Linköping University | Department of Computer and Information Science
Master's thesis, 30 ECTS | Datateknik
2022 | LIU-IDA/LITH-EX-A--2022/003--SE

A Comparison of Distance Metrics for Trace Clustering in Process Mining

– An Effort to Simplify Analysis of Usage Patterns in PACS

En jämförelse av distansmetriker för användning inom trace clustering i process mining

Christoffer Sjöbergsson

Supervisor : Jonas Wallgren
Examiner : Cyrille Berger

External supervisor : Simon Jönsson

LINKÖPING UNIVERSITY

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Abstract

This study intended to validate if clustering could be used to simplify models generated with process mining. The intention was also to see if these clusters could suggest anything about user efficiency. To that end a new metric where devised, average mean duration deviation. This metric aimed to show if a trace was more or less efficient than a comparative trace. Since the intent was to find traces with similar characteristics the clustering was done with characteristic features instead of time efficiency features. The aim was to find a correlation between efficiency after the fact. A correlation with efficiency could not be found.

Contents

List of Figures

List of Tables

1 Introduction

This chapter will outline the motivation, aim, research questions, and delimitations of this thesis.

1.1 Motivation

Today software becomes more and more prevalent in all parts of society both for entertainment and not least for different critical infrastructure applications. As software has become more widespread it has also been more and more common that software has a very large set of users. This has led to data-driven development becoming a more common method to understand user behaviors in increasingly large systems. As a further result of this event-log collection has increased.

Event-logs can present an invaluable source of information but are inherently hard to navigate and use[39]. An event log is often used in so-called business intelligence applications in order to give insight into how much different features are used, driving sources of traffic to web applications, etc. But when it comes to analyzing user behavior in a more comprehensive way, the traditional approaches often fall short. This is where process mining has shown to be a possible solution[30, 3].

Process mining can be used to generate graphs that make it easier to visualize and understand whole processes rather than individual features or events[3, 1]. This technique has already proven useful in a lot of different business process analyzing scenarios [37].

Processes can be broadly categorized into two different groups[1]. The first one being so-called lasagna processes. The second being spaghetti processes. As the names suggest the first of the two are processes that can be easily modeled in a structured way with events in an ordered and mostly linear way. Spaghetti processes on the other hand are processes with greater variability and complexity. While lasagna processes are easier to analyze it is hypothesized that a greater value can be found in being able to successfully analyze spaghetti processes[31]. When applying process mining to user event logs from software applications it is most likely that the result will be a spaghetti process. This is a problem since process mining works less well for analyzing more complex processes like these [39, 36, 35].

Because of this, it would be of great value to find a way to simplify the models in order to better understand the users' behaviors. There have been a lot of different approaches described in the literature that tries to deal with complex data. One of these techniques is trace

clustering [39, 35, 36]. Trace clustering has been proven to be able to simplify models and can also be used to create models that more accurately describe a certain process better[39, 12]. The intention with clustering is to find smaller subsets(clusters) of data where each subset contains user behaviors that are more similar to each other. The hypothesis is that data with less varied behavior will create less complex models.

1.2 Background

When developing software there is a need for developers to know their users' needs[21]. Traditional approaches to this include testing the software in a controlled environment with a relatively small group of test users. Another approach is to observe users in their regular work environment. Both of these methods can be resource-intensive and slow. Analyzing user behavior is even harder when you have thousands of users in several countries. One of the ways to analyze software on a large scale is to collect and use data about how the existing users use the system. This approach is often referred to as data-driven development. The methods that make use of data varies greatly, from simple counters of how much a specific function has been used to more advanced analysis of the data.

A picture archiving and communications system often called a PACS, is a kind of system that allows hospitals and other medical institutions to collect, save and analyze different medical images [14]. These images come from a variety of sources (modalities) like X-ray machines, CT-scans and MR-cameras just to name the most common ones. The systems have advanced features to sort and view images from different patients that are taken as part of different examinations. The software provides the medical professionals tasked with analyzing the images with tools to document, analyze and share medical images with each other in order to provide good and accurate diagnoses and treatments to patients. When a PACS was begun to be used in the hospitals it significantly changed the way that people at the hospitals worked[14].

These systems are very complex and users can interact with them in a lot of different ways[16]. Even performing the same kind of diagnosis can be done in several different ways and differ based on what instructions the user has received and also based on personal preference and experience. This makes a system like this ideal when trying to find good ways to apply process mining on a complex user interface.

This thesis is done in collaboration with Sectra. They are developing IDS7, which is one of the leading PACS used at hospitals and other care facilities in Sweden and many other countries. This system will be used as the basis for the research presented in this report. Sectra logs all commands issued by its users in a log which makes for a perfect data source to perform the process mining experiments.

1.3 Aim

Sectra is interested in being able to draw better conclusions about user behaviors from the data that they are collecting. Right now one of the primary limitations is that the data is too complex to be able to do this without some sort of pre-processing. There are two primary aims of this work. The first is simply to be able to present the data in a simpler form that can be more easily interpreted. The second aim is to see if clustering of data can create any meaningful classification of processes in terms of efficiency. If the clustering shows a clear separation between clusters there might be a possibility to find useful characteristics for the different clusters. This could in the end be used to find ways to move users' interaction patterns from a less desirable cluster to a more desirable one by making changes in the software or by user training.

1.4 Research questions

1. What distance metrics exist that would be applicable to trace clustering in process mining?
2. What distance metric appears to be best suitable to create simpler models according to the literature?
3. How well does this metric perform in terms of simplicity and fitness on usage data from a medical application?
4. Are there any simpler metrics and how do they compare to the above metric?
5. Does any of the groupings created with the compared distance metrics correlate with user efficiency?

1.5 Delimitations

There are some delimitations to this work. Firstly the selected data comes from one specific, but in this report anonymized hospital. There is also only data from a few different procedures. Another consideration is that the data is from real active systems and is noisy. This means that there might be traces that do not represent a real use case. There might also be real use cases that due to mislabeling are not included in the data.

2 Theory

In this chapter, the theory of some key concepts will be outlined and explained.

2.1 Data driven development

Data-driven development is a concept that does not only exists in software development but rather in all areas of product development that contains software [17]. The concept is rather simple in that it tries to build all important development decisions on data that is continuously collected during the lifecycle of the software [10]. Data-driven development tends to work well within software development simply because it is easy and cheap to collect all sorts of data [17]. Almost too easy, to the degree that the amount of data itself makes it hard to know what the data means.

2.2 Data and model representations

This section will describe the different data and model representations used within this thesis.

2.2.1 Event logs

An event log can contain different information depending on what information is considered to be of importance. Each row in the log represents one event that has happened. What is common and also happens to be required if the log is to be analyzed with process mining, is at least the following three pieces of information per row in the log.

- Each row needs an **event label**, which describes what has happened.
- Each row needs a **session id** to group events done in the same session. An example of a session could be *user logs in to the system→user performs some number of actions→user logs out of the system*.
- All rows in the log must have a distinct order of occurrence. This is usually achieved by having a **timestamp** for each row.

These three parts give us enough information to stitch together a full picture of what actions have been taken and in what order. At least given the assumption that all events of

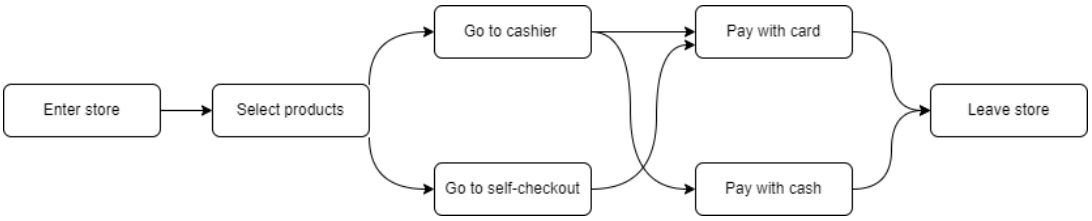

Figure 2.1: An example model

importance have been logged in the first place. While these three information parts are the necessary ones, logging additional information can be useful. This can be e.g. username or some other metadata pertaining to the events that occur. Extra data makes it possible to select data for analysis with greater precision, which in turn can lead to better models and analysis results [2, 4].

In this thesis, a session-id represents an event trace, or trace as it will be called in the rest of this report. A Trace is a set of events that are executed one after another in the software by a single user in a single session. Each trace can be considered a unique instance of a process.

2.2.2 Processes

A process is a very widely used term that can be interpreted slightly differently depending on the context. The Oxford English Dictionary defines a process as *"a series of actions or steps taken in order to achieve a particular end"*. In some cases these actions are documented in a way that is intended to be followed. In other cases processes exist without ever being documented. A naturally occurring process can also be documented as it exists. This can be done to make it easier for others to follow the same process or in order to understand the process better. The processes in this thesis will primarily be of the undocumented kind and describe the different ways that users can interact with a software

2.2.3 Process mining

Process mining is a collection of methods intended to model existing processes by analyzing data collected during the execution of these processes[3, 1]. A more comprehensive description of process mining can be found in Section 2.4.

2.2.4 Process model

A model in process mining is a graphical representation of a process[1]. There are many different ways to visualize a model but the general concept is that a model is a graph. In the graph, each node represents an event and each arc represents how different events are connected. Some ways to model processes can include constructs to visualize more advanced control flows like decisions, loops and, concurrent executions. Figure 2.1 shows a very simple example model of how the different parts of a shopping process may look.

2.2.5 Petri-Net

A Petri-net is a way to model a process commonly used in process mining [1]. Figure 2.2 shows a very simple Petri-net. In a Petri-net there exists places, transitions, and arcs represented by circles, rectangles, and arrows respectively. The black dot in *P1* represents a token. A place can hold a number of tokens and a transition consumes a set of tokens when fired. In this simple example, all transitions consume one (1) token from each place that has an outgoing arc to the firing transition. In a more complex Petri-net, the arcs can be labeled with the number of tokens consumed. A transition is said to be enabled if the ingoing places have

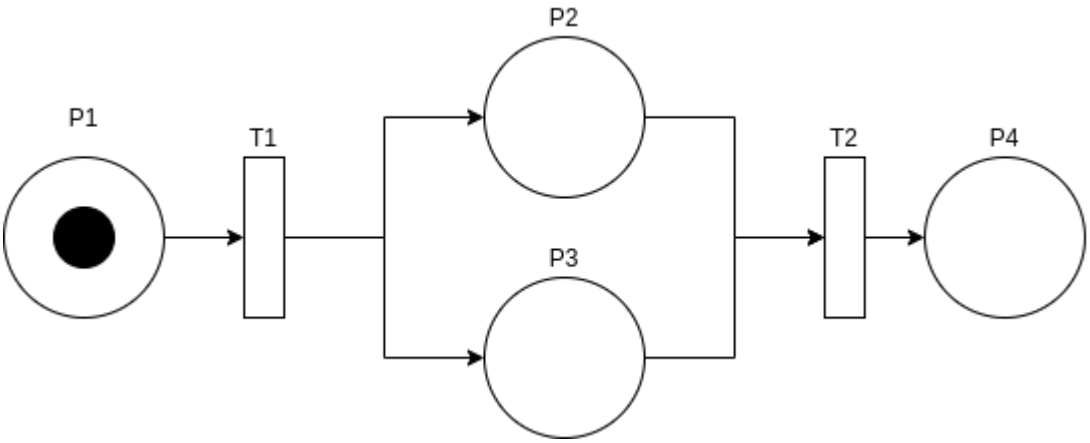

Figure 2.2: A simple petri-net

enough tokens to satisfy the ingoing required number of tokens. When a transition is enabled it can fire and when doing so, produces tokens on the outgoing arcs according to the numbers on these arcs. The outgoing tokens are placed in the places connected to the outgoing arcs. If transaction *T1* in Figure 2.2 where to fire, it would consume the token in *P1* and produce one new token in *P2* and another token in *P3*. This would in turn make *T2* enabled. For a more detailed description of Petri-nets, read the article *Petri Nets* by James L. Peterson [24].

When modeling processes in terms of a Petri-net, the transitions represent events. That means that what is traditionally called a firing sequence in Petri-nets are a trace of events in process mining. If the model allows for the sequence, each event/trace will be enabled when needed. If some transition along the way of the sequence is not enabled then that trace is not allowed by that model. In the conformance checking part of Section 2.4 it will become more apparent why Petri-nets is a suitable way to visualize processes.

2.3 Data quality

Data quality is an important factor when working with event logs from real systems. There are a few different kinds of potential quality issues to take into account [4].

2.3.1 Missing Data

Missing data is when something happened in the real world that was not recorded in the log. Missing data is problematic because it leads to blind spots. Not only by missing the event in question but also because it might lead to an incorrect context representation for the events that are logged properly.

2.3.2 Wrong Data

Wrong data is when something that did happen was recorded as something else. It could also be if an event gets logged when noting has happened. Wrong data can also be more nuanced. If for example, a user behaves irrationally in a software application, there might be logs generated that are accurate but do not represent a realistic user interaction. This would add noise to the data which might make it harder to understand and draw conclusions from the data.

