Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelor thesis

David Karwehl

Use case based introduction to process mining and current tools

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

David Karwehl

Use case based introduction to process mining and current tools

Bachelor thesis eingereicht im Rahmen der Bachelorpüfung

im Studiengang Bachelor of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Ulrike Steffens
Zweitgutachter: Prof. Dr. Stefan Sarstedt

Eingereicht am: July 11, 2018

David Karwehl

Thema der Arbeit

Use case based introduction to process mining and current tools

Stichworte

Data Science, Process-Mining, Prozessmodelle

Kurzzusammenfassung

Prozess-Mining beschreibt eine Sammlung von Techniken die Ereignisdaten nutzen um wertvolle Einblicke in Prozessabläufe zu geben. Mit Prozess-Mining können unter andererem Prozessmodelle automatisch generiert und Leistungsmetriken erzeugt werden. Die Zusammenarbeit von Menschen in einem Prozess kann veranschaulicht werden um heraus zu finden, wie gut diese zusammen arbeiten und wie die Leistung individuell ist. Die Konformität von Ergeignisdaten und Prozessmodellen kann überprüft werden um heraus zu finden ob Vorgaben eingehalten wurden, oder in welcher Instanz dies nicht getan wurde. Diese Bachelorarbeit wird die wichtigsten Prozess-Mining-Techniken vorstellen und in Fallbeispielen anwenden, die auf realen Lebensereignisdaten basieren. Drei Prozess-Mining Werkzeuge, ProM, Disco und Celonis werden vorgestellt und in späteren Kapiteln verwendet werden um die wichtigsten Prozess-Mining-Techniken anzuwenden.

David Karwehl

Title of the paper

Use case based introduction to process mining and current tools

Keywords

Data Science, Process Mining, Process Models

Abstract

Process mining is a set of techniques that use event data to provide valuable insights into processes. The techniques can be used to mine process models, and provide performance information. They can also be used to analyze how people in those processes are working together and how they perform. The conformance of a process model and an event log can be checked to analyze wether guidelines in the process were followed. This thesis will introduce the most important process mining techniques and apply them to uses cases that are based on real life event data. Three process mining tools, ProM, Disco and Celonis, will be introduced and used to apply process mining techniques.

Contents

List of Figures

1. Introduction

1.1. Introduction to Process Mining

Process mining is a set of techniques that aim to provide insights into a process using event data.

With today's systems in the business landscape a wide variety of data is available at companies. The price of data storage has dropped significantly the last decades, allowing more data to be stored in databases. Every event can be tracked, and logged in a database, and huge amounts of data can be stored because data storage became so cheap(van der Aalst, 2014). Automated steps in a process made logging of events even more relevant, because the added effort of logging the event is negligible, and logging is very important to understand what happened if something went wrong.

Process mining aims to make use of the available data to provide insights into the process. With complex processes that include up to hundreds of activities and resources that execute these activities, it is easy to lose track of what is actually happening in the process. Furthermore, it's easy to miss things going wrong, or not as intended, when a process is so complex and highly automated. To keep track of processes, process models are widespread and a useful tool. However, creating a process model and keeping it updated can be very expensive and time consuming. Especially when the process model is created without consulting all groups involved in the process, it is possible to miss less frequent, important activities in the process, or get the sequence of activities wrong.

Process mining can help to provide insights into what is actually happening in the process. It provides a wide variety of opportunities ranging from process discovery to advanced prediction algorithms that can be applied while the process is live. More advanced techniques like operational support will not be applied in this thesis. However even the more basic steps like process discovery can provide valuable insights. Process discovery is most useful, if a process model does not exist already, or if the one that exists is possibly outdated or incomplete.

Event data that is already present in IT-systems is used to create a bottom-up process model. This ensures that the created model is based on the activities that actually took place and on

the order they happened in. If a process model is already present, the events can be replayed on the data, to check wether the process model is representing the process correctly. If the process model can be considered valid, the events can be replayed to see if there were any activities that violated the rules of the process.

The use cases for process mining are not limited to analyzing or creating a process model. Given the right data, process mining can provide insights into the performance and relations of resources in the process. The way people work together to complete a complex task can be visualized. In a partly automated process there would be machines working together with people. process mining can help to provide performance statistics that aren't limited to a single task. With basic business management tools it is easy to find out who can complete the same task the quickest in a process. This is possible in a very effective way with process mining, but further than this, it is possible to compare the performance of different instances of the process, that resources were involved in.

As an example, Linda in accounting is always able to get a task done very quickly, and normally one would assume that Linda is doing a good job. But if Linda did this by cutting corners, this will likely require more work later on. Automated analysis focusing on the resources in a process can provide this kind of insight, and much faster and efficient than digging through the data manually. The opportunities that process mining provides will be described further, and in more detail, in chapter 2.

Process mining is not limited to the application in companies though. Application is possible in every field that provides the right data, and where modeling a sequence of activities as a process model makes sense. And there's plenty of data out there already and even more to come. While companies gather the data required for process mining very consciously, people generate more data everyday than they might realize.
A big part of this data is generated in the Internet of People. The internet of people includes social media like Twitter and Facebook and similar platforms. Here people share opinions, and facts about their lives, and while doing so they generate large amounts of data. This data can be combined with available location data that's being gathered in the internet of places, primarily by mobile devices. When combining this kind of data, it would be possible to analyze large events taking place like festivals, or a catastrophes(van der Aalst, 2016).

