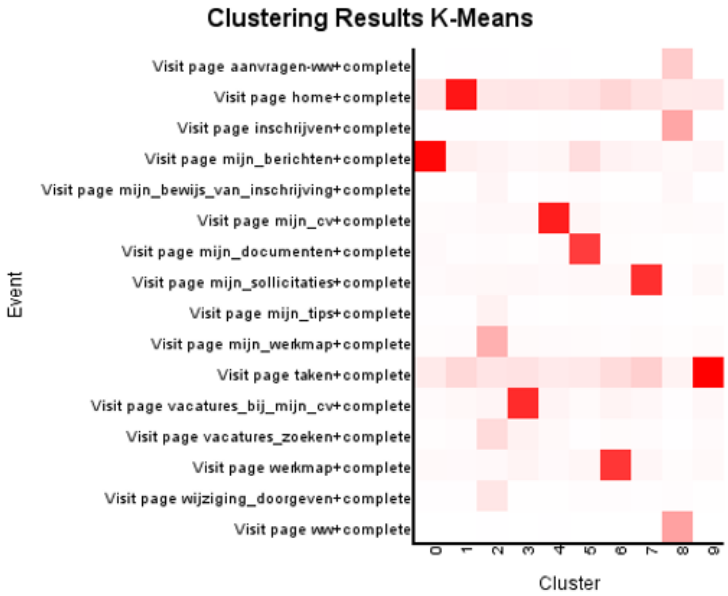

Figure 5.3: Heat map after applying the K-Means algorithm on the UWV dataset.

Cluster	Name
0	Visit page mijn_berichten
1	Visit page home
2	Visit page mijn_werkmap
3	Visit page vacatures_bij_mijn_cv
4	Visit page mijn_cv
5	Visit page mijn_documenten
6	Visit page werkmap
7	Visit page mijn_sollicitaties
8	Visit page ww
9	Visit page taken

Table 5.1: Names provided by the user for each of the clusters generated by the K-Means algorithm for the UWV dataset.

home. This resulted in an abstract event log containing of 519 cases and 17,395 events. The Heuristics Miner applied to this filtered abstract event log, results in the process model shown in Figure 5.4. This process model is more structured and easier to understand for a stakeholder compared to process model of Figure 5.1.

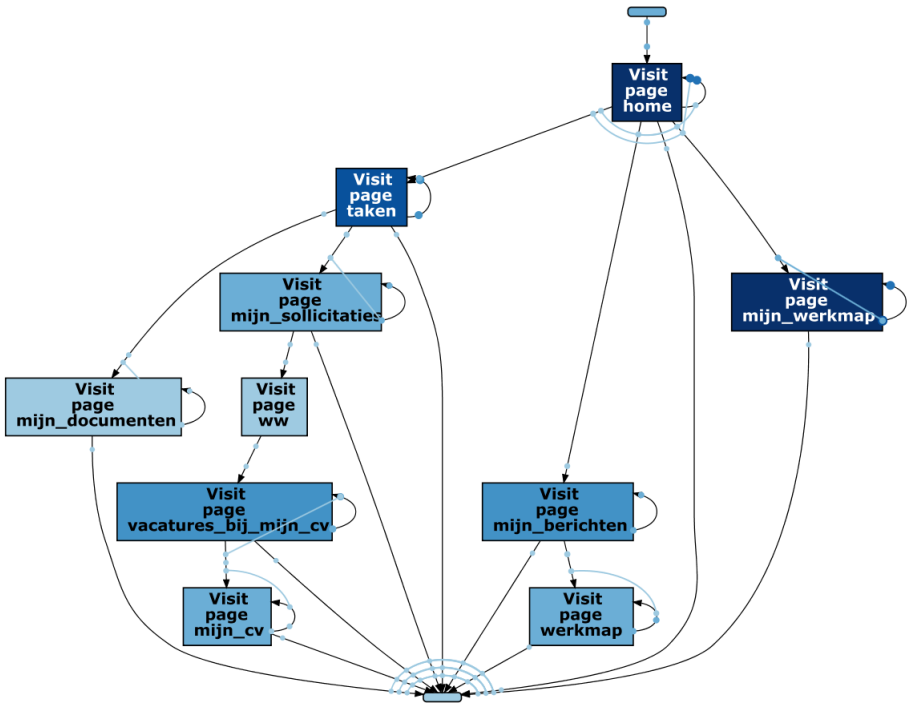

Figure 5.4: Process model of the abstract event log for the UWV dataset based on K-Means clustering.

5.1.2 DBSCAN

The first step of the DBSCAN algorithm is to determine the radius *Eps* and the $MinPts$. Initially, $Eps = 0.25$ and $MinPts = 200$ was set as the starting paramaters. The DBSCAN algorithm discovered only 4 clusters based on these parameters. Next, the *Eps* value was reduced to 0.15. Now, the DBSCAN algorithm discovered 6 clusters. It became clear that the dataset was dense, therefore the radius *Eps* was reduced further together with $MinPts$ to form clusters with less neighbor points. After several trial and errors with these parameters, it became clear that $Eps = 0.108$ and $MinPts = 160$ where the right parameters, resulting in a heat map of 10 clusters as shown in Figure 5.5. Actually, the total number of clusters is equal to 11, however the DBSCAN algorithm classified cluster 8 as a noise cluster. Note that the DBSCAN algorithm has a notion of noise and will filter these noise sessions out. However, these sessions cannot be thrown away, otherwise information will be lost. Therefore, the centroids for each of the other clusters is manually computed, the distance between the noise points and the centroids is computed and to the closest cluster.
Based on this heat map the following names are given for each of the clusters, presented in Table 5.2.

The number of events of the abstracted event log is reduced to 164,534 containing 2624 cases. In the same manner as with the K-Means abstraction, we decided to only use cases where we assume that a person starts from the homepage. Therefore the abstract log is filtered where the cases starts with the event *visit page home+taken*. This resulted in an abstracted event log containing of 889 cases and 30,858 events. The Heuristics Miner algorithm applied to this filtered

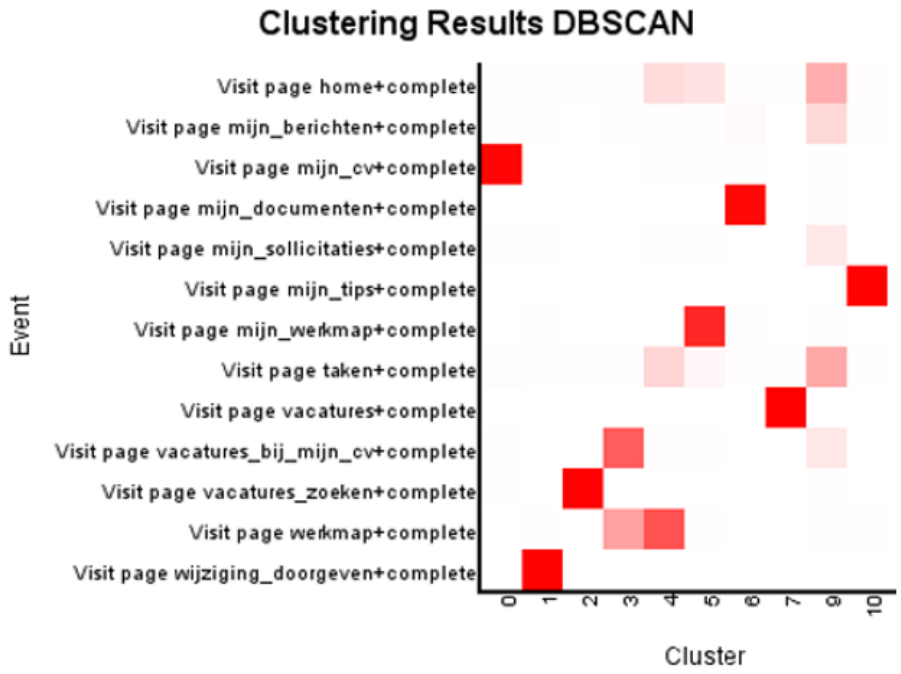

Figure 5.5: Heat map after applying the DBSCAN algorithm on the UWV dataset.

