UNIVERSITY OF TARTU
Institute of Computer Science
Software Engineering Curriculum

Yevhen Dorodnikov

Dealing With Complexity In Process Model Discovery Through Segmentation

Master's Thesis (30 ECTS)

Supervisors: Fabrizio Maria Maggi
Fredrik Payman Milani

Tartu 2019

Dealing With Complexity In Process Model Discovery Through Segmentation

Abstract:

The fundament of every successful organization is a proper business process management, as it allows maintaining the organization's production processes and the employed resources in the most sufficient way. Many noticeable problems occurred in production can be analyzed using process mining techniques and event logs obtained from tasks executed during the organization lifetime. One of the common ways to do this is to generate process models in order to study existing operations and explore processes in the organization with the aim to change them. With developing and expanding process mining technique, many methods and tools appeared which can help to solve this kind of task. However, most of the existing tools that are applied to real-life event logs produce spaghetti-like models that are difficult to understand without explanation. In this thesis we try to address this issue by filtering and sorting the logs before mining as well as adjusting model complexity, thus obtaining process models that we will measure and reform to satisfy desired complexity. A final result is a tool that produces a set of simple and understandable process models that the user can select according to his or her choices.

Keywords:

Process Mining, Process Analytics Tool, Complex Process Model, Segmented Process Model, Complexity Measures

CERCS: P170

Keerukuse Sisse Protsesside Mudelite Korrigeerimine Segmenteerimise Teel

Abstraktne:

Iga eduka organisatsiooni alus on äriprotsesside õige haldamine, sest see võimaldab parimal viisil säilitada organisatsiooni tootmisprotsesse ja kasutatavaid ressursse. Paljusid tootmise käigus esile kerkinud märkimisväärseid probleeme saab analüüsida protsessikaeve võtete ja organisatsiooni töötamise ajal tehtud ülesannete sündmustelogide abil. Üks levinud viise seda teha on luua protsessimudelid olemasolevate toimingute uurimiseks ja hinnata organisatsiooni protsesse eesmärgiga neid muuta. Protsessikaeve võtete arendamise ja laiendamise käigus kerkis esile palju meetodeid ja vahendeid, mis aitavad sellist ülesannet lahendada. Enamik olemasolevaid vahendeid, mida kasutatakse tegelike sündmustelogide puhul, tekitavad aga „spageti"-mudeleid, millest on selgitusteta keeruline aru saada. Töös käsitleme probleemi, filtreerides ja sorteerides logisid enne kaevet ning muutes mudeli komplekssust. Sellega saame protsessimudelid, mida mõõdame ja uuendame soovitud komplekssuse saavutamiseks. Tulemuseks on vahend, mis loob lihtsaid ja mõistetavaid protsessimudeleid, mida kasutaja saab soovi kohaselt valida.

Märksõnad:

Protsessikaevandamine, Protsessianalüüsi Tööriist, Kompleksprotsessimudel, Segmenteeritud Protsessimudel, Keerukusmõõdud

CERCS: P170

Table of Contents

1 Introduction

In recent years, improving business processes in organization is being a top priority task for stakeholders as it can save resources that are used in production and increase profit. Consequently, in order to achieve targeted improvement results it is required to know vital information about internal processes like performance, behavior and problems in the execution of existing tasks. Unfortunately this information is not easy to obtain.

For this reason within different successful organizations it is common to store event logs that occurred during process executions. Event logs can be stored in different formats amongst different organizations, but usually they should provide minimal information about events that occur during the execution of the process:

- The activity that was executed when the event occurred;
- Trace to which activity belongs;
- Time when event took place.

Based on these event logs, it is possible to build models of existing processes, compare and analyze them. This type of analytics is called process mining and in order to automate such difficult task, many software applications were developed. Using process mining techniques, one can investigate the current processes behavior and performance; check the conformance of the process executions or with respect to some pre-defined behaviors to evaluate their. Furthermore, through process mining performance issues can be clearly identified as well as interactions between resources which were used in production. After that, based on process mining analysis results, future corrections into existing processes can be scheduled for implementation.

However, existing applications for business process model generation do not take into consideration that the generated models should be read by humans. Most of the models, generated by these applications are hard to read and understand. In some cases, they even lack details that make the analysis of the generated model impossible and meaningless.

The aim of this thesis is to deliver an application that will be able to generate process models that satisfy specified complexity measures. Furthermore, application must be able to insert corresponding gateway into the model, producing results with high simplici-

ty. Generated models are supposed to be simpler and in the same time maintain the same effectivity as the single-generated complex model.

2 Background

In recent years there were written a lot of scientific articles that conclude different approaches in defining, searching issues and optimizing business processes based on available data obtained through records made during organization's production experience.

With the growth of the industry, increases amount and scale of problems appeared in production processes. In order to expand the scope of solutions that exist, to increase problematic process performance and keep organization's competition ability, many discussions were made and various approaches and techniques proposed. With growth of art and appearance of new methods and techniques emerges the need for their further study and improvement as they have their own limitations and there is no universal way to obtain desirable results.

The aim of this section is to provide review of existing approaches and methods of several researched articles and give meaningful answers for next questions that are listed below:

- What are the types of representation of event logs that can be used as input for segmentation process?
- What are the existing metrics that are used to quantify the difference between two model variants?
- What algorithm can be used for model generation?

The full list of reviewed literature can be seen in the reference section. Most publications were made not a long time ago and deliver fresh view on a state of the art which helps reach actual conclusions.

To search for relevant information, software engineering digital libraries were used with corresponding selection quotes: "Quality metrics for model mining" and "Process models generation based on the event logs". According to reviewed literature's content, the publications can be grouped by ones that describes existing methods of process mining and ones that provides real-life examples. The following sections were divided by performed steps in conducted research.

2.1 Data Extraction and Structuring

The first thing that requires our detailed exploration is the definition and structure of event logs. As any organization differs from another, event logs within one organization can vastly differ from others. Not only organization but each branch of the same institution can record event logs differently. Based on this the first step in the process mining should be data extraction and pre-processing tasks that allow us to assign same structure to each event log in stock.

In one of the earliest publications by van der Aalst states that event log is "an activity and a case" which relates to its activity [9]. To put it simply event log is data that was recorder by employee or system about process executions. Event logs according to van Aalst's approach should contain next information:

- case – contains information about certain process;
- activity – actions or tasks that were executed within process;
- originator – actor that executed the task;
- timestamp – date and time when task was executed.

As van Aalst approach is mostly aimed at finding out how people or procedures really work relative to each other and for Delta analysis – comparing actual and predetermined processes, the set of attributes listed above is perfectly fit for such purpose. However, this already put some limitations on this technique. First, no new measures can be added. Second, there can be excess of the information if we need only process comparison, regardless of originators that executed the task.

Another, later approach was proposed by Buijs and Reijers [8]. In context of the paper they proposed an apporach that is aimed at an actual and intended processes comparison. In order to represent data, Buijs and Reijers [8] compiled a table that contained only two attributes:

- trace id – set of activities executed in one run;
- number of runs – amount of runs that have same order of activities;

As result, we have readable representation of the process runs which can be used for comparison with initial process. However, one major flaw in this approach is that we do not have time of when each task started. This can negatively affect process comparison as we cannot track time spent on process execution.

According to Alfredo Bolt et al. [1], set of events that defined within the universe can have some set of different attributes that are related to these events. However, in terms of conducted research, Alfredo Bolt et al. [1] decided to represent event logs with the minimal amount in form of the table:

- trace id;
- activity;
- timestamp.

This minimal amount of attributes allows us to track any activity occurred within one life cycle of an instance using trace id and group them using timestamp. Using this information, Alfredo Bolt et al. [1], grouped events and built transition systems introduced in van der Aalst et al. earlier publication [4].

Transition system [4] contains states as nodes and transactions connecting them. Such an approach as representing existing event logs as transition system give us an opportunity to find out when tasks or whole process was finished and how long it took before each of activities executed. However, such an approach mostly aimed at business processes that have homogeneous behavior, meaning that comparing transition systems that have different activities would be a difficult or rather impossible task.

In order to solve the problem of homogenous behavior, van Beest et al. [2] tried to employ log delta analysis method which was developed based on a technique used in [3, 5] - model delta analysis. Log delta analysis was aimed to solve the problem in deviance mining that raised difficulties when model delta analysis technique was applied to complex logs and introduce automation. Deviance mining is the art that aimed at studying reasons that can be behind process anomaly.

Based on this flow of idea, van Beest et al. [2] introduced State Event Structures that as modified Prime Event Structures [6]. Prime event structure is a graph containing nodes that represent events, or tasks that are occurred in the process, and edges describing the sequence of the flow of the process.

However, unlike prime event structure, that labeled by task that occurred after the start of an event, state event structures represent set of all events and tasks that were prerequisite to conducting determined task. Another perceptible note about event structures is that if there is an activity that can be reached using different states, it is considered as separate nodes.

To construct event structure from event log, by van Beest et al. [2] was proposed next structure of event log:

- Trace – set of task occurred within one run.
- Ref – identifier of trace.
- N – number of times trace appeared in production.

After we formed table with the event logs, we can transform it to a set that contains ordered runs of the tasks and extract concurrency relationships between activities.

To solve this task, we can use several possible approaches. The one proposed in [2] is to use alpha concurrency approach [7]. According to [7], two activities *A* and *B* are alpha concurrent if there are at least two traces that contain *A* and *B* which are both invoked with two same events. Based on defined ordered runs and concurrency between their activities, we can build primed event structures.

If we compare the representation of event log used to build primed event structures to the one used in [1] we can conclude that they are mutually similar to each other. Event log from [1] can be interpreted as more detailed then the one from [2] with the exception that event log from [1] has additional column that represents date and time of execution of the activity. So, based on this observations we can assume that both transition systems and event structures can be modeled using same event log, therefore we can compare both structures and their outcomes. However, the major difference between transition system and event structures is that in the first case we can separately study each event that raised in our system, while in event structures we examining the process itself. The first case gives us the ability to specify features of the event we want to compare (time, frequency etc.) while in the second case we escaping the problem with the inhomogeneous behavior of the processes.

Detailed examination of data extraction and structurization showed us that event log structurization should be based on the purpose of the comparison and desariable set of event properties in outcome. Event log representation should be in easy readable form, for example table, and contain attributes that represents different event properties.

As for the event properties, we can distingish them by two types: (i) required and (ii)desirable. To required we can refer trace of the run and activities. While desarible can be any event properties which we want to use in difference quanification.