Another field where process mining can offer insights is healthcare. Hospitals need to offer a wide variety of services that can be simple, or very complex. More so, a procedure that was expected to be a simple one can turn into something very complex that takes a lot more time and resources than expected. If treatments of patients can be seen as processes, managing them efficiently requires insights into the processes, that process mining can provide.

Hospitals also often offer a good amount of data, because every step in treatment must be transparent and well documented. Treatment and patient records can be used to improve the processes. The process mining research of the Department of Mathematics and Computer Science Eindhoven University of Technology group around WMP van der Aalst has already analyzed the processes in a variety of healthcare organizations. And while these studies where merely explorative, they were able to provide some insights into missing steps in treatment of patients, and points of interest for further analysis(Mannhardt and Blinde (2017)).

Currently, not many universities are focusing on process mining. Most of the research about process mining was conducted at The Eindhoven University of Technology. Professor Wil van der Aalst is the most renowned researcher when it comes to process mining.

1.2. Motivation

With all the possible use cases and fields where process mining can be applied this thesis aims to give a brief introduction to important process mining techniques and apply these techniques in three process mining tools. The goal is to give examples of the insights process mining can provide and showcase the current state of process mining tools.

ProM, Fluxicon Disco, and Celonis will be introduced in chapter 3 and their most important features and key differences will be described. While the application of process mining may be possible in many fields, application to business processes is practical and very effective, so this thesis will focus on use cases in business processes. The most important process mining techniques will be applied to simple use cases that are based on real life event data. This will be done in ProM, Disco and Celonis to showcases the differences between the tools.

Almost all of the data sets available online are also from business processes. The best source for data sets is 4TU.nl. Synthetic and real life event logs are available. The aim is to base use cases on these real life event logs.

1.3. Glossary

1.3.1. Event Logs

The basis for the application of process mining techniques is an event log. Event logs can be extracted from a database, CSV-files and more data sources. At the least an event log requires the following information:

- A case ID
- An activity name
- A timestamp

The event log will contain many cases of a process and the case ID identifies which case each event belongs to. The activity name states which activity was executed in each event. The timestamp shows at what point in time the event takes place.

Case id	Event id	Properties				
		Timestamp	Activity	Resource	Cost	...
1	35654423	30-12-2010:11.02	register request	Pete	50	...
	35654424	31-12-2010:10.06	examine thoroughly	Sue	400	...
	35654425	05-01-2011:15.12	check ticket	Mike	100	...
	35654426	06-01-2011:11.18	decide	Sara	200	...
	35654427	07-01-2011:14.24	reject request	Pete	200	...
2	35654483	30-12-2010:11.32	register request	Mike	50	...
	35654485	30-12-2010:12.12	check ticket	Mike	100	...
	35654487	30-12-2010:14.16	examine casually	Pete	400	...
	35654488	05-01-2011:11.22	decide	Sara	200	...
	35654489	08-01-2011:12.05	pay compensation	Ellen	200	...

Figure 1.1.: Table showing a fraction of an event log

Figure 1.1 shows a fragment of an event log. Each line represents an event. This table is from van der Aalst (2016). Additional data in event logs improve the results of some process mining techniques and enable the application of others. For example performance analysis is more effective if the event log contains both a start and an end timestamp. Resource information in event logs enables resource based analysis. More on what information that can be added to event logs, and how this can improve process mining results is described in van der Aalst (2016).

1.3.2. Process Models

Process models can be used to describe the flow of a process. They can show how things should, must be done or they way they were done. A process model that is discovered by process mining techniques will showcase how things were done. Conformance checking techniques can use a model, that describes the flow every process instance must follow to find violations that took place. Depending on the use case process models can be informal or follow a well defined notation. This thesis will include informal process models that are mined by process discovery techniques in Disco and Celonis. The informal process models in Disco are named process maps. This thesis will also include Petri nets as Petri nets are used as the primary notation for models in ProM. BPMN models will also be included because Celonis requires a BPMN 2.0 model as input for conformance checking.

Process models mined in disco are called process maps, Celonis calls them process graphs. In the rest of this thesis both of these may be referred to as process models to improve readability.

2. Process mining techniques

Process mining covers multiple techniques, all operating on either an event log alone, or in combination with a process model. Process mining techniques can be divided into the following three classes.

- process discovery
- conformance checking
- operational support

The techniques in these three families differ in their respective relationship between an event log and a process model as shown in the lower half of graphic 2.1

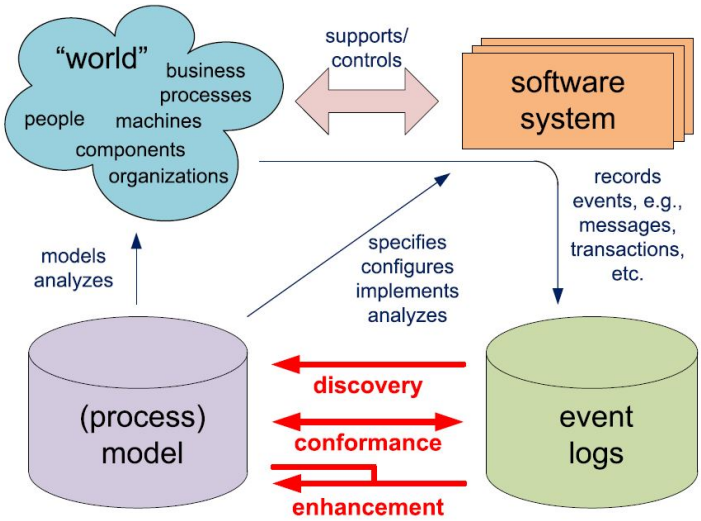

Figure 2.1.: Figure visualizing the relationship between process model and event log of process mining techniques