Cluster	Name
0	Visit page mijn_cv
1	Visit page wijzigin_doorgeven
2	Visit page vacatures_zoeken
3	Visit page vacatures_bij_mijn_cv
4	Visit page werkmap
5	Visit page mijn_werkmap
6	Visit page mijn_documenten
7	Visit page vacatures
9	Visit page taken+home
10	Visit page mijn_tips

Table 5.2: Names provided by the user for each of the clusters generated by the DBSCAN algorithm for the UWV dataset.

abstract event log, resulted in the process model as shown in Figure 5.6.

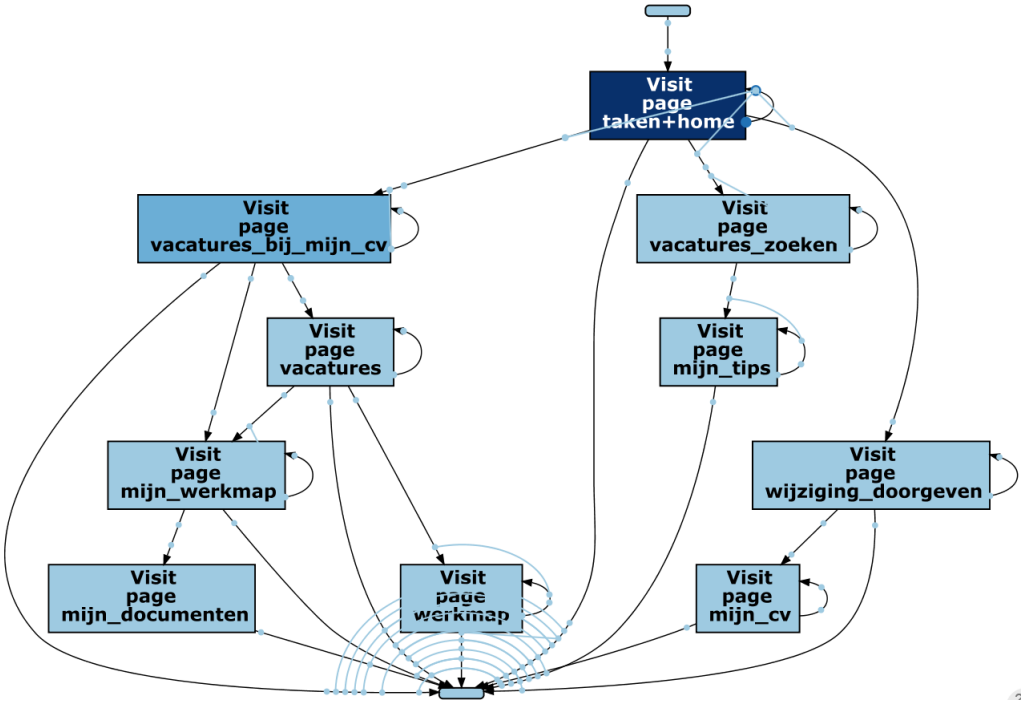

Figure 5.6: Process model of the abstract event log for the UWV dataset based on DBSCAN clustering.

5.1.3 Discussion

In this subsection a comparison between the K-Means and the DBSCAN clustering algorithms is performed. Each of the abstracted event logs is split randomly into a 70% training set to generate the models for conformance checking and a 30% test set for acquiring the *fitness, precision, generalization* and *simplicity* values, as referred in Chapter 2. The simplicity is calculated as follows:

$$simplicity = \#activities + \#bindings + \#arcs$$

For a fair comparison between the K-Means and the DBSCAN algorithm on the UWV dataset, the same parameter settings is used for the Heuristics Miner algorithm as shown in Figure 5.7. For conformance checking, a Petri Net is required. However, the Heuristic Miner gives an unsound Petri Net. After fixing the model to make it sound while preserving the behavior of the model, the Petri Net is made sound. Conformance checking is applied to verify how close the model is to the behavior observed in the event log.
Applying the Heuristics Miner algorithm to the 70% abstracted event log based on K-Means and DBSCAN resulted into the following process models as shown in Figure 5.8 and 5.10.
The DBSCAN algorithm has the notion of noise. Until this point, the noisy points are assigned to its closest nearby cluster. This is done by calculating the distance between noisy points and the centroids of the clusters and assigning the noisy points to the closest cluster.

What happens if the noisy sessions are really filtered of the abstracted event log? Filtering out noisy sessions is implemented as well. The user can choose to not calculate the distance and assigning the noisy sessions to the nearest cluster. This means that these noisy sessions will not

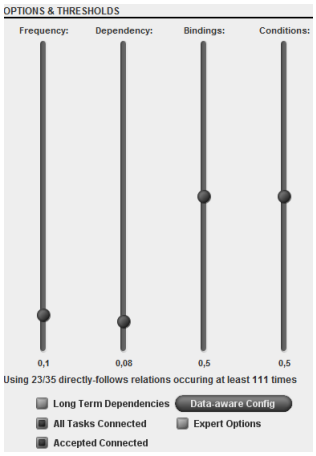

Figure 5.7: Parameter settings of the iDHM plugin for the comparison of the K-Means and DB-SCAN clustering algorithms.

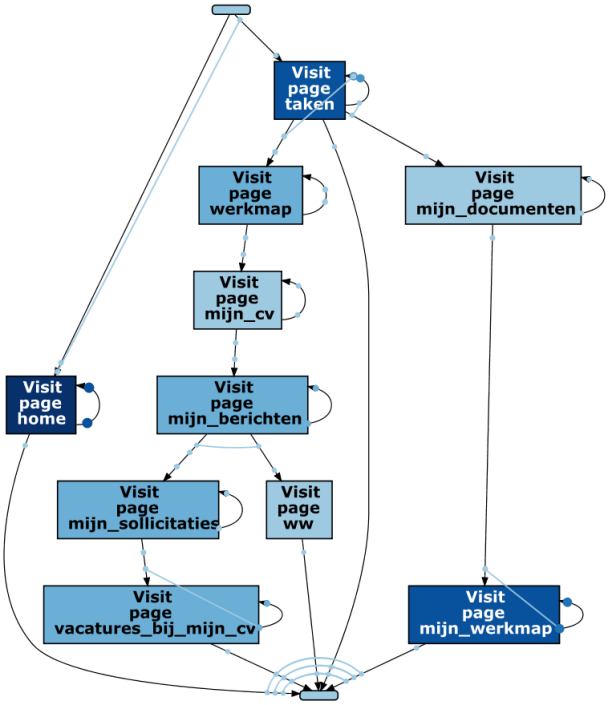

Figure 5.8: Process Model produced by the iDHM plugin on 70% of the abstract event log of the UWV dataset based on K-Means

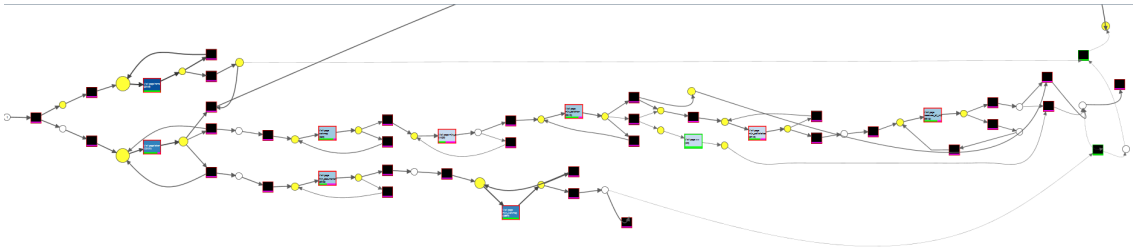

Figure 5.9: Alignment result of conformance checking on the UWV dataset based on K-Means.

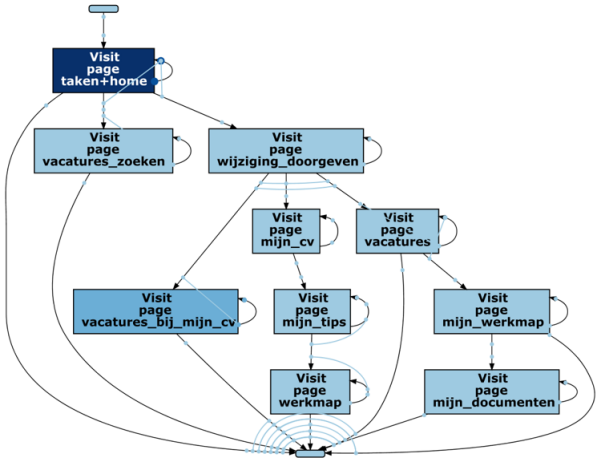

Figure 5.10: Process Model produced by the iDHM plugin on 70% of the abstract event log of the UWV dataset based on DBSCAN.