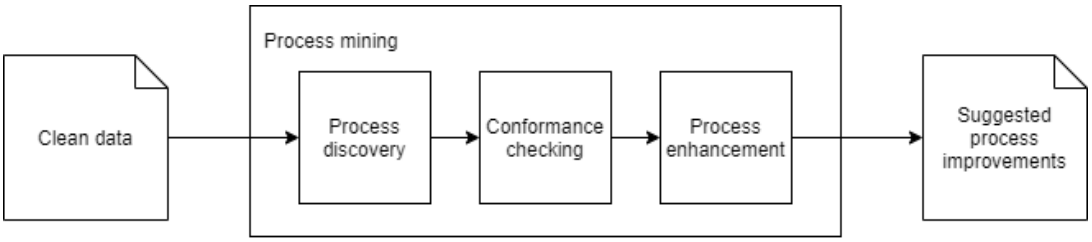

Figure 2.3: The overall structure of process mining

2.3.3 Ambiguous Data

Ambiguity is when the same recorded event means different things depending on the context. One such example is if a log describes a car inspection process. It might start with the car owner handing in the car to the inspector. It likely ends with the inspector handing off the car to the car owner. If these two events would be logged as the same kind of event, e.g. *owner meets inspector*, that can create ambiguity and as such might be the source for a bad model. This does in large part depend on what type of process the data try to capture. If the data intends to log all interactions between different human actors, then it might make sense to treat the two interactions as the same kind of event. In a case where the data represents the car inspection process, *car hand-in* and *car hand-off* might be a better way to label the events. That way it is clear that one of them can only happen at the beginning of the process while the other can only happen in the end.

2.3.4 Too Detailed Data

In some cases too detailed information can also be a problem. If there are too many events that describe a part of the process that is not interesting then the model gets unnecessarily complex by having them there.

The desired level of detail can be dependent on what data is most relevant. In IDS7 for example the commands could be divided into different kinds of categories like chat, examination, worklist related, etc. If you are interested in how users behave when they are creating new worklists, you might decide that the commands related to the chat are not of interest. In that case, you could either remove the chat events altogether or you could merge them into one event type. By merging the events into a single new event, that event would become very ambiguous. However, if the detail in that area of the process is less important it could make sense. Knowing that the user used some chat functionality may be enough detail.

2.4 Process mining

Process mining is a series of techniques that mix techniques from process analysis and data mining [1]. Data mining is when data is mined for valuable insights. One such example could be extracting key characteristcs about users of an application, these characteristcs may be what kind of web browser they use, their age etc. The difference between data mining and process mining is that rather than single characteristcs, process mining tries to capture the characteristics of entire processes. I.e. how do different events interact and take the order of events into consideration while drawing conclusions.

Figure 2.3 visualizes how the process mining process works on a very simplified high level. Clean data is supplied in one end and the result is process improvements in the other end. It can be argued that cleaning the data is a part of the process discovery phase but for clarity, it is broken out in its own step. Overall process mining can be said to be divided into three different stages, process discovery, conformance checking, and process enhancement, which are described in more detail below [1].

2.4.1 Process discovery

Process discovery can be described as the core part of process mining [1]. It is the part where a model is discovered from the event log. The event log must fulfill some basic requirements to be suitable for process mining, described in section 2.2.

There are many different algorithms, often called miners, for process discovery. Some of the more common miners include the alpha miner, the heuristic miner, and the inductive miner [1]. Miners are used to create a visual model for what process is generating the data. All of the miners have their own ways of working and are suitable for different kinds of processes. One of the deficiencies of the alpha miner for example is that is can not accurately describe processes with loops in them. On the other hand, it is rather simple to implement. While different miners are good for different things not all miners are created equal and more recent algorithms are often more powerful with fewer drawbacks than the older ones. Miners tend to have a standard way of visualizing the resulting model. The visualization is often a natural result of the inner workings of the miner and can vary between different miners. While processes can be visualized in different ways, it is also common to transform one visualization to another. This could be done for example if the intended audience of the models is already familiar with a certain modeling standard [1]. Some of the used ways to visualize a model are Petri-net, heuristics net, decision trees, and the Business Process Modelling Notation (BPMN). In this study, the Petri-net model presented in Section 2.2 will be used to visualize different models.

Some event logs are hard to mine for accurate models. This is especially true for processes with many choices, parallel executions paths, many different kinds of events and if they contain ambiguous events [1, 3].

Inductive miner

The miner used in this report is the inductive miner. There exists a few different implementations of the inductive miner, the one referenced in this thesis is the one defined and described by Leemans, Fahland and van der Aalst in the paper *Scalable Process Discovery with Guarantees* [20]. It is reasonably efficient and avoids most of the common problems that exist in the earlier and more simple algorithms. The inductive miner always guarantees a sound model. In some circumstances, it can create a model with a low generalization value, but that is a problem common for many of the existing miners. All ways of discovering a process from data start with the traces. The default output of the inductive miner is a so-called action tree, but before that can be created the miner first creates a directly follows graph (DFG). A DFG is a graph that visualizes all events that occur directly after each other, see an example in Figure 2.4. The creation of a DFG is simple and can even be done by hand given a small enough collection of traces. A DFG has events as nodes and the edges connect all events that occur after each other at least once in one or more traces in the data. This graph is created by finding events in the log and checking what preceding events they have. The edges in the DFG also have numbers indicating how many times that event pairing occurs. In order to turn the DFG into an action tree, the miner works by recursively iterating over the graph and creating smaller and smaller sub-graphs. The division into smaller graphs can be done with four different cuts. Exclusive-or cut, Sequence cut, redo-loop cut, and parallel cut.

- An **exclusive-or cut** is a cut that divides the DFG into two parts where the control flow requires one or the other part to be selected, i.e. there are no arcs between the two parts.
- The **sequence cut** can be used in a place where all arcs are outgoing to another part of the graph, i.e. any of the events in part A will always be followed by some event in part B.

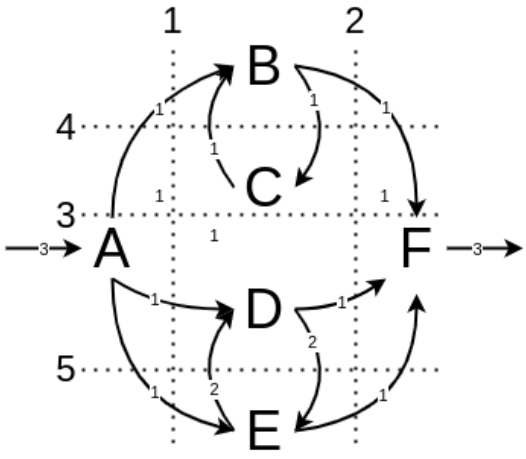

Figure 2.4: DFG created from example traces.

- A **redo-loop cut** can be used when all outgoing arcs from part B are connected to the same nodes of part A as part A's ingoing arcs, i.e, part B can be executed only after A and if executed, A has to be executed again.
- A **parallel cut** can be used when all nodes in both parts are connected by bidirectional arcs, i.e, all nodes can be reached in any order.

See the following example to better understand how the different cuts can be used in practice.

The following is a simple example of the different steps of the inductive miner. In this example a log containing the following three traces will be used: $[A, B, C, B, F], [A, E, D, E, F], [A, D, E, D, F]$. Constructing a directly follows graph is in essence just counting all occurrences of connections between all pairs of events. In this example, all three traces start with A, and all traces end with F i.e. three traces. The rest follows the same pattern, counting how many connections there are between all the different kinds of events. A is followed by B one time etc. The result of this step can be found in Figure 2.4.

When a complete DFG has been constructed, the actual inductive miner algorithm can start. This miner works by recursively iterating over the graph and creating smaller and smaller sub-graphs. In Figure 2.4 each cut is marked with a dashed line. Cut 1 and 2 are sequence cuts, cut 3 is an exclusive-or cut, cut 4 is a redo-loop cut and cut 5 is a parallel cut. Each kind of cut has a corresponding symbol in the action tree that will be created after this step.

To create an action tree, the atomic subparts created by cutting are connected by the appropriate symbol in a tree structure. This results in a graph like the one in Figure 2.5. In Figure 2.6 the action tree has been transformed into a Petri-net.

2.4.2 Conformance checking

When a model has been generated it might be of interest to check how well the generated model corresponds to the data that created it or reality [19]. This is called conformance checking. Conformance is closely related to the different quality criteria described below

Model quality metrics

The quality of a generated model is most often determined based on four criteria: Fitness, Simplicity, Precision and Generalization [11]. Fitness is how well a model fits a data set, i.e. to what degree do the data fit the model[28] The definition of fitness used in this thesis can be seen in Equation 2.1. Missing, consumed, remaining, and produced tokens are the total

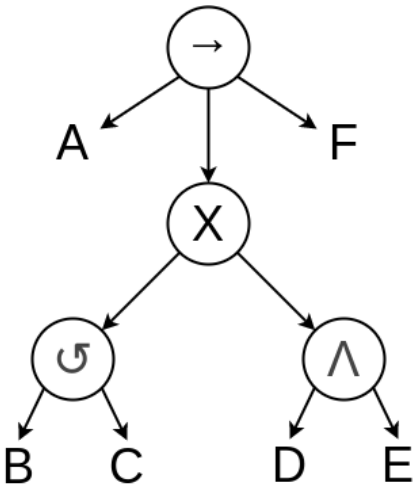

Figure 2.5: Action tree created from example DFG.

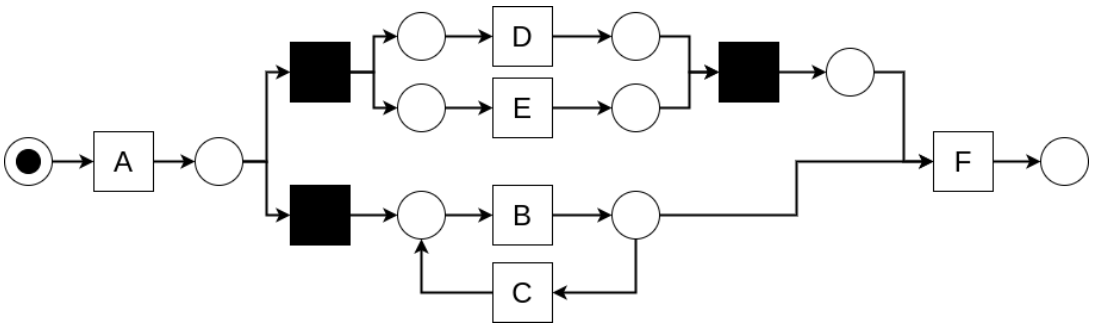

Figure 2.6: Action tree converted into petri-net.

of all tokens in each category for all traces. These tokens are counted after the data has been replayed on the model acording to the token-replay method explained in the next section.

$$fitness = 0.5 * \frac{1 - missing\_tokens}{consumed\_tokens} + 0.5 * \frac{1 - remaining\_tokens}{produced\_tokens} \tag{2.1}$$

Simplicity is usually calculated based on the number of splits and joins of the control flow in the model [11]. The definition of simplicity used in this thesis is found in Equation 2.2. This definition is in essence a comparative metric. *mean_degree* is the mean number of ingoing and outgoing arcs for each node in the model. The parameter k is a value between 0 and infinity that is intended to be a reference arc-degree. I.e. the arc-degree of the model that the simplicity is compared with.

$$simplicity = \frac{1}{(1 + max(mean\_degree - k, 0))} \tag{2.2}$$

Generalization is how well a model can handle traces that were not present in the data but are valid variations of the modeled behavior [28]. One example of this could be to allow for a user to use the zoom function 3 times in a row while the data might only have shown users using it 1, 2, and 4 times. Precision is the opposite of Generalization, i.e. how much more than the observed behavior does the model accept. One common way of measuring precision is model soundness. Model soundness is the percentage of traces, that can be generated by the model, that is acctualy found in the log. Neither Generalization nor precision will be used in this thesis, therefore no exact definitions will be included.

Token-Replay

As with every stage of the process mining process, there are several different ways to go about conformance checking. One of the methods is token-replay, sometimes only called replay[19, 9]. Replay is based on a Petri-net representation[9]. Since a Petri-net has the concept of tokens one can fire the events in a log and consume and generate tokens accordingly in a model. When a trace in the log has been replayed, the model can be analyzed to see where the tokens are. If the trace fits then all tokens must be in the end state, no tokens can be left in the other places in the model. It also requires that all necessary tokens have been available during the replay, i.e. that the replayed actions have been enabled when they have been fired. Fitness is the primary metric that can be calculated by replaying an event log on a model. While fitness is only represented by a number between 0 and 1, the replay process can give more detailed insight into where the model deviates from the traces in the log. This can be useful in process enhancement. The fitness score is based on counts of enabled and disabled transition that has been triggered as well as the number of tokens left in the model by the end of the replay run.

