UNIVERSITY OF TARTU
Institute of Computer Science
Software Engineering Curriculum

Erdem Toraman

Visualizing Business Process Deviance With Timeline Diagrams

Master's Thesis (30 ECTS)

Supervisor: Marlon Dumas, PhD

Tartu 2018

Visualizing Business Process Deviance With Timeline Diagrams

Abstract:

The thesis poses two main questions regarding to the notion of "*deviance mining*" and proposes a new technique to visualise the differences of two event logs in terms of temporal dynamics in order to perform deviance mining. The objective of deviance mining is to pinpoint the origin of the problems and the deviance. Throughout the research of the related work it's observed that most of the existing methods focus on the main structure of the process which is the order of the tasks being executed. The new technique brings out the relative durations i.e temporal dynamics of the activities in the *normal* and *deviant* traces by drawing a timeline diagram of the variants. Additionally the proposed technique puts forward set of different settings such as the performance measure and the granularity level of the process to customize the diagram. Lastly, a proof-of-concept tool abiding by the proposed approach is implemented which is served on the web.

Keywords:
Business process, Business process management, Process mining, Deviance mining, Process visualisation

CERCS Code: P175 - Informatics, systems theory.

Äriprotsessi hälvete visualiseerimine ajaskaala skeemidega

Lühikokkuvõte:

Lõputöös püstitatakse kaks peamist küsimust mõiste "hälvete kaevandamine" kohta ja tehakse ettepanek uue meetodi jaoks, mis näitavad kahe sündmuse logide erinevusi ajalise dünaamika mõttes, et täita hälvete kaevandamist. Hälvete kaevandamise eesmärk on täpsustada probleemide päritolu ja kõrvalekaldeid. Kogu sellega seotud töö uurimisel on täheldatud, et enamus olemasolevatest meetoditest keskenduvad protsessi põhistruktuurile, mis on ülesannete täitmise järjekord. Uus tehnika näitab tavapäraste ja hälbivate jälgede tegevuste suhtelisi kestusi, st ajalist dünaamikat, joonistades variantide ajakava. Lisaks pakutakse välja tehnikat, mis näitab diagrammide kohandamiseks erinevaid seadeid, nagu tulemuslikkuse mõõdik ja protsessi üksikasjalikkus. Lõpuks on välja arendatud kontseptsiooni vahed, mis järgib välja pakutud lähenemisviisi ja on veebis saadaval.

Võtmesõnad: Business process, Business process management, Process mining, Deviance mining, Process visualisation

CERCS: P175

Table of Contents

1. Introduction

The term business process is defined as a collection of inter-linked tasks performed according to predefined rules in order to deliver a service or a product to a customer [9]. A typical example of a business process is an order-to-cash process in a manufacturing company. Such a process involves a number of tasks performed by several resources, such as examining the purchase order, checking the stock availability, ordering missing raw materials, manufacturing the product, shipping the product, invoicing and payment collection.

Business Process Management (BPM) is a set of principles, methods, and tools for managing business processes in order to maintain and improve their performance. In order to enable the effective management of business processes [1], it is common to start by modeling the process in detail, so as to be able to trace down issues in the process to individual tasks, events, or decisions. Business processes are commonly modeled using the Business Process Model and Notation (BPMN) [2].

The BPM discipline incorporates numerous techniques for analyzing, re-designing, implementing and executing business processes using process models. This thesis specifically focuses on a subset of the BPM discipline called *process mining*. The goal of process mining is to discover patterns and deduct knowledge out of the *event logs* that are recorded during the execution of a business process using a process-aware information system such as a Business Process Management system. An event log is composed of set of traces which is a set of events that occur during the execution of a single instance within the process.

Process mining techniques include four main use cases that are a*utomated process discovery, conformance checking, performance mining* and *variant analysis*. Briefly, automated process discovery aims to produce the business process model using the event logs. Conformance checking methods require both event logs and expected business process model and they output the level of conformance by listing the exceptional non-conforming cases and differences between the model and the actual logs. Performance mining techniques, using the logs and business process model generate another process model that visualises the performance related aspects. Variant analysis, which can be also named as *deviance mining* is about listing the differences between two logs. The overall schema of process mining can be found in Figure 1.

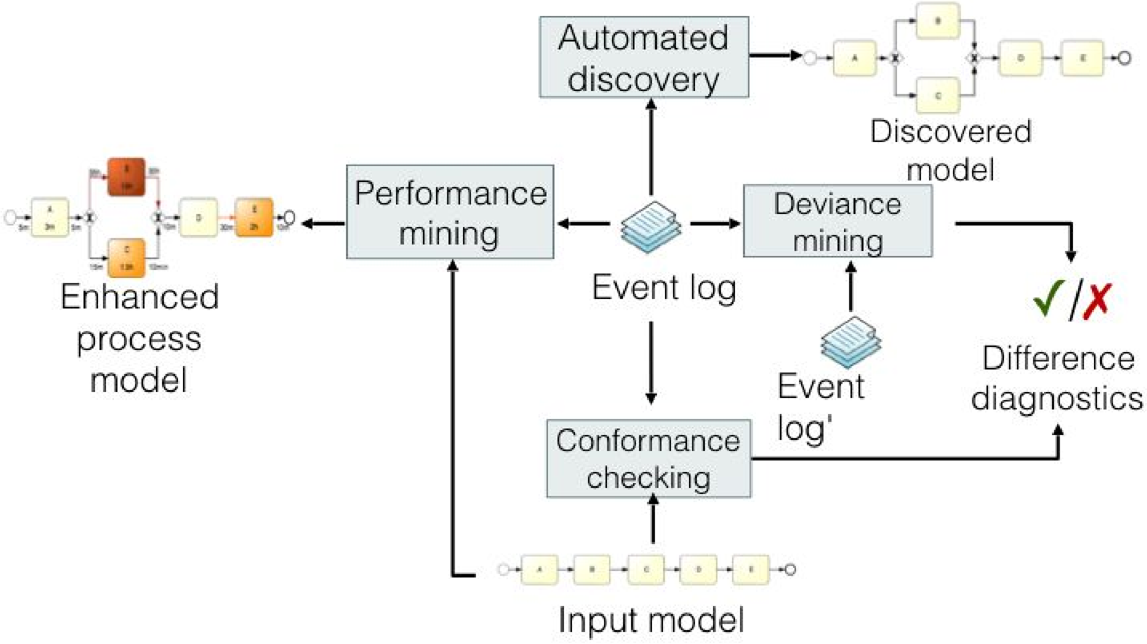


Figure 1: Process mining Steps [1]

Deviance mining refers to mining of possible reasons for deviant cases in business processes utilizing the event log [3]. Perception of deviance might vary with respect to the definition of the performance measures such as time and quality. These performance criteria are defined by the process owner. To illustrate, an insurance company might be suffering from the fact that resolving a case takes too much time which ultimately yields customer dissatisfaction. In this case, they might want to investigate the possible causes of longer than the anticipated processing times. On the other hand, in a lead-to-order process, some leads might not end up ordering the product which is considered as an unsuccessful case.

A deviance mining tool is expected to receive two different event logs (the log of normal cases and the log of “deviant” cases) of the same process. The output of deviance mining is a set of differences between these two event logs [3]. This output can be interpreted by an analyst in order to determine to what extent the deviant cases can be explained by these differences. One key issue in deviance mining is that it should not simply tell the differences of two event logs but only the ones that are statistically significant in order for the analyst to extract useful insights concerning the origin of the deviance. A concrete example can be to detect that in one variant a certain activities average duration is much more than the other one or one activity is repeated many more times in one variant. This kind of information could assist us in diagnosing problems such as bottlenecks and lack of resource as well as in eliminating exceptional cases stemming from external factors. It is beneficial in terms of improving the current state of the process thereby reducing the customer dissatisfaction and possibly boosting the profitability. The size of the data is a paramount factor to be able to conceptually generalize the patterns of the process. However, as the data gets bigger, it is infeasible to handle these tasks manually without the support of automation.

This thesis aims to present the techniques that are used in deviance mining. Accordingly, the first research questions of this thesis is, *how can the deviance mining techniques be classified and what are the characteristics of each category of deviance mining techniques?* To this end, we examine the related papers and find out the advantages and disadvantages of the current methods as well as classifying them.

After an analysis of existing deviance mining techniques, we found that one aspect that the existing techniques neglect is the temporal dimension. In other words, existing techniques for deviance mining do not allow one to readily compare the detailed temporal dynamics of an event log (e.g. normal cases) against the temporal dynamics of the other event log (deviant cases). This raises a second research question, namely: *How to visualize the temporal dynamics of an event log concisely, and how to compare the temporal dynamics of two event logs?* To answer this question, we reviewed existing techniques for visualization of event logs, we analyze their scope and limitations, and we propose a new technique for event log visualization and comparison. The main contribution of the thesis is a new technique in order to visualise, summarise and compare process variants with regard to temporal dynamics as well as the associated tool implementation. Temporal dynamics in this context refers to the duration of the tasks or stages being executed or being stalled.

The rest of the document is structured as follows. The next chapter covers the background information regarding the event logs and the available data. The third chapter gives a comprehensive review of the aforementioned approaches to the problem as well as answers the research questions which will be followed by a conclusion. The fourth chapter describes the proposed approach and technique with reasoning. The fifth chapter provides the implementation details of the tool in addition to a use case scenario explaining how tool works and interpreting the diagrams. Lastly they are followed by a conclusion and references sections.

2. Background: Event Logs

Business processes can be very complex and convoluted due to process' nature even for the one who designs them. Considering that, extracting knowledge out of event logs is naturally difficult as well. In order to comprehend the core of the problem, a clarification on the structure of the available data is essential. The data is called "event logs". An event log is a collection of events with timestamps. Extensible event stream (XES) [4] is considered as the universally valid syntax and grammar for event logs in order to capture the behavior of systems. In this thesis the terms log, case and event are used frequently and descriptive definitions of those are required to clarify the following analysis.

An event represents a lifecycle transition of a task in the process. For instance, an event might represent the following sentence "The task A for the case C1 is completed by the resource R1 at 01.01.2017 14:21:52." The lifecycle diagram of a task is shown in Figure 2. This figure shows that a task can be scheduled, started, suspended, resumed and completed. An event in the log, with an identifier of the task, indicates one of these actions.