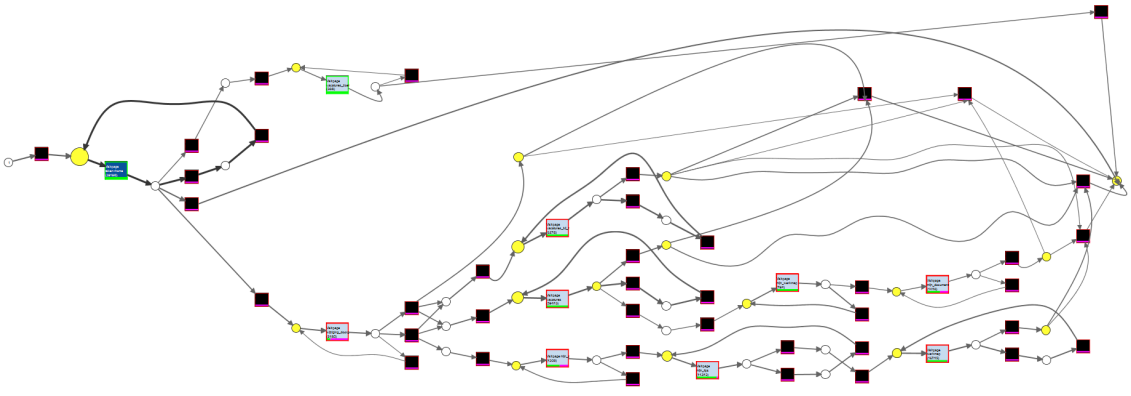

Figure 5.11: Alignment result of conformance checking on the UWV dataset based on DBSCAN.

be mapped to the higher-level activity. The consequence is that the noisy sessions in the trace of the event log will not be replaced by an higher-level activity and, thus, will be filtered.

The abstract event log with filtering of noisy sessions contains 543 cases and 1,918 events in total. The same procedure of splitting this event log into a training and test set is applied as well. The result of the process model without noise based on 70% of the abstract event log is shown in Figure 5.12.

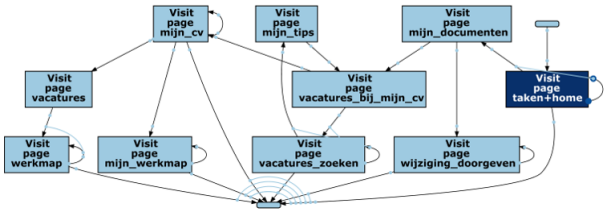

Figure 5.12: Process Model produced by the iDHM plugin on 70% of the abstract event log of the UWV dataset based on DBSCAN without noise.

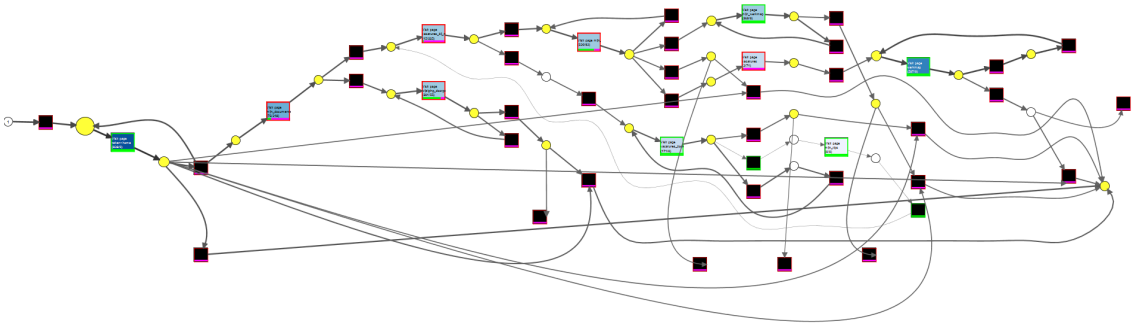

Figure 5.13: Alignment result of conformance checking on the UWV dataset based on DBSCAN without noise.

The results of the conformance checking for the K-Means, DBSCAN and DBSCAN with noise filtered, are summarized in Table 5.3.

	K-Means	**DBSCAN**	**DBSCAN Filtered**
Fitness	0.6637	0.6270	0.2785
Precision	0.33192	0.74779	0.68247
Generalization	0.99962	0.99996	0.99998
Simplicity	81	91	79

Table 5.3: Results of conformance checking for the K-Means, DBSCAN and DBSCAN with noise filtered.

Discovering a process model based on an event log without abstraction resulted in a "Spaghetti-like" process model, which is hard to understand. Clustering low-level events into high-level activities based on the K-Means and DBSCAN clustering algorithms resulted in structured process models.

For all of the process models, the generalization is high. The K-Means algorithm has the best fitness value compared to DBSCAN. This means that the process model discovered with the K-Means algorithm, can better reproduce the cases recorded in the event log. On the other

hand, DBSCAN without the noisy sessions, has a poor fitness value. The reason for this is that information is removed from the event log which describes the behavior of the process model. This affected the fitness of the process model.

DBSCAN scores the best in precision. It is noticeable that the precision for the K-Means is very low. This indicates that behavior that is not recorded in the event log is allowed by the model discovered by the K-Means algorithm.

To get more insight in terms of business values, these results are analyzed with a process stakeholder. First of all, the process model discovered by the K-Means algorithm was simpler. It has a clear sequential loop, as opposed to the model discovered by the DBSCAN algorithm. There, the activities are performed in a more parallel way, where most of the activities are occurring at the same time. The model of DBSCAN with noise filtered has no parallel execution. In addition, the behavior is different.

Secondly, it is not really clear what is going on just by looking at the models. The models represent the behavior of the people in general. However, it can not be distinguished, e.g., how older people are browsing than younger people. In addition, these process model does not really show the browsing behavior of people during their UWV period. So profiling customers based on their behavior during their period in UWV is still not clear.

Lastly, some data preparation could be done to get even better clusters and thus higher-level activities. For example, *mijn_werkmap* and *werkmap* actually belong together and, thus, could be merged.

All in all, the process models after abstraction revealed new insights. The models are simpler than the model discovered from the original event log. In general, it is a good starting point to get hidden insights in the data and for doing further analysis.

5.2 Retail

The online retail dataset is a transnational dataset which contains all the transactions occurring between December 1st 2010 and December 9th 2011 for a UK-based and registered non-store online retail. The company mainly sells unique all-occasion gifts and many customers of the company are wholesalers. A fragment of the dataset can be found in Appendix A.

The first naive application of the Inductive Visual Miner on the original online retail dataset, resulted in a very large and hard to read process model. Figure 5.14 shows the process model on a sample of the original dataset, because the dataset was too big to be handled by the miner. However, even with this sample dataset the process model is big.

The transactions in this dataset all occur at the same time. To apply the time -and frequency based abstraction technique, each row in the dataset is duplicated based on the quantity of the item. An example is in Table 5.4: the top event with quantity 4 is transformed into 4 events. Now, the *frequency* of this event is used as a value for the sessions for the abstraction technique instead of the time difference between events.

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
554065	22752	SET 7 BABUSHKA NESTING BOXES	4	22-5-2011 10:39	8,5	1827	United Kingdom
↓			↓			↓	
554065	22752	SET 7 BABUSHKA NESTING BOXES	1	22-5-2011 10:39	8,5	1827	United Kingdom
554065	22752	SET 7 BABUSHKA NESTING BOXES	1	22-5-2011 10:39	8,5	1827	United Kingdom
554065	22752	SET 7 BABUSHKA NESTING BOXES	1	22-5-2011 10:39	8,5	1827	United Kingdom
554065	22752	SET 7 BABUSHKA NESTING BOXES	1	22-5-2011 10:39	8,5	1827	United Kingdom

Table 5.4: Duplication of each of the rows based on the quantity.