2.4.3 Process enhancement

When it comes to process enhancement, techniques start to blend together more and more with process and network analysis techniques. Different methods can be used to read, analyze and make sense of the models that have been generated and checked. The goal of this stage is to take the new-found knowledge and apply it to the targeted system, software, company, or other process[1]. Methods here can vary greatly depending on what kind of process has been analyzed. In software, it can give an understanding of user behavior that can be used when developing new versions of the program[30]. In business processes, it could be some other analysis that needs to be done. Knowledge might be gained from the models directly or by applying some sort of process analysis technique.

2.4.4 Use within software and user behavior analysis

Several papers have been written about using process mining to better understand how users use different software. Rubin et al. propose to use process mining to automatically create user interaction flow models [30]. Calderia and Abreu describe how they intend to analyze software development processes by collecting usage data from IDE software used by developers [13]. Closely related to this thesis is the work by Forsberg et al. that has explored the usage patterns of PACS users before with process mining [16]. A little further away but still in the domain of process mining in software are the works by Cook and Wolf as well as Rubin et al. that focuses on process mining to analyze the software development process [15, 29]. Furthermore, process mining has also been suggested as a tool to understand communication between web services [5]. Holmstöm and Bosch argue that the collection of post-deployment data, i.e. data from systems in use, is an untapped source of information to understand how users use products and software[17].

While process mining in business processes and for analyzing software may try to answer similar questions and use similar techniques there also exists a few differences. Most prominent is that when analyzing business processes it is common to model parallel data flows between different actors. In such a system it might be interesting to do bottleneck analysis. I.e. does one path in the model receive too much traffic. Or find specific nodes that do not manage to handle the traffic fast enough. When analyzing user behavior in a software application, the model represents how the user interacts with the software but the rate of flow in the model would not reflect the performance of the different actors of the system but rather the user's ability to complete that step in the application. In the business process, we have multiple cases of the process interacting in the same system, in the user behavior case we have one user per instance of the software. As such, a bottleneck does not represent conges-

tion of a node but rather a difficulty for the user to perform their task quickly. It could be that the application is slow at that point or that the task might be more time-consuming in nature.

2.5 Clustering

Clustering is often categorized as an unsupervised machine learning technique because it can be used to categorize unlabeled data[18]. The basic concept is to divide data into different groups (clusters), based on different features in the data. What features to use depends on what kind of groupings that are desired. In a machine learning scenario, the data used initially would be so-called training data. i.e. data with known group belongings. The interesting result of this clustering would not be the grouped data itself but rather where the edges of the groups land. These edges can then be used to categorize new, never-before-seen data.

In the case of process mining, on the other hand, the clustering would be done directly on the main data source [39]. In this case, it is not of interest to teach a model to categorize new traces, instead, the biggest benefit is to get smaller chunks of data to discover processes from. One could be fooled to think that this makes the features irrelevant as long as the data can be divided into more than one cluster. That is however not entirely true. If the generated models are to be useful, there has to be a relevant similarity between the traces in each cluster to produce a less complex model [25].

2.5.1 Distance metrics

Several clustering methods are used in process mining contexts and usually, the clustering methods themselves are not very special[25, 39]. What is special is the different distance metrics that are used. For clustering to work you have to be able to decide the distance between two data points [18]. Depending on what the distance is measured between, this has to be done in a suitable way. In general, distance metrics can be divided into relative and absolute distance metrics. An absolute metric means that each point has a distance with a static reference point. E.g. if the number of events in each trace was to be measured. In this case, a trace with 4 events and a trace with 3 events would always be one event apart. In a relative distance metric, however, all distances are relative between each pair of data points. One example of this could be to count the number of non-shared events.

The general form of a distance metric can be described using the notation $Distance : D(a, b)$, where a and b are traces.

In an absolute distance metric (D_a) it would be possible to use addition and subtraction of distances to calculate unknown distances, see Equation 2.3. In a relative distance metric (D_r) on the other hand that would not be the case, see Equation 2.4.

$$D_a(a, b) + D_a(b, c) = D_a(a, c) \tag{2.3}$$

$$D_r(a, b) + D_r(b, c) \neq D_r(a, c) \tag{2.4}$$

More specific details about the distance metrics used in this thesis can be found in section 4.3.

2.5.2 Clustering methods

There are a lot of different methods to cluster data. This report will mainly include two of them, K-means and Agglomerative Hierarchical Clustering. Both of these methods work well for absolute distance metrics but only the last of the two works for relative distances.

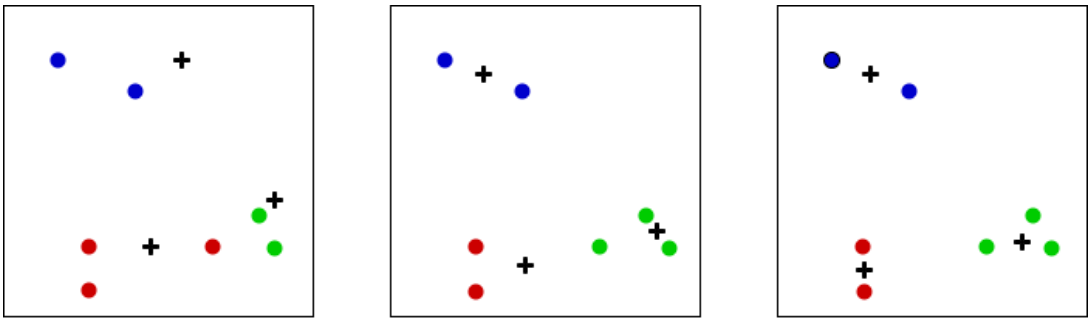

Figure 2.7: K-means steps.

K-means

K-means clustering divides n data points into k clusters by finding the least common mean[18]. This is done by first assigning k random points, called centroids. Each data point is then assigned to the closest centroid. The centroids are then moved to the mean of each cluster. Then the data points are again assigned to a centroid. This process is repeated until the centroids no longer move, see Figure 2.7. In some cases, it can take a very long time for the centroids to stop moving entirely which is why the algorithm often has an iteration limit when used in practice. Another point worth mentioning is that the solution finds a local optimum solution and not necessarily a global optimum. One way of trying to find the best possible solution is to perform the clustering multiple times with different starting points for the centroids and then choose the best clustering result. That requires that there exists such a thing as best clustering which might be hard to know depending on what data is clustered.

Agglomerative Hierarchical Clustering

Agglomerative Hierarchical Clustering (AHC) is a bottom-up clustering method that starts with all data points in their own cluster[18]. Clusters are then joined together to form larger clusters until all data points are in the same cluster. This is often presented in a dendrogram, Figure 2.8. The tree can be cut at the desired level to give the desired number of clusters. In Figure 2.8 for example, a potential cut is shown as a dashed line. That cut would give three clusters. The joining of clusters can be based on different criteria, such as the least possible spread between min and max distance values or minimal total distance between all data points. I.e. the distance between two clusters considered for merging can be done either on one or more data points in the respective clusters.

2.5.3 Cluster significance

Cluster significance is a way to determine the quality of a clustering method. When evaluating how well a clustering method works, it is important to have some kind of metric. Custer significance is an aggregated measurement based on individual cluster quality measurements. In practice, this means that each cluster is evaluated according to some metric first and then the results of this metric for all clusters are combined into a single score that determines the quality of the clustering method itself rather than the individual clusters.

There exist different ways to define cluster significance depending on what factors are of interest. Often cluster significance is based on the shape or separation of the clusters. This can be useful in some cases.

In the case of trace clustering to generate process models with less complexity, however, it is more important to measure how the clustering affects the resulting models. Therefore cluster significance in this report will be based on the definition presented by Chandra Bose and van der Aalst [25]. They propose to use the aggregate fitness and complexity metrics of the

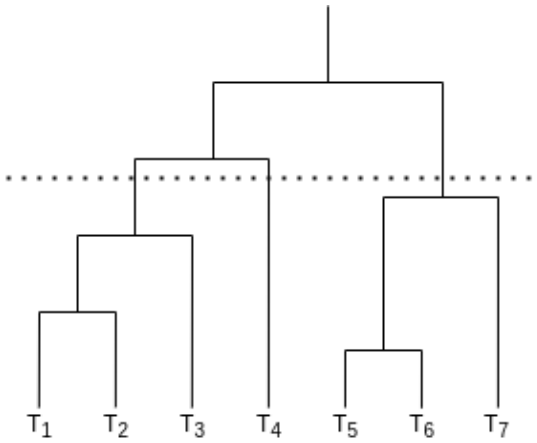

Figure 2.8: Example dendrogram.

models generated by the data in the different clusters. They do this by calculating a weighted average complexity and fitness for all models generated by the different clusters. In practice, this means that a model is generated for the data in each cluster. These models are evaluated for fitness and complexity and a weighted average is calculated for each of the metrics. The number of traces in each cluster is used as weights. The resulting average complexity and fitness values can then be used to compare different ways of clustering. Higher fitness and lower complexity mean a better clustering method.

2.6 User testing and efficiency metrics

Efficiency is often considered to be part of usability engineering [21]. While there exist many slightly different definitions of efficiency, it is most commonly defined by the time it takes to successfully perform a well-defined task. Shorter durations indicate higher efficiency and longer durations indicate lower efficiency. One common way to determine user efficiency is by giving a test user a set of assignments and then studying the user while they perform the assignment. When studying the user a test supervisor writes down what the user does and what they remark. The supervisor also counts successful assignments as well as the number of errors and measures the time that it takes for the test user to perform the tasks. There is often more than one task but each task is given to the user one at a time. This process can be repeated to more test users to gain better confidence in the results.

The benefit of these kinds of tests is that they can give much details and information about how a potential user experiences the software interface and many different improvements can be identified immediately. The problem may be that it is slow and takes many resources to perform [8]. A test of this kind usually needs two to three persons; one instructor that tells the test user what to do, one observer that writes down the result of the tests, and in some cases a so-called driver that maneuvers and controls the moving parts of the tests. There is also an inherent time constraint in that the tests take a relatively long time for each test person which makes it hard to perform tests with too many test users.

3 Method

In this chapter, the method used will be described in detail.

3.1 Pre-study

This study started with a pre-study to gather information about the domain in order to find and select the methods to compare. The pre-study is described in detail in Chapter 4.

3.2 Data selection and pre-processing

The basis for this report is data provided by Sectra. More specifically the data is from, what Sectra calls, the command usage log. This is a database that contains logs from all user interactions done in Sectra's IDS7 PACS system. The information is structured in a few different tables but the important part is how the data looks when it has been exported. The exported data is a simple text-based log file (CSV) where each row corresponds to an event by a user in the program. Each row has six columns, one for the event name, one for the timestamp, one for the session id, one for the body part tag, and lastly two columns that identify which user and what role that user has. There is more data per row than is strictly needed for process mining as described in section 2.2. This makes it possible to filter the data further before process mining. The body part column contains descriptions of what kind of procedure is being examined. The user role can also be of interest to be able to filter out only one kind of user role e.g. Radiologist.

3.2.1 Data collection

The data used in this thesis has been collected from data logs generated by the IDS7 PACS. IDS7 creates these logs collecting almost all of the interactions performed by the user in the software. These logs are primarily intended to be used in business intelligence applications, i.e. more of a traditional data mining application and not for process mining. As such some transitional events that could indicate what way the user arrived at the destination may be missing. These logs have been exported and a subset meeting the data selection criteria outlined in the next section has been used for the analysis. No modification has been done to the data collection in IDS7.

3.2.2 Data selection

Since the command usage log contains all interactions by all users from all hospitals that use IDS7 it is necessary to select what data to perform the tests in this study on. The following things have been taken into consideration when selecting the data.

- **Covid19 -** Data has been selected from before Covid19 had any major impact on how the hospitals worked.
- **IDS7 Version -** Since Sectra updates its software regularly the data has been selected in a way that all data in a dataset is from the same version of IDS7. Furthermore, the user interaction logging has been extended and improved along the way, therefore data from the latest possible version have been selected. That does not mean that the data comes from the latest IDS7 version because Covid19 made the most recent data unsuitable to use and because some hospitals do not update straight away.
- **Hospital -** Data has been selected in a way that each dataset does not contain data from more than one hospital or region. This is mostly because the labeling of body parts and roles differ between different sites making it hard to do a comparison without domain expertise. Combining data from different hospitals in the same dataset would be even harder.
- **User role -** All dataset contains data from only one user role.
- **Body part -** All dataset contains data tagged with the same body part.
- **Duration -** In order to filter away traces that do not represent real use, only traces between 5 seconds and 30 minutes were chosen. This is to avoid cases where the user might have left the computer mid analysis or mistakenly created a very short session by doing a wrong click.

Given these considerations, data were selected between January and March of 2020 from a hospital in Sweden. According to Sectra who provided the data, one of the more standardized workflows performed in the software at the selected hospital is Mammography screening. This data will be used as it is one of the least complex datasets available. Furthermore, it is also one of the workflows with the most data available. This will be compared to data from radiologists analyzing images of lungs to see how the results vary based on input data complexity. The datasets with images of lungs should be a little less homogenous and it is also the dataset with the second most traces available.