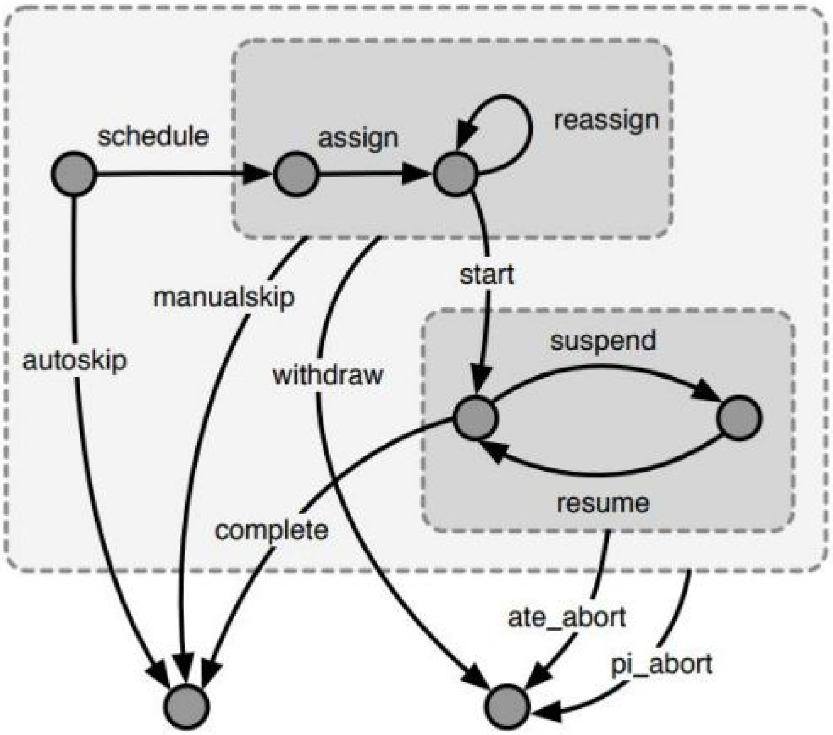
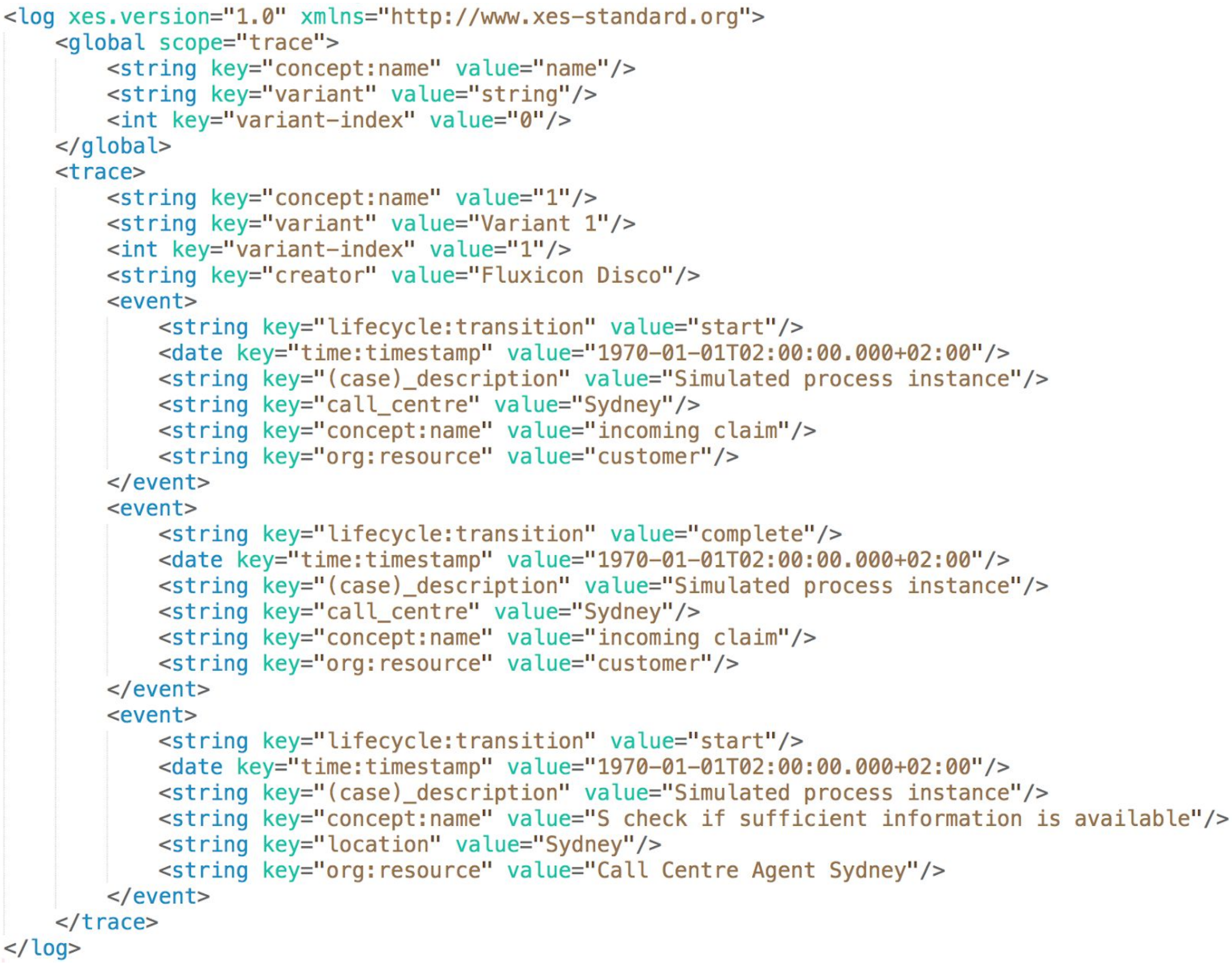


Figure 2: Lifecycle of a Task [4]

A trace (case) is a sequence of events happening regarding to a single customer or point of concern. For example, all the events occurring that concern a single prescription in a pharmacy belong to the same case, therefore, each event of the same case will have the same case identifier in the event log. An event log is a set of cases of the same process. It is assumed the union of all the tasks of the events in all cases is the universe of tasks in that process.

An event, in addition to fundamental elements of time, task, and lifecycle, might have a payload. This payload can be used to embed extra information into an event. For instance, an event could be carried out by a resource like an employee as well as it could have associated costs. This helps us to widen our area of investigation which might yield more accurate information. To exemplify, if a resource is underutilized in the normal cases and over-utilized in the time-wise problematic cases then this indicates a lack of resource problem. Figure 3 is an example of a log with one trace which is composed of 3 events.



```
<log xes.version="1.0" xmlns="http://www.xes-standard.org">
    <global scope="trace">
        <string key="concept:name" value="name"/>
        <string key="variant" value="string"/>
        <int key="variant-index" value="0"/>
    </global>
    <trace>
        <string key="concept:name" value="1"/>
        <string key="variant" value="Variant 1"/>
        <int key="variant-index" value="1"/>
        <string key="creator" value="Fluxicon Disco"/>
        <event>
            <string key="lifecycle:transition" value="start"/>
            <date key="time:timestamp" value="1970-01-01T02:00:00.000+02:00"/>
            <string key="(case)_description" value="Simulated process instance"/>
            <string key="call_centre" value="Sydney"/>
            <string key="concept:name" value="incoming claim"/>
            <string key="org:resource" value="customer"/>
        </event>
        <event>
            <string key="lifecycle:transition" value="complete"/>
            <date key="time:timestamp" value="1970-01-01T02:00:00.000+02:00"/>
            <string key="(case)_description" value="Simulated process instance"/>
            <string key="call_centre" value="Sydney"/>
            <string key="concept:name" value="incoming claim"/>
            <string key="org:resource" value="customer"/>
        </event>
        <event>
            <string key="lifecycle:transition" value="start"/>
            <date key="time:timestamp" value="1970-01-01T02:00:00.000+02:00"/>
            <string key="(case)_description" value="Simulated process instance"/>
            <string key="concept:name" value="S check if sufficient information is available"/>
            <string key="location" value="Sydney"/>
            <string key="org:resource" value="Call Centre Agent Sydney"/>
        </event>
    </trace>
</log>
```

Figure 3: Event logs example

3. Related Work

This section is structured as presenting the overall categorization of the techniques and explaining them under the corresponding the titles by referring the related work. As the different methods of the same techniques approximately have the same advantages and disadvantages, they are mentioned under the subsections. In addition to the academic papers, in the last section of this chapter commercial tools are presented.

3.1. Literature Search Method

Process mining is a very a popular subject as the searched words in Google Scholar hit more than 400 thousands results. On the other hand, deviance mining is a relatively smaller subset of process mining hitting around 9 thousand results. As the objective is to evaluate and present the state of the art, the publication date is a very important factor, hence the publications older than 2013 were filtered out.

In addition to the date, the scope of this thesis enforces us to focus on the papers which introduce a method. In order to successfully analyze the scope of this paper, systematic review principles were applied. The widely-known search engine, Google Scholar was used for this process. Some of the keywords such as “business process”, “deviance mining” and “discriminative patterns” were combined. Some of the papers were clearly out of the scope. For the remaining papers, reading the abstracts and skimming were sufficient to eliminate some of the papers. Moreover, snowballing and reverse snowballing techniques were utilized meaning that in addition to checking out the cited publications from an identified and accepted paper, the articles which cite that paper were also examined.

The most effective way of finding the new and related articles was snowballing technique as the terms of business process and deviance is very broad.

The selected work is about having 2 different sets of event logs belonging to the same process and summarising the differences between them in order to discover the core of the problem which is referred as the “deviance”.

This section summarises the selected papers by explaining the methods used. There are two different mainstream approaches to the problem of mining the logs. One of those considers an event log as only as a sequence of events whereas the other one takes the other parameters into consideration as well such as the timestamp. The former one is referred as control-flow based and the latter one is called payload analysis. Figure 4 shows the classification of the different techniques.

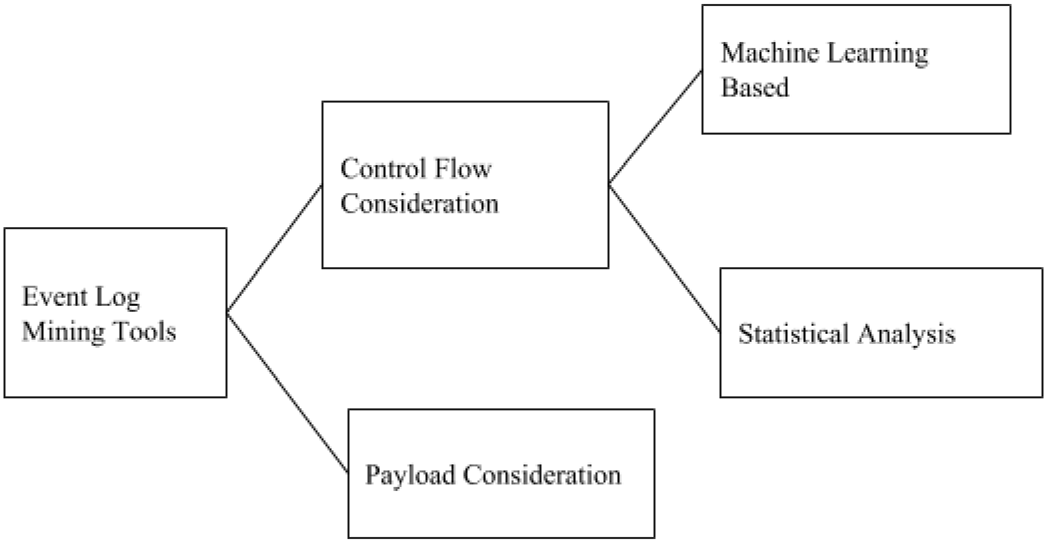


Figure 4: Taxonomy of Deviance Mining Techniques

3. 2. Search Results and Analysis

This section presents the results of the research and their analysis. The structure of this analysis follows the taxonomy of the techniques shown in Figure 4.

3.2.1. Control-Flow Consideration

Control flow can be interpreted as a directed graph where each node represents a task. It can contain loops and parallelism because some of the tasks can be carried out concurrently whereas others can be interdependent. Control-flow is the fundamental element of a process hence these methods can be considered as a baseline approach. The biggest advantage of them is the fact that they're fault-tolerant. They don't consider any other parameter but the order and the number of the events happening and this prevents them to be misled due to errors of the payload data in the logs. Besides, they are more universal in terms of application. It's because they require less information and the format of the data is not uniform everywhere.

On the other hand, not using the available data disables us to do deeper analysis. This might be important to detect the significant differences of the techniques. For instance, if in some cases tasks of some type considerably take longer time then this might indicate a possible problem related to handling of that task. Control-flow based methods fail to identify such differences.

The methods that lie under this category can also be divided into two different sub-categories. One is to develop a classifier function which receives a case and determine if it's a deviant case [3]. On the other hand, there are different techniques to summarize descriptive and discriminative patterns. The latter one focuses on in-depth comparison of two subsets whereas the former one's struggle is to extract correct features from event logs and represent them in numerical vectors so that mainstream machine learning algorithms can be applied such as knn, decision trees or naive [3].

3.2.1.1. Machine Learning Based Techniques

These techniques represent the event logs as mathematical vectors by using the events in every case. The specific compression methods vary with respect to the considerations of representing the information in an effective and accurate manner. However, they all only consider the sequence of the events.

This branch of the deviance mining is evaluated in the paper published by Nguyen et al. 2016 [3] in a broader manner which gives an overall taxonomy. In their work, the common techniques under this category are classified into 3 groups namely "*individual activity*", "*set-based*" and "*sequence-based*". The first one can be regarded as the baseline approach. It only considers the number of occurrences of each event in a "case" and a "case" is represented as a vector of each event and their number of occurrence. The second and the third ones enhance this by applying extra feature selection methods. The second one, for instance, takes only the frequent sets of activities into account and the order of those are neglected. The third one aims to detect the patterns within the process as well such as loops or sub-processes. Hence, they select the common sequences across different cases.

Their empirical evaluation demonstrates that sequence-based techniques which heavily focus on patterns, outperform the individual activity approach in the case of the business process being strictly structured. On the other hand, if the business' itself entails a high level of variability, the increase of success rate is negligible.

Moreover, in the paper by Nguyen et al. 2014 [5] fitness of the common sequence classification techniques for the problem of analyzing deviance is evaluated. The struggle in this area is to represent the events and cases in a more descriptive way so that classification of a case would be more accurate. They mentioned the common methods to represent cases in features, but what they compared is different combination of those methods. In addition to the representation techniques, different machine learning based classification algorithms are also tested. The success criteria is defined as accuracy and AUC values. The result has shown that the combination of "activity frequency" and "discriminative pattern" as feature extraction method and knn as the classifier algorithm yields the best AUC score whereas the combination of "activity frequency" and "tandem repeat" with knn classification is the best in terms of accuracy though the accuracy scores for both of those are very close.

3.2.1.2. Techniques Based on Statistical Analysis

The variety in this section is wider than the previous one. The main goal in this approach is not to represent the events as vectors but to discover statistically significant differences between two event logs. In order to judge the significance a variety of data mining techniques are utilized. The presented techniques' fundamental approaches are similar yet the outcome of those in terms of presentation differ substantially.