Process discovery techniques use only the event log to mine a proces model.
In conformance checking techniques both event log and model are already present. As this

technique is based on the event log and the model, insights into both the model and the real life process, that created the event log, can be gained. This relationship between event log and process model is shown in 2.1(van der Aalst, 2016) Operational support differs from process discovery and conformance checking in the sense that these techniques are applied *off-line*, i.e., techniques are applied afterwards to further understand or improve the processes. Process mining techniques, that are applied online, i.e., while the process is running, are referred to as operational support.

2.1. Process Discovery

Process discovery techniques use an existing event log to create process models. The discovery of process models is the most common and important technique. The creation of a process model for a process can already create a lot of value, but process discovery offers a lot more. The process model becomes even more useful when it is annotated with performance information.

Frequency information that is added to a process model helps to provide insights into the flow of a process. A quick study of the model will reveal if the process is *flowing right*, i.e., following the planned activities in the majority of the traces. An unintended flow can also be discovered. A process model can also be annotated with performance information, showing the total, median or mean duration of a step in the process. This makes it possible to identify bottlenecks in processes and and if an activity is being executed more often than it should. **example**.

As stated before, how much effort was put into correctly logging the events of the process, ensuring an event log of high quality, may determine the quality of process mining results. Recording additional information might also result in a higher quality of the analysis. This becomes clear when looking at how the time information of events may be recorded, and the resulting difference in the performance information. To log an activity in a process it is sufficient to record a single timestamp for every activity. This timestamp would usually be the time the activity was finished. However, if *both start and end* timestamps are recorded for every event, it is possible to tell precisely how long an activity took in the proces and also *how long the duration between activities was.* This can be very helpful, because the bottleneck might not be an activity itself, but the time between two activities.

However process discovery is not limited to constructing process models. The result of another process discovery technique that will be featured in this thesis is a social network instead of a process model. One of these techniques analyzes the handover of work in a process and the result is visualized in a social network. Such a network might show that a larger group

of resources is handing over work to a single resource, who hands it back to them. The larger group clearly relies on the single resource. A possible insight would be that the the process would come to a halt if this resource isn't available, e.g., an employee getting sick. A possible action that might be taken is to provide another resource that is able to perform the required step, e.g., train another employee for this task.

2.2. Conformance Checking

Conformance checking can be used to check both the event log, or the model. To gain meaningful insights one should be confident that either the event log, or the process model can be considered valid. If the event log can be considered valid, conformance checking can prove if the process model is appropriate to explain the behaviour of the process. This is done by replaying the event log on the model. The conformance checking algorithm will try to execute the sequence of activities in the event log on the process model. Not every trace of the event log needs to be observed in the model. It is likely that some deviations are based on false data in the event log, or some exceptions in the process. A very important application of conformance checking is to check wether guidelines are being followed. Not following guidelines could have severe consequences in every process and especially in healthcare. It is therefore valuable to find out if guidelines aren't being followed, and why this might be the case. This would be done by replaying the event log in question on a process model where these guidelines are correctly implemented. This might involve one or more activities that need to be executed *before* a certain activity.

2.3. Operational Support

Process discovery and conformance checking are done by analyzing event logs of traces that have completed the process. Operational support aims to provide information and help while the process is *live*, i. e., currently being executed. Using techniques, that might be not unlike those in earlier steps, it provides insights into how the process can be improved as it is running. It can help to answer questions like which step is to be taken next to ensure the best run through the process, or to recommend a resource for the execution of the next activity. This could help making single traces of processes more efficient as they run. By adding guards to the process model, violations can be perceived the moment their information enters the system.

As the application of operational support requires a running process these techniques will not part of this thesis.

2.4. Preparation for process mining

The sheer amount of data that can be available offer opportunities as well as problems. One of the problems is that considering all the data available will often lead to a process model that is too complex to understand, severely limiting the insights process mining can provide. It is therefore often necessary to specify what the goal of applying process mining is. By doing that, it is possible to filter the event log for the analysis and only consider a fraction of the data. Commercial tools like Disco and Celonis PI offer this step during the analysis. In these tools it is possible to first explore the default model and create or adjust a filter on the log, and discover the new process model for the filtered log instantly. This can be really convenient. Specifying the scope of the process mining analysis in advance can be very challenging, making incremental adjustments necessary.

An example of such a step in preparation of the analysis, specifying the scope of the project early, is the choice of Felix Mannhardt and Daan Blinde, to focus their study in a hospital environment on the trajectories of sepsis patients. The hospital, that was not named in the study for privacy reasons, had 700 beds in multiple locations and is visited by about 50,000 patients a year. Considering all these patients when discovering a process model would not result in a process model that allows for much insights, as the model would simply be too complex. Instead Felix Mannhardt and Daan Blinde chose to focus on analyzing 'the trajectories of patients in a Dutch hospital from their registration in the emergency room until their discharge' Mannhardt and Blinde (2017)

The same principle of zooming in on areas of the process can be applied to other complex processes, to improve the insights process mining will provide.