- **Dataset 1** January - March 2020 - *Mammography screening workflow analyzed by Mammography specialist radiologists*
- **Dataset 2** January - March 2020 - *Images of lungs analyzed by radiologists*

In order to be able to process the data in a reasonable time 3000 traces were randomly selected from each dataset by using Pandas[33] data sampling feature. This makes sure that the data sampling results in a new dataset that retains as much of the original characteristics as possible.

3.2.3 Data Pre-processing

In the context of this study, a session starts when a user executes the command *Set Current Exam* and stops when the user does one of two things, either issue the *Set Current Exam* again and thereby starting a new session, or by logging out of the software. I.e. a user finds an examination in a list of examinations and selects it, the user then interacts with the selected

examination until they select a new one or stop by logging out. At the end of all traces, there is an event called artificial end, or *artend* for short. This event is created during Preprocessing to have a common ending event for all traces since this makes the generated models easier to read. The definition of a session used in this study is not the same as the definition used by IDS7. In IDS7 a session lasts between the user login in and until the user logs out. A session directly from the IDS7 log can therefore include more than one exam selection per trace. In an effort to make valuable comparisons each IDS7 trace is divided into smaller traces where each sub-trace can describe similar instances of the same process, the exam analysis process.

3.2.4 Data Baseline

In order to have something to compare the results to, the unclustered data were analyzed to find some baseline numbers and results. The first thing that was explored was the number of unique event orders that existed in the data. The findings of the data analysis can be seen in section 5.1.2 and in table 5.1. The data was also plotted against the axis of total trace duration and deviation score. See Section 3.4.3 for more information about deviation score. Lastly, the two datasets were mined using the inductive miner to generate a model for each dataset. These models were measured with the same quality metrics as were to be used in the clustered data model evaluation.

3.3 Implementation

The implementation of the tests is done in Python and uses the libraries PM4PY[7], Pandas[32], and SciKit-learn[23] for process mining, data representation, and clustering respectively. The test has been performed on a decently powerful PC. The initial selection, filtering, and preparation of data have been done with Pandas. Both the k-means clustering and the AHC have been done with standard algorithms included in SciKit-learn[34]. It has been determined in the pre-study, Section 4.2.2, that the clusters have to have a human-understandable grouping. Therefore this thesis has decided to try with efficiency as the feature on which to separate the data in the clusters. Therefore there will only be two clusters in all experiments. The intention is to have one cluster with more efficient traces and one with less efficient. The methods implementations described below were selected for evaluation during the pre-study, see Chapter 4 for more details about how the methods were selected and how they work.

3.3.1 Bag-of-Events

BOE is implemented by combining a few standard implementations from Pandas and SciKit-learn. First, a data frame is calculated containing the number of each kind of event as columns and each row represents a trace. Then these values are used to perform a standard k-means clustering. The K-means clustering uses the default values of SciKit-learn i.e. 10 clustering cycles with a maximum of 300 iterations each. K is set to 2 in order to find 2 clusters.

3.3.2 Generic Edit Distance

The GED implementation is a little more complicated. Mostly since the generic edit distance is a relative distance metric. The first step, therefore, has to be to create a distance matrix that contains all the distances between all pairs of traces. To do the actual GED calculations a library called *textdistance*[22] is used. The distance matrix is then used together with the SciKit-learn implementation of hieratical agglomerative clustering to get the clustered logs.

3.3.3 Weighted Generic Edit Distance

As described in Chapter 4, the Weighted Generic Edit Distance is to some degree related to GED. It is different enough that no preexisting libraries could be found and as such had to be implemented from scratch.

The implementation consists of a few different steps. First, the different costs have to be calculated. There are both substitution scores as well as insertion and deletion scores. Insertion and deletions are essentially the same things in reverse order and, therefore, only share the same implementation.

The implementations used in this thesis are made by more or less translating the formulas presented in Section 4.3.6. It is done by applying row and column operations in Pandas on data frames in order to calculate the required costs and distances. In order to improve performance, the distance for a given trace composition is only calculated once even if that same trace composition may have many occurrences in the data. If that is the case the score will be reused for all occurrences.

3.4 Evaluation

In order to be able to compare results from the different methods a couple of different metrics are needed. The intention is to use metrics that can decide the cluster significance i.e. how well the different methods create clusters with well-defined characteristics. In the case of this report, the main focus is to find out if the clustering improves model simplicity.

3.4.1 Model complexity

In order to see if clustering helps in creating models with less complexity the complexity needs to be measured. This is done by first modeling the data in each cluster with the inductive miner. The complexity for each model is then calculated separately with the simplicity metric included in PM4PY. The definition of this metric can be found in Equation 2.2. In this case, the PM4PY default value of k will be used which is 2. In practice, this means that all models will be compared to a model where all nodes have one ingoing and one outgoing arc. This could be seen as the simplest possible model.

When this is done the weighted average can be calculated. The weighted average is the average weighted by the number of traces in each cluster. So if we have the complexity of the models from two clusters, $c_1 = 0.5$ and $c_2 = 1$ where the first of the models contain twice as many traces as the second. In this case the resulting complexity would be $c_{tot} = \frac{c_1 * 2 + c_2 * 1}{3}$. The reason for using a weighted average is that clusters with low complexity and a low number of traces would skew the results towards higher simplicity which would favor clustering techniques that result in uneven distributions. Since clusters with few traces inherently create less complex models, this has to be avoided.

3.4.2 Fitness

Fitness will be calculated for each model but this metric is mostly to make sure that it does not decrease in any major way. As such it is expected to remain high. As mentioned in the theory chapter there are many ways to define fitness. Token replay is essentially when you use the concept of tokens in a Petri-net to analyze how well the Petri-net corresponds to the data, more details can be found in section 2.4.2. The exact version of token-replay used in this case is the one included in PM4PY created by Berti and van der Aalst [9]. After the replay has been performed the fitness is calculated as described in Equation 2.1. The weighted average fitness will be used in this case as well to be able to make relevant comparisons.

3.4.3 Deviation score

As described in Section 2.6, user efficiency is often measured by measuring the time duration of successfully performed tasks. There are two problems with using this definition in this thesis. The first problem is that the logs contain no definition of successful completion. The second problem is that the logs may contain several different tasks of an unknown distribution. There will be no effort put into trying to separate different tasks from each other in this thesis. Rather the efficiency will be regarded as an aggregated measurement over all kinds of tasks.

In regards to determining successful completion of tasks, it exists possibilities to do this by classifying certain events as markers for success. Forsberg et. al. do just this by deciding to only include traces that include a write-report-event [16]. In this thesis, the traces will be analyzed on a group level rather than on an individual trace level. Therefore the assumption will be made that the majority of the traces analyzed represent successful traces in some regard. This decision has been made with the selected datasets in mind, which represents repetitive tasks made by experts and as such should in majority consist of successfully performed tasks.

Based on the considerations above, a new evaluation metric has been developed for this study specifically. It is called average mean deviation or deviation score for short. It aims at measuring relative efficiency between traces with the mean as a baseline.

The approach consists of two main stages:

1. Create a lookup table with the mean duration of each kind of event.
2. Calculate the scores for one or more traces using the lookup table.

The first stage works by finding the duration of all events in the data. Duration is defined as the time between the starting timestamp of one event to the starting timestamp to the next event. If event A was logged at 10:45 and Event B at 10:46 the duration for event A would be one minute. When all durations have been calculated the mean for each kind of event is calculated. E.g. the mean for all events of the type A may be 1.3 minutes.

The second part is scoring one or many traces. When scoring a trace, first each event's deviations ratio is calculated. This is done by taking the mean duration of that kind of event (e_m) and the actual duration of the specific event (e_d) and calculating a ratio as in Equation 3.1.

$$r = \frac{e_d - e_m}{e_m} \tag{3.1}$$

This gives a ratio of how far from the mean that specific event is. The trace then gets the score by taking the sum of the deviation ratios divided by the number of events in the trace, Equation 3.2 and 3.3. E.g. mean times for a hypothetical dataset is $A = 10$, $B = 2$, and $C = 5$. If a trace, $T = [A, B, C, B]$ with durations, $E_d = [3, 4, 6, 2]$ were to be scored the score $T_s = 1/8$ as described by equations 3.2 and 3.3. The intention of this metric is to get a sense of whether the average event in a trace is faster or slower than the mean given the events that happen in the trace.

$$D_t = \sum_{\forall e \in T} \frac{e_d - e_m}{e_m} \tag{3.2}$$

$$T_s = \frac{D_t}{|T|} \tag{3.3}$$

This evaluation metric is intended to give an indication of how well the different clustering techniques manage to capture relative user efficiency between clusters.

4 Pre-study

This chapter will describe the method and result of the pre-study as well as some brief discussion and conclusion.

4.1 Aim

The pre-study aims to answer three main questions.

1. What clustering methods are there that can be applied in order to simplify models typically generated from user event logs?
2. In what way can efficiency be measured to compare the clustering methods' ability to classify traces according to efficiency?
3. Which of these methods would be of interest to test and why?

4.2 Method

This section will describe the method of the pre-study.

4.2.1 Finding clustering methods

To find suitable methods for clustering all ways of performing trace clustering were initially considered.

The search started in a structured way by searching the article databases of *ACM*. *IEEE Xplore* and *Google scholar* for articles meeting the keywords, *process mining* and *trace clustering*. Each paper was evaluated by reading the abstract and conclusion. If the paper appeared to provide a concrete method for clustering event traces in a process mining context or provided a method that could be adapted to be used in such a context, it was saved. The saved articles were read through more thoroughly. Papers that were deemed to provide possible value were also used as a source for further articles by examining the references used in the already found papers. This method continued by this methodology until the paper, *Contex Aware Trace Clustering Towards Improving Process Mining Results* by Jagadeesh and van der Aalst, was found[25]. After deliberation with Sectra, it was decided that the method provided within the

paper were highly interesting and, due to time constraints, further search for methods was halted. The found methods will be presented in the pre-study result.

4.2.2 Selection of methods

The selection of methods to compare in the experimental part was based on which methods would present the best opportunity to answer the research questions, i.e. each method's expected ability to create trace clusters in a way that creates simpler models. There are two aspects to simple models, less complexity in the models and making sure that the model represents a known domain. The first of these aspects is rather easy to see and measure as it is directly measured on the resulting model [11]. Knowing the domain is about knowing what the model represents. A model could represent all users or maybe a subset of them. The key here is that the person that reads the model knows what it represents. Depending on the distance metric this might be easier or harder to know.

4.3 Result and discussion

This section will present the findings of the pre-study. First, a brief comment about a common theme throughout the found models. Then the methods for trace clustering are present and explained.

What can be said about the different methods is that the clustering technique itself usually is not very special or different between the different techniques. The distance metrics on the other hand differ quite a bit. Below follow a brief description of all distance metrics that were found and how they work.

4.3.1 Simple manual time segmentation

The first method is not so much a distance metric as it is a way to perform manual data selection. In this method traces are selected that have a duration that falls within a set time-range. This method can be used to filter away noise in the data, in that case only one time range is keept[16]. Hypothetically more time ranges could be kept and mined for processes separately. To do this traces are manually or semi-automatically divided into different groups based on time constraints. E.g all traces that have a duration of between 5 to 30 seconds in one, 31 seconds to 5 minutes in the next. The groups of traces are then modeled with, e.g. the inductive miner, and the models could be analyzed to figure out how the processes in that time range behave. This method produces a known domain in terms of time but will not capture any other behaviors. Each cluster could be less complex due to less data but this is not guaranteed.

4.3.2 Cluster based on time

The second approach is an extension to the first one in that the segmentation is based on time. Rather than manualy dividing the traces into groups it might be possible to use using k-means clustering on the time axis [18]. This will try to find naturally occurring clusters instead of predetermined time ranges. This method will likely show similar results in terms of simplification. It may even perform worse since the clusters may gather larger amounts of traces in some clusters. It will produce clusters with a rather clear domain though.

4.3.3 Bag-of-Events

The bag-of-events (BOE) method decides the difference between two traces by counting the occurrences of each event in each trace[31]. If [a,b,b,c,a] would be compared to [a,c,c,b] they would have the BOE vectors [2,2,1] and [1,1,2] respectively, i.e. trace one contains two a,

two b, and one c. This would give the difference vector [1,1,1]. Then the clustering can be performed with each kind of event as a dimension in a multidimensional graph. The two first methods presented both have a known domain in terms of time. BOE tries to find categorize traces based on common events rather than time. This has the benefit that it may find traces with greater similarity which in turn have a greater possibility to generate simpler models. The drawback is that it is harder to figure out what the model represents.