In the work presented by Bolt et al. 2016 [6] for instance, a visual summarization technique is introduced by exploiting the process metrics on the control flow which means that the frequency of events occurring in cases as well as their transitions. Their system uses transition system to capture the behavior and detect the distinctions. In order to avoid the increased complexity and noisy data they applied a filter to eliminate the rare cases. The system draws the map of the process by showing the more frequent tasks with darker colors. Moreover, the thickness and the colors of the arrows between tasks are adjusted with respect to the frequencies of those transitions.

Another attempt of summarizing event logs is carried out in Dijkman and Wilbik 2017 [7]. They worked on a tool which compiles the interesting patterns in the event log in a natural language. This work does not claim to be a tool for summarization of differences of two event logs but only to explain the visible and interesting patterns of a single event log. However, one can claim that their system can be used for this purpose as well by feeding two event logs separately and manually checking the differences that are presented in natural language. An example of a sentence produced by the tool could be "for most cases that contained the sequence ABC, the throughput time was long" [7].

They introduced two protoforms of sentences namely simple and extended protoforms. Simple protoforms consist of a quantifier and a summarizer. A quantifier is a word that describes the amount of these cases such as many, most or almost all. A summarizer, on the other hand is a statistically interesting pattern; e.g. low cost, high frequency. In addition to what simple protoforms contain, extended protoforms include a qualifier which is a linguistic value for an attribute. It is used to narrow down the cases with a specific condition being met. These protoforms represent a basic structure of a linguistic summary of a pattern albeit being vague. In order to resolve the issue of sentences being ambiguous, they introduced a formula called truth value. This formula considers the strength of the quantifier word and the ratio of the summarizer word's case being correct. For instance, if the quantifier is "many" and the ratio is 0.6 then the truth function returns a value of 1.0 which means that it is completely true. Conversely, if the quantifier is "most" with the same ratio then the truth value is 0.33 indicating that it does not reflect the truth. Figure 5 demonstrates the function briefly.

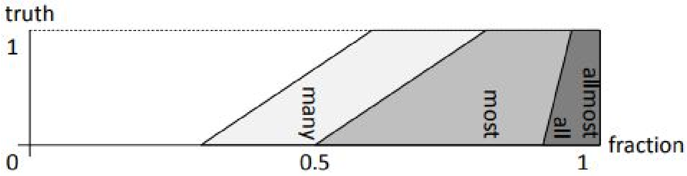


Figure 5: Quantifier Truth Value [7]

Van Beest et al. 2015 [8] developed a system which compares two different event logs from the perspective of event structure. They present a novel lossless encoder for event logs as event structures in combination with a frequency-enhanced method which is called Frequency-enhanced Prime Event Structure (FPES). In order to evaluate their system, synthetic event logs are generated regarding a specific business process that contains different control-flow patterns such as 'xor' and 'and' gates. On the contrary to other methods, theirs only considers the order of the events in the cases. The developed system does not take the time and duration aspects into consideration. Therefore, the results of the system are only regarding the occurrence of the events such as "In variant 2, "Assess eligibility" occurs after "Assess loan risk" and "Appraise property", while in variant 1 it does not occur". Though it gives useful information with respect to differences in terms of the events' orders, this method fails to consider timestamps and event payloads as it treats the event log as only sequences of events.

3.2.2. Payload Consideration

These methods are mostly an improvement over the control-flow based ones. In addition to control-flow related pattern analysis they also consider the timestamps of the events which allows us to have deeper insights regarding the internal dynamics of the process.

One of the disadvantages of these techniques is to difficulty of implementation and computational burden. If the event log is very big and the process is complex then there might be too many different possible points to examine. However, this might be reduced by limiting the considered parameters only to time aspects. In the case study published by Syamsiyah et al. 2017 [9] a more generic methodology is applied. They set forth a step based technique in order to extend their work to broader terms. Figure 6 illustrates the steps.

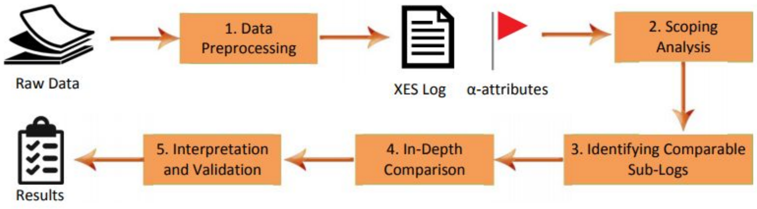


Figure 6: Process Comparison Methodology (PCM)

For the sake of simplicity and consistency first step can be skipped as the other methods assume to have a formatted event log. The second step is carried out in order to eliminate the non-relevant parts of the log which enables us to extract more accurate and pertinent results. The paper also focuses on the real-life problem of 2 different event logs having disjoint events and high variance. If the variants do not have sufficient commonalities then the comparison will be obsolete, thus identification of comparable logs is essential. The third step precisely targets this problem and the solution they suggested is to manually select the sub-logs in case if the domain knowledge is available. Otherwise, clustering methods can be used depending on the similarity attributes. This step is followed by the main function called in depth comparison which analyzes the differences between two sets and finds out the relevant and impactful ones. This step produces a list of interesting differences in terms of control-flow and performance parameters such as cycle time and processing time. Lastly, the fifth stage is about interpreting the raw facts generated in an automated manner and combining them in such a way that they would be more relevant in business level.

The implementation of the case study is conducted on Xerox Services real event logs. In order not to deviate from our 1 scope and only in-depth comparison part is relevant to us, the remaining phases are skipped. In order to pinpoint control-flow related discrepancies, they applied the technique mentioned in [Bolt et al. 2016] which is an addition to statistical and payload data related comparison. However, the analysis with respect to payload is not very deep as it considers the effects of them in terms of only the duration of the events.

Bolt et al. 2017 [10] also in their work try to tackle the same problem from the same angle but another implementation. Though their objective is to discover the interesting variants instead of analyzing the possible reasons of the deviance, this method can be used for the same purpose by manually checking the distinctions between the deviant and the normal event logs. The tool that they made is publicly available which enables the user to split the event log of the process by any combination of the event variables. This possibility gives a chance of semi-automated analysis of the process by simply filtering and splitting the logs via desired variables and checking the process variants. However, the system also lacks a comprehensive statistical analysis in terms of performance, quality and other factors.

3.3. Summary of Literature Review

The results have shown that many methods revolve around two main aspects and the techniques only considering the control flow is a baseline in terms depth of the analysis. Most of the methods heavily focus on the control-flow analysis. In some cases this might actually provide the essential distinctions between the two sets but if the causality is an important part of the investigation then in order to enlighten the reasons a deeper analysis is required. Some of the papers examined in this paper used a few parameters such as cycle time which is the term for the duration of one case in addition to control-flow.

The objective of this research was not to compare the common methods in terms of the results but rather the methodologies and it became evident that a convincing comparison among different techniques is very hard to achieve. Most of them depend on a specific use-case scenario and since the output of their systems is a summarisation of the event logs, there is not a concrete parameter or performance metrics for to be used in a comparison. Moreover, even if a metric was present then the studies are not carried out on the same kind of processes let alone the same data.

Merely the results' themselves are not sufficient in order to fully comprehend the real causes of the deviance and it requires further manual analysis. Hence, considering the real-life scenarios in which those methods are utilized, the presented output must be human readable. That's why most of the examined techniques focus on the presentation layer. That's another aspect of exhibiting the difficulty of comparing the methods as the measurement of goodness in terms of the presentation level is subjective.

All the works that are examined in this paper have their own limitations. One of the biggest common problems is the variability of the processes and the possible errors of the logging that could lead information loss or inaccurate results. Besides, the outlier cases must be carefully handled as they could indicate a possible leak or mishandling in the system or they can simply be extremely rare case which needs to be discarded. To overcome this problem Bolt et al. 2016 [6] applied a filter to omit the rare cases as they could influence the output in a misleading way.

Deviance mining in business processes is a developing area which aims to solve the existent problems regarding the processes' handling mechanism. There are different mainstream approaches to the problem of identifying the main differences between two event logs. This chapter presented an overview of those different methods by inspecting papers from those areas.

It became apparent that the domain specific dynamics have a very big impact on not only the work's itself but also the output. While control-flow based applications fail to identify the causality of the deviance, the methods considering the payload data are better at judging the impacts of those flow-based differences in terms of performance. Another disadvantage of the flow based applications is to reveal the obvious. For instance, in an order-to-cash process if the matter of deviance whose causes are to be investigated is about processes ending up with cancellation of the order, the application might point out that the unsuccessful cases contain the task "cancel the order" more. This kind of output brings no useful information, because instead of giving us insights regarding the cause of the "cancellation", it simply states that failure cases do fail.

The deeper statistical analysis is expected to result in better and more acute information. Those might contain but are not limited to duration of the cases, tasks occurrence number, waiting and processing times. These are all intra-case meaning that the statistical analysis is about the tasks within the cases. On the other hand, we might actually consider the inter-case parameters such as throughput, arrival rate, and resource utilization. None of the examined papers worked on such detailed analysis in terms of the payload of the case and processes.

Moreover, the existing techniques are not designed to take the temporal dynamics of the activities into consideration. Though the control flow and the payload of variables help us to pinpoint the differences, it is often the case that the most indicative parameters of performance related differences are the temporal aspects i.e the relative processing and cycle times of the activities.

3.4. Event Log Visualisation Techniques

In order to address the second research question posed in Section 1, we analyzed the existing visualisation techniques for the event logs. This section introduces two techniques while specifically discussing the beneficialness of those visualisation techniques in the context of deviance mining and variant analysis.

3.4.1. Process Maps

Process maps are widely used to visualise the event logs [1]. They are generated by the directly-follows relation. While process map visualisations give a good understanding of the process model, they fail to consider the durational aspects. Another big problem the process maps encounter is the complexity of the graphs and the so-called *"spaghetti"* graphs. The figures 7 and 8 exemplify the concept.

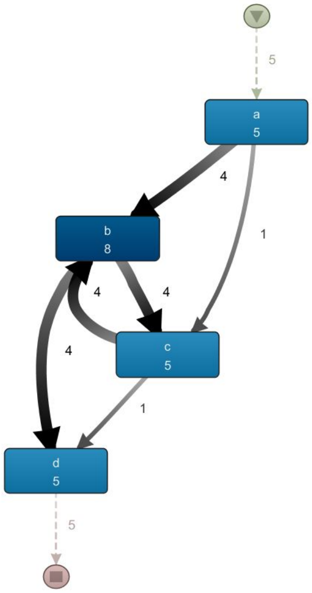


Figure 7: An example of process maps[11]

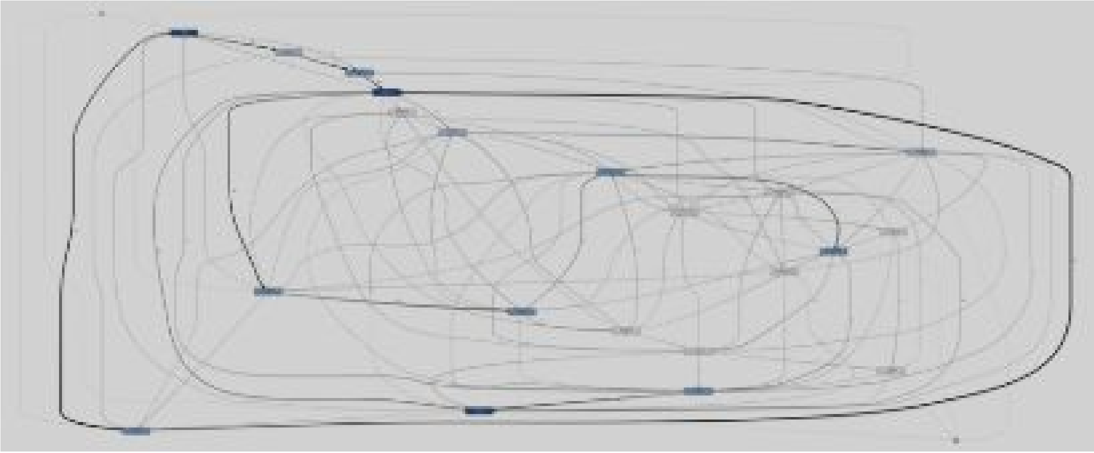


Figure 8: An example spaghetti-like process maps [12]

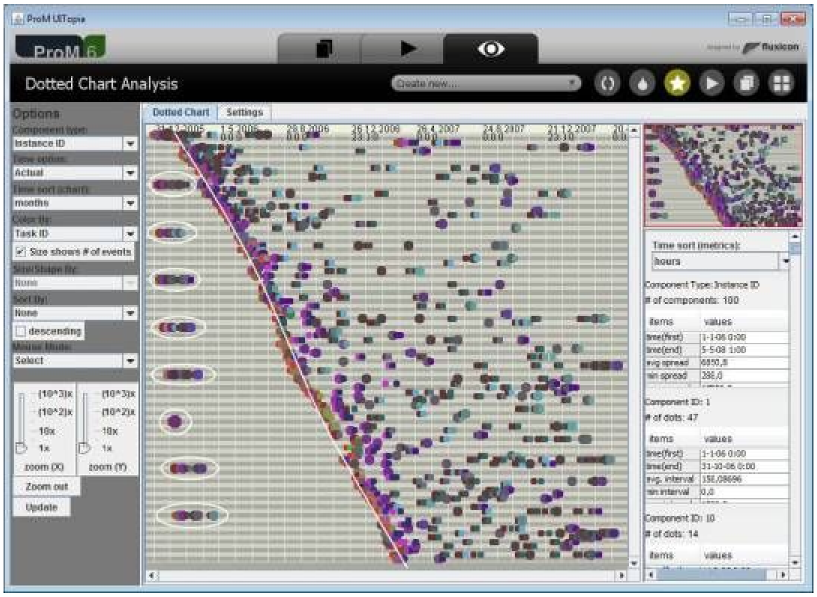
Figure 7 is a process map derived from a well known process mining tool called Disco[13]. For simple processes, it is an effective method to understand which paths are more common, but as the processes get more activities the method loses its effectiveness. To avoid spaghetti like maps generally a filtering is applied on the logs and it makes it easier to compare.