Note: To realize one of these techniques in a tool like ProM there needs to be a specific plug-in for this technique. Commercial tools like Disco and Celonis PI will apply the same algorithm for every technique and will not show the user any details of this. The chosen algorithms of these two tools will meet the most common requirements, but for academic purposes, or to apply process mining on a higher level, it may be useful to be informed about different algorithms for the process mining techniques. As an example, an academic research group may be able to explore the reality of a process in more detail by choosing the algorithm and its parameters carefully. This thesis will not go into these different algorithms in detail

but will provide a simple example of how different the resulting view on the process can be with different discovery algorithms.

3. Tools

There are available tools for process mining, that differ significantly in scope, their target user, and their ease of use. This thesis will focus on presenting and comparing three different of these tools; ProM, Disco, and Celonis PI.

3.1. ProM

ProM is an academic tool, that was developed at the University of Eindhoven, whose Process Mining research group is leading in the field. ProM is a framework that supports a wide variety of process mining techniques. These techniques are provided in the form of plug-ins. Plug-ins come in the form of different algorithms for process discovery and other techniques. As research in process mining progresses, new plug-ins are being developed. Because of the variety of plug-ins available, ProM is able to offer more extensive process mining techniques than the other two. The best resource to learn more about process mining research and working with ProM is processmining.org[1] But as it is a purely academic, ease of use is not the first priority. The user himself has to choose the plug-ins that suits his purpose best. For example, process discovery is not as easy in ProM as it is in other tools, because there are many plug-ins for the techniques, some of which may not provide a very good representation of the process because of the limitations of the algorithm. Other plug-ins are able to discover very accurate process models, but the user has to choose the right plug-in in ProM and configure it correctly. It can be difficult to tell if the discovered process model is accurate, when there is nothing to compare it to and without in depth knowledge of the process. Analyzing it manually would also be very time consuming. In the process discovery chapter all discover plug-ins that produce a Petri net as their output will be applied to the same event log.
At the date of this thesis ProM is available as the full framework,current build 6.8, or as the 'Lite' version, current build 1.2. According to the release notes[2] over 1.900 plug-ins are available

[1] processmining.org
[2] ProM 6.8 release notes
List of plug-ins available in ProM 6.8

in 6.8. The release notes[3] of ProM Lite 1.2 only lists available 152 plug-ins.

This thesis will use ProM Lite 1.2 because the 'Lite' version contains plug-ins for the most important process mining techniques and pre-processing of event logs and is therefore sufficient for the purposes of this thesis. Mentions of ProM in the rest of this thesis will refer to ProM Lite 1.2.

3.2. Fluxicon Disco

Disco was developed by former students of the TU Eindhoven and professor Wil van der Aalst is still acting an an advisor. Disco is easy to use and guides the user through the first steps of process mining. Process models can be discovered and filtered to only show the most frequent cases or a selection of cases that fit certain criteria. Individual cases can be inspected meeting the criteria can be inspected. As a more user friendly commercial tool Disco does not require the user to choose algorithms for the process mining techniques he wants to apply. A discovered process will also always have the same visualization. Out of the three tools that are considered in this thesis, Disco has the smallest team, with only three people working on it. While conformance checking techniques are not available in Disco, the tool can provide a variety of insights into a process as even more basic techniques like process discovery and performance analysis can provide very useful insights. The great filtering options Disco provides also The user interface of Disco is optimized which can be a great benefit, especially for users new to process mining. Fluxicon provides academic licenses, example event logs, insights into the development of Disco, and tutorials and tips for process mining.

3.3. Celonis

Developed by Celonis SE in Germany Celonis is a process mining tool, that is widely used by big companies today[4]. Among their customers names like Vodafone, Deutsche Bahn and the REWE group can be found. It is a web-based application, that offers effective process mining techniques. When analyzing an event log, the user will upload his event log, and can choose between pre-built apps, of process mining techniques, or choose to built a new app. Building a new app enables the user to use the techniques Celonis has to offer, but arrange the user interface in a way that will serve him or his team best. Academic licenses are available

[3] ProM Lite 1.2 release notes
List of plug-ins available in ProM Lite 1.2
[4] Celonis customers and customers success stories

and provide the user with a few example processes to showcase an effective way of applying process mining techniques in this tool The user can also upload custom data sets, and create his own analysis. Celonis employees have published a paper(Veit et al., 2017), presenting the features of the Proactive Insights engine available in Celonis.

With all of its Plug-ins, ProM offers the application of the most process mining techniques, and the most algorithms, to do so. Celonis comes second in the amount of process mining techniques that are realized in this tool and new techniques were added to the tool in the last six months, most notably an app for conformance checking . Similar to Disco it is neither possible nor necessary to choose an algorithm for the analysis.

3.4. Introduction to the user interfaces of the tools

To make the application of process mining techniques in the following chapter 4 more comprehensible the user interfaces of ProM[5], Disco and Celonis will be introduced briefly, using annotated screenshots.

3.4.1. ProM

ProM Lite 1.2 uses the user interface of ProM 6 versions. The workspace view shown in figure 3.1 lists a selection of resources and has buttons for importing and working with the resources.

[5] A detailed documentation of the ProM 6 user interface is available at promtools.org .