2.2 Existing Metrics

To evaluate model complexity various metrics exists. In order to select suitable metrics that can be used to evaluate model simplicity, various papers were reviewed. According to Irene Vanderfeesten et al. [10] using metrics as guiding principals can help to design less error-prone, effective and easy to manage and to understand the model. Irene Vanderfeesten et al. [10] came to the conclusion that a business process model, whether it have been modeled as a BPMN or a Petri Nets have a lot of common with software development. Usually, the application that is developed, is separated into several modules or functions that change given input to present some output. Based on the given idea, they proposed to use the same set of the metrics used for software code complexity evaluation to evaluate the simplicity of the process model. Those metrics are:

- Coupling – in software development coupling metric represents a number of the connection between different modules. The high value of coupling signifies that the program is more likely to have some errors. In order to apply a coupling metric to the process model, we have to consider the activities from the process model as modules from the program. So, coupling in the process model represents a number of the connections between the activities in the process model. [10]
- Cohesion – represents a measure of the relationship of the methods or functions in the one module. Low cohesion value shows that the application is more likely to have errors. To use cohesion as a complexity measure for the process model, we have to use operations instead of the functions and methods in the application. [10]

- Complexity – calculated by taking the number of modules in the program. Probability of error appearing in the program increases when the value of complexity is high. The complexity of the process model can be calculated by taking the number of activities in the model. [10]
- Modularity – shows the degree of the modularization. Under-modularization, as well as over-modularization, indicates that there are more errors in the application that it is desired. [10]
- Size – shows the size of the modules or deepness or their nesting. Big size value increases error probability in the application. [10]

Among complexity measures described above, coupling and cohesion considered as the most important in the evaluation. Taking this into account, Hajo A. Reijers and Irene T.P. Vanderfeesten [11] discussed the applicability of these two metrics in the process designing. In their paper, they use cohesion and coupling to solve the issue of the activity design in the administrative process. In the research paper, they came to the conclusion that cohesion and coupling can be used during workflow design to find the best solution among existing.

An alternative metrics for model complexity evaluation were proposed by Antti M. Latva-Koivisto [12]:

- Coefficient of Network Complexity (CNC) – described as a ratio between edges and nodes in the process model.
- Cyclomatic Number – calculated a number of independent cycles in the process model.
- Complexity Index – calculated by taking a number of node reductions.
- Restrictiveness estimator – measures network complexity in the context of the resource-constrained project scheduling problem (RCPSP).
- Number of trees in the graph – classifies graphs by the number of trees they contain.

The idea of using alternative metrics comes from applying them to the graph as the process model can be easily represented as a simple graph where its activities serve as the nodes and associations are the edges.

2.3 Model Discovery Algorithms

The goal of the process discovery algorithm is to analyze the data and produce a process model that corresponds to the input log.

In the research paper, Albana Roci et al [14] proposes an extended α-algorithm which is aimed to deal with the problem of the short-loops and to process large event logs with a better time compared to the other versions. The main idea of this algorithm is to create the incident matrices based on the events in the log.

Another algorithm was reviewed by Andrea Burattin, Alessandro Sperduti and van der Aalst [15]. Heuristics Miner is a process mining algorithm that calculates different types of frequencies to discover relationships between the activities. The main advantage of this algorithm is that it the most suitable for mining stationary streams due to the ability to cope with streams exhibiting concept drift.

The third reviewed algorithm is Split Miner. According to Adriano Augusto et al [5], Split Miner is aimed to produce simple, deadlock-free process models that correspond to the event logs that are used as input. The main advantage of the algorithm is a filtering method for directly-follows graphs and its ability to determine split gateways which are necessary for traces with parallel or alternative flows.

3 Approach

This section explains the meanings of the complex and simple models, complexity metrics, discovery methods and algorithms used in this thesis.

3.1 Complex Model Definition

First of all, it is necessary to define, what the meaning of the “complex model” is and how it differs from the “easy to understand model”. For that reason please refer to Figure 3.1.1.

Figure 3.1.1 demonstrates an example of a complex process model. It has many nodes that represent activities or gateway and many arcs that describe the process flows. The excess amount of the nodes and arcs makes the model hard to read and analyze by the human eye as it usually requires an understanding of each possible flow that goes through the process.

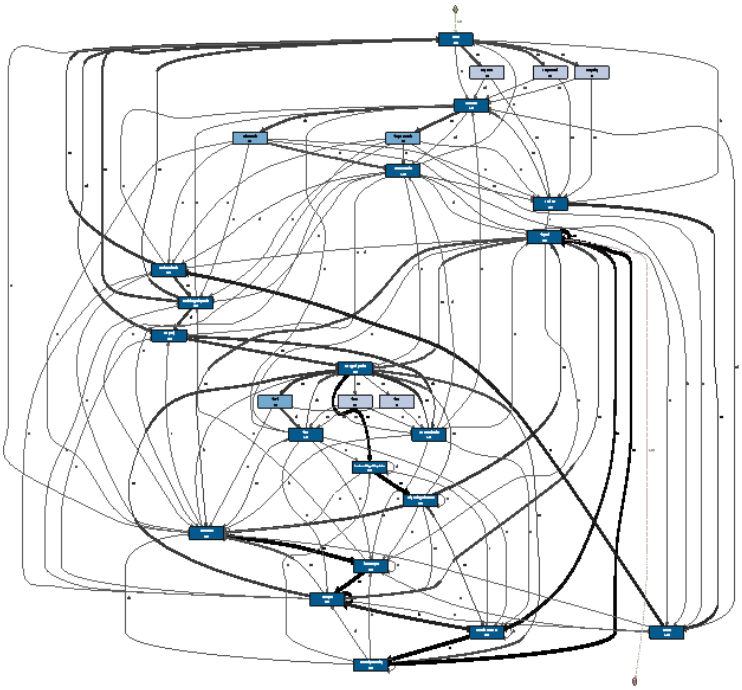


Figure 3.1.1 Example of the complex process model

When it comes to the process management, it is more welcoming to have as little activities in the process as possible. It requires less funding, consumes less time for process execution and produces the same result. For that reason, many analysts prefer a simple model that describes the behavior of the process. However, on practice, the workflow of the process may differ from the one described in the model and events that usually are not required for correct output appear and recorded in the event log. As a result, the event log may contain the excess events that may not be needed for the main flow of the process.

If this kind of log will be used for the model generation, we will receive a spaghetti-like complex process model which is hard to analyze. So the complex process model is supposed to contain excess activities that are not needed for the analysis and in order to receive a simple process model the excess events should be defined and removed from the model or event log.

3.2 Process Discovery Methods

One way of removing excess events would be creating segmented models based on the certain set of traces that is likely to contain only necessary events.

To obtain a set of the traces we can use the following discovery methods:

- A method based on the trace frequency. Sorts all traces variants by the total amount of times they appear in the log.
- A method based the time taken, starting from the first activity till the last one. Builds all possible traces and compares their average time taken to execute each one of them. Next, the fastest traces are selected and the model that satisfies complexity measures is build.
- A method based on provided resources. Builds a model based on traces in which resources satisfies some performance criteria defined by the user.
- A method based on artifacts involved. Builds a model based on traces that uses specified nodes, resources or sets of them.
- A method based on the longest or the shortest traces. Selects traces with the smallest or the biggest amount of nodes. Generates segmented models based on them.
- A method based on endpoints. Selects traces that have specified by the user starting and ending activities. Generates segmented models based on these criteria.

3.3 Complexity Measures

After the traces in the log are sorted or filtered according to one of the methods described above, we can take a certain amount of the traces at the beginning of the list to generate a segmented model.

To define that amount we can use the following complexity measures:

- A number of the nodes (N). Shows the total amount of activities, gateways, and events in the generated model.
- Density (D). Calculated by dividing all arcs on the model by the amount of all possible arcs between the nodes.
- Cohesion (Ch). Shows the rate between relative cohesion and informational cohesion [13].
- Coupling (Co). Represents the amount of the nodes that are connected to each other [11].
- Coefficient of network connectivity (CNC). Displays rate between arcs and nodes in the model [12].

First, we generate the model, then we calculate its measures and check if they satisfy desired values or not. After we found a segmented model that satisfies complexity measures, we take the next group of the traces that are left in the list and generate the next models. Then we repeat the action until we use all traces in the log. As result, we will receive a list of the segmented models with desired complexity.

4 Tool Description

The tool can be downloaded from:
https://bitbucket.org/teaCat-/sbpm/src/master/

In this section the architecture of the implemented tool is presented including its modules and features with respect to its front-end and back-end parts. The aim of this section is to explain the selected technologies and the process of realization and implementation of the features in the application.

In order to set on the architecture of the application, the first important step is to decide what type of the application will be developed. The use of desktop application implies some limitations like dependency on the specific platform for which tool was developed, necessity to download and install the system or supporting and updating outdated versions of application. To avoid these problems it was decided to create a web-based application which can provide immediate access regardless of the platform or device selected by the user.

The application is composed of two parts: a front-end which is responsible for the interactions with the user and a back-end where all main logic and data are stored. The communication between those two parts is implemented through json requests that are generated by the front-end part. This kind of implementation provides accessibility to the tool services for the user without a need to install it on any devices. Furthermore, this gives portability to the application modules: the data produced by modules on the back-end part can be also used in separate projects, regardless of the programming language that was used.

4.1 Front-end Application

4.1.1 Front-end Architecture

Front-end application was developed with the aim to serve as a separate and independent tool. For that reason application operates as the separate server, using Vue.js technology.

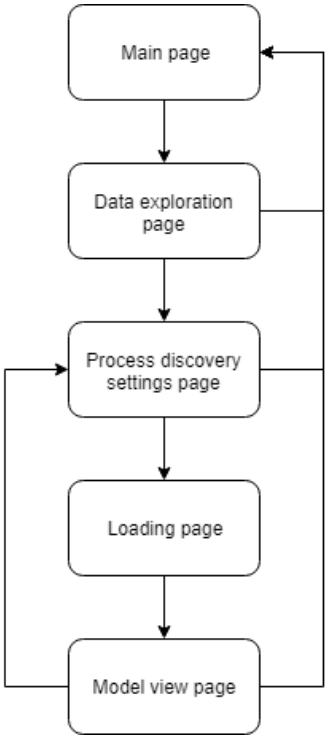


Figure 4.1.1 Map of the site

Vue.js is a JavaScript framework which can be used for creating custom interfaces. The main advantage if Vue.js framework is that it creates a single-page application which can change its content without reloading the page. This approach releases the user from necessity to load pages thus decreasing the time spent on page loading.

Although front-end is a single page application, the next components can be considered as the separate pages:

- Main page.
- Data exploration page.
- Process discovery settings page.
- Loading page.
- Model view page.

The complete site map can be seen in Figure 4.1.1 with all redirections from page to page.

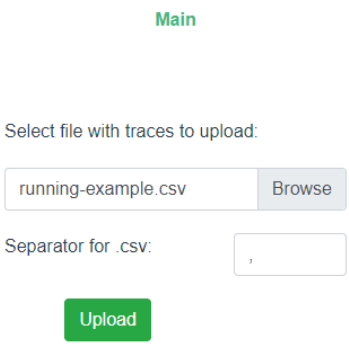


Figure 4.1.2 Screenshot of the main page

4.1.2 Views Description

The overall design of the pages was developed to be structured and as simple as possible. To satisfy that decision Bootstrap library is used. Bootstrap is popular open source library that contains a lot of different styles and components that can be used in site construction.