Figure 8 below is an example of such a case. The same event logs are used to visualise the process and to detect the possible problems. *Disco* allows users to add multiple filters to the process to give a customizable way of visualising the logs. Left side of the figure is the process map of the traces ending with *A_Complete* activity whereas the right side is the ones ending with *A_Denied*. The analyst who is familiar with the dynamics of the process might detect certain patterns with this visualisation. However, the tool does not give an intelligent way of finding out the deviations or outliers, it assumes that the user has knowledge of the process and has suspects regarding the reasons of the deviance. What's more it does not visualise any temporal aspect of the process though it's possible to filter the traces based on cycle time. In some cases, process maps can be useful, however, this approach is quite cumbersome and not practical when the process map is very large.

3.4.2. Dotted Charts

Time measures of processes are important to analyze the general performance of those and a visualisation based on the temporal variations can bring out many different aspects the other methods fail to detect. The events in the event logs are generally associated with a timestamp allowing us to measure the durations in between. The dotted chart technique relies on the point that events occur in the timeline. The dotted chart is basically a 2 dimensional area where y axis represents the case id and x axis represents the time. It's generally a better idea to show the relative durations rather than the absolute time values.

The following figure is an example of such chart from a well known framework called Prom[14]. Prom is a framework dedicated to process mining researches and tools. It allows developers to use their framework free of charge and extend it with plugins.



Options

Component type: Instance ID

Time option: Actual

Time sort (chart): months

Color By: Task ID

☑ Size shows # of events

Size/Shape By: None

Sort By: None

□ descending

Mouse Mode: Select

Time sort (metrics): hours

Component Type: Instance ID

# of components: 100

items	values
time(first)	1-1-06 0:00
time(end)	5-5-08 1:00
avg spread	6850,8
min spread	288,0

Component ID: 1

# of dots: 47

items	values
time(first)	1-1-06 0:00
time(end)	31-10-06 0:00
avg. interval	150,08696
min interval	0,0

Component ID: 10

# of dots: 14

items	values

Figure 9: Prom dotted chart[14]

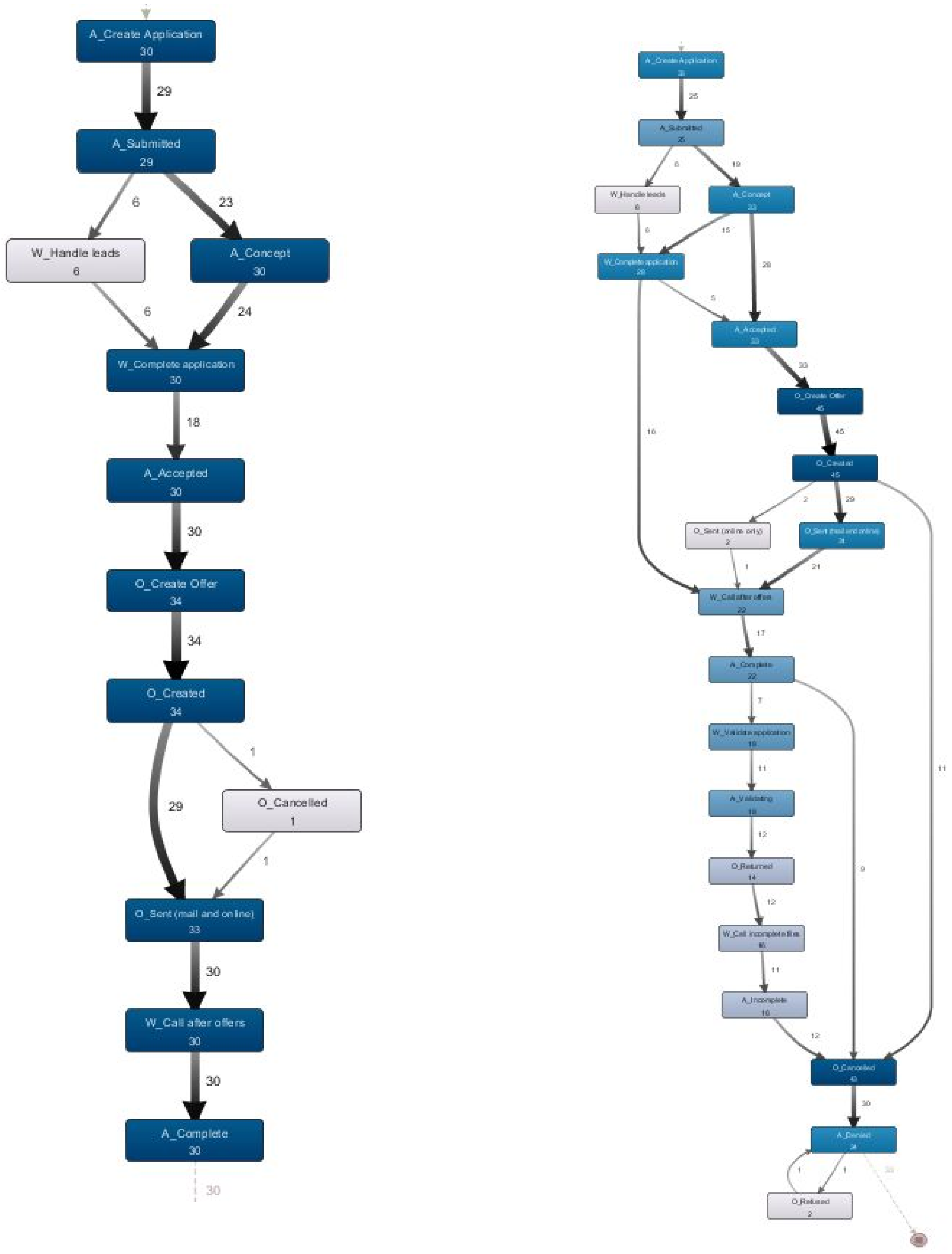


Figure 10: Accepted cased (left), denied cases (right)

4. Approach

This section discusses the approach for temporal analysis and comparison of process variants based on timeline diagrams. It starts with explaining the timeline diagrams and the basics then builds upon the initial concept by the adding and combining different techniques to reach the ultimate result out of which the tool is implemented.

4.1 Timelines

The basic idea is to visually display the temporal dynamics of an event log using a timeline diagram. We start by assuming that the activities in the process always appear in the same order. By the nature of business processes some activities might not be present in some traces but at the end the order of the activities remain unchanged.

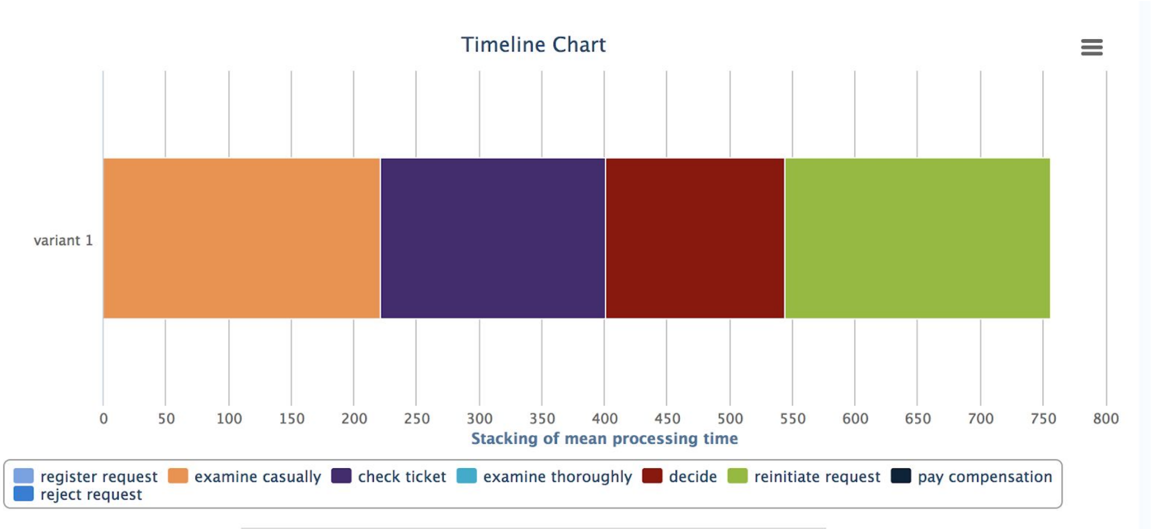


Figure 11: Example of timeline chart

The figure 11 illustrates the overall idea. In a single line of visualisation we can understand the general structure of the process and have an idea of amount of time each activity takes. Since timeline diagrams allow to represent a process variant using only one line, it becomes possible to display multiple variants on top of each other, which enables us to perform deviance mining from both control flow and temporal perspectives.

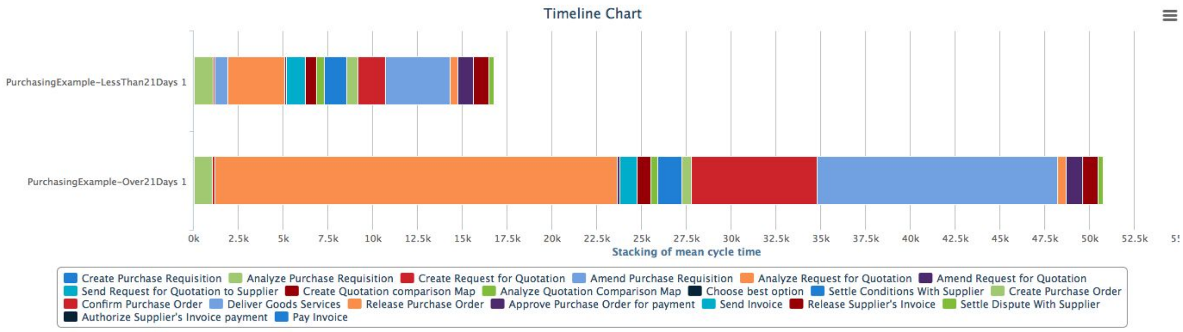


Figure 12: Example for comparison in hours

The example displayed in Figure 12 shows how to compare two variants in terms of the absolute time that each activity takes on average. The figure is extracted using two variants of the same process. The first log is the collection of traces which took less than 21 days which is considered as the threshold for cycle time success criterion. On the other hand, the second log is the collection of traces longer than 21 days. The diagram clearly states the difference between these two. The numbers on the x-axis indicate the number of hours spent on the specific activity. The summation of those hours gives a rough approximation of the cycle time of the process variant as well. While the absolute values of the duration gives a more distinctive and sharp visualisation, it fails to grasp the most important aspect of this tool which is to give a comparative analysis. Absolute values might over-emphasise numerical the difference between two variants in terms of the duration. Using the relative values instead of the absolute values would clarify the real difference between the variants. To clarify, let's assume in *variant 1* the activity *A* takes 10 days in average. The mean cycle time of the *variant 1* is 60 days whereas in *variant 2* the activity *A* takes 20 days with the mean cycle time of 180 days. The absolute values tell a different story but if we take the relative values, it's apparent that activity *A* takes relatively less time.