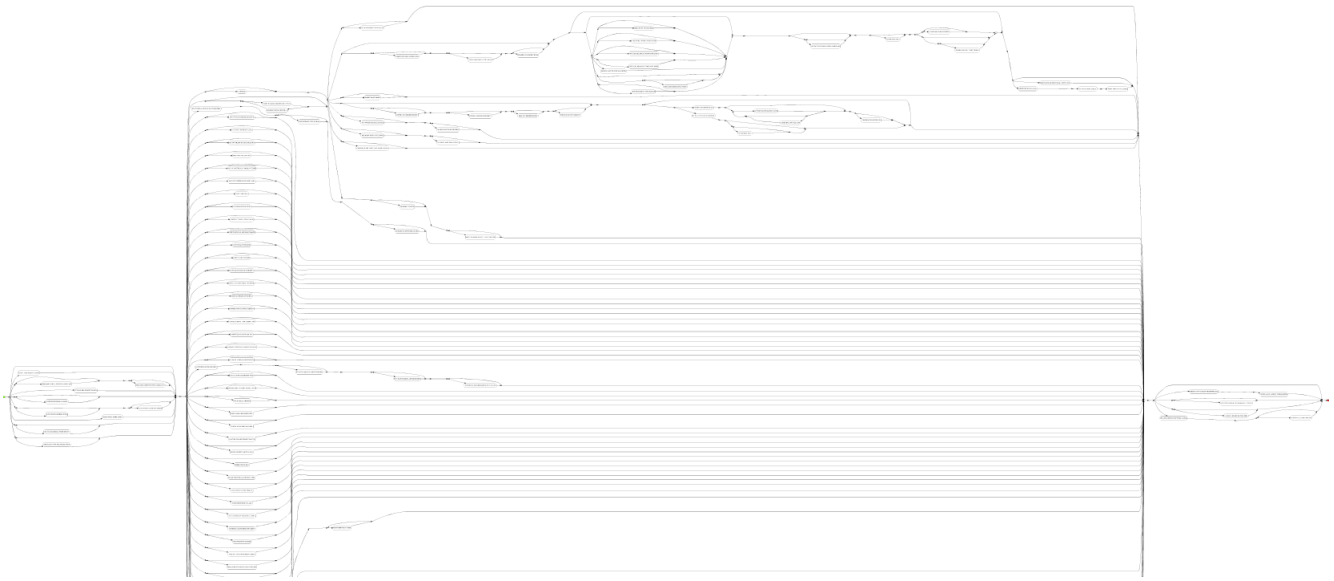

Figure 5.14: An excerpt of the unstructured process model obtained from the initial process discovery step for the online retail dataset.

5.2.1 K-Means

The first step is to determine the number of clusters beforehand. Unfortunately, the elbow graph in Figure 5.15 does not give a clear answer. The Within Sum of Squared Errors is not decreasing even when the number of clusters is increased. The explanation could be that the sessions points are very dense and closely packed together. Therefore, we conclude that the K-Means algorithm is not suitable for this dataset. A better option would be the DBSCAN algorithm, because it is density based and does not one require to determine the number of cluster a priori. However, to test the K-means algorithm on this dataset, $k = 15$, has been chosen as the number of clusters. This resulted in a heat map of 15 clusters as shown in Figure 5.16.

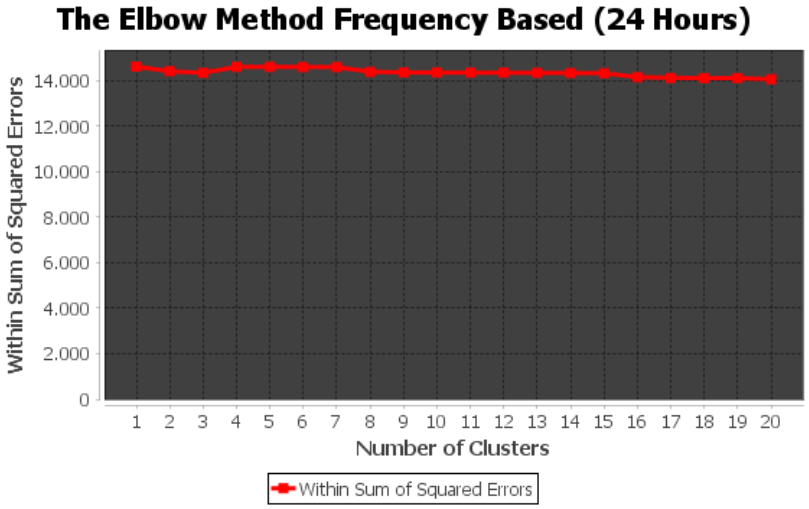

Figure 5.15: The elbow graph for determining the optimal number of clusters for the K-Means algorithm for the online retail dataset.

Based on this heat map, the following names are given for each of the clusters, as presented in

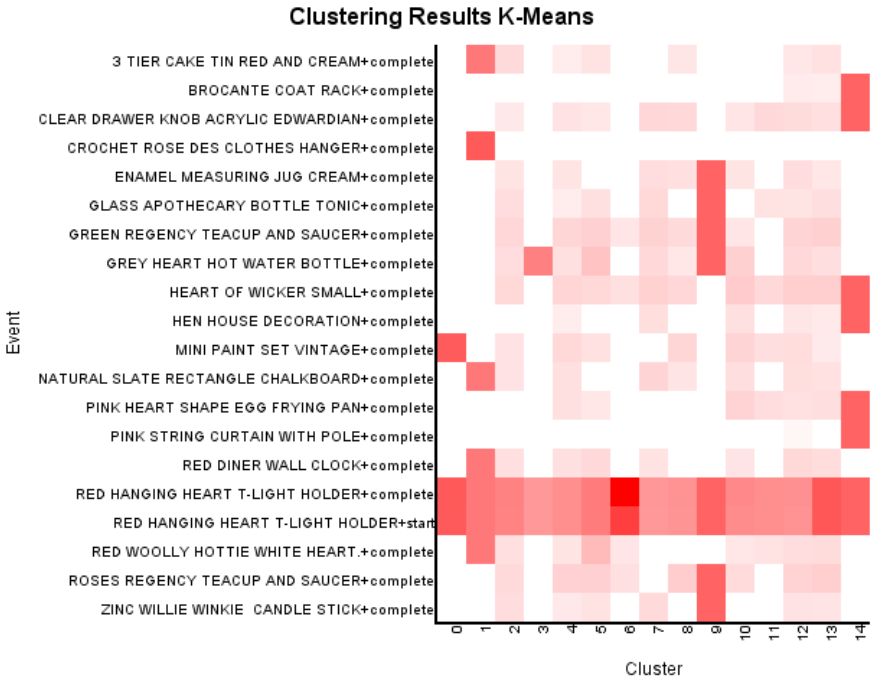

Figure 5.16: Heat map after applying the K-Means algorithm on the online retail dataset.

Table 5.5.

Cluster	Name
0	RED HANGING HEART T-LIGHT HOLDER
1	CROCHET ROSE DES CLOTHES HANGER
2	RED HANGING HEART T-LIGHT HOLDER
3	GREY HEART HOT WATER BOTTLE
4	RED HANGING HEART T-LIGHT HOLDER
5	RED HANGING HEART T-LIGHT HOLDER
6	RED HANGING HEART T-LIGHT HOLDER
7	RED HANGING HEART T-LIGHT HOLDER
8	RED HANGING HEART T-LIGHT HOLDER
9	ENAMEL+GLASS+GREEN+GREY+RED+ROSES+ZINC
10	RED HANGING HEART T-LIGHT HOLDER
11	RED HANGING HEART T-LIGHT HOLDER
12	RED HANGING HEART T-LIGHT HOLDER
13	RED HANGING HEART T-LIGHT HOLDER
14	BROCANTE+CLEAR+HEART+HEN+PINK+T-LIGHT

Table 5.5: Names provided by the user for each of the clusters generated by the K-Means algorithm for the online retail dataset.

The number of events of the abstracted event log is reduced to 25,540 containing 3223 cases. The Inductive Miner applied to this abstracted event log resulted in the process model of Figure 5.17.