Whenever the user accesses an application, he or she is redirected to the main page. The main page serves as a starting point in the process of segmented model generation. On this page, the user can upload a file with the event log that he or she wants to use in his analysis. The file can be of the two types:

- Comma-separated values (CSV)
- Extensible Event Stream (XES)

In case of CSV file, the application will propose the user to type a separator for the values listed in the file. If the user leaves the field with separator without any changes, default value "," for the separator will be used. The screenshot of the main page can be seen in Figure 4.1.2.

After user has decided on the file with the log he wants to process, he can click the submit button and application will send the request to save the uploaded file along with the separator in case of CSV file format. After file successfully placed on the server, the application will read the first eleven lines and show them to the user on the data exploration page, as it displayed in Figure 4.1.3.

Regardless of the format of the file that user uploaded, the application will present data in the form of the table making easy for the user to verify file content. However, since in files with CSV format column types are not defined, the application will ask the user to select appropriate types for listed columns. This is necessary in order to generate correct XES file from CSV file since discovery algorithms can support data only in the form of the objects that can be crated based on the content of XES files.

Possible choices for column types are:

- trace – defines a column as the id of the trace to which corresponding line or event belongs. At least one column with given column type should be specified.
- activity – defines a column as a name of the activity which will be displayed on the nodes of the model. At least one column with given column type should be specified.
- timestamp – defines a column as time and date of the event execution. If selected, an additional field will appear where the user will have to type format of the time and date. At least one column with given column type should be specified.

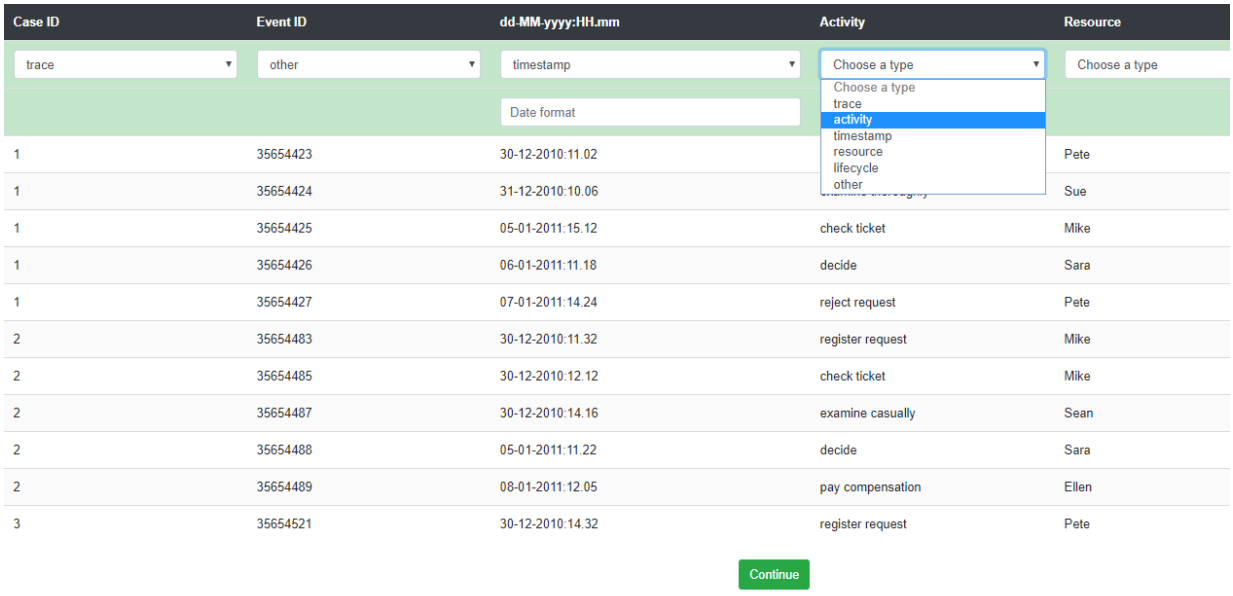


Figure 4.1.3 Screenshot of the data exploration page

- resource – defines a column that contains resources used for activity execution. At least one column with given column type should be specified. Resources will be counted for each activity and result will be displayed along with models on the appropriate page.
- lifecycle – indicates the start or end of the activity. This column type can be ignored by user otherwise there can be only one column of the given type.
- other – defines a column as additional data for the activity. Values of these columns will be counted along with the resources and will be displayed on the appropriate page. Can be several columns with the given type.

After the user have finished with data verification and clicked submit button, the application will redirect him to the discovery settings page. In Figure 4.1.4 a screenshot of the page is placed.

Discovery settings page serves as the last step in data preparation for process model generation. Here user can specify how exactly application supposed to process provided logs and which metrics should be used for segmentation.

Among discovery methods that the user can select are the following (all methods are described in section 3.2):

- Process discovery by trace frequency.
- Process discovery by included/excluded artifacts.
- Process discovery by the shortest traces.
- Process discovery by the longest traces.
- Process discovery by the most time consuming trace.
- Process discovery by the least time consuming trace.

By selecting one of the discovery metrics, the application will sort or in some cases remove event logs from the list that is used for process models generation.

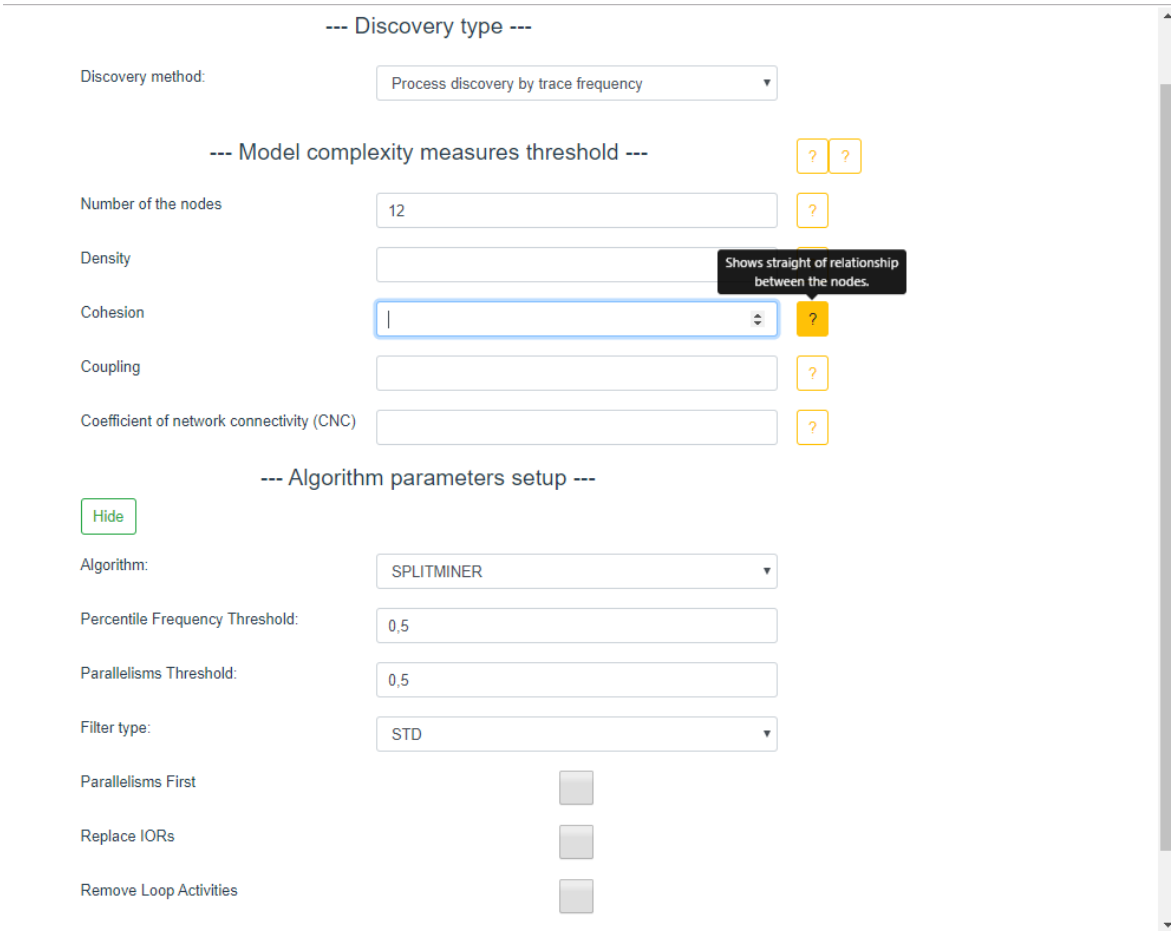


Figure 4.1.4 Screenshot of the discovery settings

Aside from the discovery method, the user can specify various metrics that will be used as a threshold for next model generation (see 3.3 section for the details):

- Number of nodes.
- Density.
- Cohesion.
- Coupling.
- CNC.

Whenever metrics of the generated model will reach but not the cross threshold of specified by user values, the model will be added to the list and algorithm will proceed to another model generation.

If fields of some metrics are left empty, they will not be included in the process. Or the user can leave all metrics fields empty and in this case, one model will be generated based on all traces in the event log.

In case the user is not sure about what certain metrics mean, he or she can read the explanation in the small window that appears whenever help sign near the field of the interest is clicked.

Aside from the discovery method and metrics threshold values, it is also possible to change the algorithm that will be used to generate a model from the event log and its settings. By default Split miner algorithm is selected and corresponding algorithm settings are displayed:

- Percentile Frequency Threshold. Used to specify how frequent one edge supposed to appear in order to be retained by the algorithm.
- Parallelisms Threshold. Used to specify a threshold for the decision whether gateway will be of type parallel or not.
- Filter type. Filtering type used by the algorithm.
- Parallelisms First. Defines whether the gateway should be considered parallel by default.
- Replace IORs.
- Remove Loop Activities. Signifies whether loop activities should be removed during the filtering process.
- Structuring Time.

After the user finished filling fields of interest, he may click a submit button and application will redirect him to the loading page. Loading page does not contain any input fields and serves as a process indicator showing that the application is processing the data. While application busy with the model generation, the user is allowed to navigate through the application or even start another discovery process. The server will respond to all requests, as the process of the model discovery runs in a separate thread.