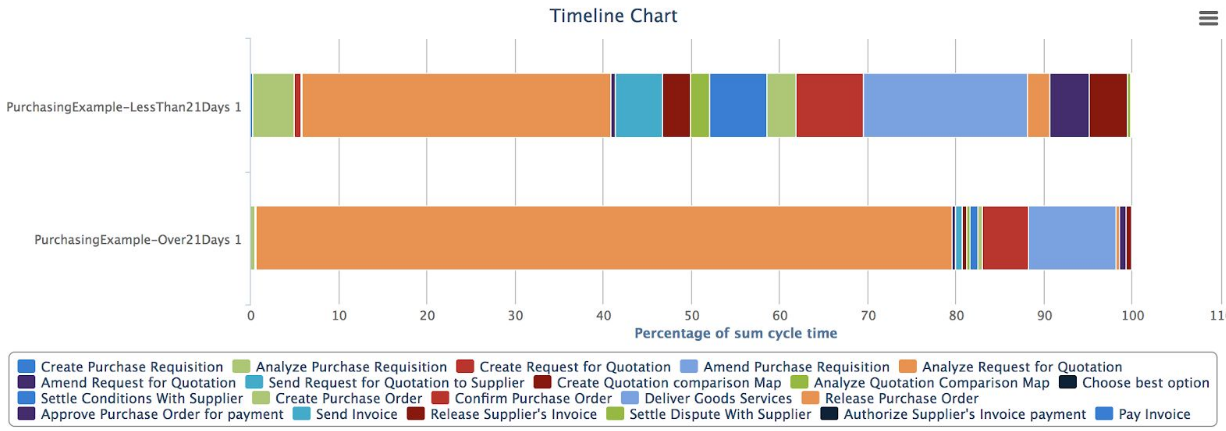


Figure 13: The chart with relative values

Figure 13 is drawn using the same data but instead of absolute this time relative values are used. The x-axis demonstrates the percentage of the time spent on the activity. Comparing the previous figure it's much clear to see that the activity "*Settler Conditions With Supplier*" takes relatively less time in the second variant. This wasn't trivial to infer from the previous graph.

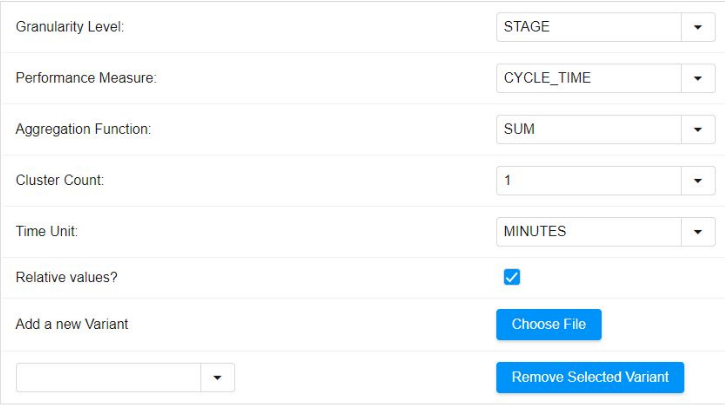
It's mentioned in the beginning of this section that traces do not contain all the possible activities. This poses another problem of ordering the activities in the graph. They should be ordered in a way that from left to right the order of the activities being displayed represents the order of occurrence in the process. To overcome this problem a new index called "*Mean First Occurrence Index (MFOI)*" is calculated. For each activity, the average of the index of the first occurrence is calculated. This gives a rough estimation of the order of the activities happening in traces. There are always exceptional cases like a single activity occurring only one time in a single trace which results in having a index of one. However, since the width of a process in the diagram is directly related to its duration with respect to sum of the durations of all the traces, exceptional cases are visually filtered out.

4.2 Clustering and Multiple Logs

Sometimes the process variants are given explicitly in the form of two or more separate event logs. However, in some cases the user does not have an explicit criterion for splitting the process into variants, but instead the user would like that the variants be automatically determined based on their relative speed. To deal with such situations, we extended the tool by applying clustering algorithm. Traces are clustered using K-Means clustering technique. A trace is represented as a vector where each activity has an index and the value is the occurrence count. To utilise the algorithm a distance measure needs to be defined between the feature vectors. In our case we used “Canberra Distance”. Essentially K-means algorithm efficiently finds the vectors that are close to each other and groups them up. This helps user to find possible deviance problems without actually knowing about them.

The tool we developed is a customisable timeline chart drawing tool. The user can add multiple variants as well as asking the system to cluster the log. By displaying multiple logs as multiple timelines in the same diagram, the too enables users to perform deviance mining from a temporal perspective.

Figure below is a view of the form that user can customize their preferences of how and what to display.



Granularity Level:	STAGE
Performance Measure:	CYCLE_TIME
Aggregation Function:	SUM
Cluster Count:	1
Time Unit:	MINUTES
Relative values?	☑
Add a new Variant	

Figure 14: Form for the user preferences

The parameters are explained in the Table 1.

Parameter	**Description**
Granularity Level	How to perceive the traces, sequence of “stages” or “activities”.
Performance measure:	Cycle time, waiting time, processing time, activity occurrence
Aggregation function	Sum, mean
Relative values	A boolean parameter for the timeline to be scaled to percentage or show the absolute values.
Time Unit	To decide the granularity of the calculations and representation of time. Depends on the process, it varies from seconds to days.
Cluster count	How many clusters does the user want to have. Useful if there are no deviant variants as separate log and user wants to see the deviant traces.
New Variant	New variants can be added or removed for the same process to visually compare them.

Table 1: Parameters and Brief Explanations

For the calculation, for each activity in the process the chosen performance measure is calculated using the aggregation function. For instance, processing time and mean are selected. Each activity in each trace has a processing time duration field which is simply calculated as the time between event_start and event_end of that activity in that specific trace. Since the aggregation function is mean, then the average of the processing times of each activity is taken. Some activities do not occur in some traces and in this case this is ignored. So the average is not based on how many traces there are but based on how many times the activity occurred. Afterwards using the activity MFOI calculations the activities are sorted so that logically they are ordered in the way they would have occurred in real life. The following graph illustrates the output.

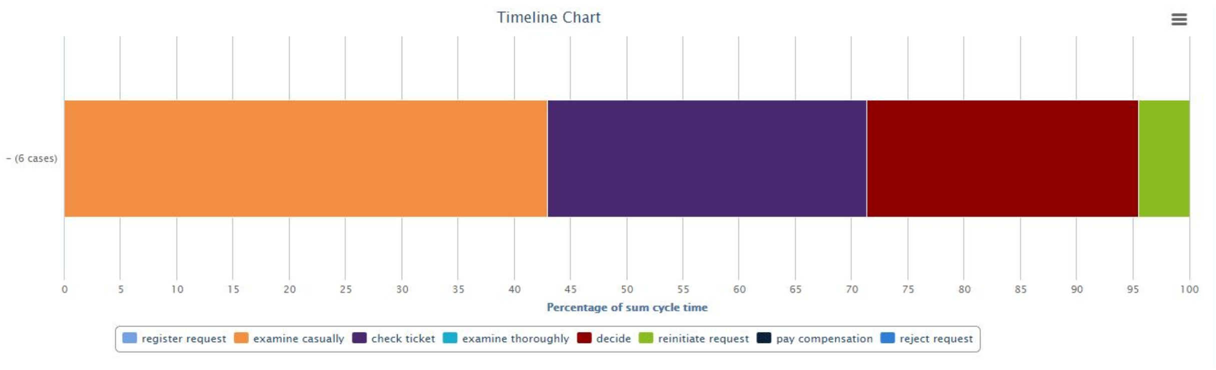


Figure 15: MFOI ordering

4.3 Stage Decomposition

The event logs belonging to real life processes contain many activities and even though the timeline diagram filters the uncommon ones by simply allocating them a very small room in the diagram, increased overall complexity is inevitable. Besides, many processes can be divided into sub-stages where essentially each stage covers one logical bigger step of the process. Perceiving the process from the further angle provides user the opportunity to detect the problems or possible distinctions of the bigger picture. The tool is extended to provide this feature by simply allowing user to choose the granularity level of the process. While it is trivial to extract the activities from the event logs, figuring out the mapping of activity to stage requires effort. The purpose is to ultimately find out a way to find sequence of stages in the process where each stage contains multiple activities thus the process contains less stages. Nguyen, Hoang, et al. 2017 [15], addresses to this issue and provides an algorithm to find a stage decomposition such that it maximises the modularity. In the following section, we will describe the algorithm suggested in the paper and implemented in the tool in detail in order to give a better comprehension.

The proposed technique is composed of two main steps. First one is to create a flow graph out of the event logs. The event logs are reduced into set of events where each event is a type of either "start" or "complete" of a specific activity. The assumed input event logs is a set of such logs and even though the .XES format allows other lifecycle events, those are ignored and the input logs are reduced to required format. Then a *flow graph* which is a directed-weighted graph is created using "directly-follows" relations between activities. If the activity *b* occurs right after activity *a* then a directed edge from a to b is added. The weight of the edge is solely based on the frequency of the aforementioned relation occurring. Figure 16 is an example of illustration of such graph.

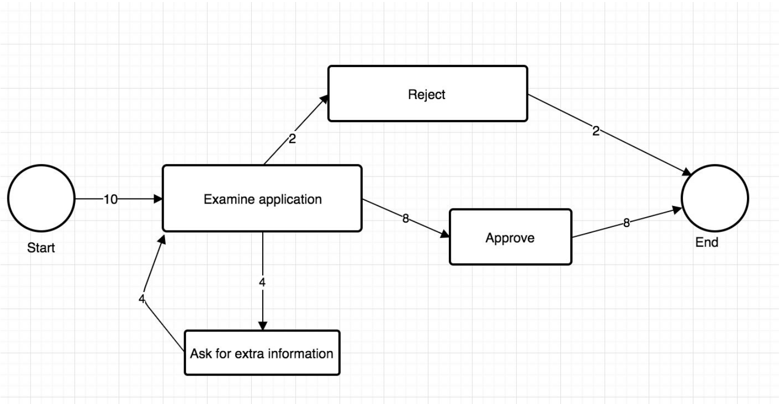


Figure 16: A flow-graph example from process logs

Given such a graph, stages are considered as a “quasi-SESE” fragments [15]. *SESE* stands for single entry single exit and quasi-SESE is a relaxed version of it where multi entry multi exits are conditionally allowed. To clarify, a quasi-SESE fragment has a single entry point with high influx and single exit point with high outflux where influx is the sum of the weights of all incoming edges and outflux is the sum of the weight of all outgoing edges. To evaluate the quality of the stage decomposition, they suggest to use “modularity”[16] which detects the community structures in the networks. A higher modularity score represents more of a community structure. To be specific, the modularity score considers the number of edges within the community and the number of edges from the community to another community.

The algorithm goes as following, first candidates nodes are select for being the transition nodes between stages. Selection process is based on the well-known *min-cut* concept. For each node in the graph, the node and all edges are removed from the graph and min-cut value is calculated, if it’s less than the mincut value of the original graph then it’s selected as a candidate node. Once all the candidate nodes are selected, the graph is cut into stages. This is done by removing the candidate node and it’s edges from the graph and find the reachability graph from the source node. The source reachability graph represents one stage and the remaining part represents another stage and the candidate node is a transition node. This stage decomposition is assessed by the modularity score and if there is an improvement on the score the new best decomposition is assigned. When the ultimate result is achieved when there is no improvement over “*modularity score*” anymore, each possible activity in the process is assigned to a stage and the process can be degraded to the sequence of stages providing a more compact overview.

5. Tool

The proposed timeline-based event log visualization technique has been implemented as a prototype. The source code and online version of the prototype are publicly available on github[1] and http://timelines.nirdizati.org respectively.

This section provides a comprehensive analysis and description of the tool in terms of both usage and development

5.1. Sample Use Case

The user interface at the very beginning requires an event log to be uploaded to the system right away to be used. Once the user uploads the file, the system checks if the file is in the required format. This validation is followed by parsing the logs and storing the data in the server-side memory which triggers statistics calculation phase based on the settings the user has selected. Finally, the timeline diagram is presented to the user.

The fact that the tool is served as a web server is a great advantage for the users as it does not require any software installation and it is available on any platform. The user needs to provide the event logs in xes format in order for the application to accept the logs. Provided that the user is already aware of the deviations and problems regarding the process, they can supply the application with multiple sub-logs of the same process. Additionally, the user can easily remove the previously uploaded files so that they can reuse the system for different variants or for different processes.

The charts provide many functionalities to the user to customize what to see there, such as filtering out the activities helps user to visualize the differences of some specific activities. These re-calculations are carried out in the client side without requiring and server connection. It can be very beneficial in certain use cases.