4.3.4 N-Gram

If the order of events is to be captured in some way, events can not be tracked without context. One of the methods to achieve this is to count n-grams[25]. This method is similar to BOE. The difference is that instead of singular events, n-grams are counted, i.e. n number of sequential events in a particular order. This has the benefit of capturing a sense of context for each occurring event. If we were to use 3-grams and have the trace [a, b, b, c, d], The 3-grams would be *abb*, *bbc*, *bcd*. So instead of having one dimension for each kind of event, there is one dimension for each n-gram. It is worth considering that this method will have to deal with significantly more dimensions than BOE which could present issues both in terms of performance[25] and distances metric accuracy degradation[6].

This method has the potential to produce better clusters than BOE in terms of capturing trace event characteristics. In terms of the known domain, it is still hard to find out what the models represent. If however, the clusters are better it might be easier to manually analyze the models to draw relevant conclusions about the model.

4.3.5 Generic Edit Distance

A popular method for measuring the distance between strings is Generic Edit Distance (GED) which could be described as a way to measure how many edits it takes to make one string into another[27]. GED works by calculating the cost of editing one of the traces to be exactly the same as the other. The actions that can be taken are addition, deletion, and substitution. In the simplest of cases, an edit has a cost of one and the parts of the trace that is already correct cost nothing. This simple version is often referred to as Levenstine distance.

While this metric is usually used for string comparison, e.g. in spell checking and autocomplete contexts, it could easily be adapted to work on event traces. simply treat each event as a character in a string. This gives a good estimate of distance with a good capture of event order.

The simple cost function may not work very well in complex scenarios because the distance between traces with variations with similar context can be the same as traces with a similar number of differences but the events have a weaker relation. Once again this show more potential than the previous methods to create models with meaning and as such less complexity.

4.3.6 Weighted Generic Edit Distance

Using GED with different weights has been done before to create a distance metric that can treat different symbols in the alphabet differently. This is intended to enhance the performance in regards to including real-world event context[25]. This could be done e.g. to make a substitution more expensive if the substitution is between two events that are unrelated to each other.

It is possible to calculate these costs manually by people with great domain expertise[25]. But given the sensitivity to how the weights are set it may be hard even for domain experts.

Therefore Jagadeesh and van der Aalst have proposed an automated way to calculate these weights[25]. There are two different algorithms involved, one for calculating substitu-

tion scores and one to calculate insertion and deletion scores. The substitution score calculation is based on how likely an event is to occur in each possible context.

The next step in the algorithm is to give every pair of events a score based on how likely they are to occur in the same context. This leads to high scores in cases where the events are likely to occur in the same context and a low score if the events are not likely to occur in the same context.

Substitution Score

This section will describe how the score is calculated when one event is substituted with another event.

Firstly this concept is dependent on the context of the existing event. A context is defined as the leading and trailing event in a tri-gram, i.e. if xay and xby exists in the set of tri-grams generated from the log, that would be the context xy shared between a and b. Contexts are used to calculate, what is known as co-occurrence scores. Co-occurrence scores are calculated by first calculating the number of co-occurrences of each context and pair of events, Equation 4.1. I.e. how many times is a specific context shared between any two given events?

$$C_{x,y}(a,b) = \begin{cases} \frac{n_{xay}(n_{xay}-1)}{2}, & \text{if } a = b \\ n_{xay} * n_{xby}, & \text{if } a \neq b \end{cases} \tag{4.1}$$

The co-occurrence values defined in Equation 4.2, is a measurement of the frequency of co-occurrence in the same context as another event, i.e. how many contexts co-occurrences are shared between event a and event b. $X_{(a,b)}$ is the set of all contexts common between a and b.

$$C(a,b) = \sum_{xy \in X_{(a,b)}} C_{x,y}(a,b) \tag{4.2}$$

M, defined in Equation 4.3, is a matrix containing normalized co-occurrence values.

$$M(a,b) = [C(a,b)/Nc] \text{ where } Nc = \sum C(a,b) \tag{4.3}$$

p_a, as defined in Equation 4.4, is the probability that event a occurs. It is important that all probabilities sum up to 1 since some event will always occur.

$$p_a = M(a,a) + \sum_{b \neq a} M(a,b) : \sum p_a = 1 \tag{4.4}$$

$E(a,b)$ is a matrix containing pairwise values of the expected value of occurrence defined in Equation 4.5.

$$E(a,b) = \begin{cases} [p_a^2], & \text{if } a = b \\ [2p_a p_b], & \text{if } a \neq b \end{cases} \tag{4.5}$$

Finaly, the formula for the substitution score is a log-odds ratio and is defined as seen in Equation 4.6. This function returns the substitution score for substituting a with b.

$$S(a,b) = log_2(\frac{M(a,b)}{E(a,b)}) \tag{4.6}$$

Insertion and Deletion Score

The algorithm used to calculate the insertion and deletion scores is also based on likelihood, but in this case, it is the likelihood that a certain event occurs after another certain event.

The way the comparison process works make insertion and deletions complementary to each other. Because of this, there is no specific score for deletion but rather the two actions

use the same score The reason behind this will be explained in more detail further down in this description.

$G_{xy}(a)$ is the count of three-grams xay. $Z(a,x)$ represents the number of times an event of type a exists directly to the right of an event of the kind x in the data. See definition in Equation 4.7

$$Z(a,x) = \sum_{y|xy \in \text{contexts of } a} G_{xy}(a) \tag{4.7}$$

In Equation 4.8, $norm(a)$ is defined as the sum of $Z(a,x)$, i.e. the total count for how many times a given event a occurres to the right of any other event.

$$norm(a) = \sum_{\forall x} Z(a,x) \tag{4.8}$$

Equation 4.9, defines $Z_{norm}(a,b))$ which is a matrix containing normalized values of event occurences to the right of all other events.

$$Z_{norm}(a,b) = \frac{Z(a,b)}{norm(a)} \tag{4.9}$$

As with the substitution score, the insertion score is a log-odds ratio, Equation 4.10.

$$R(a,b) = log_2(\frac{Z_{norm}(a,b)}{p_a p_b}) \tag{4.10}$$

The definition of p_a is shared with the one for the substitution scores.

Distance Calculation

Contrary to how regular GED works, the algorithm proposed by Jagadeesh and van der Aalst, produces similarity scores, i.e. a higher similarity results in a higher score. This means that a slightly modified version of GED has to be used. The first step is a definition of what it means when two traces are similar. The similarity function has a recursive definition, see Equation 4.11.

$$Sim(U^m, T^n) = max\begin{cases} s(U(m), T(n)) + Sim(U^{m-1}, T^{n-1}) \\ i(U(m), -) + Sim(U^{m-1}, T^n) \\ i(-, T(n)) + Sim(U^m, T^{n-1}) \end{cases} \tag{4.11}$$

Given the traces U and T, U^m and T^n are subsets of the traces including all events from index m and n to the end of the trace. $S(U(m), T(n))$ is a function that returns the substitution cost for substituting the event with index m in S with the event at index n in T. $i(U(m), -)$ and $i(-, T(n))$ are two uses of the indel cost function, the first is inserting $U(m)$ and the second is deleting $T(n)$. This is why there is no need for a separate cost function for deletion, as the edit distance algorithm simply tries to make the two event traces the same. This can be done by modifying either of the two traces. As such deletion in one of the traces can be achieved by inserting the same character in the other trace instead.

The second step is to calculate the distance between two traces. This is done by transforming each similarity score into a distance, see Equation 4.12.

$$Dis(U,T) = \frac{|U| + |T|}{Sim(U,T)} \tag{4.12}$$

If all traces in the dataset are compared pairwise, a matrix where the distance between all different pairs of traces could be generated. This cost matrix can then be used together with agglomerative hierarchical clustering to find clusters with the most similar traces together.

This approach can create clusters with specific behaviors. This should be able to create clusters with good domain coherence although it might still be hard to figure out what domain it represents.

4.4 Conclusions

Given the methods listed above, two out of six methods are based on time and the remaining four methods are based on trace composition. If the models generated are to be of use they have to represent a usable domain. In this case, useable can be defined as a way to classify users' behaviors. While time can be used to find different user behaviors, it would be more interesting to see what time characteristics different user behaviors have. As such the selection of methods will be limited to the four later techniques. The research question asks for the most promising method as well as simpler methods to compare. Of the presented methods, WED seems to be most promising. The remaining three are simpler to implement and as such interested to compare. However, the n-gram version is deemed to create too many dimensions to cluster. As such the selected methods will be BOE, GED, and WED.

5 Results

This chapter will describe the results of this thesis. First, the results of the data quality analysis will be presented and then the experimental results.

5.1 Data

This section will outline the observations made about the data quality as well as data characteristics that have been done along the way of the project.

5.1.1 Quality

The quality of the data has not been analyzed in any structured way because the data provided by Sectra is sourced from real users using real software in the designated environment and collected in a robust way. It seems unlikely for data to be missing or added incorrectly due to any transfer issues like network or software malfunction. In this regard, the data quality is assumed to be very high.

Despite this, some observations have been made regarding data quality. The negative aspect that has been observed is that some data is simply not collected. As of this moment, Sectra is mostly using its data in more classical business intelligence applications. In such systems, there is little to no interest in collecting all interaction events that occur in the software. So in this regard, the dataset is missing some data that would have been useful in a user behavior analysis. There are interactions and clicks in the system that is not collected at all. This has been observed when manually creating logs for testing purposes by interacting with the software. The events that are collected seem to include all the important main events when analyzing and diagnosing a patient but not all the small interactions in-between. In the simplest forms of user interactions, this means that the trace can consist of little more than the events that show that a new examination has been selected.

5.1.2 Characteristics

During the testing of the different distance metrics some general characteristics of the datasets were discovered which can be observed in table 5.1. One of the most critical discoveries that should be taken into consideration when observing the results is that both of the datasets

Table 5.1: Dataset Characteristics

Dataset	Datatype	User Role	#Samples	#Total	#Unique of total
1	Mammography screening	Mammoradiologist	3000	48244	800
2	All images of lungs	Radiologist	3000	48177	753

Table 5.2: Dataset Metrics

Dataset	Datatype	Log fitness	Model Simplicity
1	Mammography screening	0.985	0.567
2	All images of lungs	0.998	0.591

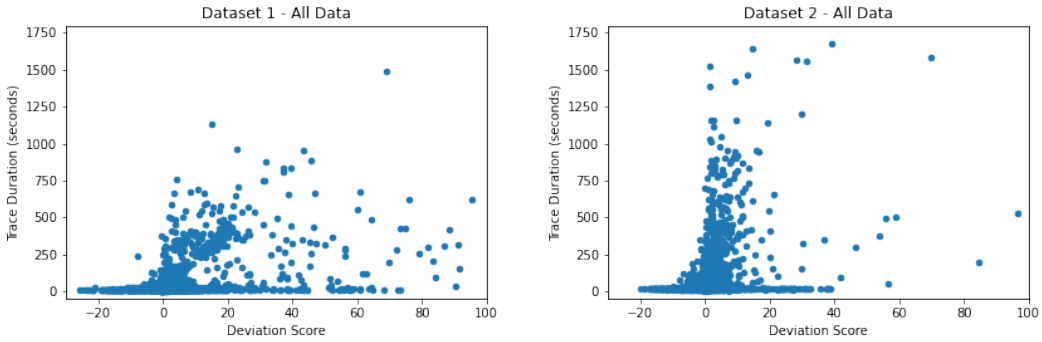

Figure 5.1: Datasets duration and deviation score

consist of more than 50% of a single trace order. This does not mean that all these trace instances are the same, nor does it mean that they share all parameters. What it does mean is that more than half of the traces consist of the same events in the same order. The duration of the trace may vary, as well as which user performed the trace and many other aspects.

The simplicity metrics of the two selected datasets without clustering can be seen in Table 5.2.

All traces have been plotted in a graph on the axis total duration and mean deviation score. All the traces of the two datasets can be seen plotted in figure 5.1 The datasets contain traces between 5 seconds and 30 minutes in duration. As can be seen, there are no obviously separated clusters visible in either of the datasets.

5.2 Experiment Results

Overall it can be said that the results in the different tests look very similar between both datasets. Below will be a summary of the results for each method as well as a table and figures showing the results.

5.2.1 Bag-of-Events

Table 5.3 shows the result for the BOE method. The BOE distance metrics together with k-means clustering puts almost all traces in a single cluster and separates just a handful of traces into the other cluster. This is true for both datasets. For dataset 1 it can be seen that both cluster 0 and 1 shows an improved simplicity score. The fitness is virtually unchanged in cluster 1 and perfect in cluster 0 resulting in an overall improved fitness score.

Table 5.3: Clustering Results Bag-of-Events

Dataset	Cluster	#Traces	Simplicity	Fitness	Avg. Fitness	Avg. Simplicity
1	0	2980	0.600	~1	~1	0.600
	1	20	0.625	0.994		
2	0	7	0.606	1.000	~1	0.579
	1	2993	0.579	~1		

For Dataset 2 the overall fitness score is slightly improved and the simplicity score is worse than for the unclustered data.