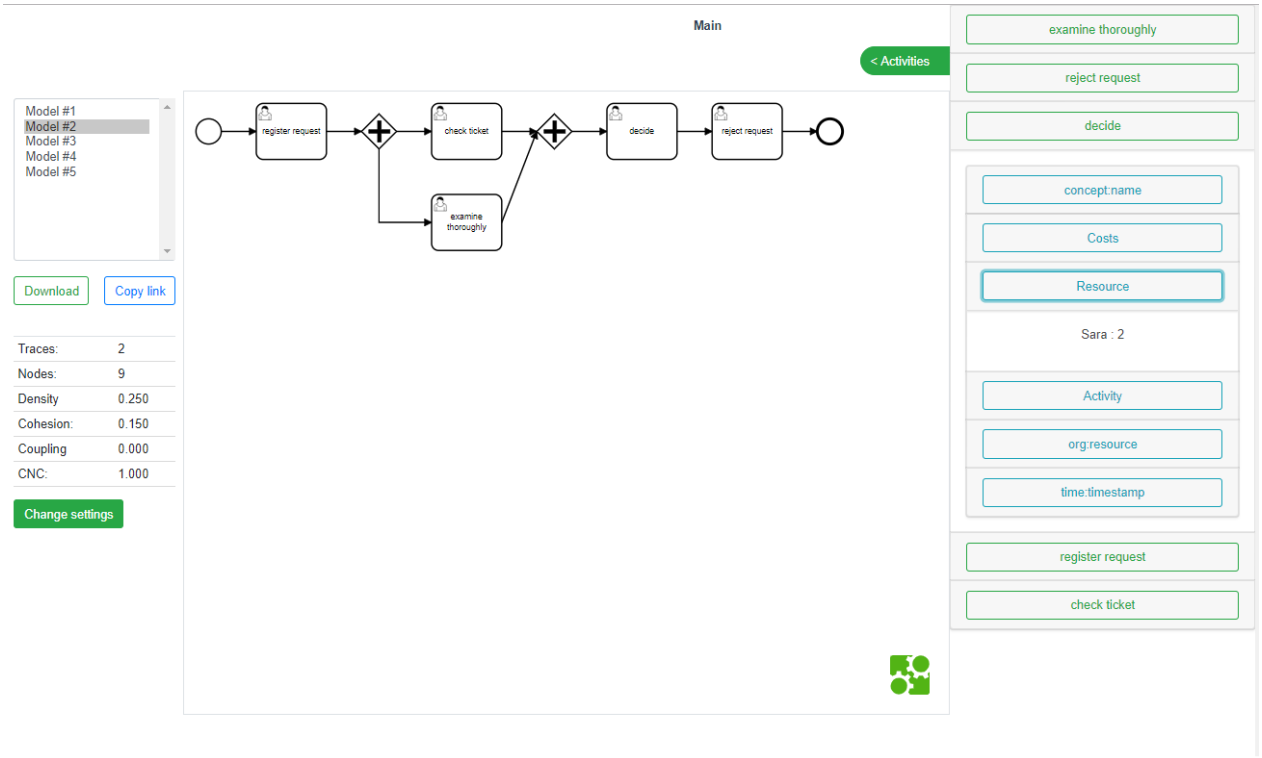


Figure 4.1.5 Screenshot of the model view page

As soon as all possible models generated, the application will redirect the user from the loading page to the model view page. A screenshot of the page can be seen in Figure 4.1.5. Model view page contains area with the displayed model which can be scrolled, zoomed in and out; list of the models that were generated; a table with statistical measures calculated for displayed diagram and a hidden tab with a list of activities. For each activity displayed in the tab, the user can find information about different resources related to the corresponding activity.

In order to switch between the models, the user may click on the model he or she would like to explore. Whenever a model from the list is clicked, diagram, statistical measures, and activity list will be changed.

4.1.3 Mobile Device Support

Another advantage of Vue.js is that application automatically adjusted for mobile devices usage. Providing mobile support increases application value for the user as it gives

him more flexibility in choosing suitable device for his work. Figure 4.1.6 shows a screenshot of an application on the mobile device.

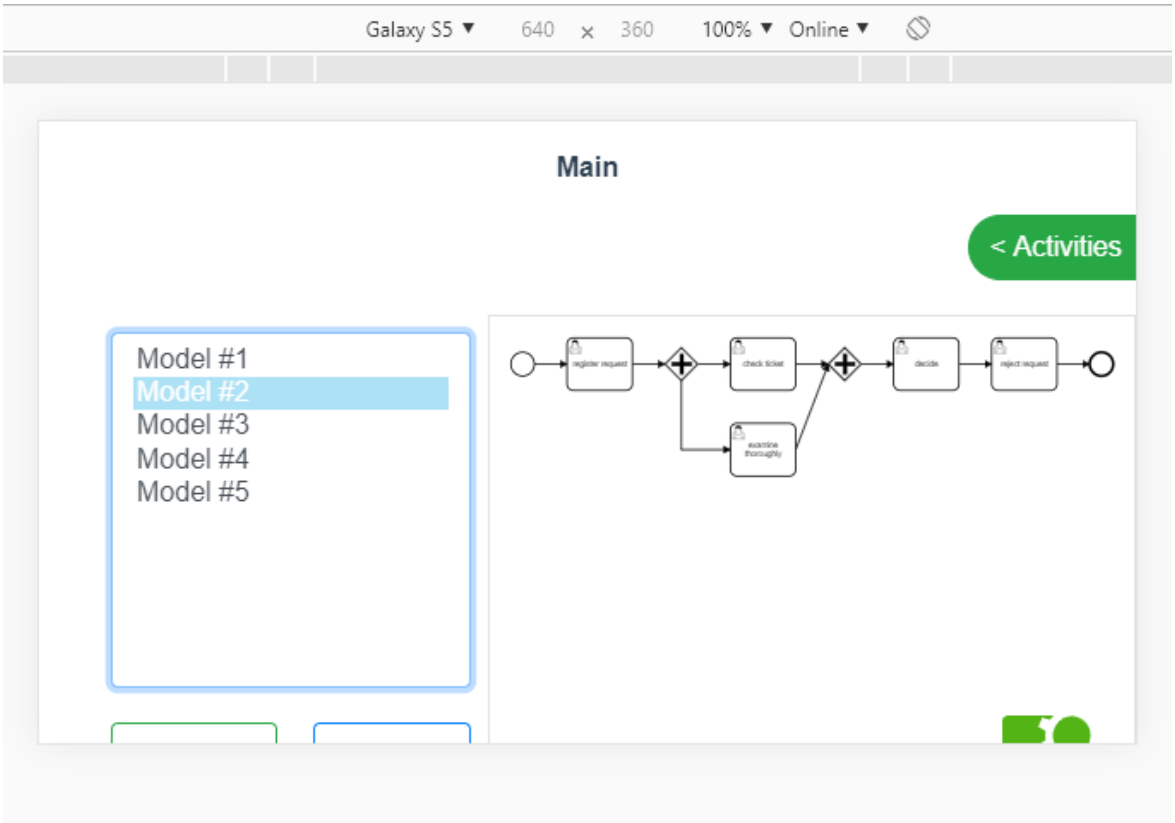


Figure 4.1.6 Screenshot of the model view page

4.2 Back-end Application

4.2.1 Back-end Architecture

For the back-end part of the application “Spring” framework was used. “Spring” is an application framework that was developed for the Java platform. It contains various features which can be used for developing web applications and supports both SQL and NoSQL databases.

Back-end application uses Model-View-Controller scheme that is commonly used for interface development. The idea of the architecture is to separate application into three modules: model, view and controller.

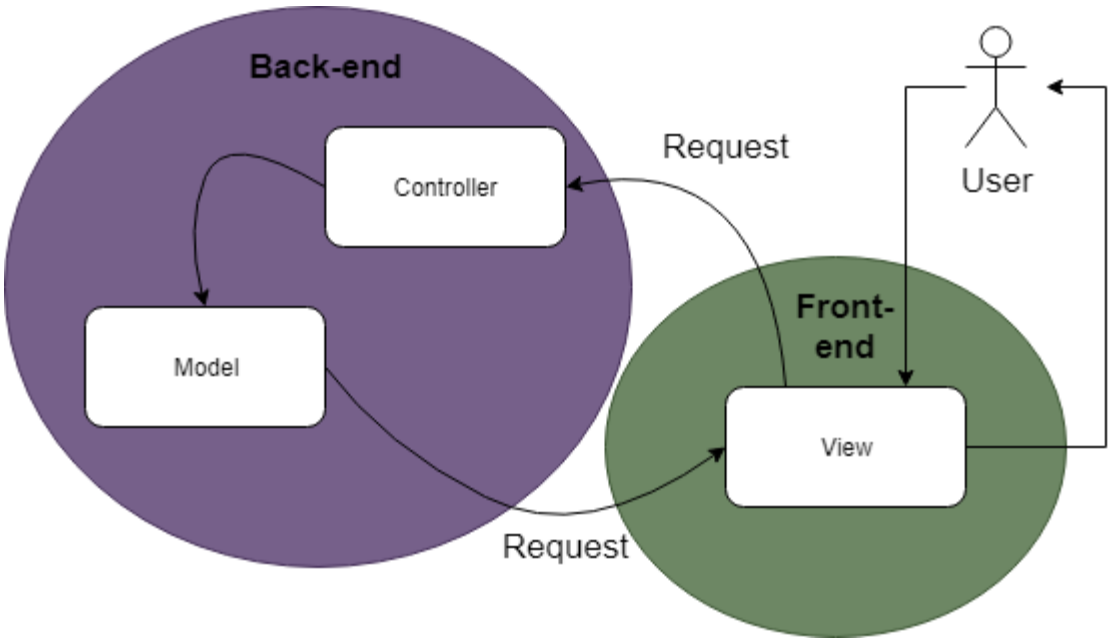


Figure 4.2.1 MVC scheme

Whenever the server receives a request, it uses a controller to select appropriate logic to process the request. Model part of the application is responsible for the data and work with the data: requests to the database and verifications. After all the necessary data was received; it is up to View part of the application to present data to the user on his demand. Both the controller and the model are stored on the back-end part of the application, while the view is realized as a separate server in the front-end part of the application. The complete architecture of MVC scheme can be seen in Figure 4.2.1.

In order to be able to easily update or change the application without problems, the code of the back-end part was separated into modules that are responsible for processing different user's requests and execute certain tasks.

The back-end can be separated into three main modules:

- Data loading module.
- Filtering module.
- Processing module.
- Exporting module.

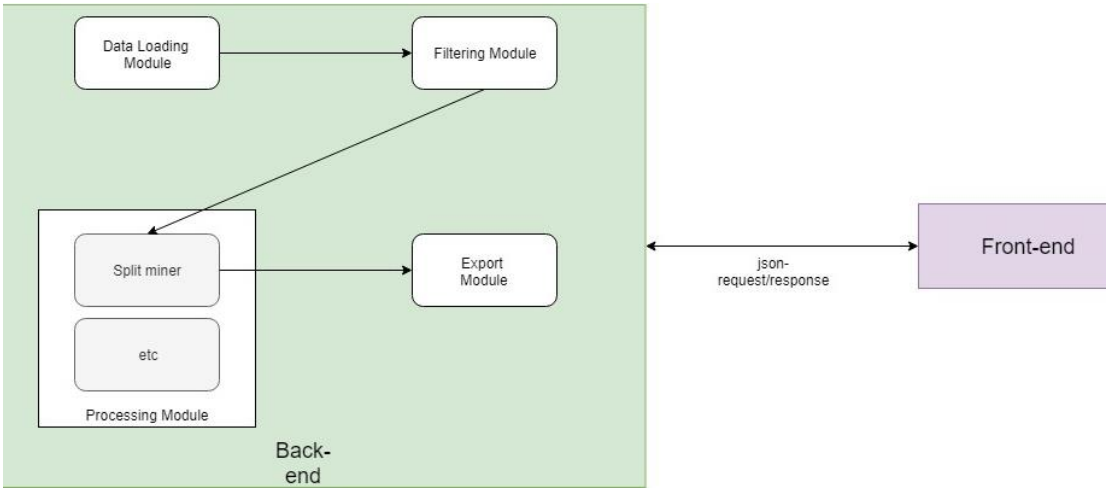


Figure 4.2.2 Architecture of the back-end

Each module executes different tasks and processes different requests. The entire architecture of the application can be seen in Figure 4.2.2.