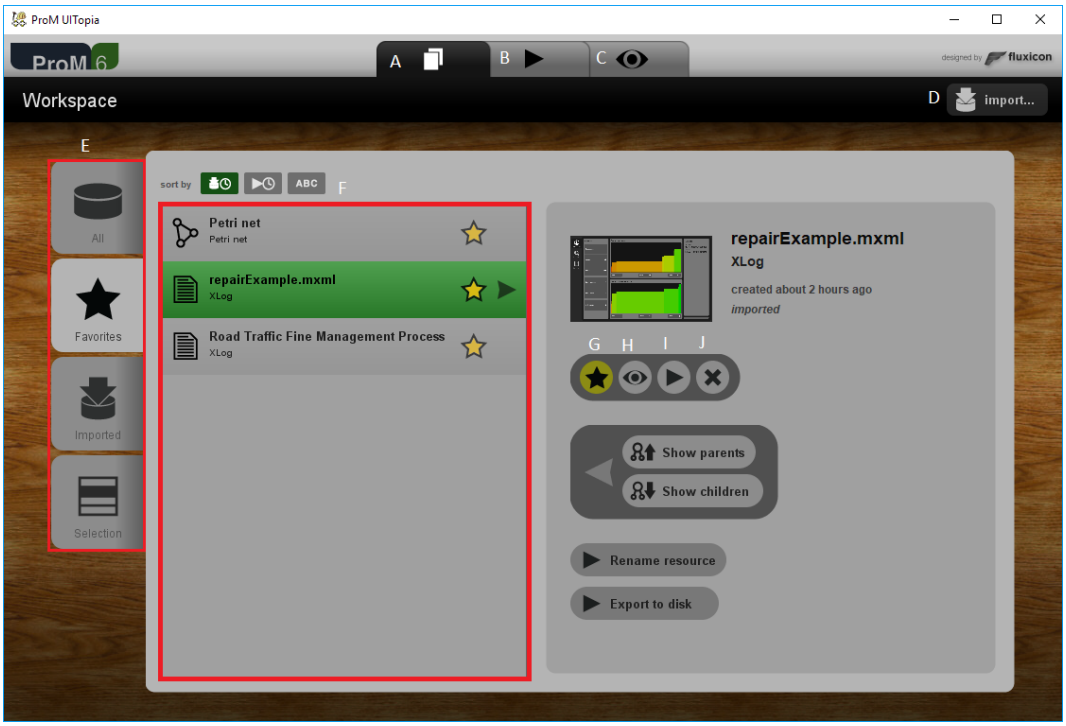

Figure 3.1.: Annotated screenshot of the workspace view in ProM 6

A 'Workspace' tab brings the user to the workspace view.

B 'Action' tab brings the user to the action view.This view will be described in more detail in the next graphic.

C 'View' tab brings the user to the view view. This view visualizes resources.

D Button to import resources into the workspace.

E Tabs to select which resources are displayed in the resources. Listing all resources also shows resources like 'initial marking' and 'final marking' that are not interesting on their own. The favorite tab shows imported resources and the main resulting resources of plug-ins. Resources can be added or removed from this selection using button 'G' on the graphic.

F List of the selected resources.

G Button to toggle the favorite flag of a resource.

H Button to view the selected resource.

I 'use resource' button to use the selected resource or resources in a plug-in, this will select one or more resources, switch to the action view and add the resources to the Input selection.

J Button to remove the selected resource from the workspace.

In the actions view shown in figure 3.2 plug-ins can be used. Input resources and the output type for plug-ins can be specified.

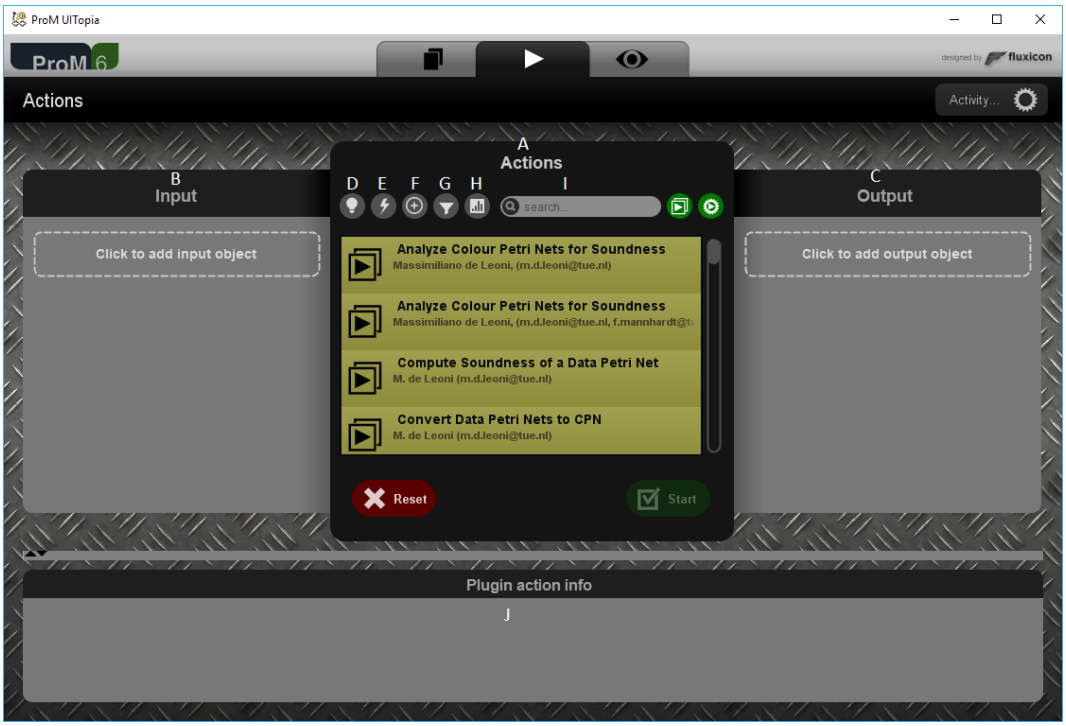

Figure 3.2.: Annotated screenshot showing the actions view in ProM 6

A List of available plug-ins when no input resource and output type are specified this list will contain all plug-ins. When input resources are specified the list will show only plug-ins that use all of the input resources. When one or more output types are specified the list will only show plug-ins that produce these outputs. Plug-ins in the list are shown as green when an input resource of all required input types are selected. Plug-ins are shown as yellow when the plug-in requires additional resources.