As can be seen, the process model of the abstracted event log is very small and simple. In addition, it is noticeable that the activity *RED HANGING HEART T-LIGHT HOLDER* is very frequent. The reason for this is this online retail dataset is quite skewed. After examining the log

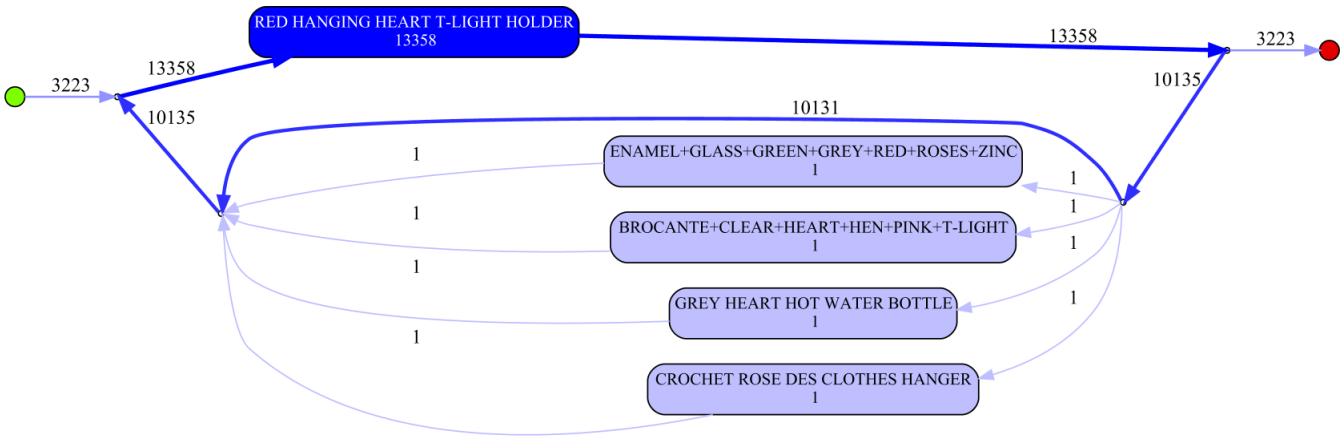

Figure 5.17: Process model of the abstract event log for the Online Retail dataset based on K-Means clustering.

summary, it appears to be that this activity captures 7.4% of the events in the original event log, whereas the other activities only capture below 1% of the events. The consequence is that the K-Means algorithm discovered clusters where this activity is very frequent which is also seen in the heat map in Figure 5.16. Most of the clusters are thus mapped to this activity resulting in a very small, skewed and simple process model.

5.2.2 DBSCAN

The first step is to determine the Eps and $MinPts$ parameters. Initially, we tried generating the heat map with $Eps = 0.25$ and $MinPts = 1000$ as the parameters. This resulted in a heat map with only one cluster. In the next try, the Eps parameter was reduced to $Eps = 0.1$. This resulted in a heat map with only 2 clusters. After several trial and errors, it became clear that this dataset was very dense and the points where closely packed together. The conclusion was that the right parameters for this dataset was with $Eps = 0.031$ and $MinPts = 7$, resulting in a heat map of 15 clusters as shown in Figure 5.18.

Based on this heat map, the following names are given for each of the clusters, presented in Table 5.6.

The number of events of the abstracted event log is reduced to 26,172 events containing 3223 cases. The Inductive Visual Miner applied to this event log results in the process model shown in Figure 5.19.

As can be seen the process model is easier to understand and there is some kind of structure in it. Here again, it is noticeable that the activity *RED HANGING HEART T-LIGHT HOLDER* is very frequent.

5.2.3 Discussion

In this subsection a comparison between the K-Means and the DBSCAN algorithms is performed based on the Online Retail dataset. Each of the abstracted event logs is split randomly into a 70% training set to generate the models for conformance checking and a 30% test set for acquiring the *fitness, precision, generalization* and *simplicity* values. The simplicity for these models are calculated as follows:

$$simplicity = \#activities + \#arcs$$

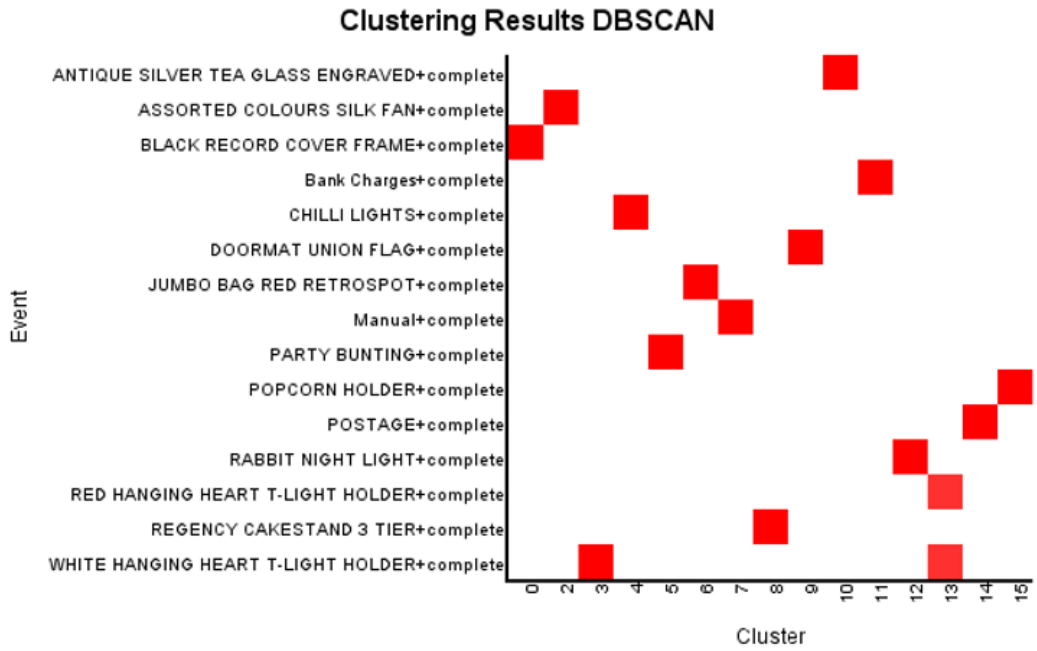

Figure 5.18: Heat map after applying the DBSCAN algorithm on the online retail dataset.

Cluster	Name
0	BLACK RECORD COVER FRAME
2	ASSORTED COLOURS SILK FAN
3	WHITE HANGING T-LIGHT HOLDER
4	CHILLI LIGHTS
5	PARTY BUNTING
6	JUMBO BAG RED RETROSPOT
7	Manual
8	REGENCY CAKESTAND 3 TIER
9	DOORMAT UNION FLAG
10	ANTIQUE SILVER TEA GLASS ENGRAVED
11	Bank Charges
12	RABBIT NIGHT LIGHT
13	RED HANGING T-LIGHT HOLDER
14	POSTAGE
15	POPCORN HOLDER

Table 5.6: Names provided by the user for each of the clusters generated by the DBSCAN algorithm for the online retail dataset.

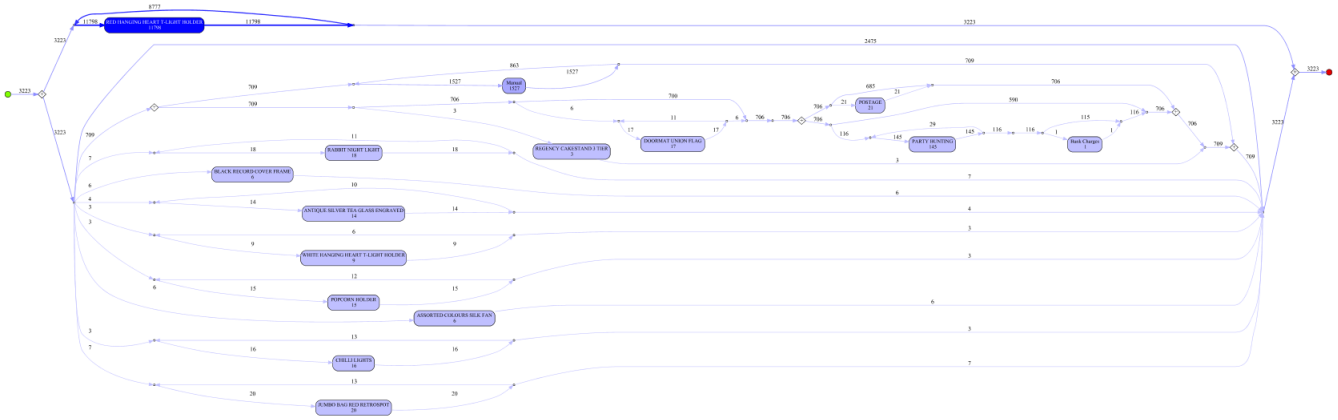

Figure 5.19: Process model of the abstract event log for the Online Retail dataset based on DBSCAN clustering.