[1] https://github.com/erdemtoraman/Deviance-Miner

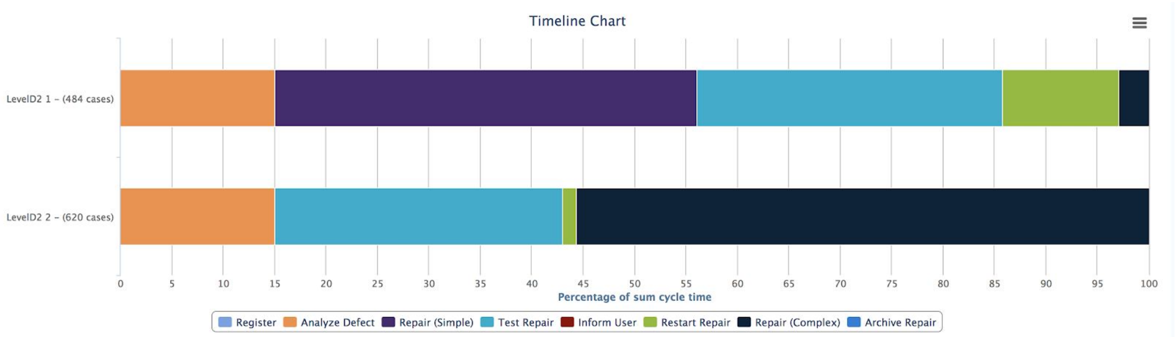


Figure 17: Relative time diagram example

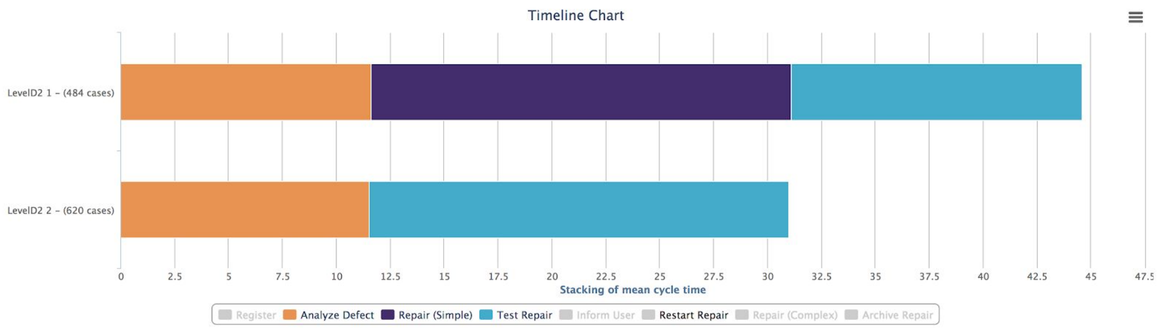


Figure 18: Absolute time diagram with filtering applied

The above-figures demonstrate the usefulness of filtering feature. By simply filtering out the other activities (in gray), the clutter is removed and the chart provides much more lean and clearer picture of the differences between 2 variants.

In the following section we will demonstrate a specific use case of this application by explaining how to use the tool and indicating the key points.

In this scenario we will assume that we want to examine the root causes of problems within a process and try to enhance it in temporal aspects by simultaneously displaying the diagrams produced by the tool. The selected process is a insurance company claim handling scenario and the logs[2] contain around 3500 unique claims.

Firstly we will use the commercial product *Disco* [13] to visualise and familiarise with the process. When we insert the logs into Disco, figure X, process map, is generated. From the process map we can understand that there are two kinds of claims and they're fairly equivalent in terms of frequency. In some cases the information is not sufficient and the case ends right away. After the case is registered a initial assessment is applied to

[2] http://www.processmining.org/event_logs_and_models_used_in_book

avoid unnecessary work for unreasonable claims. This is followed by the actual evaluation of the request which branches out to either closing the claim or proceeding with payment or advising the claimant. The process map is a clear way of visualisation of general flow of the process but while it shows very clearly how the process works, we need to combine the tool and Disco to discover more insights regarding the performance.

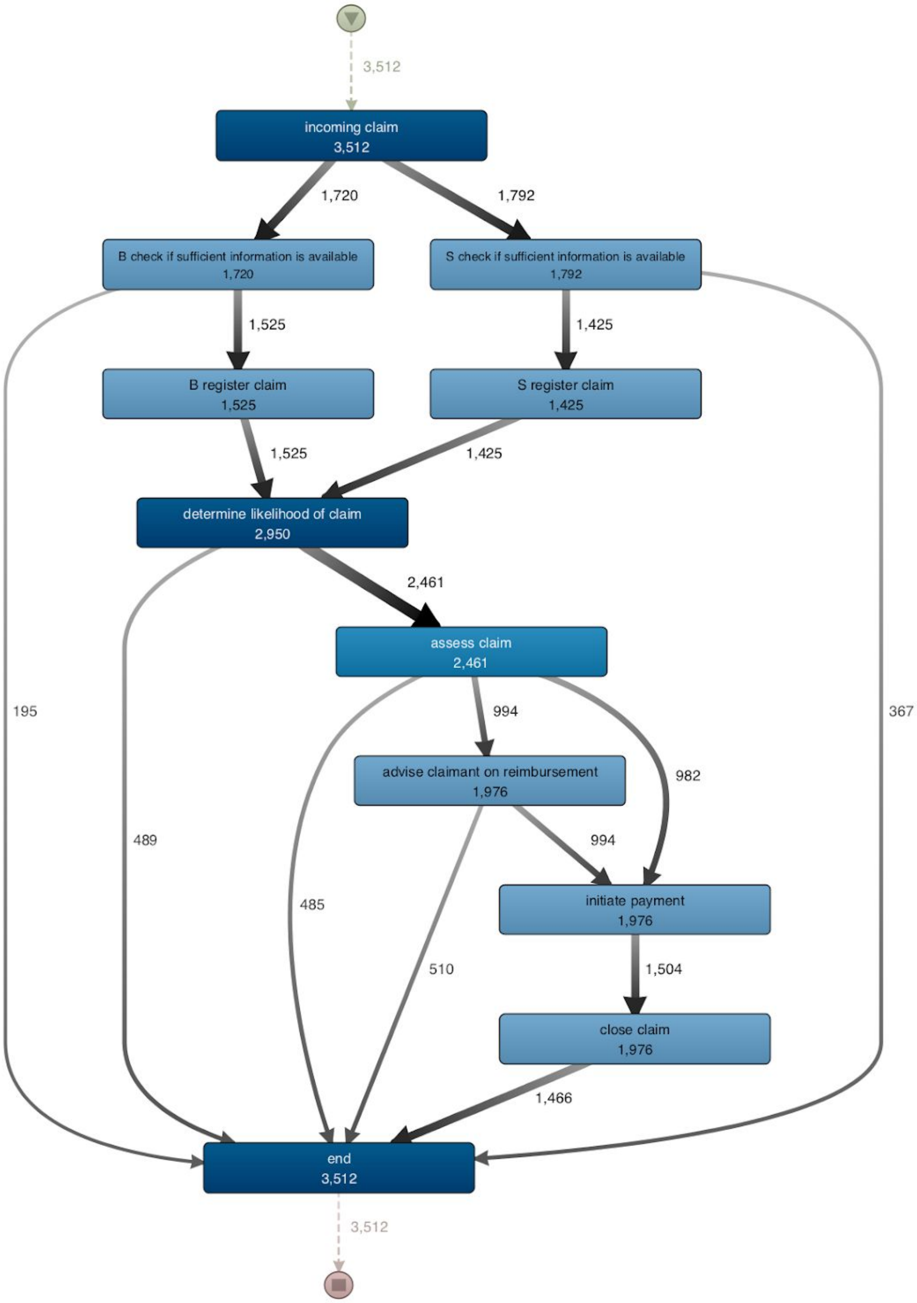


Figure 19: Process map generated by Disco

When we feed the tool with the logs with initial parameters it produces the following output.

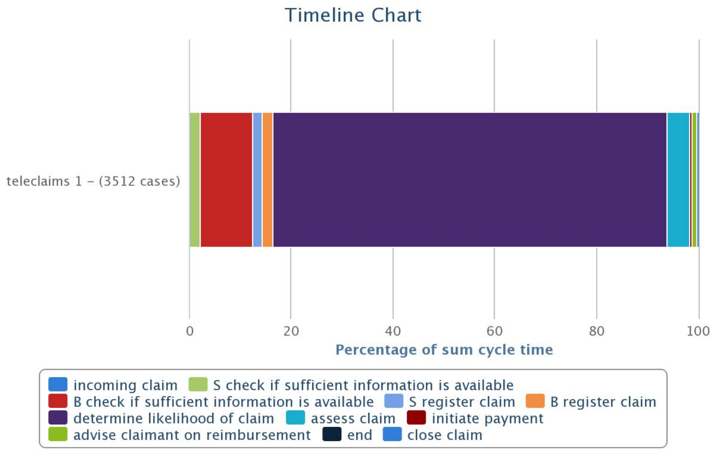


Figure 20: Cycle time distribution

It’s apparent at first sight that the activity *“determine likelihood of claim”* is the most time-consuming activity within the process as it takes approximately 75 % of the cycle time. One important point to reconsider here that this view actually shows the cycle time of each activities that is summation of both processing times and waiting times.

Visualising the distribution of the processing times of activities is as easy as changing a single parameter, from cycle time to processing time which yields the figure X.

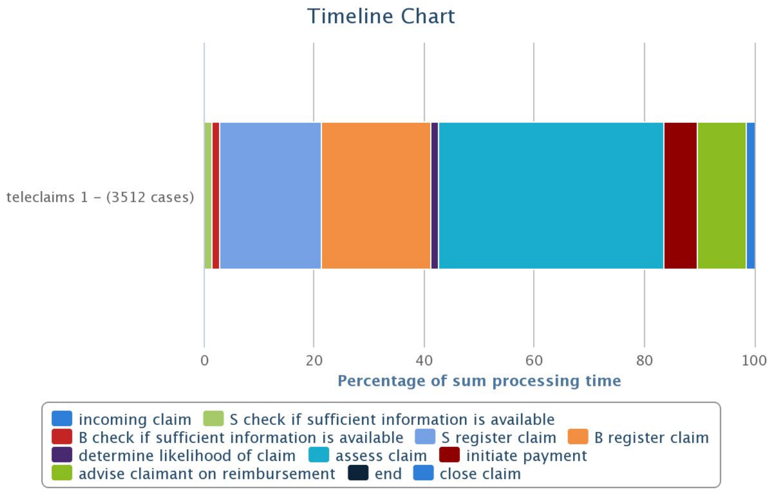


Figure 21: Processing time distribution

The output diagram can confidently be interpreted as the activity *"determine likelihood of claim"* is a big problem as processing time actually takes significantly less than the case stalling for that activity to be executed. Even though it's obvious that there is tremendous amount of waiting time in this activity it's very difficult to make an accurate estimate regarding the root cause of it as we're not familiar with the internal dynamics and the exact granularity level of the process. It might indicate that there is a resource shortage which executes this activity or there are other waiting times included in the sub steps of the execution of this activity such as waiting for information from an external stakeholder.

Another very practical functionality of the tool is clustering. It clusters the cases using the active settings. Clustering is done by vectorization of the cases where each activity is a dimension and the values of each activity is calculated from the active settings in the tool. If the user wants to cluster while performance measure is "cycle time" and aggregation function is "sum" then the values of each dimensions (activities) in the vector will be the sum of the cycle time of those activities within the case. This opens up many different possibilities of clustering the process thereby offering different results.

Cluster count is also an important parameter, too big of a cluster count value can yield unimportant and outlier cases which don't provide any business value. On the other hand, if the cluster count is held too small then some important distinctions can be merged together yielding an average result. The optimal way is to try to increase the cluster count as long as the number of cases within the cluster is relevant. The following diagram is produced with parameters of "cycle time", "sum" and cluster count of 5.

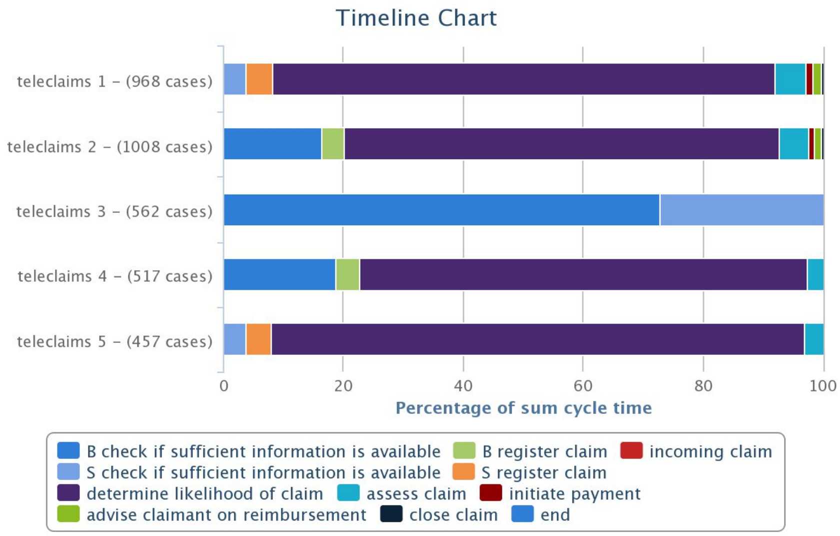


Figure 22: Clustering example 1

As it can be seen from the diagram, the tool clustered the cases based on the given parameters and it displays distinctions very well. The activity *"determine likelihood of claim*" is not executed at all in cluster-3, for instance, it takes up plenty of time in cluster-5.