B Selection of the resources that will be used as the input for the chosen plug-in. If the chosen plug-in requires additional input resources the type of the missing resources will also be shown here.

C Optional selection of one or more output types. Output types have to be selected from a list that contains all the resource types in ProM. This means that the user must know the name of the resource type in ProM if he does not want to search through a list with over a hundred different resource types.

D Toggle to select wether process discovery plug-ins are shown.If none of the toggles D, E, F, G, H are switched on, all plug-in types are shown. If one toggle is switched on. Only the selected types of plug-ins will appear in the list.

E Toggle for conformance checking plug-ins.

F Toggle for enhancement plug-ins. Enhancement plug-ins enhances a given model using a given event log.

G Toggle for filter plug-ins.

I Toggle for other analytic plug-ins.

J Showing the package, author, category of a selected plug-in and a short description of the plug-in. Package information and a short description is not available for all plug-ins.

3.4.2. Disco

The map view shown in figure 3.3 is shown after successfully importing an event log. The map view shows the proces map, buttons for navigation and many sliders and buttons to adjust the process map.

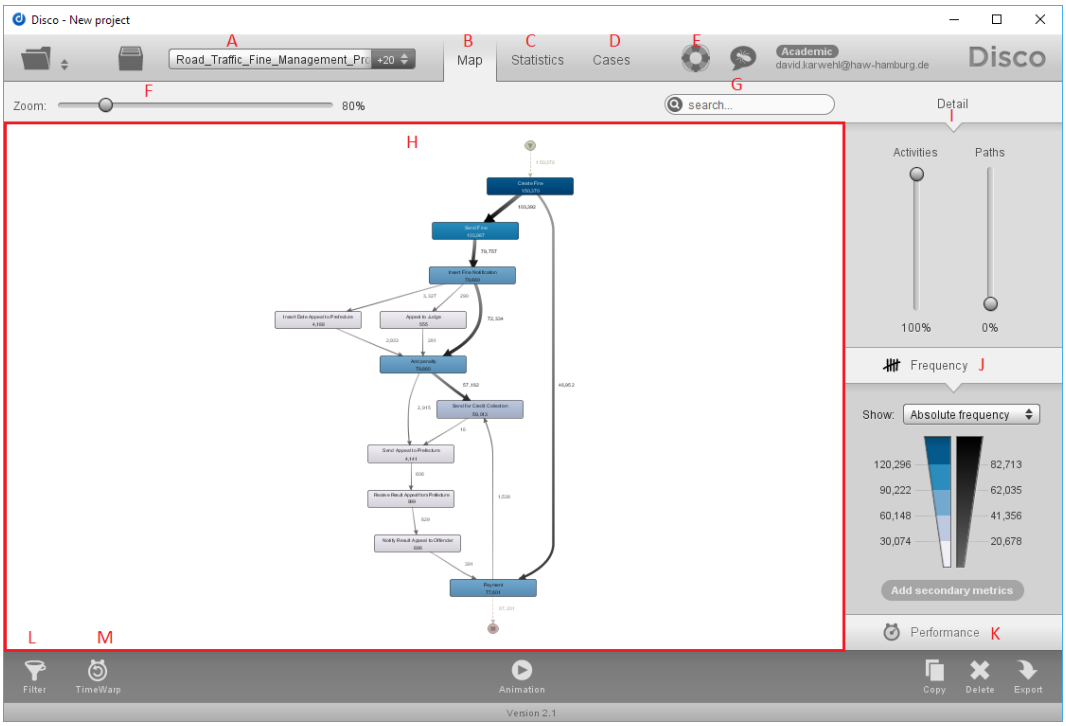

Figure 3.3.: Annotated screenshot showing the map view in Disco

A Clicking on button 'A' will open a list of all data sets in the current project. Clicking on another data set in the list will switch to the data set and to the view that data set was in last displayed in.

B 'Map' tab: clicking on the tab will switch to the map view.

C 'Statistics' tab, Clicking on the tab will switch to the statistics view.

D 'Cases' tab: Clicking on the tab will switch to the cases tab, where case variants and individual cases can be inspected in more detail.

E Clicking this button will display some notes with hints on them to help beginners.

F Zoom slider to zoom in or out of the process map.

G Searchbar to find activities in the process map. Typing the name of an event or an event group will highlight and zoom in on one or more events that fit the input.

H The process map in Disco. This shows the process according to the configured detail level and filter level. Metrics can be chosen and added to the process map.