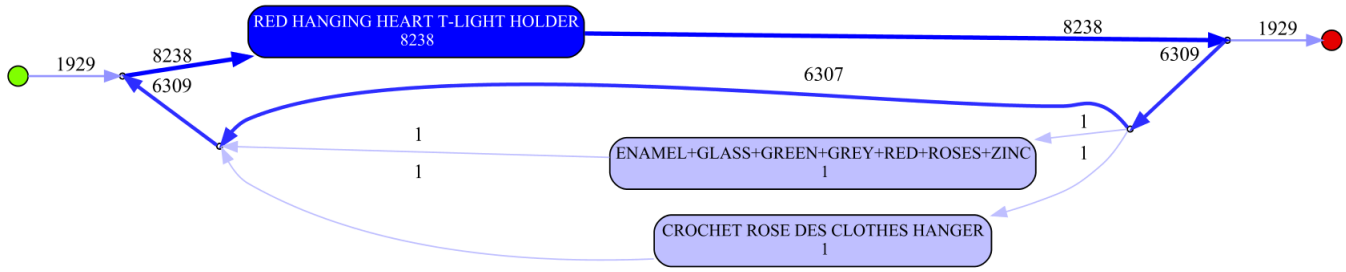

Figure 5.20: Process Model produced by the Inductive Miner on 70% of the abstract event log of the online retail dataset based on K-Means.

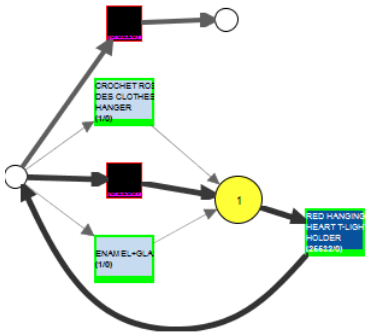

Figure 5.21: Alignment result of conformance checking on the online retail dataset based on K-Means.

Applying the Inductive Miner to the 70% abstracted event log based on K-Means resulted into the following process model as shown in figure 5.20.

Applying the Inductive Visual Miner plugin to the 70% abstracted event log based on DBSCAN results into the following process model as shown in Figure 5.22.

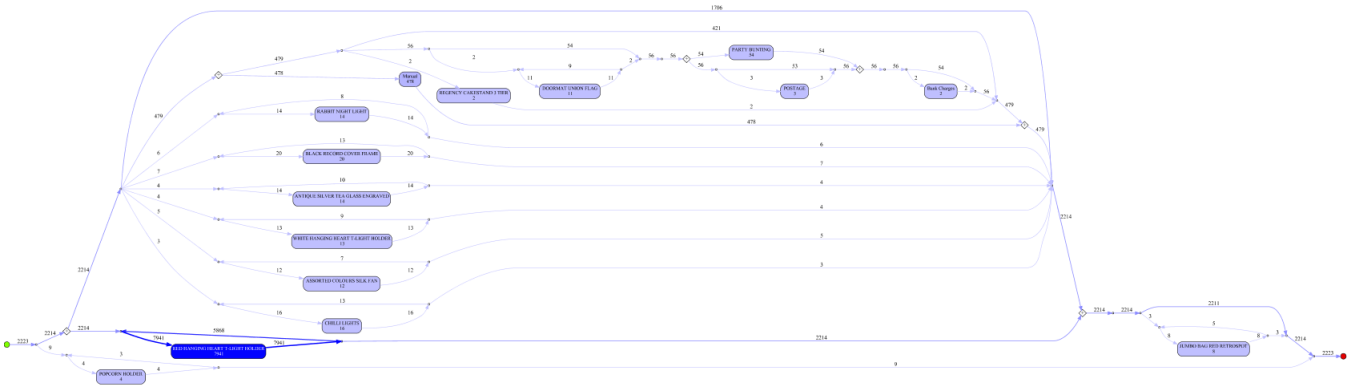

Figure 5.22: Process Model produced by the Inductive Visual Miner plugin on 70% of the abstract event log of the online retail dataset based on DBSCAN.

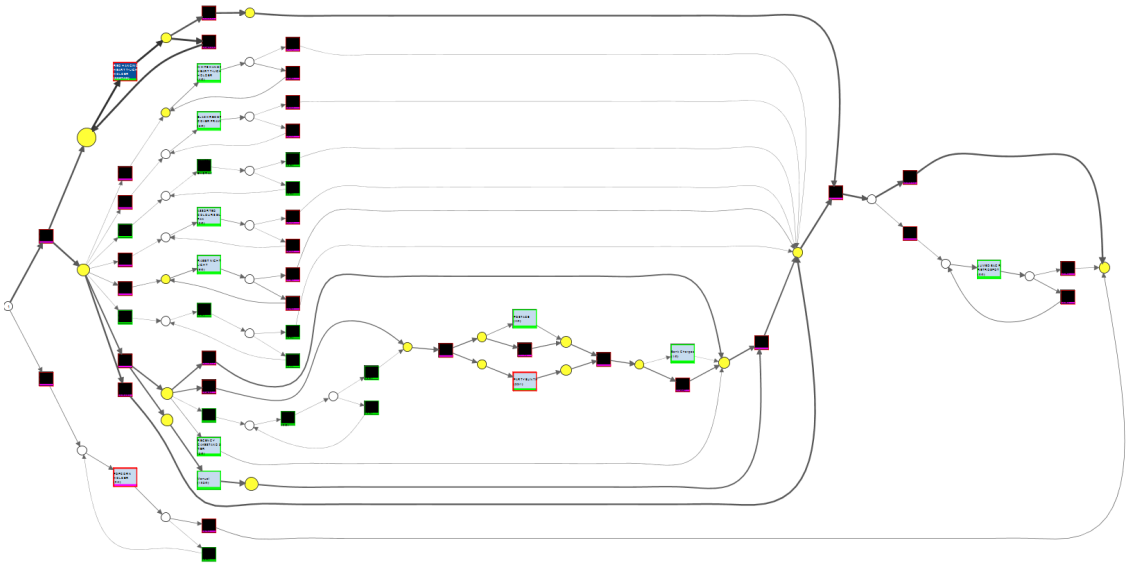

Figure 5.23: Alignment result of conformance checking on the online retail dataset based on DBSCAN.

Applying the Inductive Miner to the 70% abstracted event log based on DBSCAN without noise results into the following process model as shown in Figure 5.24.

Discovering a process model based on an event log without abstraction resulted in a "Spaghetti-like" process model, which is hard to understand. Clustering low-level events into high-level activities based on the K-Means and DBSCAN clustering algorithms resulted in a more structured process models as well. For this dataset the K-Means algorithm was not suitable. The process model of the abstracted event log based on K-Means was too small and too simple. This is also reflected in the conformance checking results. Because of the simple model the generalization and fitness is optimal, whereas the precision is poor.

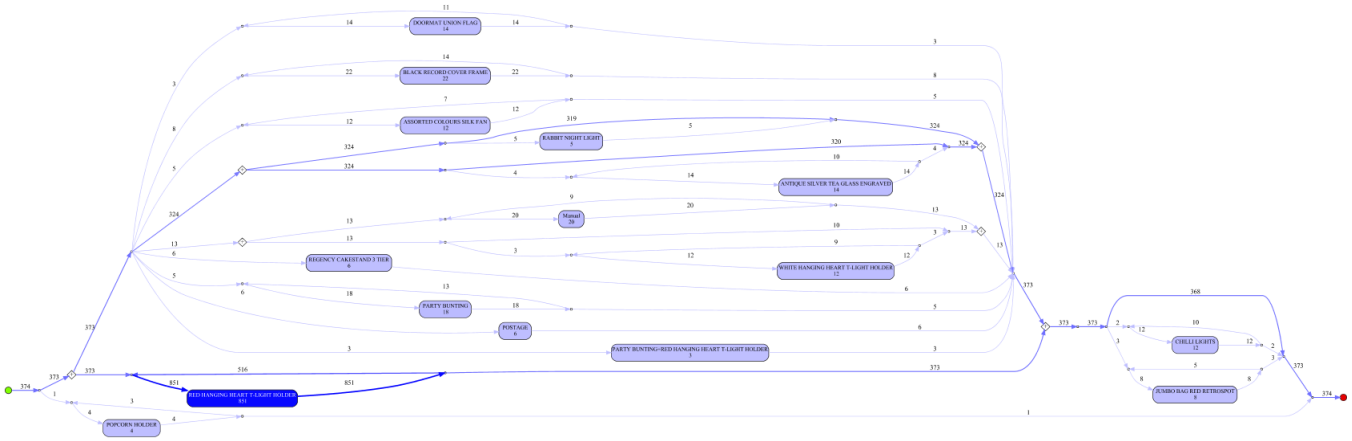

Figure 5.24: Process Model produced by the Inductive Miner on 70% of the abstract event log of the online retail dataset based on DBSCAN with noise filtered.