To illustrate the idea of tweaking the parameters and discovering different aspects we will change the parameters to "*activity_occurrence*", *"mean"*, "*absolute values"* and cluster count to 3.

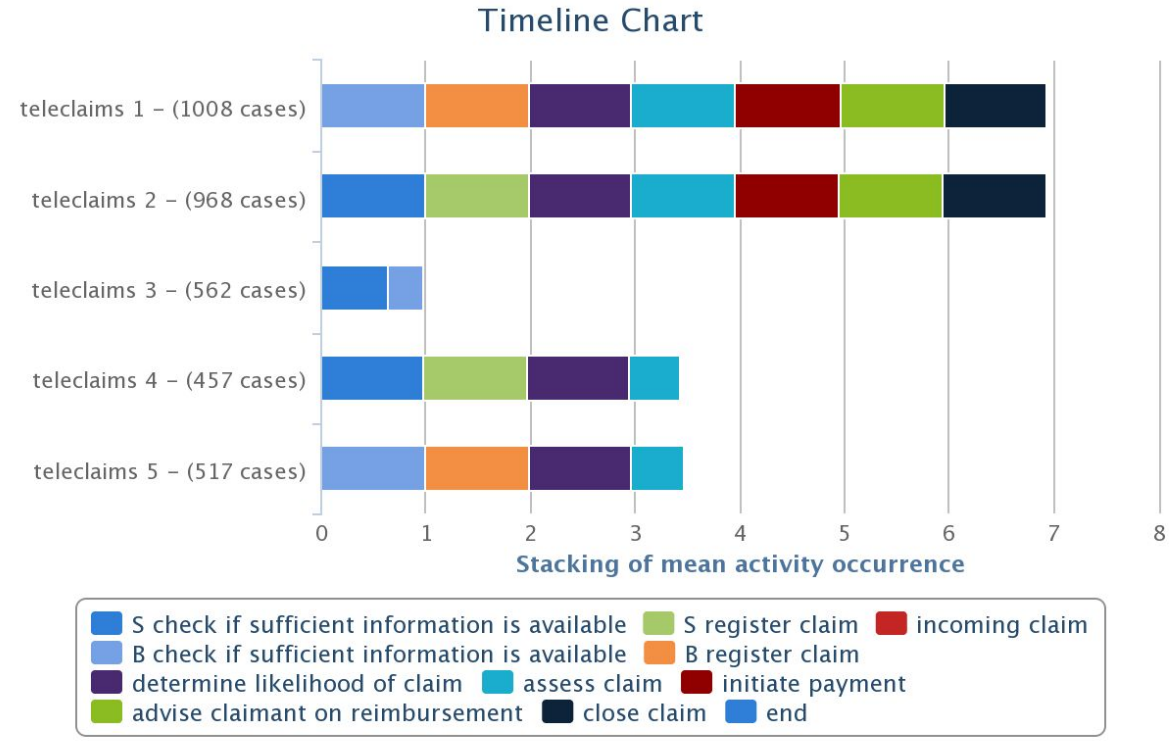


Figure 23: Clustering example 2

Cluster-1 and cluster-2 in these settings for type B and type S claims respectively, represent the so called happy path which is the sequence of activities that lead to the most expected final result. The third cluster, on the other hand, shows the paths that finalize with not sufficient information condition. Finally the last 2 clusters are the rejected ones for both types after the assessment is concluded.

Until this point we used the tool to find out the interesting patterns of the process without the pre knowledge of deviance, now we will show the functionality of adding new variants. The tool is capable of managing multiple logs of the same process at once. To enable this feature, using Disco, we splitted the initial logs such that one contains only the short cases while the other one only is composed of the longer cases.

The split is based on the total duration of the cases, while the short ones are up to 1 hour long whereas the long cases are between 8 hours to 12 hours. Figure X visualises the mean cycle time with absolute values in minutes which clearly exhibits the difference.

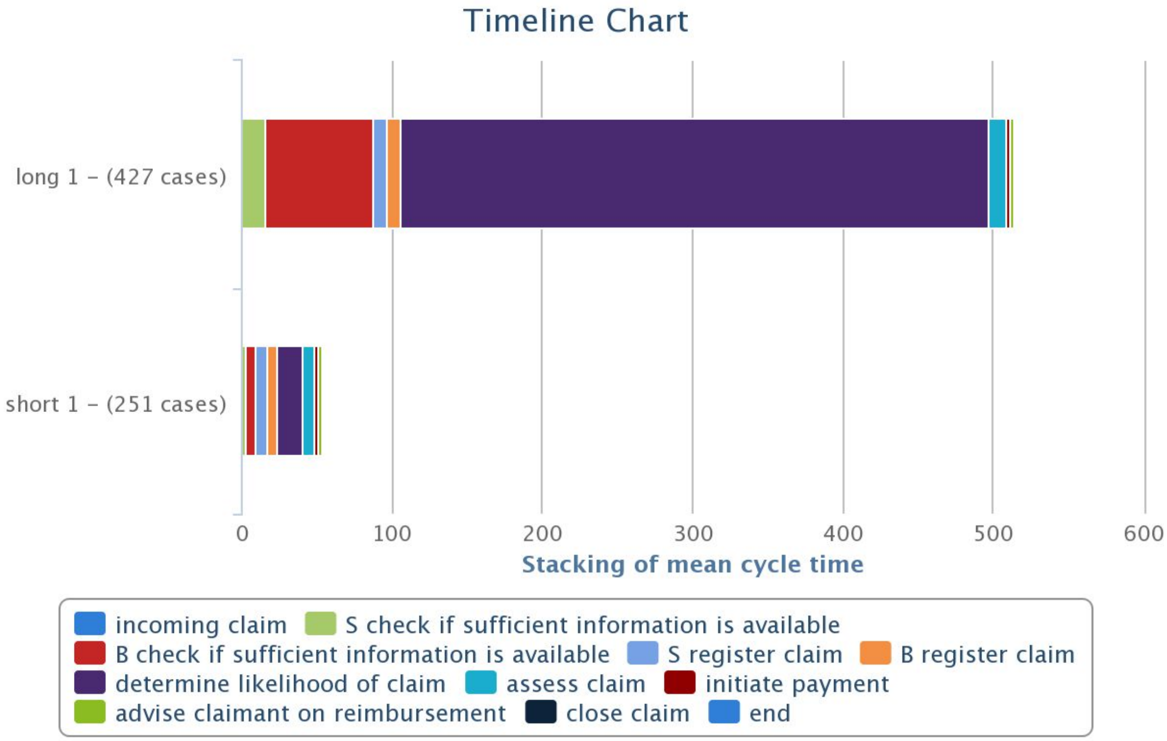


Figure 24: 2 different logs with absolute values

Now we will show how to use the granularity level setting which might be a great help in certain cases. While activity level examination gives a deep level of understanding it might be helpful to avoid seeing unnecessary small tasks but decompose the graph into logical stages. In our example for instance, we can right away see there are 4 to 5 different stages. To list a few, initial check of type S, initial check of type B, determination of likelihood, assessing and lastly the steps executed after successful assessment can be the logical stages of this process.

Figure X shows after decomposition how the variants look. At this point, the tool does not show which activities are listed under stages but that is left as future work. However, the difference between these processes are very clear that stage 5 takes much more time for long cases. After selecting "*STAGE*" as the granularity level all of the previous steps can be applied, the system considers each stage as an activity after that point while calculating everything in background including clustering.

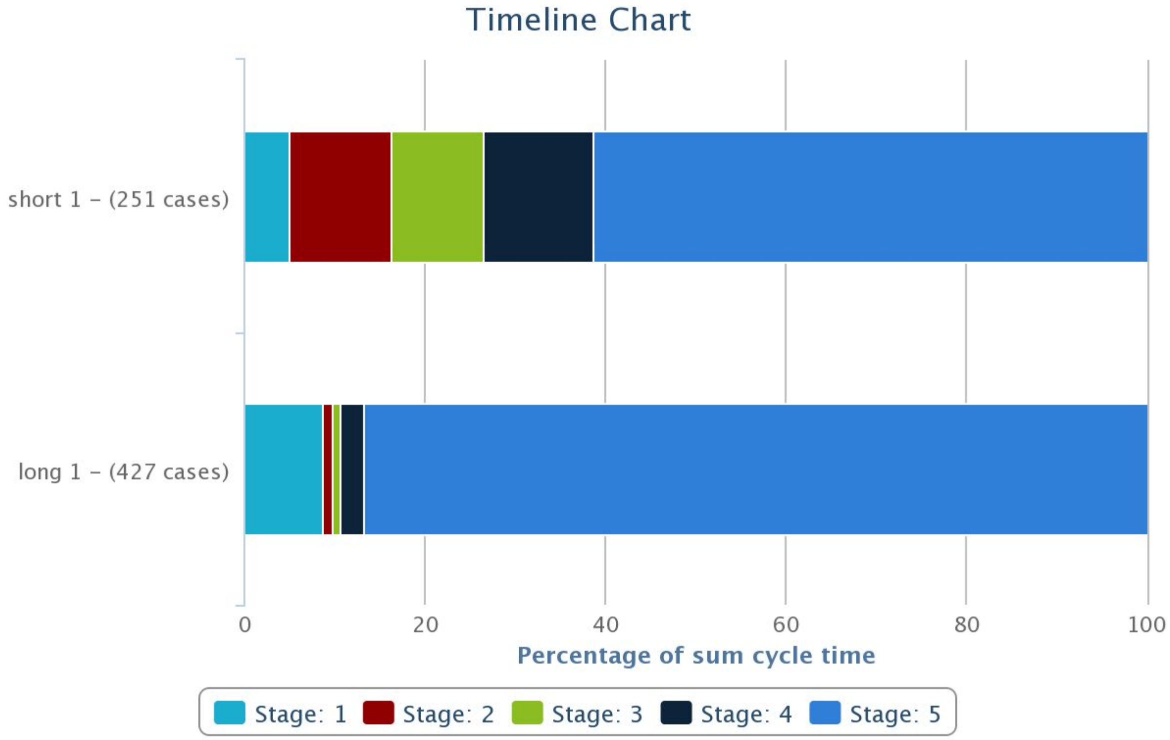


Figure 25: Stage based visualisation

5.2. Implementation Details

The application consists of four pipelined components as the Figure 26 depicts. The first component parses the event logs given in XES format. For this component, OpenXES [17] library is used. The library is written in Java and available in .jar files. Second part is to process the objects parsed by OpenXES libraries, so that they're in the best format of usage. The third component is to calculate the statistics and the last one is the presentation layer which handles user inputs and serves them the output. The last part is implemented using ZK framework [18] & charts and the system is served on web.

After this point, users can upload/remove event logs or customize the settings to see the logs from other perspectives. Figure 26 demonstrates the overview of the internal workflow.

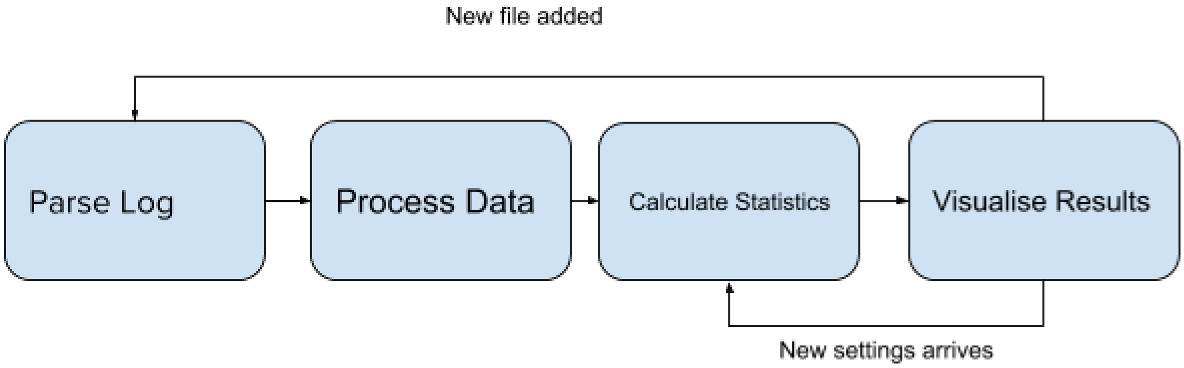


Figure 26: overall flow of the tool

The system is developed in a modular manner such that the presentation layer can be separated and replaced by simply abiding by the core api. The programming language used for the core api which calculates the statistics and prepares the data for the presentation layer is Kotlin and Java. Kotlin is a JVM language which can be interoperated with Java meaning that Java libraries and classes can be used in Kotlin and vice versa. Many beneficial sides of functional programming paradigms are enabled in Kotlin which helps tremendously during the development phase.