I The detail sliders for the process map. When the 'Activities' slider is set to 100% all activities present in the event log will be shown in the process map. Lowering the slider will remove less frequent activities. When the 'Activities' slider is set to 0% the process map will show only the most frequent activities. Similarly to the 'Activities' slider , the 'Paths' slider configures the frequency threshold for paths between activities that will be shown. When Disco generates the default process map the 'Paths' slider will usually be set to a low value to improve readability of the process map.

J Frequency metrics configuration. Different frequency metrics can be added to the process map as primary metrics and frequency or performance metrics can be added as secondary metrics.

K Performance metrics configuration. Clicking on this button will change the primary metrics of the process maps to a performance metrics. Performance or frequency metrics can be added as secondary metrics.

L Clicking will open the filter settings for the current data set. New filters can be added or existing filters can be adjusted or removed.

M 'TimeWarp' settings allows the user to set custom business hours and holidays for each data set. This will improve the performance information provided by Disco.

3.4.3. Celonis

The process explorer app shown in figure 3.4.3 is displayed after successfully importing a data model and creating a default analysis. The map view features the proces map, buttons for navigation and many sliders and buttons to adjust the process map.

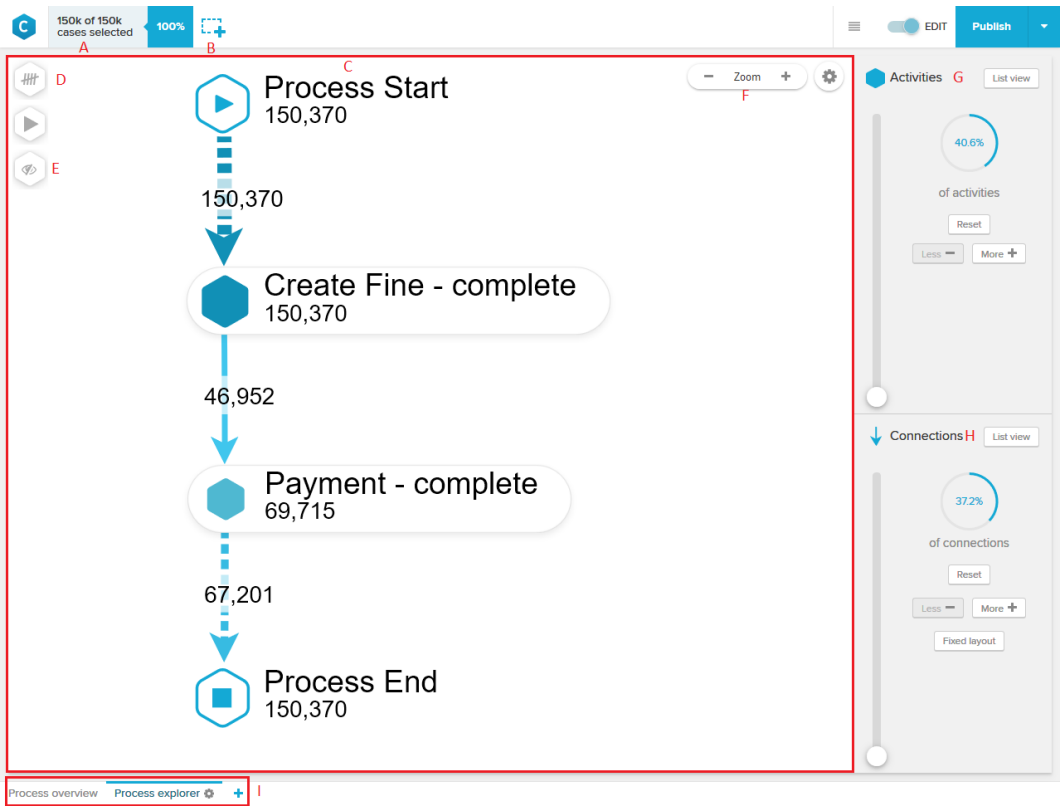

Figure 3.4.: Annotated screenshot showing the process explorer app in Celonis

A Shows how many cases are displayed in the process model. Adding a filter will result in a selection of the numbers that will be displayed.

B Adds filters, called selections in Celonis, to the analysis. Only cases that fit the filter criteria will be considered in the analysis.

C Shows the process model. The process model is adjusted instantly when the configuration of the process explorer is changed.

D Offers a selection of KPIs that can be added to the process model. The default KPI is 'Case Frequency' other options are 'Activity Frequency', 'Throughput Time(Median)', 'Throughput Time(AVG)' and 'Throughput Time(Trimmed mean)'.

E Allows the user to hide any of the activities in the process model.

F Zooms in or out on the process model. Clicking in the middle will reset the zoom level on the process model.

G Configures how many activities are shown in the process model. At 100% all activities are shown. Setting the slider to the lowest value will show only the most frequent activities. The percentage value next to the slider shows how many activities of the event log are shown in the model (total frequency)

H Same functionalities as 'I' for connections between activities in the model.

I Shows all apps in the current analysis and allows the user to add more apps to it.

4. Use case based application of process mining techniques

4.1. Use cases in this thesis

As stated by Alistair Cockburn in his book 'Writing effective Use Cases' (Cockburn (2001)), use cases do not need to be very complex, and do not always need to follow a certain structure to be useful. This thesis focuses on the study of process mining techniques in current tools instead of the writing of complex use cases. Less formal, simple use cases will serve as examples for the effective application of process mining techniques.
The use cases presented in this thesis will be based on real life event logs available at data.4TU.nl[1]. These event logs come from projects of the IEEE CIS Task Force on Process Mining.
In these projects members of the task force and the Eindhoven University of Technology have applied process mining techniques in several fields. Because of the complexity of these studies, the goal of these will not be considered in the uses cases in this thesis. Instead the thesis will present simple use cases that show effective application of process mining techniques and the limits of the tools presented in this thesis.