4.2.2 Data Loading Module

The web application supports two extensions of the file that can be used in order to upload event logs:

- Comma-separated values (CSV)
- Extensible Event Stream (XES)

Files of the both extensions must contain information on events that were executed in the process:

- ID of the trace
- Name of the activity
- Resources
- Timestamp

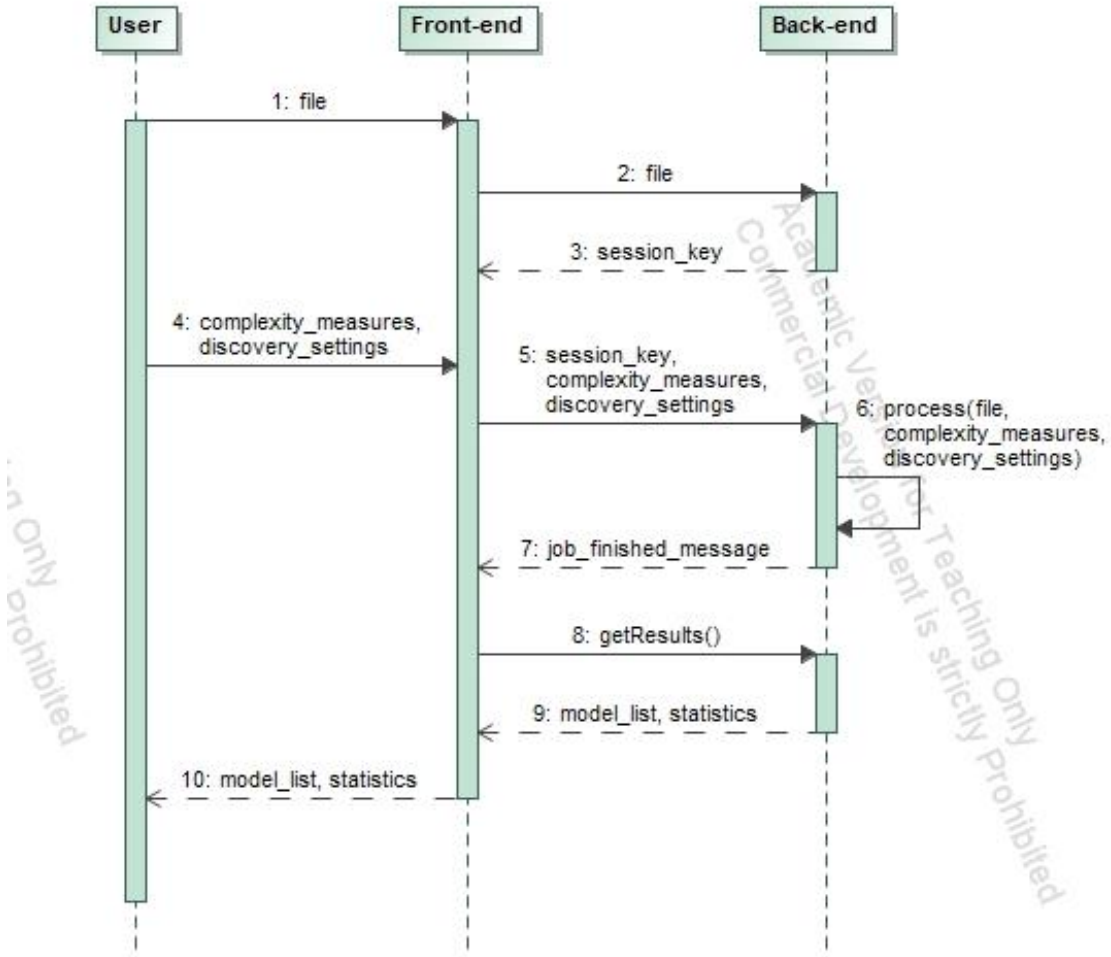


Figure 4.2.3 Example of use case

To use the application, the user loads a log file into the system through the front-end interface. Once the file is selected, it is sent to the back-end part where the type of the file is defined and corresponding library is used in order to parse and extract the first 6 rows of the file. After that, examples of the events are returned back to the front-end where in case of the CSV file user can specify which column represents activities, traces, and resources and forward desired selections to the back-end. The complete version of the use case can be seen in Figure 4.2.3.

If user uses CSV file, after columns types were specified, server will convert CSV file to XES to use the same library for the process exploration. This was made in order to simplify the process of adding new algorithms for process generation into the application. For successful file transformation, csv file should have next columns specified:

- Trace id;
- Activity;
- Resources;
- Timestamp.

After user has verified the content of the file, the application proposes to select discovery type to filter traces that will be used in the model generation and specify desired metrics threshold for model segmentation. In addition, it is also possible to replace default algorithm's parameters if needed.

Complexity measures that user can use to set complexity of the models are the following (see description in the section 3.3):

- Number of nodes.
- Density.
- Cohesion.
- Coupling.
- Coefficient of network connectivity (CNC).

Whenever the value of one of the specified complexity measures is reached, selected algorithm stops processing traces and returns models that were generated. Specifying these measures is not required, and in case all measures were ignored by the user, algorithm will process all traces and return one model.

4.2.3 Processing the Event Log

The processing module responds to the main functionality of the system. To be more precise, it generates the business process model based on the provided event logs and complexity measures defined by user. Aside from generating a traditional business process model based on the frequency of the traces in event logs, the user may specify one of the following types of discovery process (see description in the section 3.2):

- Business process model based the time taken, starting from the first activity till the last one.
- Business process model based on provided resources.
- Business process model based on artifacts involved.
- Business process model based on the longest or the shortest traces.
- Business process model based on endpoints. Selects traces that has specified by user starting and ending activities.

Another noticeable feature of the application is that complexity measures can be used as a separator between models. For example, in case the user decides to generate a business process model based on the frequency of the traces in the event log, the application may generate several models. The first model is generated from the part of the event logs including the traces with the highest frequency when the model reaches one of the thresholds specified by one of the complexity measures, the algorithm stops and saves this model as output. However, as there are records in the event log that were not processed, the system will start generating another model, which will not exceed the complexity measures. This loop is repeated until all records in the event log are processed.

To define the highest amount of the traces needed to satisfy complexity measures, dichotomy search method is used. Total amount of traces is divided by half and the process model is generated. Complexity measures of the model are compared to the defined by user values and decision is made. It is either to select first half of the traces and continue search of optimal amount there. Or take the second part. Or just add created model to the list in case it satisfies complexity measures and adding more traces will exceed value of the measures defined by the user.

Another major feature of the application that is worth mentioning it is ability to specify algorithm that has to be used in order to process event logs and generate the set of the process models. In the current version of the application the following algorithms are supported:

- Split miner. Split miner is an algorithm that builds the model based on traces from event logs and filters out the least frequent connections between activities. One noticeable feature is that it is possible to define gateways in the model by using this algorithm and another feature is its high speed of processing, which is why it can be considered as one of the best algorithms.

As the tool was designed to be a web application, data extraction and processing are executed on the server. By designing tool to be this way the user's device is released from complex computations and operations that allow users to execute other tasks and activities in background. However, there are also downsides of this solution.

The most critical issue is that it may take a lot of time to process event logs provided by one user, and this can create waiting time for another user. Furthermore, the system may be immersed into a frozen state, not allowing any actions from the user's side, unless processing on the back end is done and models are generated. In order to solve this issue, asynchronous requests are used: whenever a new request to process data is received, the system creates a separate thread. This way the waiting time for the processor to be free is terminated and provides the user with the ability to continue using the system when the event log is processed.

4.2.4 Data Storing and Manipulation

Properly storing data that is used in the application is an important matter that should be approached, as this determines how exactly the application works and which functions it will be able to execute. A poor design of data storage can lead to problems with the development of the application and high execution times. In particular, this applies to a web application, as it has shared resources and data across multiple users. This requires control of the access to the particular information that was requested by a specific user and separation of the stored data.

Unfortunately, there is no predefined structure for even log file: it may contain activities along with several resources and that makes it impossible to calculate the exact amount of columns that should be stored in the table. One possible solution to solve this issue would be storing each row that has its own activity, trace and a list of resources, but

it will make database hard to read, as rows from different event logs could be mixed in the same table.

On the other hand, in terms of application, there is no need to access separate parts of the event log from the client side. That is why the best solution would be to store records of event log as they are, in the file and add a link to the database. Of course, reading from the file takes more time than from the database, but it gives a big advantage when it comes to large storages of the data. Searching for all rows that are related to specific event log may take some time; however, if we store only the link instead, we will search only for the one row that contains this link.

Another downside of the selected solution is that storing event log files may take a part of the server's memory and eventually lead to running out of the space. Even though files are not big, continuous usage of the service will lead to huge amount of this files and filled memory. To solve this, data cleaning is required. If files are deleted regularly, there will not be any problem with memory.

4.2.5 Export of the Results

The ability to export the data helps user to execute further analysis or share the results. As models are stored as files, it is easy to export them.

Models can be exported in BPMN format. Files with extension BPMN were developed by Business Process Management Initiative and can be used to graphically represent BPMN models. They are widely used in different process mining tools, and ability to export files in this format provides additional value to the tool.

5 Evaluation and Validation

The following section contains an evaluation of the tool describing the process and presenting the results. In order to verify application capabilities, we check how the algorithm that generates segmented models works as well as the quality and complexity of the models that were generated.

The evaluation process is structured as follows:

1. First, we load the event log and read its properties.
2. Then we generate the process model that describes all traces in the event log.
3. In the next step, we set complexity measures and generate segmented process models.
4. In the end, we perform conformance analysis to verify that segmented models have the same correspondence to the log as the process model that was generated based on all process variants.

All results are presented in the tables.

5.1 Conformance Metrics

Conformance analysis is a process mining technique that is used to compare a process model with the event log. In our case, as we generate the model based on the event log, the model based on all traces variance should satisfy majority traces in the log. This way we can prove tool usefulness if the segmented models that are generated can also satisfy majority traces from the event log.