The frontend of the application or the last component in figure 26 is implemented using the ZK framework. ZK framework [18] is a commercial framework which generates web applications without having to write any client-side javascript application but only Java application. The presentation layer of the tool is fully integrated with ZK framework. Kotlin being interoperable with Java enabled to use the core api directly instead of tcp sockets. Domain model diagram can be found in figure 27.

The core process modelling consists of four classes namely, *Process, Trace, Activity, Process-Statistics* and *Stage.* A process contains set of traces, process id and has a relation with process statistics and has stages to which each activity is connected. A trace has an ordered list of activities and a trace id where an activity has a start time, end time and enablement time. In addition to the core classes, presentation layer is composed of 2 classes. One is the UI composer which is an extension class to a ZK framework provided class which simply handles the view of the chart and the other one is a simple data class storing the user's settings in the form called *ChartPreferences*. The overall architecture is finalised by a *Connector* class which orchestrates the whole system by analyzing the new *ChartPreferences* and manages the new statistics and sends the results to the UI layer. The class diagram and relations between the components can be found in the following figure.

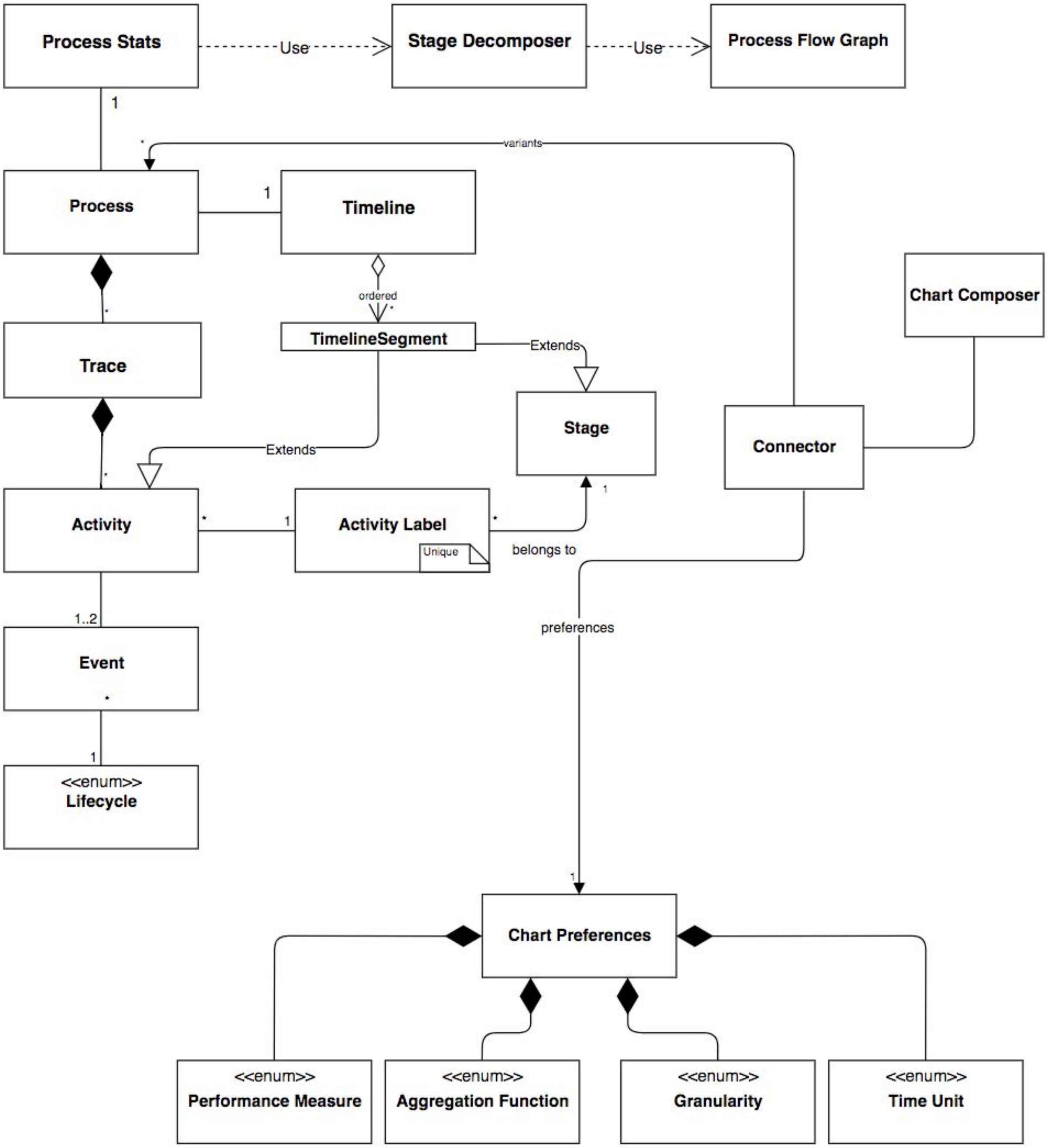


Figure 27: Domain Model Diagram of the System

5.3. Tool Limitations

The set of techniques that are introduced in section 4 are implemented in the tool yet it comes with certain limitations. The limitations can be classified under 2 categories one of which is performance and the other is coverage of use cases.

Initially, since the tool is not developed to be a commercial product but rather as a proof-of-concept, it fails to meet some of the performance related expectations including total memory consumption and caching. The application's logic is being executed on a server and anyone can access the link given which means that multiple users can access the application concurrently. The caveat regarding that is the number of concurrent users that can be handled is hardly bounded to the physical memory the server possesses. While enabling the users to easily and setup-free use the application, performance suffers and it's the inevitable trade-off between server-side and desktop client. One way to avoid such issues is to apply filtering such as random selection in advance for the event logs in another tool.

Secondly, the application depends on the temporal aspects of the activities, thus it is assumed that the activities in the input event logs contain both *start* and *end* events. If the event logs do not provide the required events then the application would still run and result in but there'd be no temporal data to analyse.

Another usability problem of the application is error handling mechanism which is not optimized but only enabled. Any unexpected input would fail the application and the error displayed would be the Java exception message. The error code and the message are caught by the ZK framework and presented to the user directly.

As to the coverage of use cases, the tool cannot diagnose the nuances between the variants in terms of control flow. For instance, it does not explicitly capture the notion of *repetitive* tasks. A task is simply represented as a segment in the timeline even though the tasks is repeated multiple times. Hence, if the distinction between deviant and normal cases stem from the fact that some tasks are repeated multiple times the visualisation technique would not demonstrate it.

Another limitation the tool has is associated to ordering of the events. Though the aforementioned method "*MFOI"*, successfully orders the activities in a meaningful manner. The tool cannot fully represent the control flow and if the difference between some tasks occurrence index is small or does not have any impact on a level that would shift the mean value then the tool struggles to visualise.

6. Conclusion

In this thesis a new technique to visualise and summarise the differences between two event logs is introduced and a proof of concept implementation is presented. The source code is published at github publicly and the running version can be found at http://timelines.nirdizati.org/.

The tool enables user to visualise a process using the event logs. The most common use case of the tool is to display the main differences between the deviant and the normal cases in the logs. The tool accepts the event logs in XES format and draws the time diagram. Timeline diagram displays the cycle time of one or more processes decomposed into either activity level or stage level which are logical combinations of the activities within the process. The diagram form provides multiple functionalities such as which aspect of the activities of stages to be considered in terms of durations. The options are processing times, waiting times and cycle times which is the sum of previous two. The activities or stages are displayed in the order that they occurred in the logs representing a timeline of events happening. If the user cannot provide a second XES file containing only the deviant cases or simply is not aware of what kind of deviance might have happened, the tool provide an option of clustering the event log so that similar traces are put together yielding a possible list of deviant traces.

In this thesis, also, we showed that combination of different techniques can bring out better outputs such as clustering and decomposition. The main focus of the new approach is not only to visualise the differences but also to navigate the user to find out a variety of differences by mainly relying on relative speeds of the different variants and applying certain data processing techniques.

As almost any other software, this tool depends on certain other libraries and frameworks as well. However, other than the presentation component the tool is mainly framework free. Thus can be later adapted to other presentation layers by applying minor modifications.

The thesis posed 2 main research questions.

RQ - 1: How can the deviance mining techniques be classified and what are the characteristics of those categories?

This question is answered in the section 3.3 as the common techniques are classified and explained. There are 2 main approaches to deviance mining that are control flow based and payload based. The motivation of the thesis was the lack of temporal dynamics based analysis.

RQ - 2: How to visualize the temporal dynamics of an event log concisely, and how to compare the temporal dynamics of two event logs?

The question is addressed at section 3.4 and the section lists the 2 most common visualisation methods namely process maps and dotted charts. The visualisation techniques are utilised in order to compare two different variants. Both of the techniques presented in this thesis mark the differences in terms of control flow, a timeline and relative speed diagram is the contribution of this thesis.

The tool and the provided concept surely requires further work to cover all exceptions as well as to provide a better user experience. One big improvement in terms of performance would be to switch the presentation layer with a desktop client such that the users can use the tool with larger event logs. From the perspective of effectiveness, it'd be a great improvement over the system if the tool could visualise the differences in terms of control flow. Another big enhancement could be to make use of the payload variables which are ignored in the tool. The payload variables such as the cost of the activity or the associated personnel might be visualised as well in addition to the duration. Besides, a filtering option can be added to generate the variants within the app instead of relying on other applications.

7. References

1. Dumas, M., La Rosa, M., Mendling, J., & Reijers, H. A. (2013). Fundamentals of business process management (Vol. 1, p. 2). Heidelberg: Springer.
2. Documents Associated with Business Process Model and Notation^{TM} (BPMN^{TM}) Version 2.0." BPMN 2.0, Object Management Group, Inc, Jan.2011,www.omg.org/spec/BPMN/2.0/.
3. Nguyen, H., Dumas, M., La Rosa, M., Maggi, F. M., & Suriadi, S. (2016). Business Process Deviance Mining: Review and Evaluation. arXiv preprint arXiv:1608.08252.
4. start | XES. (2017). Xes-standard.org. Retrieved 22 November 2017, from http://www.xes-standard.org/
5. Nguyen, H., Dumas, M., La Rosa, M., Maggi, F. M., & Suriadi, S. (2014, October). Mining business process deviance: a quest for accuracy. In OTM Confederated International Conferences" On the Move to Meaningful Internet Systems" (pp. 436-445). Springer, Berlin, Heidelberg
6. Bolt, A., de Leoni, M., & van der Aalst, W. M. (2016, June). A visual approach to spot statistically-significant differences in event logs based on process metrics. In International Conference on Advanced Information Systems Engineering (pp. 151-166). Springer, Cham
7. Dijkman, R., & Wilbik, A. (2017). Linguistic summarization of event logs–A practical approach. Information Systems, 67, 114-125.
8. Van Beest, N. R., Dumas, M., García-Bañuelos, L., & La Rosa, M. (2015, August). Log delta analysis: Interpretable differencing of business process event logs. In International Conference on Business Process Management (pp. 386-405). Springer, Cham.
9. Syamsiyah, A., Bolt, A., Cheng, L., Hompes, B. F., Bose, J. C. B., van Dongen, B. F., & van der Aalst, W. M. P. (2017). Business Process Comparison: A Methodology and Case Study.
10. Bolt, A., van der Aalst, W. M., & de Leoni, M. (2017, September). Finding process variants in event logs. In OTM Confederated International Conferences" On the Move to Meaningful Internet Systems" (pp. 45-52). Springer, Cham.
11. Broucke, Seppe vanden. "A PRocess Mining Tour in R." Data Mining Apps, 2 Nov. 2017, www.dataminingapps.com/2017/11/a-process-mining-tour-in-r.
12. Delias, Pavlos. "Process Mining in Healthcare – Case Study No. 3." Fluxicon, 21 July 2013, fluxicon.com/blog/2013/07/process-mining-in-healthcare-case-study-no-3/.
13. "Disco." Fluxicon, fluxicon.com/disco/.
14. Prom Tools, Process Mining Group, Eindhoven Technical University, www.promtools.org.
15. Nguyen, H., Dumas, M., ter Hofstede, A. H., La Rosa, M., & Maggi, F. M. (2017, June). Mining business process stages from event logs. In International Conference on Advanced Information Systems Engineering (pp. 577-594). Springer, Cham.
16. Newman, M. E., & Girvan, M. (2004). Finding and evaluating community structure in networks. Physical review E, 69(2), 026113.
17. XES. (2017, June 16). Retrieved July 22, 2018, from http://www.xes-standard.org/openxes/start
18. ZK, Leading Enterprise Java Web Framework. (n.d.). Retrieved July 22, 2018, from https://www.zkoss.org/

X. Licence

Non-exclusive licence to reproduce thesis and make thesis public.

I, **Erdem Toraman**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

   1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

   1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright, of my thesis,
   Visualizing Business Process Deviance With Timeline Diagrams, supervised by Marlon Dumas.

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, **09/08/2018**