As for the time/deviation score-plots, Figure 5.2 and 5.3, there are no obviously visible separation in groups.

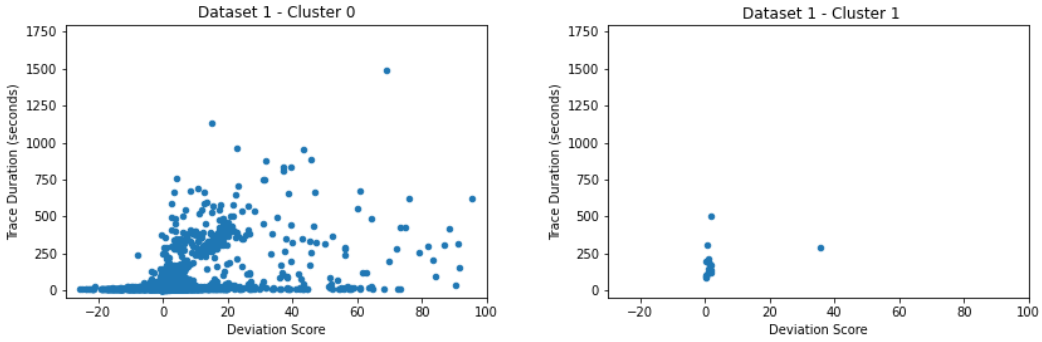

Figure 5.2: Dataset 1 duration and deviation score, BOE distance metric used

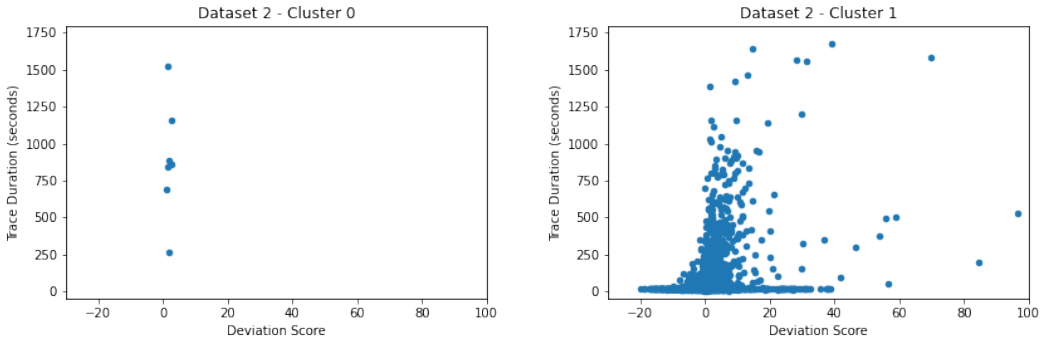

Figure 5.3: Dataset 2 duration and deviation score, BOE distance metric used

5.2.2 Generic Edit Distance

The results for the GED method can be seen in Table 5.4 and Figures 5.4 and 5.5. Similar to how the BOE distance metric worked, this method put all but a few traces in the same cluster. The resulting model simplicity and fitness are greatly improved in the small clusters of each of the two datasets. The large clusters have metrics that are very close to the metrics observed in the unclustered data. Overall the averages are mostly unchanged for all datasets and metrics.

Since the majority of the data is in one cluster, it is impossible to find a clear grouping in the duration/deviation score-plot.

Table 5.4: Clustering Results Generic Edit Distance

Dataset	Cluster	#Traces	Simplicity	Fitness	Avg. Fitness	Avg. Simplicity
1	0	2997	0.567	0.986	0.985	0.567
	1	3	0.846	1.000		
2	0	2999	0.590	0.998	0.998	0.590
	1	1	0.630	1.000		

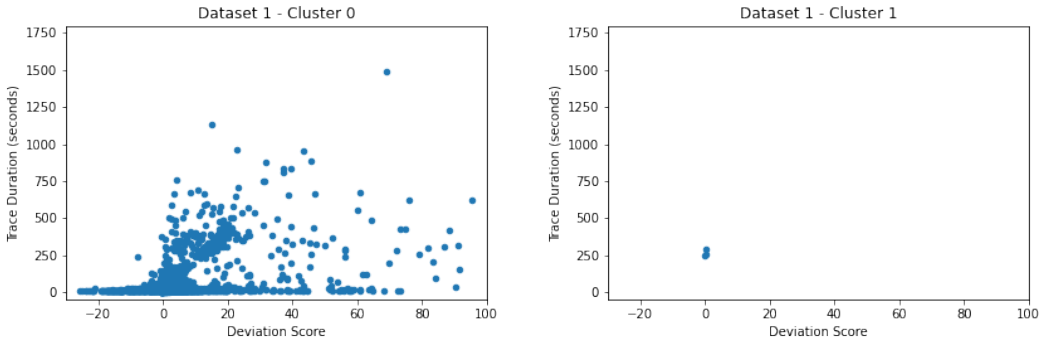

Figure 5.4: Dataset 1 duration and deviation score, GED distance metric used

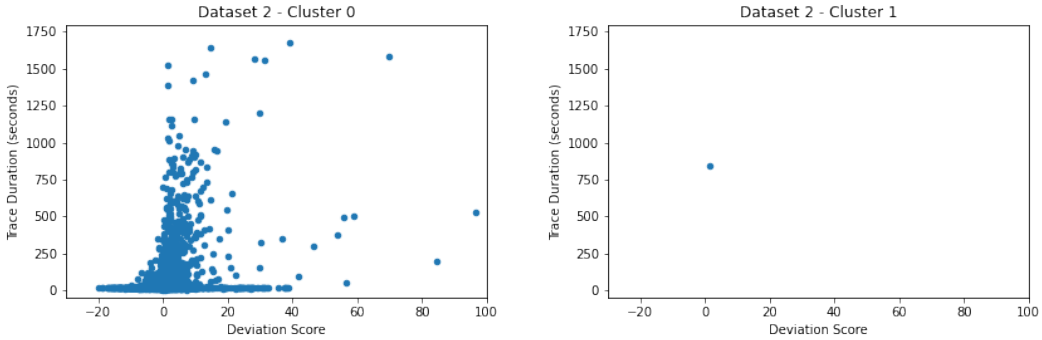

Figure 5.5: Dataset 2 duration and deviation score, GED distance metric used

5.2.3 Weighted Generic Edit Distance

Table 5.5: Clustering Results Weighted Generic Edit Distance

Dataset	Cluster	#Traces	Simplicity	Fitness	Avg. Fitness	Avg. Simplicity
1	0	1632	0.594	~1	~1	0.594
	1	1368	0.593	1.000		
2	0	1172	0.593	0.981	0.993	0.614
	1	1828	0.628	1.000		

Table 5.5 shows the results for the WED method. This method unlike the other two divided the traces into more equally sized groups. The simplicity scores show improvements for both datasets. The fitness scores are improved for dataset 1 but slightly worse for dataset 2. The increase in simplicity for dataset one is marginal but the increase for dataset 2 is among the best in this study.

When it comes to the duration/deviation score-plot, see Figures 5.6 and 5.7, it is not possible to see a clear separation for dataset 1. Dataset 2 on the other hand does show some sort of separation. It can be seen in Figure 5.7 that cluster 1 contains almost exclusively traces with a total duration close to 0 but a wide range of deviation scores. Cluster 0 on the other hand contains traces within a narrower band of deviation score but a wide range of total durations.

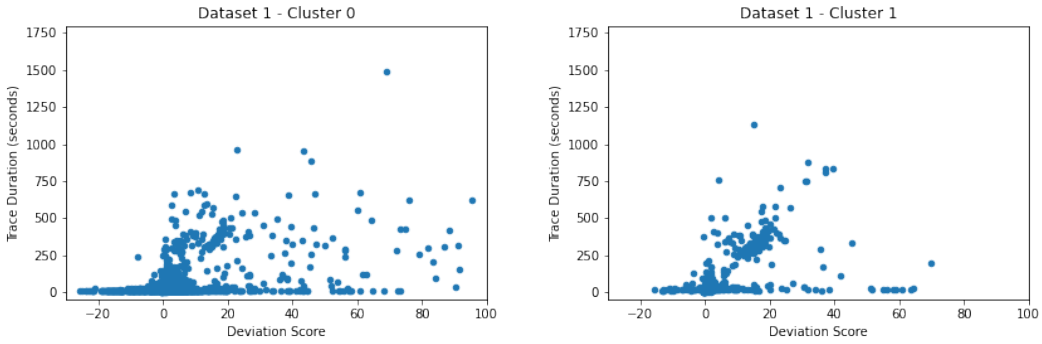

Figure 5.6: Dataset 1 duration and deviation score, WED distance metric used

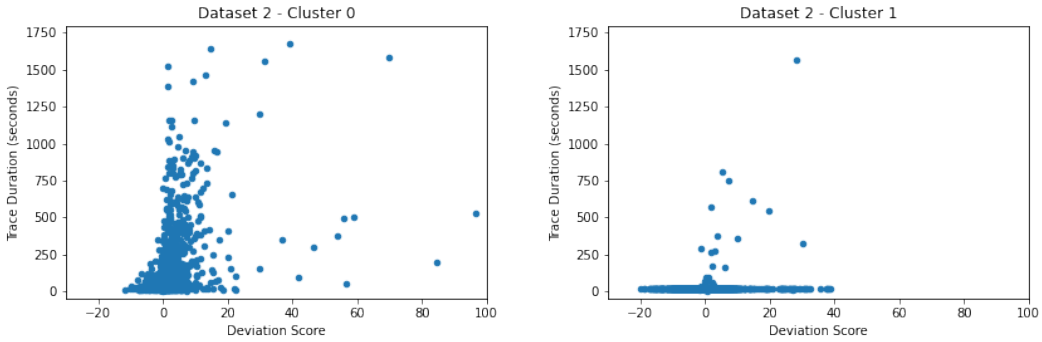

Figure 5.7: Dataset 2 duration and deviation score, WED distance metric used

5.2.4 Comparison of results

Tables 5.6 and 5.7 show a comparison between the results of the simplicity and fitness respectively. What can be seen is that almost all results are unchanged or improved, only fitness for WED and simplicity for GED showed a worse result than the comparative numbers.

Table 5.6: Simplicity Result Comparison

Dataset	Baseline	BOE	GED	WED
1	0.567	0.600 (+0.033)	0.567 (0.000)	0.594 (+0.027)
2	0.591	0.579 (-0.012)	0.590 (-0.001)	0.614 (+0.023)

Table 5.7: Fitness Result Comparison

Dataset	Baseline	BOE	GED	WED
1	0.985	~1 (+0.015)	0.985 (0.000)	~1 (+0.015)
2	0.998	~1 (+0.002)	0.998 (0.000)	0.993 (-0.005)

6 Discussion

This chapter will discuss the results, method, and work in a wider context as well as areas of possible further study.

6.1 Results

This section will discuss the different findings.

6.1.1 Improved Simplicity

As can be seen in the results chapter, the results indicate that complexity can be lowered. This while maintaining the same or even increased log fitness in most cases. The increase in simplicity is most pronounced for the BOE method followed by the WED method with dataset 1. Dataset 2 shows an increased simplicity with WED but does at the same time have the greatest decrease in fitness of all the performed tests.

One of the interesting parts to analyze is that of the difference in cluster sizes. In the case of the BOE method, Table 5.3 shows that both clusters indicate a noticeable improvement but also that the clusters are very uneven in size. For WED, Table 5.5, on the other hand, there are rather evenly sized clusters. While there is an improvement in both WED clusters, it is marginal in comparison to the improvement of the BOE method for dataset 1. It could be expected that smaller clusters create models with less complexity due to fewer traces and subsequently less probability for diversity [31]. But the more interesting part here is that cluster 0 for BOE shows the second greatest improvement of the four clusters mentioned here while also being the largest cluster. The best is cluster 1 with BOE which is a significantly smaller cluster. As Models of data with more similarity between traces tend to be less complex [39], it can be assumed that the clustering made by the BOE method creates better clusters in that regard than the WED method in this particular case. This is a little surprising since the WED method was expected to categorize the traces more accurately based on similarity and subsequently produce clusters with less complexity [25]. If we compare BOE and WED for dataset 2 We can see just that, WED performing better than BOE by quite a bit. As a matter of fact, for this dataset, the BOE method leads to a decrease in simplicity.

When it comes to GED there is barely any difference from the unclustered data models. Dataset 1 shows no difference for simplicity and dataset 2 has a 0.001 decrease. Fitness is

completely unchanged. This most likely means that there has been no useful segmentation at all. When looking at the cluster sizes the reason for this is very apparent, there are only 3 and 1 trace respectively for the two datasets that are not contained in the same cluster. The data could basically be considered the same as before clustering.

6.1.2 Deviation score and user efficiency

When considering the results of the user efficiency aspects of this thesis, one could divide it into two parts; the usability of deviation score in and of itself and whether or not the score gave any interesting insights into the data.