To evaluate the algorithm, we use the following metrics:

- Execution time. Amount of the time taken for the algorithm to execute and to generate the process models.
- Fitness. Shows the rate of the traces from the log that can be reproduced using the model.
- Precision. Shows the rate of how much there is behavior that exists in the model and does not exist in the log.

- Generalization. Shows the rate of how well the model explains behavior that does not exist in the log.

5.2 Data

For tool evaluation, real-life data was used. Data was taken from the data archive of 4TU.Center and comes in the form of tree files of different BPI challenges conducted in the following years: 2012, 2013 and 2019. All event logs have XES file format.

5.3 Results

5.3.1 BPI Challenge of the 2013 Year

The aim of this experiment is to use a light-weight real life event log to verify application correct work.

The BPI challenge file of the 2013 year contains event logs of Volvo IT incident and problem management. The total amount of the traces in this log equals to 1487. After the log upload, the application has defined the following fields:

- concept:name (of the trace) – trace;
- time:timestamp – timestamp;
- org:group – other;
- resource country – other;
- organization country – other;
- org:resource – resource;
- organization involved – other;
- org:role – other;
- concept:name (of the event) – activity;
- impact – other;
- product – other;
- lifecycle:transition – lifecycle.

Model	Complexity measures					Traces variants	Generation Time	Fitness	Precision	General-ization
	N	D	Ch	Co	CNC					
All traces	12	0.24	0.22	0	1.33	183	1.41s	0.88	0.88	0.99
S1	12	0.23	0.25	0	1.25	92	5.9s	0.88	0.86	0.99
S2	10	0.27	0.28	0	1.2	18		0.66	0.95	0.99
S3	11	0.24	0.22	0	1.18	2		0.77	0.89	0.99
S4	12	0.23	0.25	0	1.25	18		0.66	0.9	0.99
S5	12	0.23	0.25	0	1.25	6		0.6	0.84	0.99
S6	12	0.23	0.25	0	1.25	9		0.87	0.74	0.99
S7	12	0.23	0.25	0	1.25	19		0.88	0.7	0.99
S8	12	0.23	0.25	0	1.25	10		0.88	0.86	0.99
S9	10	0.27	0.28	0	1.2	8		0.86	0.88	0.99
S10	8	0.32	0.31	0	1.13	1		0.86	0.98	0.99

Table 5.3.1. Evaluation results for BPI challenge 2013

As the next step, we select discovery method by the trace frequency and do not specify any complexity measures and generated the model taking into the account all traces variants from the log. The model generation took 1.41 seconds. As result, we received the model (All traces) described in Table 5.3.1.

After the generation of the model based on all traces variants, we decided to use the same settings and only decrease the value of the CNC (1.25) for segmented models. Model generation took 5.9 seconds.

After the segmented models were generated we added them to the table (S1-10) with their complexity measures and calculated conformance metrics for each of them separately. The results of the evaluation can be seen in Table 5.3.1.

The model with all traces variants can be found in the Appendix 1. Examples of the segmented models (S1, S2) can be found in the Appendix 2 and 3.

Model	Complexity measures					Traces variants	Generation Time	Fitness	Precision	General-ization
	N	D	Ch	Co	CNC					
All traces	113	0.037	0.126	0	2.04	4366	9.64s	1	0	0
S1	100	0.035	0.1	0	1.75	940	35.05s	0.99	0.46	0.99
S2	100	0.036	0.1	0	1.8	622		0.86	0.44	0.99
S3	106	0.034	0.1	0	1.8	406		0.87	0.43	0.99
S4	104	0.035	0.1	0	1.79	676		0.88	0.46	0.99
S5	107	0.034	0.1	0	1.79	521		0.82	0.42	0.99
S6	100	0.036	0.09	0	1.77	331		0.87	0.5	0.99
S7	104	0.035	0.1	0	1.8	379		0.86	0.4	0.99
S8	100	0.036	0.1	0	1.79	380		0.88	0.46	0.99
S9	96	0.037	0.09	0	1.75	110		0.81	0.52	0.99
S10	30	0.076	0.058	0	1.10	1		0.59	0.88	1

Table 5.3.2. Evaluation results for BPI challenge 2012

5.3.2 BPI Challenge of the 2012 Year

The aim of the second experiment is to take a big even log and slightly decrease the complexity of the initial model and see how conformance metrics will change.

The BPI challenge file of the 2012 year contains an event log of a loan application process. The total amount of the traces in this log equals to 13087. After the event log was loaded, the tool defined the following attributes of the event:

- concept:name (of the trace) – trace;
- time:timestamp – timestamp;
- lifecycle:transition – lifecycle;
- concept:name (of the event) – activity.

We repeat the previous experiment structure. We set discovery method to the most frequent traces and leave all complexity measures empty. The generation of the first model that is based on all traces variants took 9.64 seconds. All complexity measures of the generated model were added to the Table 5.3.2.

To generate the segmented process model we used the same settings and only decreased the value of CNC (1.8) and as result we received 10 process variants. The model generation took 35.05 seconds for all models together. After calculation of the conformance metrics, all models were added to the result Table 5.3.2 (S1-10).

5.3.3 BPI Challenge of the 2019 Year

The aim of the third experiment is to take a large event-log and decrease the complexity of the model as much as possible to see the changes in values of the conformance metrics.

The BPI challenge of the 2019 year contains an event log from a large multinational company making sales in the area of coatings and paints from The Netherlands. The total amount of the traces in this log equals to 31509.

The following properties of the evens we defined after the event log was loaded into the application:

- concept:name (of the trace) – trace;
- concept:name (of the event) – activity;
- time:timestamp – timestamp;
- org:resource – resource;
- User – other;
- Cumulative net worth (EUR)- other.

We have left the complexity measures fields empty and selected discovery by the most frequent trace. The generation of the model using all traces took 52.08 seconds. After the generation, all model's complexity measureas were added to the Table 5.3.3 (All traces).

As the next step we used the same discvoery type and set a value of CNC metric to 1.15 and generated the segmented models. After algorithm execution with defined complexity measures, 4480 models were generated. For the evaluation first 10 models were taken. The result can be seen in the in Table 5.3.3 (S1-10)

The model with all traces variants can be found in the Appendix 4. Examples of the segmented models (S1, S2) can be found in the Appendix 5-6.

Model	Complexity measures					Traces variants	Generation Time	Fitness	Precision	General-ization
	N	D	Ch	Co	CNC					
All traces	70	0.04	0.04	0	1.41	251734	52.08s	0.77	0.97	1
S1	15	0.16	0.14	0	1.13	5	1.2h	0.82	0.9	1
S2	16	0.15	0.17	0	1.12	4		0.77	0.98	1
S3	15	0.16	0.2	0	1.13	9		0.76	0.92	1
S4	17	0.14	0.14	0	1.125	6		0.73	0.95	1
S5	16	0.15	0.14	0	1.125	4		0.66	0.96	1
S6	13	0.18	0.15	0	1.07	3		0.56	0.98	1
S7	15	0.16	0.14	0	1.13	3		0.71	0.98	1
S8	20	0.12	0.13	0	1.15	6		0.31	0.76	1
S9	13	0.18	0.14	0	1.07	2		0.67	0.97	1
S10	11	0.2	0.14	0	1	2		0.6	1	1

Table 5.3.3. Evaluation results for BPI challenge 2019

5.3.4 Summary

By looking at the results we can conclude that there is no significant difference in fitness, precision, and generalization between the model generated from all traces variants and segmented models. All segmented models except a few have nearly the same fitness, precision, and generalization values as the model with all traces variants.

Compared to execution time from the BPI challenge 2013, BPI 2012 and BPI 2019 consumed more time; however, it is still within expected calculations given the log sizes.

Evaluation of the generated models showed that segmented models are more useful for the user than the model build using all trace variants, as segmented models are simpler and lose only little in terms of correspondence to the event log.

6 Related Work

The following section provides an overview of the existing process mining tools, their capabilities, features, and disadvantages. There are many process mining tools that can be used for the automated process discovery; however, they take the whole event log as an input and produce complex, spaghetti-like models making it hard for the user to analyze them.

According to B.F. van Dongen et al [3], ProM framework is an open-source framework with all kinds of plugins developed by the community. The ProM framework is capable of generating process models in different forms like BPMN or Petri Net. The generated models can be used as input for the different actions defined buy community-created plugins, for example, conformance checking.

Another advantage of the ProM framework is that it is capable of dealing with large event log files, and sorting or filtering events before the model discovery process. However, ProM framework has no possibility to omit certain traces or set the desired complexity of the generated model. The screenshot of the ProM framework can be found in Figure 6.1.

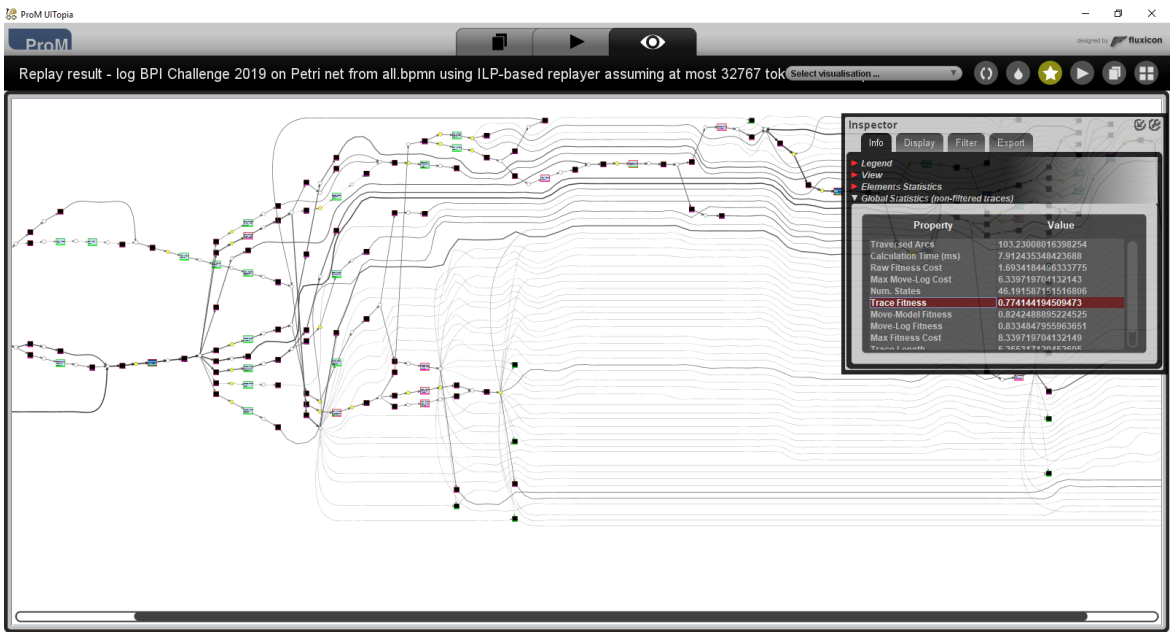


Figure 6.1. A screenshot of the ProM framework

Another process mining tool that is capable of the filtering the logs is Disco. Disco is a process mining tool that is capable of creating process maps based on the event log. The screenshot of the Disco can be seen in Figure 6.2.