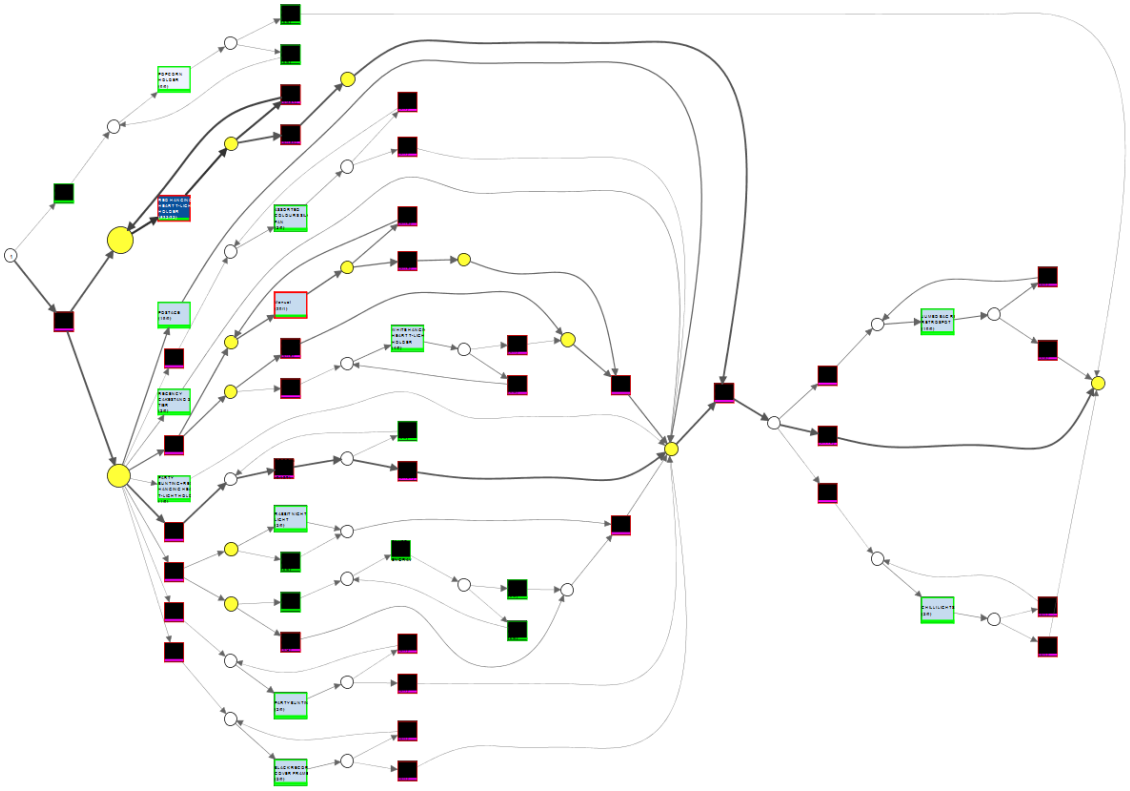

Figure 5.25: Alignment result of conformance checking on the online retail dataset based on DBSCAN without noise.

	K-Means	**DBSCAN**	**DBSCAN Filtered**
Fitness	0.9998	0.9412	0.8923
Precision	0.33917	0.22949	0.24689
Generalization	1	0.9919	0.9620
Simplicity	13	105	100

Table 5.7: Results of conformance checking for the K-Means, DBSCAN and DBSCAN with noise filtered on the Online Retail dataset.

The DBSCAN algorithm was more suitable for this dataset, for the reason that it can find arbitrary shaped clusters. This was also reflected in the heat maps with distinct clusters. Also for the DBSCAN algorithm, the fitness is very good, but it even scores lower on precision than the K-Means algorithm. Filtering out noisy sessions will result in a lower fitness because of the information lost in the abstract event log.

All in all, the abstracted event logs, except for the K-Means, are suitable for this dataset. However, the abstracted models allow for more behavior which is not captured in the event log.

Chapter 6

Conclusions

This thesis presents a method to cluster low-level events into high-level activities based on the notion of *time* and *frequency*. Traces are broken down into sessions, which are clustered. Sessions are considered as concluded when the subsequent event is further in time then a given threshold. The user is able to provide meaningful activity names with the aid of a heat map, where the cluster centroids provide meaningful information for a stakeholder. While the methodology is independent of any clustering technique, the K-Means and DBSCAN clustering algorithms are elaborated in this thesis.

The time -and frequency-based abstraction is implemented as a plugin in the ProM framework. Clustering parameters and several options for creating the sessions can be set in the plugin. The plugin produces an abstract event log, which can be further analyzed in the ProM framework.

An extensive case study on the UWV and the online retail datasets was conducted in order to discover real-life processes for the evaluation of the abstracted event logs. Our abstracted event logs have proven to discover a simpler and a structured process models compared with the unstructured "Spaghetti-like" process models that are discovered with and without abstraction.

Conformance checking is applied to evaluate the discovered process models based abstraction. The results have shown to produce process models that have good *fitness*. Thus, the discovered models can quite accurately reproduce cases recorded in the abstracted event log. In addition, for all of the process models the *generalization* is high and thus the abstracted models will be able to reproduce future behavior of the process. Unfortunately, the *precision* of the abstracted event log based on K-Means is relatively low. More behavior is allowed by the model that is never observed in reality. An explanation for this is that complex structures in the original process is abstracted to a simpler process. Hereby, this information of the hidden complex structures in the original process is lost. On the other hand, the *precision* is relatively high for the abstracted event log based on DBSCAN.

From a business viewpoint, the results obtained from the abstracted event log reveals insight that cannot be obtained when using the original event log and thus is a good way to start for process stakeholders to gain a better insight in their process.

6.1 Future Work

This thesis has shown the practical feasibility and relevance of abstracting low level-events into high-level activities based on time and frequency. The research in this thesis results in multiple opportunities for future work and research. This section presents these ideas for future work in this area.

First of all, in this thesis the sessions are clustered based on the K-Means and DBSCAN clustering algorithms. It would be interesting to test this method with other clustering algorithms, e.g. the hierarchical clustering algorithm, and see if the results differ from the evaluation in this thesis.

Secondly, in Chapter 3 the creation of the sessions is discussed. For the events that are the last of a session, the average time duration of that particular activity in the entire log is used as a value. An idea could be that a stakeholder can put a value that is more realistic, for the reason that he has more knowledge about the process. Next to that, extending the creation of sessions with weights could be an option. An example is the web page *visit page home* of the UWV dataset. It could be the case that people are just opening this web page and do other work that are not related to the session. For example, they can take a break or just go to another website. This will result in a time difference that is high and actually not right compared to the other activities. Such activities get more priority and this will definitely affect the clustering of the sessions, which will result in a different abstract event log and thus process models. Punishing these activities could be an option. Thirdly, it would be an opportunity to discover process models for the sessions in a cluster. Hereby, it can be analyzed if sessions belonging to a cluster have similar model patterns than sessions belonging to another cluster.

Lastly, our work could be tested on multiple real-life datasets. In this thesis, we extra evaluated our work with the Online Retail dataset provided by the UCI Machine Learning Repository. However, this dataset was quite unbalanced. It would be interesting, to use this method on real-life event data, such as of a supermarket where each session is the time when a customer goes to that supermarket and do grocery shopping.