In Figure 5.1 the unclustered data is plotted with trace duration on the y-axis and deviation score on the x-axis. In the graphs, higher deviation scores equal a relatively longer time to perform the task. It could be seen rather clearly in the graph for dataset 1 that there seems to be a trend in the upper limit of total duration towards higher deviation scores for the traces with higher total duration. This is not as easy to see in the graph for dataset 2.

This difference between the two datasets could be a confirmation of the hypothesis that dataset 2 is less homogenous than dataset 1. This is because of the nature of the metric. If a dataset were to contain only the events a, b and c in different orders and different event durations each trace would have a deviation score related to every other trace. This would result in a graph where there would be a strong correlation between deviation score and total duration. In a case on the other extreme where no traces share any event at all, each trace would only be compared to itself in the deviation score dimension. As such, the deviation score needs at least some overlap in the events between traces to be useful. The fact that dataset 2 might have less overlap may explain that the data is more spread out along the time axis and less so along the deviation score dimension.

Both datasets have in common that there are rather few traces with a deviation score lower than 0 and those that exist have a very short total duration. Traces with negative deviations scores are on average quicker than other similar traces so it is reasonable that these traces have a short total duration.

When analyzing the clustering results in regards to the user efficiency the results are less clear. What can be said for certain is that all but one of the tested methods for clustering created no clear clusters at all. The hypothesis was that data with only a few different types of interactions would lead to the clustering techniques finding different variants of doing the same thing in the software. The hope was then that the resulting clusters would correlate to efficiency.

There are a few reasons why this hypothesis could have failed. The first is that there simply is no correlation between different ways of doing the same thing and efficiency. Another reason could be data quality. As described in section 5.1.1 the data showed signs of missing some events that could be performed by a user in the software. If the data is missing too many events that affect user efficiency and only contains key-events that would occur no matter how you performed the action[8], then this would mean that subtle differences in users behavior would be indistinguishable in the log. If this is the case then it would probably not be possible to perform this kind of analysis without first making changes to the logging.

A third possible problem could be that the selected metric simply does not capture user efficiency in a realistic way or that the relative comparison makes different ways of performing the same task appear the same in the deviation score by mostly comparing instances to other instances done in the same way.

One thing that can be concluded is that the datasets contain more homogenous traces than was first hypothesized. The data was by all means selected to maximize homogeneity but that there would be more than 50% traces with the same order of events was not expected. This could suggest that the users simply work in a quite similar way which suggests that the users do not have as much freedom to work in different ways as previously thought. Or the selected procedures are too simple to show the real variances that this method was intended

to find. It could also be that the data is simply only representing users working in a similar way. Dataset 1 for example represents only users that are specialists in mammography image analysis and work for the same hospital. Chances are that these users are colleagues and may even have been trained in the software by the same instructor. So while the software makes it possible to interact with pictures and perform similar examinations in different ways, the users might not choose to do so. Mammography screening is also a highly repetitive and quick analysis procedure in most cases. This could make this problem even more likely since there are most likely fewer possible ways to do things than for a more complex workflow. Of course, the lack of variety between traces could also be another indication that the logging is lacking details.

6.1.3 Cluster usefulness

The importance of simplifying process models has been researched rather diligently in the business process world where business models have been used for a long time. Much of this research and tooling can be used when creating and analyzing models in a software context. Reijers and Mendling have for example made a study that concludes that dividing process models into smaller subprocess models can improve the understandability, especially when the part that is of value to understand is on a lower abstraction level[26]. There are however studies that indicate that the cognitive load on the person reading the model increases if they know that the smaller model is part of a larger model[38]. This indicates that for a simplified model to be useful it has to be rather self-contained.

While there are measurable improvements in simplicity using some of the clustering methods, it could be questionable how useful the clustered models are considering that the common characteristics for traces in a cluster are largely unknown. If it with certainty could be said that the traces in one cluster comes from the 10% most efficient users and that the traces in the other cluster belong to the rest of the users then it would be highly interesting to put in some effort into comparing the two models and see if any key differences could be identified. But in this case, no clear characteristic difference between the clusters has been identified which makes it hard to state such claims. The best performing method in this report differentiate based on the number of each kind of event. If some effort would be put into finding characteristics there might very well exist a reasonable mapping between the clusters and real-life differences in user interaction patterns. This would however have to be done by someone with sufficient domain knowledge. But it might very well be two or more different types of analysis that have different time requirements.

6.2 Method

The method used in this project can in hindsight be considered to be somewhat naive. It lacks some robustness in terms of what could be concluded in a scenario where the hypothesis was not proven true. In this case, some improvements in simplicity could be found but since the thesis only searched for a connection between the different methods and efficiency it falls short when this connection does not exist.

6.2.1 Pre-study

The pre-study in this thesis was rather broad in order to encompass as much as possible of the existing methods. There was, however limited time to go deep in all the areas that could be of interest and the thesis at large probably has to be seen as rather exploratory. Therefore the results of the pre-study have an important role to play when it comes to this and further studies. It outlines several avenues that were not focused on in this thesis. Despite this broad coverage, there is much more to be explored and while the field of process mining is rather narrow in software development terms it is wide and complex in other process areas.

The pre-study also focuses on trace clustering only and did not explore any other ways of achieving simpler models. This may have led to missing techniques that could have been useful to achieve one or both of the thesis's main objectives. As for the selection of methods from the ones found in the pre-study this was done in a somewhat subjective way. And rather than selecting based on the same criteria, the three methods were selected based on different goals. While this seemed logical at the time it might not have been the best since they have been measured and compared by the same methods and standards. As such, no considerations have been made when testing the methods that they where selected partly based on the difference in complexity to implement as per research question 4.

6.2.2 Complexity analysis

Quality metrics of process models can be done in many ways. Fitness and simplicity were selected in this case. Fitness was selected as an indicator to make sure that the different attempts at simplifying the model would not lead to a worse representation of the logs. Selecting simplicity as a metric to measure a decrease in complexity seems obvious. The selections of metrics seem reasonable as such but it is a little bit more complicated. For the sake of convenience the metrics available in PM4PY was used in this thesis. This may have been limiting and may have led to other potentially better metrics being missed. Another important aspect is that neither of the selected metrics has a singular accepted definition. The implementations used do however have detailed documentation and their implementations are based on relevant and published research papers. As such one can be reasonably certain that their quality is at least decent and provide a good enough indicator of the performance of the tested methods.

6.2.3 User efficiency analysis

The only two user efficiency metrics used are total trace durations and deviation score. The problem with these is that Total duration does not tell very much on its own and deviation score is a new metric as far as is known to the author of this thesis. The intention was to be able to compare the data with data collected from users at the hospital that the data came from. The Covid-19 pandemic resulted in this being unfeasible. The lack of comparison combined with the fact that the traces in the logs lack information about the users' intended outcome and as a result, there exists no clear definition of trace success. Without a definition of success, there is no way to use the definition of efficiency presented in Section 2.6. Instead, the result has to be analyzed from another perspective. The data primarily comes from professionals that work with this every day and have proper training. This likely means that most of the interactions are made by users that know what they are doing. With this assumption, the results of the method can be put in relative terms instead and as such comparison of efficiency can be done rather than efficiency in a single instance.

The chosen way of dividing traces into smaller traces where one trace only contains data from one examination prevents the analysis of how users structure sessions at a higher abstraction level. This thesis has focused on the narrower perspective but future research could be of interest in the domain of higher abstraction levels.

6.2.4 Sources

The sources in this paper have been selected to reflect the current state of the domain as well as to give a well-founded theory on which the current state of the domain relies. One issue that makes this effort hard to succeed with is that the domain of process mining brings together a lot of different domains. Add the fact that this thesis tries to use process mining to analyze efficiency in a usability context. This makes it hard to find, collect and consider sources that reflect all relevant aspects of all the domains. As such there might be relevant sources that have been missed in this thesis.

Despite the cross-discipline nature of process mining, there is one person that has been more active than any other, Wil van der Aalst. He is listed as an author of an impressive amount of articles within process mining, to the degree that it is hard to find relevant papers that he has not been involved in writing. There has been an active effort to include sources in this thesis that are not authored by van der Aalst and the hope is that this will make this thesis less prone to bias. With that being said there is no way of denying that Wil van der Aalst has had an enormous impact on this field and many of the ground truths that all other papers are based on come from papers published by him or his colleagues.

6.3 The work in a wider context

When looking at this study from an ethical point of view there are some interesting aspects to take into consideration. The use of clustering to make simpler models of user data might not be cause for concern by itself but everything pertaining to user data needs to be handled with care. This in combination with developing software intended for diagnostic use within the medical domain makes it even more sensitive. It can be argued that no third party should have access to employee efficiency information. If the intent is to use the information to improve the software though it is hardly useful if the company developing the software does not have access to it.

The question of data collection in itself is, by all means, an interesting topic but in the context of this study, it might be more relevant to discuss the interpretation of the data analysis. The different interpretations and how these findings are used are intended to affect the users' daily life for the better. One potential problem though, could be that an overly emphasized focus on efficiency might lead to a work environment that becomes too focused on performance leading to stressed personnel and poorer care for patients. Another possible drawback is if the data is misinterpreted in a way that makes the most common workflows optimized in a way that causes less used, but is still important features, to become less user-friendly.

While there exist risks for negative impact, improving efficiency by improving user experience is almost always universally positive. It also has the added benefit in the medical field that it makes the diagnostic workflow quicker which in turn may lead to the patient getting the right treatment more quickly. In big systems, data collection and analysis can be valuable tools to achieve this[17].

6.4 Further study

There are a lot of different aspects to study further in this area of research. While this particular study did not find any suitable ways to separate the behavior of effective and less effective users there is potential in some of the findings. As stated in the delimitations, this study did not try to find more than two clusters. Trying to cluster the data into more clusters may show that there are more to be found if the granularity is finer. Since the clustering was based on the characteristics of the traces, using the same number of clusters as the number of different user workflows in the data could be interesting.

Other parameters could also be varied. E.g. it is known that WED is very sensitive to how the weights are applied [25]. Since WED did not perform as well in this study as it has in previous studies it would suggest that there might be a possibility to alter the weights in some way to gain better results. There also exist more metrics that could be explored.

It would be of interest to do more comparative studies on deviation score. For example, comparing to user testing. This could create a better understanding of whether this is a good metric or not. It could also be interesting to search through the literature more thoroughly to see if something similar already has been used before.

One of the primary areas for further study is to continue where this study stops, i.e. try to figure out what the models represent and put them in a context where they can provide concrete value to de software development process.

Lastly, it could be worth mentioning that while this thesis has focused on trace clustering, i.e. clustering the data before it has been transformed into a model. There exists research that tries to divide the models into smaller parts after the model has been generated.

7 Conclusion

The intention of this study was to validate if clustering could be used to simplify models generated with process mining. The intention was also to see if these clusters could suggest anything about user efficiency. To that end a new metric where devised, average mean duration deviation. This metric aimed to show if a trace was more or less efficient than a comparative trace. Since the intent was to find traces with similar characteristics the clustering was done with characteristic features instead of time or efficiency features. The aim was to find a correlation between efficiency after the fact. A correlation between efficiency could not be found. There are a lot of aspects involved in this. The first being how efficiency is defined. In this case, time was the only evaluated part of efficiency although in two different ways. A more robust way would most likely be to take into account whether the trace was a successful trace or not. In this case that was not possible due to the data not containing that level of information.

What distance metrics exist that would be applicable to trace clustering in process mining? The result part of the Pre-study chapter lists the distance metrics found and considered during this thesis. Out of these, the following three were selected for experimental evaluation.

- Bag-of-Events
- Generic Edit Distance
- Weighted Generic Edit Distance

The fact that it probably exists more methods that are not covered in this study is brought up in the discussion. This means that the list presented in this study is most likely not complete and has to be regarded as a subset of available metrics.

What distance metric appears to be best suited to create simpler models according to the literature? Of the methods found in the pre-study, the one that showed the most promise where Weighted Generic Edit Distance. This one was selected based on the fact that it in at least one previous study had performed better than the other methods that were found. Here, performance is defined as the ability to cluster traces based on event composition and

the context of the event in a way that would group similar user behavior together.

How well does this metric perform in terms of simplicity and fitness on usage data from a medical application? Weighted Generic Edit Distance showed an improvement of 0.027 for dataset 1 and 0.023 for dataset 2 in terms of simplicity. These were the second and third best improvements of all results. In terms of fitness, there was an improvement for dataset 1 and a very slight decrease for dataset 2.

Are there any simpler metrics and how do they compare to the above metric? Yes, there are metrics that are simpler to implement. In this thesis Bag-of-Events and Generic Edit Distance were chosen to use as a comparison to Weighted Generic Edit Distance. They represent two different levels of simplification. Bag-of-events is a very simple metric that only counts the number of each kind of event while Generic Edit Distance is very similar to Weighted Generic Edit Distance but without the weights. Clustering based on N-grams was also suggested in the pre-study but was not evaluated because of the many dimensions created. On top of these methods, a few methods based on the time domain were listed but not selected for evaluation since clustering with duration as a feature was deemed to be less likely to create models with usable domain representations.