The main advantage of the Disco is a broad range of the log analysis features that help in the process model exploration:

- The time frame filter – user can select a certain period of the time from the log for the process analysis.
- Variation filter - helps to filter either the most common or exceptional behavior in a process.
- Performance filter – allows specifying the performance criteria for the cases.
- The endpoint filter – uses specific events as start and the end of the process.
- The attribute filter – allows selecting events based on the resources or activity.

Unfortunately has two main disadvantages when it comes to process map generations. First of all, it does not display the splits or gateways in the map-making it impossible for the user to define the behavior is some places of the process. Th second is a disadvantage is that it can not build a model based on a set of the trace variants. Only separate variants are present in the variation filter.

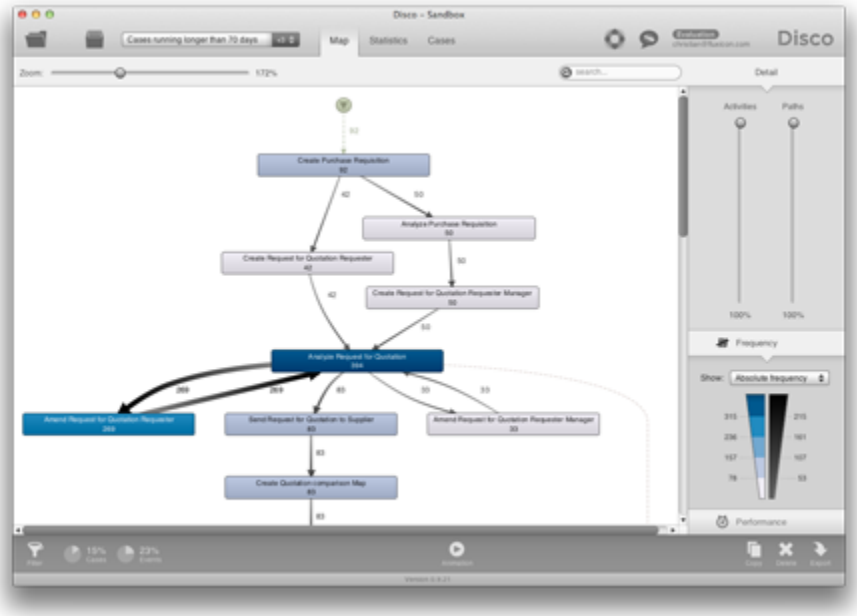


Figure 6.2. A screenshot of Disco taken from the Fluxicon site

APROMORE is an open-source process analytics platform designed for different process mining functionality including process discovery. According to [16] APROMORE is capable of evaluation, ranking, designing and presenting the models. With APROMORE it is possible to build a process model based on the event log, evaluate it with different features and even calculate its complexity measures. However, there is no possibility of sorting or filtering the initial event log, and it is impossible to create segmented models based on the traces variants that are placed in the same log.

Another process mining tool that is considered as one of the best process mining tools is Celonis. This tool contains a basic set of features that make possible to load the log into an application and process it. When it comes to the process discovery Celonis has a lot of common with Disco tool. Namely, it generates a fuzzy process map that can be analyzed with the help of different charts and filters. However, this tool is inefficient when it comes to the process segmentation.

The last tool that was reviewed for its process discovery capabilities is called ARIS developed by SoftwareAG. According to the website, ARIS has wast amount of features including process mining. The tool is capable of creating the process models in the BPMN format based on the provided event log and provide an overall statistics, however, it can not evaluate model complexity and produce segmented models based on the traces variants. The screenshot of the tool can be fount in Figure 6.3.

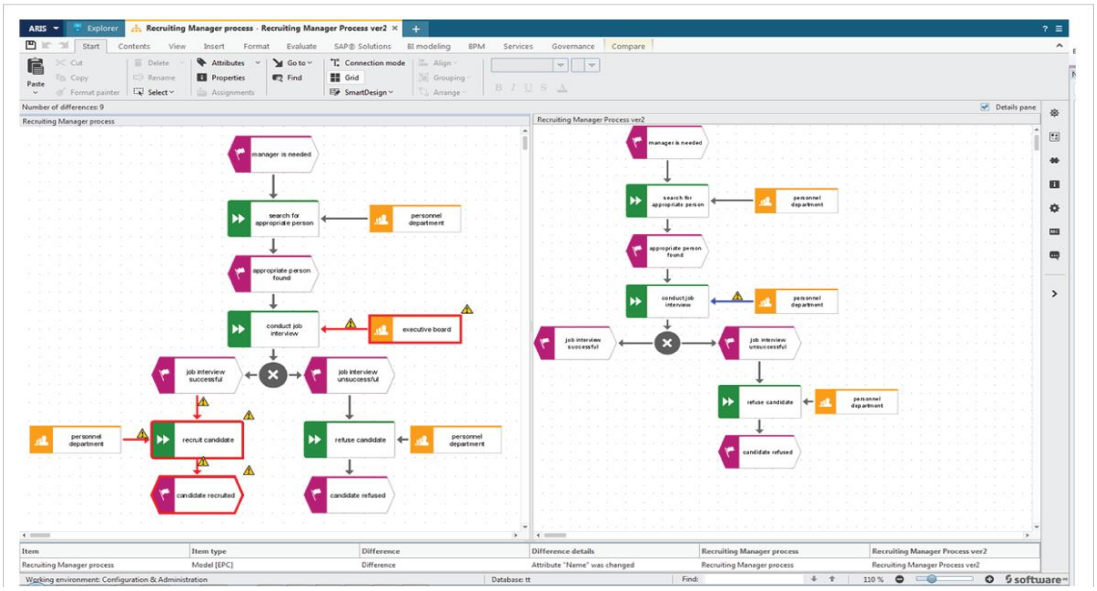


Figure 6.3. A screenshot of ARIS taken from the developer site

7 Conclusion

This thesis proposes an approach for the process discovery from the event logs that can produce simple human-understandable process models. The idea behind the approach is to use a complexity measures to evaluate and adjust process model complexity. To support this approach, various complexity measures and process mining algorithms were researched as well as existing mining tools. The research showed that all existing tools can build a process model or process map from the software log, however, they cannot change the model's complexity or generate segmented models.

The final product of this master thesis is a web-based application that is capable of generating segmented process models from the event logs. Using this application user can set the desired complexity of the model and select how exactly the event log will be processed by specifying discovery type. Based on the specified complexity metrics and discovery method a list of segmented models is generated along with a short statistics on the event resources.

The evaluation of the tool showed that it is capable of dealing with large event logs and produces the results that can be used in the process management instead of the complex models.

There are several ways to improve the application for future use. First of all, it is adding new complexity measures and algorithms. More options make the process of the model discovery more agile and more precise. The application can also be improved by adding more features for activities and resource exploration. Helping user to explore the model that he has generated without a need for support of the third-party tools will improve application value. Another way of improving would be making the application interface more responsive. For example, displaying the progress of the model generation and time left for the process to finish to the user.

8 References

[1] Alfredo Bolt, Massimiliano de Leoni, Wil M. P. van der Aalst: A Visual Approach to Spot Statistically-Significant Differences in Event Logs Based on Process Metrics. CAiSE 2016: 151-166.

[2] N. R. T. P. van Beest, Marlon Dumas, Luciano García-Bañuelos, Marcello La Rosa: Log Delta Analysis: Interpretable Differencing of Business Process Event Logs. BPM 2015: 386-405.

[3] van Dongen B.F., de Medeiros A.K.A., Verbeek H.M.W., Weijters A.J.M.M., van der Aalst W.M.P. (2005) The ProM Framework: A New Era in Process Mining Tool Support. In: Ciardo G., Darondeau P. (eds) Applications and Theory of Petri Nets 2005. ICATPN 2005. Lecture Notes in Computer Science, vol 3536. Springer, Berlin, Heidelberg

[4] van der Aalst, W.M.P., Schonenberg, M.H., Song, M.: Time prediction based on process mining. Information Systems36(2) (2011) 450 – 475 Special Issue: Semantic Integration of Data, Multimedia, and Services.

[5] Adriano Augusto, Raffaele Conforti, Marlon Dumas and Marcello La Rosa: Split Miner: Discovering Accurate and Simple Business Process Models from Event Logs. 2017

[6] Nielsen, M., Plotkin, G.D., Winskel, G.: Petri Nets, Event Structures and Domains, Part I. TCS13. 1981. 85–108

[7] van der Aalst, W.M.P., Weijters, T., Maruster, L.: Workflow mining: discovering process models from event logs. IEEE TKDE16(9) (2004) 1128–1142

[8] J.C.A.M. Buijs and H.A. Reijers: Comparing Business Process Variants using Models and Event Logs. 2014: 154-168.

[9] W.M.P. van der Aalst: Business Alignment: Using Process Mining as a Tool for Delta Analysis and Conformance Testing. 2005

[10] Irene Vanderfeesten, Jorge Cardoso, Jan Mendling, Hajo A. Reijers, Wil van der Aalst: Quality Metrics for Business Process Models. 2007

[11] Hajo A. Reijers and Irene T.P. Vanderfeesten: Cohesion and Coupling Metrics for Workflow Process Design. 2004

[12] Antti M. Latva-Koivisto: Finding a complexity measure for business process models. 2001

[13] H.A. Reijers: A Cohesion Metric for the Definition of Activities in a Workflow Process. 2003

[14] Albana Roci et al: A Polynomial-Time Alpha-Algorithm For Process Mining. 2018

[15] Andrea Burattin, Alessandro Sperduti, Wil M. P. van der Aalst: Heuristics Miners for Streaming Event Data. 2012

[16] Marcello La Rosa, Hajo A. Reijers, Wil M.P. van der Aalst, Remco M. Dijkman, Jan Mendling, Marlon Dumas, Luciano García-Bañuelos: APROMORE: An advanced process model repository. 2011

9 Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, Yevhen Dorodnikov

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Dealing With Complexity In Process Model Discovery Through Segmentation,

   supervised by Fabrizio Maria Maggi and Fredrik Payman Milani.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