Bibliography

[1] W.M.P. Aalst, van der. *Process mining : discovery, conformance and enhancement of business processes*. Springer, Berlin, 2011. ISBN 978-3-642-19344-6. doi: 10.1007/978-3-642-19345-3. 1, 5, 6

[2] W.M.P. Aalst, van der, B.F. Dongen, van, C.W. Günther, R.S. Mans, A.K. Alves De Medeiros, A. Rozinat, M.S. Song, H.M.W. Verbeek, and A.J.M.M. Weijters. Prom and the challenges of process mining. 2007. URL `http://www.win.tue.nl/~tcalders/dbdbd07.htm`. Dutch-Belgian Database Day 2007 (DBDBD 2007), November 29, 2007, Eindhoven, The Netherlands, DBDBD 2007 ; Conference date: 29-11-2007 Through 29-11-2007. 2

[3] Thomas Baier. *Matching Events and Activities*. PhD dissertation, University of Potsdam, 2015. 19

[4] Thomas Baier and Jan Mendling. Bridging abstraction layers in process mining by automated matching of events and activities. In Florian Daniel, Jianmin Wang, and Barbara Weber, editors, *Business Process Management*, pages 17–32, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-40176-3. 19

[5] Thomas Baier, Andreas Rogge-Solti, Mathias Weske, and Jan Mendling. Matching of events and activities - an approach based on constraint satisfaction. In Ulrich Frank, Pericles Loucopoulos, Óscar Pastor, and Ilias Petrounias, editors, *The Practice of Enterprise Modeling*, pages 58–72, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. ISBN 978-3-662-45501-2. 19

[6] Joos CAM Buijs, Boudewijn F Van Dongen, and Wil MP van Der Aalst. On the role of fitness, precision, generalization and simplicity in process discovery. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 305–322. Springer, 2012. 6

[7] B.F. Dongen, van, A.K. Alves De Medeiros, H.M.W. Verbeek, A.J.M.M. Weijters, and W.M.P. Aalst, van der. The prom framework : a new era in process mining tool support. In G. Ciardo and P. Darondeau, editors, *Applications and Theory of Petri Nets 2005*, Lecture Notes in Computer Science, pages 444–454, Germany, 2005. Springer. ISBN 978-3-540-26301-2. doi: 10.1007/11494744_25. 2

[8] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996. 10, 12

[9] Bettina Fazzinga, Sergio Flesca, Filippo Furfaro, Elio Masciari, and Luigi Pontieri. A probabilistic unified framework for event abstraction and process detection from log data. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 320–328. Springer, 2015. 19

[10] Diogo R Ferreira, Fernando Szimanski, and Célia Ghedini Ralha. Mining the low-level behaviour of agents in high-level business processes. *International Journal of Business Process Integration and Management 8*, 6(2):146–166, 2013. 19

[11] Christian W Günther, Anne Rozinat, and Wil MP Van Der Aalst. Activity mining by global trace segmentation. In *International Conference on Business Process Management*, pages 128–139. Springer, 2009. 19

[12] Hari Krishna Kanagala and VV Jaya Rama Krishnaiah. A comparative study of k-means, db-scan and optics. In *Computer Communication and Informatics (ICCCI), 2016 International Conference on*, pages 1–6. IEEE, 2016. 12

[13] Sander J. J. Leemans, Dirk Fahland, and Wil M. P. van der Aalst. Discovering block-structured process models from event logs containing infrequent behaviour. In Niels Lohmann, Minseok Song, and Petia Wohed, editors, *Business Process Management Workshops*, pages 66–78, Cham, 2014. Springer International Publishing. 7

[14] S.J.J. Leemans, D. Fahland, and W.M.P. Aalst, van der. Exploring processes and deviations. In F. Fournier and J. Mendling, editors, *Business Process Management Workshops : BPM 2014 International Workshops, Eindhoven, The Netherlands, September 7-8, 2014, Revised Papers*, Lecture Notes in Business Information Processing, pages 304–316, Germany, 2015. Springer. ISBN 978-3-319-15894-5. doi: 10.1007/978-3-319-15895-2_26. 6, 7

[15] S.J.J. Leemans, D. Fahland, and W.M.P. Van Der Aalst. Using life cycle information in process discovery. In M. Reichert and H.A. Reijers, editors, *Business Process Management Workshops*, Lecture Notes in Business Information Processing, pages 204–217, Germany, 2016. Springer. ISBN 978-3-319-42886-4. doi: 10.1007/978-3-319-42887-1_17. 7

[16] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967. 9

[17] Felix Mannhardt and Niek Tax. Unsupervised event abstraction using pattern abstraction and local process models. *arXiv preprint arXiv:1704.03520*, 2017. 19

[18] Felix Mannhardt, Massimiliano De Leoni, Hajo A Reijers, Wil MP Van Der Aalst, and Pieter J Toussaint. From low-level events to activities-a pattern-based approach. In *International Conference on Business Process Management*, pages 125–141. Springer, 2016. 19

[19] Shi Na, Liu Xumin, and Guan Yong. Research on k-means clustering algorithm: An improved k-means clustering algorithm. In *Intelligent Information Technology and Security Informatics (IITSI), 2010 Third International Symposium on*, pages 63–67. IEEE, 2010. 9

[20] Michael Palmer. Data is the new oil. *ANA marketing maestros*, 2006. 5

[21] Wil MP Van Der Aalst, Hajo A Reijers, and Minseok Song. Discovering social networks from event logs. *Computer Supported Cooperative Work (CSCW)*, 14(6):549–593, 2005. 6

[22] W.M.P. van der Aalst, A.J.M.M. Weijter, and L. Maruster. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16: 2004, 2003. 6

[23] Boudewijn F van Dongen and Arya Adriansyah. Process mining: fuzzy clustering and performance visualization. In *International Conference on Business Process Management*, pages 158–169. Springer, 2009. 19, 20

[24] M. L. van Eck, N. Sidorova, and W. M. P. van der Aalst. Enabling process mining on sensor data from smart products. In *2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS)*, pages 1–12, June 2016. doi: 10.1109/RCIS.2016.7549355. 20

[25] H.M.W. Verbeek, J.C.A.M. Buijs, B.F. Dongen, van, and W.M.P. Aalst, van der. Prom 6 : the process mining toolkit. In M. La Rosa, editor, *Proceedings of the Business Process Management 2010 Demonstration Track (Hoboken NJ, USA, September 14-16, 2010)*, CEUR Workshop Proceedings, pages 34–39. CEUR-WS.org, 2010. 2

[26] A. J. M. M. Weijters and J. T. S. Ribeiro. Flexible heuristics miner (fhm). In *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 310–317, April 2011. doi: 10.1109/CIDM.2011.5949453. 8

[27] AJMM Weijters, Wil MP van Der Aalst, and AK Alves De Medeiros. Process mining with the heuristics miner-algorithm. *Technische Universiteit Eindhoven, Tech. Rep. WP*, 166:1–34, 2006. 7, 8

[28] A.J.M.M. Weijters, W.M.P. Aalst, van der, B.F. Dongen, van, C.W. Günther, R.S. Mans, A.K. Alves De Medeiros, A. Rozinat, M.S. Song, and H.M.W. Verbeek. Process mining with prom. In M. Dastani and E. Jong, de, editors, *Proceedings of the 19th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC 2007), 5-6 November 2007, Utrecht, The Netherlands*, Netherlands, 2007. Utrecht University. 2

[29] Jyoti Yadav and Monika Sharma. A review of k-mean algorithm. *International Journal of Engineering Trends and Technology (IJETT)–Volume*, 4, 2013. 9

Appendix A

Online Retail Dataset

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
554065	22752	SET 7 BABUSHKA NESTING BOXES	4	22-5-2011 10:39	8,5	1827	United Kingdom
554065	22583	PACK OF 6 HANDBAG GIFT BOXES	6	22-5-2011 10:39	2,55	1827	United Kingdom
554065	22582	PACK OF 6 SWEETIE GIFT BOXES	6	22-5-2011 10:39	2,55	1827	United Kingdom
554065	84584	PINK GINGHAM CAT WITH SCARF	6	22-5-2011 10:39	2,55	1827	United Kingdom
554065	84920	PINK FLOWER FABRIC PONY	4	22-5-2011 10:39	3,75	1827	United Kingdom
554065	22807	SET OF 6 T-LIGHTS TOADSTOOLS	6	22-5-2011 10:39	2,95	1827	United Kingdom
554065	47421	ASSORTED COLOUR LIZARD SUCTION HOOK	24	22-5-2011 10:39	0,42	1827	United Kingdom

Table A.1: A fragment of the Online Retail dataset