The Bag-of-Event method showed a slight increase in fitness for both datasets. The most interesting result of this method was the simplicity score on dataset 1, which increased by 0.033. But the same method also lead to a decrease of 0.012 for dataset 2.

Generic Edit Distance showed barely any changes at all in any of the measured metrics.

Overall can be said that the Weighted Generic Edit Distance showed more consistent improvement in simplicity than the simpler methods.

Do any of the groupings created with the compared distance metrics correlate with efficiency? Weighted Generic Edit Distance also provided rather evenly sized clusters for both datasets, contrary to the other methods. For dataset 2 the clusters are even visually separated in the time/deviation score scatter plot. This is the only clustering method that shows any indication of a correlation between efficiency and the clusters. The dividing line is also almost completely along the x-axis, meaning that the cluster is divided by total duration and not deviation score.

Bibliography

[1] Wil van der Aalst. *Process Mining: Data Science in Action*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016. ISBN: 978-3-662-49851-4. DOI: 10.1007/978-3-662-49851-4_1. URL: https://doi.org/10.1007/978-3-662-49851-4_1.

[2] Wil van der Aalst. "Process Mining: Overview and Opportunities". In: *ACM Trans. Manage. Inf. Syst.* 3.2 (July 2012). ISSN: 2158-656X. DOI: 10.1145/2229156.2229157. URL: https://doi-org.e.bibl.liu.se/10.1145/2229156.2229157.

[3] Wil van der Aalst, Arya Adriansyah, Ana Karla Alves de Medeiros, Franco Arcieri, Thomas Baier, Tobias Blickle, Jagadeesh Chandra Bose, Peter van den Brand, Ronald Brandtjen, Joos Buijs, Andrea Burattin, Josep Carmona, Malu Castellanos, Jan Claes, Jonathan Cook, Nicola Costantini, Francisco Curbera, Ernesto Damiani, Massimiliano de Leoni, Pavlos Delias, Boudewijn F. van Dongen, Marlon Dumas, Schahram Dustdar, Dirk Fahland, Diogo R. Ferreira, Walid Gaaloul, Frank van Geffen, Sukriti Goel, Christian Günther, Antonella Guzzo, Paul Harmon, Arthur ter Hofstede, John Hoogland, Jon Espen Ingvaldsen, Koki Kato, Rudolf Kuhn, Akhil Kumar, Marcello La Rosa, Fabrizio Maggi, Donato Malerba, Ronny S. Mans, Alberto Manuel, Martin McCreesh, Paola Mello, Jan Mendling, Marco Montali, Hamid R. Motahari-Nezhad, Michael zur Muehlen, Jorge Munoz-Gama, Luigi Pontieri, Joel Ribeiro, Anne Rozinat, Hugo Seguel Pérez, Ricardo Seguel Pérez, Marcos Sepúlveda, Jim Sinur, Pnina Soffer, Minseok Song, Alessandro Sperduti, Giovanni Stilo, Casper Stoel, Keith Swenson, Maurizio Talamo, Wei Tan, Chris Turner, Jan Vanthienen, George Varvaressos, Eric Verbeek, Marc Verdonk, Roberto Vigo, Jianmin Wang, Barbara Weber, Matthias Weidlich, Ton Weijters, Lijie Wen, Michael Westergaard, and Moe Wynn. "Process Mining Manifesto". In: *Business Process Management Workshops*. Ed. by Florian Daniel, Kamel Barkaoui, and Schahram Dustdar. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 169–194. ISBN: 978-3-642-28108-2.

[4] Wil Aalst, A. Weijters, and Laura Măruşter. "Workflow Mining: Discovering Process Models from Event Logs". In: *Knowledge and Data Engineering, IEEE Transactions on* 16 (Oct. 2004), pp. 1128–1142. DOI: 10.1109/TKDE.2004.47.

[5] Wil van der Aalst. "Service Mining: Using Process Mining to Discover, Check, and Improve Service Behavior". In: *IEEE Transactions on Services Computing* 6.4 (2013), pp. 525–535. DOI: 10.1109/TSC.2012.25.

[6] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. "On the Surprising Behavior of Distance Metrics in High Dimensional Space". In: *Database Theory — ICDT 2001*. Ed. by Jan Van den Bussche and Victor Vianu. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 420–434. ISBN: 978-3-540-44503-6.

[7] Fraunhofer Institute for Applied Information Technology. *PM4PY*. Version 2.1.1. 2020. URL: https://pm4py.fit.fraunhofer.de/.

[8] Richard Atterer, Monika Wnuk, and Albrecht Schmidt. "Knowing the User's Every Move: User Activity Tracking for Website Usability Evaluation and Implicit Interaction". In: *Proceedings of the 15th International Conference on World Wide Web*. WWW '06. Edinburgh, Scotland: Association for Computing Machinery, 2006, pp. 203–212. ISBN: 1595933239. DOI: 10.1145/1135777.1135811. URL: https://doi.org/10.1145/1135777.1135811.

[9] Alessandro Berti and Wil van der Aalst. *Reviving Token-based Replay: Increasing Speed While Improving Diagnostics*. Tech. rep. Process and Data Science group, Lehrstuhl für Informatik, Aachen University, Germany, 2019.

[10] Christian Bird, Brendan Murphy, Nachiappan Nagappan, and Thomas Zimmermann. "Empirical Software Engineering at Microsoft Research". In: *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work*. CSCW '11. Hangzhou, China: Association for Computing Machinery, 2011, pp. 143–150. ISBN: 9781450305563. DOI: 10.1145/1958824.1958846. URL: https://doi-org.e.bibl.liu.se/10.1145/1958824.1958846.

[11] Fabian Rojas Blum. *Metrics in process discovery*. Tech. rep. Computer Science Department, University of Chile, June 2015.

[12] Alejandro Bogarín, Cristóbal Romero, Rebeca Cerezo, and Miguel Sánchez-Santillán. "Clustering for Improving Educational Process Mining". In: *Proceedings of the Fourth International Conference on Learning Analytics And Knowledge*. LAK '14. Indianapolis, Indiana, USA: Association for Computing Machinery, 2014, pp. 11–15. ISBN: 9781450326643. DOI: 10.1145/2567574.2567604. URL: https://doi-org.e.bibl.liu.se/10.1145/2567574.2567604.

[13] João Caldeira and Fernando Brito e Abreu. "Software Development Process Mining: Discovery, Conformance Checking and Enhancement". In: *2016 10th International Conference on the Quality of Information and Communications Technology (QUATIC)*. 2016, pp. 254–259. DOI: 10.1109/QUATIC.2016.061.

[14] R H Choplin, J M Boehme, and C D Maynard. "Picture archiving and communication systems: an overview." In: *RadioGraphics* 12.1 (1992). PMID: 1734458, pp. 127–129. DOI: 10.1148/radiographics.12.1.1734458. URL: https://doi.org/10.1148/radiographics.12.1.1734458.

[15] Jonathan E. Cook and Alexander L. Wolf. "Discovering Models of Software Processes from Event-Based Data". In: *ACM Trans. Softw. Eng. Methodol.* 7.3 (July 1998), pp. 215–249. ISSN: 1049-331X. DOI: 10.1145/287000.287001. URL: https://doi.org/10.1145/287000.287001.

[16] Daniel Forsberg, Beverly Rosipko, and Jeffrey L Sunshine. "Analyzing PACS usage patterns by means of process mining: steps toward a more detailed workflow analysis in radiology". In: *Journal of digital imaging* 29.1 (2016), pp. 47–58.

[17] Helena Holmström Olsson and Jan Bosch. "Towards Data-Driven Product Development: A Multiple Case Study on Post-deployment Data Usage in Software-Intensive Embedded Systems". In: *Lean Enterprise Software and Systems*. Ed. by Brian Fitzgerald, Kieran Conboy, Ken Power, Ricardo Valerdi, Lorraine Morgan, and Klaas-Jan Stol. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 152–164. ISBN: 978-3-642-44930-7.

[18] A. K. Jain, M. N. Murty, and P. J. Flynn. "Data Clustering: A Review". In: *ACM Comput. Surv.* 31.3 (Sept. 1999), pp. 264–323. ISSN: 0360-0300. DOI: 10.1145/331499.331504. URL: https://doi-org.e.bibl.liu.se/10.1145/331499.331504.

[19] Sander J. J. Leemans, Dirk Fahland, and Wil M. P. van der Aalst. "Scalable process discovery and conformance checking". In: *Software and Systems Modeling* 17 (2016), pp. 599–631.

[20] Sander J. J. Leemans, Dirk Fahland, and Wil M. P. van der Aalst. "Scalable Process Discovery with Guarantees". In: *Enterprise, Business-Process and Information Systems Modeling*. Ed. by Khaled Gaaloul, Rainer Schmidt, Selmin Nurcan, Sérgio Guerreiro, and Qin Ma. Cham: Springer International Publishing, 2015, pp. 85–101. ISBN: 978-3-319-19237-6.

[21] Jakob Nielsen. *Usability Engineering*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994. ISBN: 9780080520292.

[22] orsinium. *textdistance*. Version 4.2.0. 2020. URL: https://pypi.org/project/textdistance/.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[24] James L. Peterson. "Petri Nets". In: *ACM Comput. Surv.* 9.3 (Sept. 1977), pp. 223–252. ISSN: 0360-0300. DOI: 10.1145/356698.356702. URL: https://doi.org/10.1145/356698.356702.

[25] Jagadeesh Chandra Bose R.P. and Wil Aalst. "Context Aware Trace Clustering: Towards Improving Process Mining Results". In: Apr. 2009. DOI: 10.1137/1.9781611972795.35.

[26] Hajo Reijers and Jan Mendling. "Modularity in Process Models: Review and Effects". In: Sept. 2008, pp. 20–35. ISBN: 978-3-540-85757-0. DOI: 10.1007/978-3-540-85758-7_5.

[27] E.S. Ristad and P.N. Yianilos. "Learning string-edit distance". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20.5 (1998), pp. 522–532. DOI: 10.1109/34.682181.

[28] Anne Rozinat, A Medeiros, C Günther, A. Weijters, and Wil Aalst. "Towards an evaluation framework for process mining algorithms". In: *Reactivity of Solids* (Jan. 2007).

[29] Vladimir Rubin, Irina Lomazova, and Wil Aalst. "Agile Development with Software Process Mining". In: May 2014. DOI: 10.1145/2600821.2600842.

[30] Vladimir A. Rubin, Alexey A. Mitsyuk, Irina A. Lomazova, and Wil M. P. van der Aalst. "Process Mining Can Be Applied to Software Too!" In: *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. ESEM '14. Torino, Italy: Association for Computing Machinery, 2014. ISBN: 9781450327749. DOI: 10.1145/2652524.2652583. URL: https://doi-org.e.bibl.liu.se/10.1145/2652524.2652583.

[31] Minseok Song, Christian Günther, and Wil Aalst. "Trace Clustering in Process Mining". In: vol. 17. Sept. 2008, pp. 109–120. DOI: 10.1007/978-3-642-00328-8_11.

[32] The pandas development team. *pandas-dev/pandas: Pandas*. Version 1.2.0. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[33] the pandas development team. *Pandas documentation pandas.DataFrame.sample*. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.sample.html. Accessed: 2022-01-20.

[34] the scikit-learn development team. *scikit-learn documentation sklearn.cluster: Clustering*. https://scikit-learn.org/stable/modules/classes.html#module-sklearn.cluster. Accessed: 2022-01-20.

[35] Gabriel M. Veiga and Diogo R. Ferreira. "Understanding Spaghetti Models with Sequence Clustering for ProM". In: *Business Process Management Workshops*. Ed. by Stefanie Rinderle-Ma, Shazia Sadiq, and Frank Leymann. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 92–103. ISBN: 978-3-642-12186-9.

[36] Jochen Weerdt, Seppe vanden Broucke, Jan Vanthienen, and Bart Baesens. "Active Trace Clustering for Improved Process Discovery". In: *IEEE Trans. Knowl. Data Eng.* 25 (Dec. 2013), pp. 2708–2720. DOI: 10.1109/TKDE.2013.64.

[37] A. Weijters and Wil Aalst. "Workflow Mining: Discovering Workflow Models from Event-Based Data". In: Jan. 2002, pp. 78–84.

[38] Michael Winter, Rudiger Pryss, Thomas Probst, Julia Bass, and Manfred Reichert. "Measuring the Cognitive Complexity in the Comprehension of Modular Process Models". In: *IEEE Transactions on Cognitive and Developmental Systems* PP (Oct. 2020), pp. 1–1. DOI: 10.1109/TCDS.2020.3032730.

[39] Fareed Zandkarimi, Jana-Rebecca Rehse, Pouya Soudmand, and Hartmut Hoehle. *A Generic Framework for Trace Clustering in Process Mining*. International Conference of Process Mining. ICPM 2020. Oct. 2020.