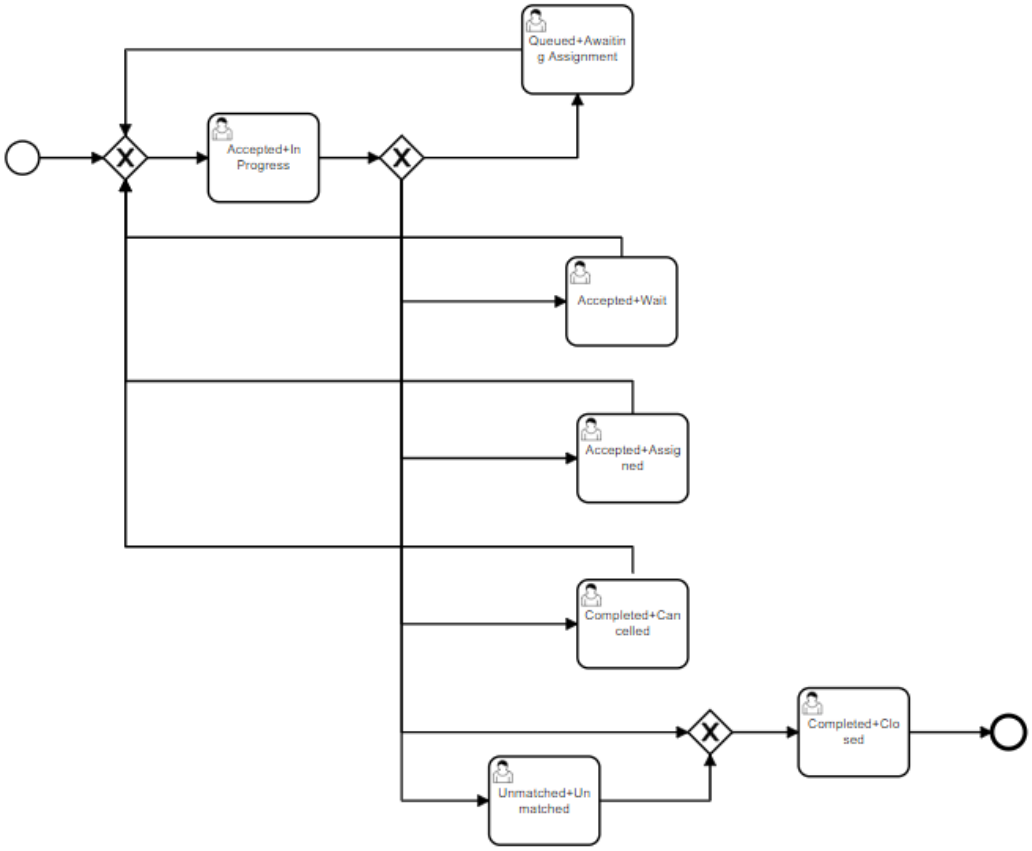
4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Yevhen Dorodnikov

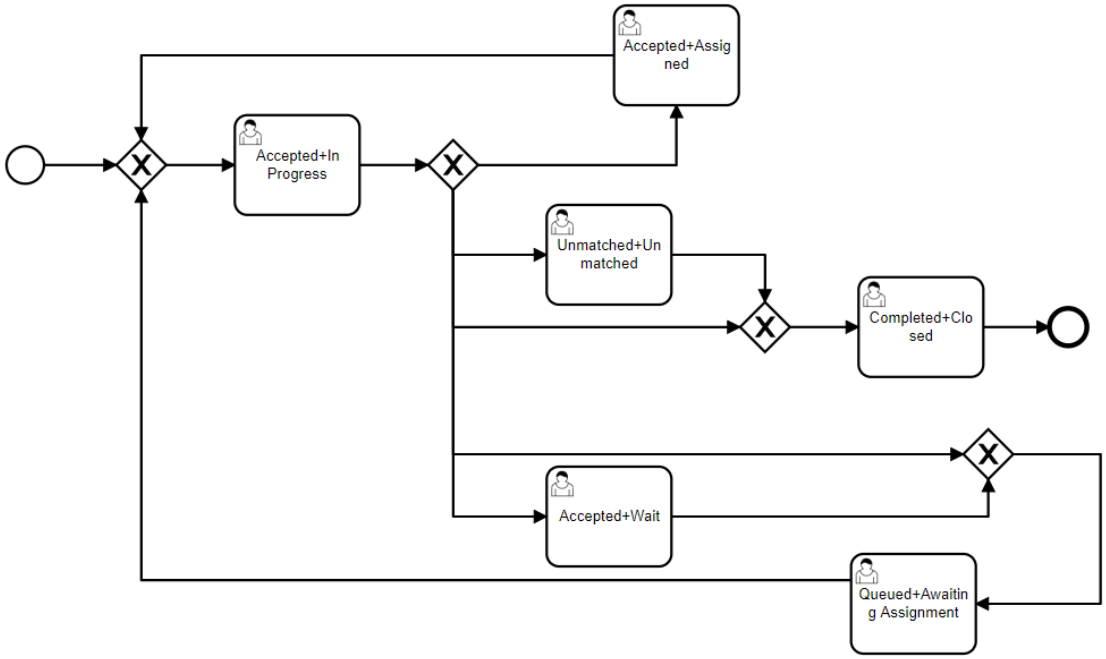
10/08/2019

Appendix

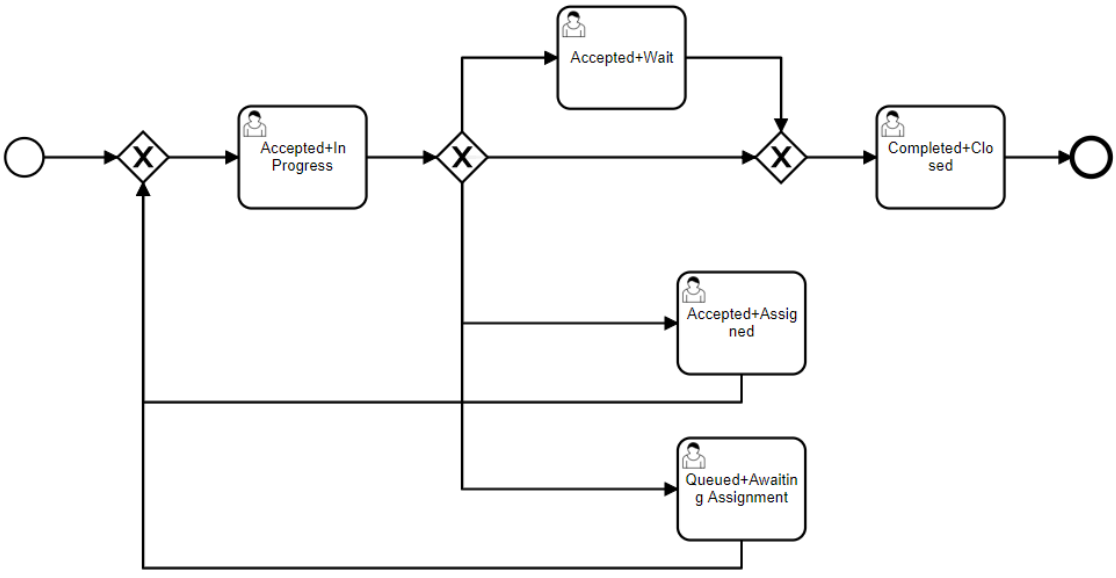
Appendix 1



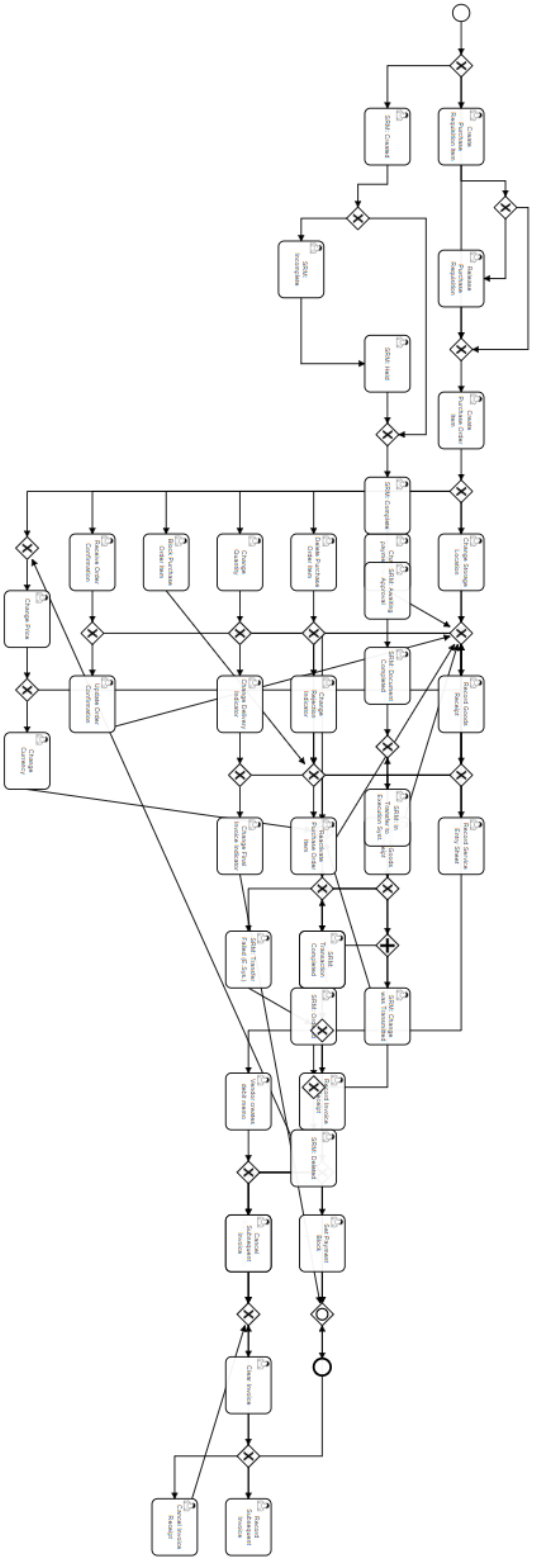
Appendix 2



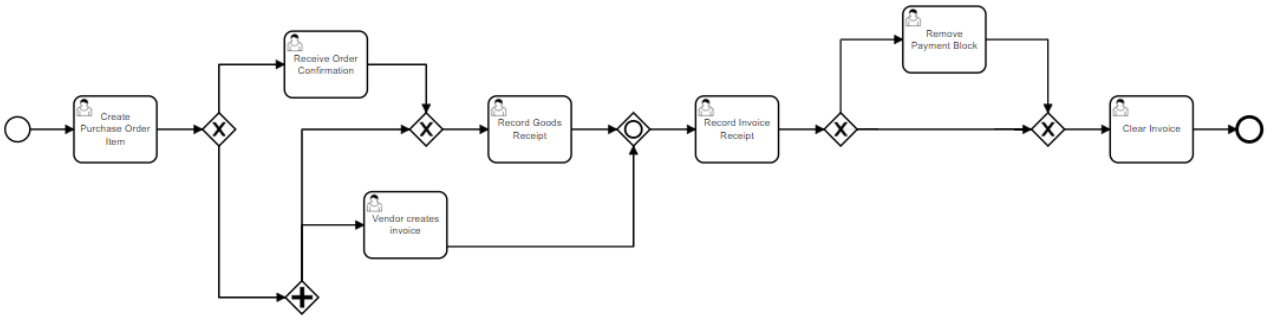
Appendix 3



Appendix 4



Appendix 5



Appendix 6