It is easy to imagine a context where process mining can be useful when thinking of large companies where many employees work together on complex processes. In a worst case no one really knows what the process looks like or where responsibilities end or overlap. Process mining can provide insights into the details of processes by clarifying what the reality looks like.

In the rest of this chapter simple uses cases will and expectations to the tools will be described. The tools are then used to apply process mining techniques. Each section ends with a short conclusion on wether the tools met the expectations.

The configuration of the plug-ins in ProM will be described in more detail. Hopefully this will give readers new to ProM an idea of how the plug-ins applied in this thesis can be

[1] Real life event logs available at data.4TU.NL

configured to improve their results. In the following use cases the application of the process mining techniques will be described in the tools Disco and Celonis before ProM. This is done to avoid starting the use case based analysis with a long explanation of the configuration of a plug-in in ProM.

4.2. Discovery of a process model from an event log

Use case: Application of process discovery techniques to mine a process model from an event log.

Expectations:

- The tools should support the discovery of different process models, where activities and connections are removed from the model based on frequency, to improve readability of the model.

This use case assumes the simple scenario of applying process discovery techniques to an event log. This use case is based on the real life event log 'Road Traffic Fine Management Process', available on data.4tu.nl[2]. This event log was used in a study conducted at the Eindhoven University of Technology about application of conformance checking techniques. The event log was provided by the local police of an Italian city. The log contains information about over 150.000 road traffic fines. Events in the log describe activities such as creating the fine, sending out the fine and receiving payment(Mannhardt et al., 2016).

[2] 'Road Traffic Fine Management Process' event log.

4.2.1. Disco

Disco requires the fewest steps to create a process model. The user needs to choose which event log should be imported, relevant columns in the event log like case ID, activity name, and timestamps will automatically be identified. Disco will parse the event log and automatically create a process model(called process map in Disco). The process map shows a process model with additional frequency and performance information about the process, and identify different case variants. As stated before, the process model can be adjusted, depending on the level of detail needed. The standard process model discovered by Disco will show all the activities, but only the bare minimum of paths between activities.

As shown in section 3.4, Disco offers a process map, statistics, and overview over case variants and cases as part of the analysis. In the user interface these are split into the *Map, Statistic and Cases* tab. The detail level of the model can be adjusted with two sliders. One for the level of detail for the activities, and one for the paths. If both sliders are at a 100%, the full process map will be shown. In this process model all behaviour that was observed in the event log can then be seen in the process map. Lowering the sliders will result in a lower level of detail and less frequent activities or paths will be removed from the model. At 0% only the most frequent activities or paths are still present in the process model. This can make it easier to understand the process, especially if the process is very complex.

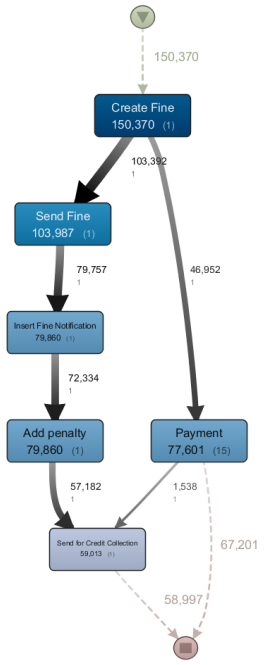

Figure 4.1.: Process model discovered in Disco, showing only six most frequent activities and the most frequent paths

To discover a process model that shows only the most frequent activities the activities slider is lowered until all activities with an absolute frequency of less that 50.00 are removed from the model. The paths slider is also set to 0% to show only the bare minimum of paths.

For comparison, the process model Disco would discover with all activities and paths included can be seen in figure A.1 on page 55.

4.2.2. Celonis

In Celonis users will have access to their own or shared workspaces to import event logs and create analyses. In the first step the the data model is named and uploaded. When the data model is uploaded. Celonis will attempt to detect the data types of columns in the data model. The user is then prompted to review if the automatic detection identified the data types of the columns correctly. In the next step the user can select the sorting and end-timestamp columns. This is an optional step. it is now possible to create a new analysis for this data model or declare the user column (called resource column in other tools and papers) or a cost column, declaring the cost of individual activities. When creating an analysis, the user needs to choose the data model and a name for the analysis. The user can then create a new analysis in a workspace. Data models and analyses will be stored in the workspace. The workspace is organized by separating the analyses and the data models. It is also possible to organize the analyses in folders to provide a better overview. It is possible to create a default analysis which will consist of the process overview app and the process explorer app. If the user chooses not to create a default analysis he can still add prebuilt apps to his analysis. In Celonis apps offer different functionalities for the analysis. The Process explorer app will discover process models of varying levels of detail. Similar to Disco, the process explorer app in Celonis hast two sliders for the user to choose the level of detail. One for the activities and one for the connections between them(paths in Disco). The process explorer will show a process model with the minimum of activities and connections first. In addition to the slider, Celonis features three buttons for each settings. The "Less -" "More +" and "Reset" button. The ‘More +’ button will add either one activity(with two connections) or one additional connection to the model. The added activity or path will always be the next frequent.

The activities sliders is adjusted until only activities with a case frequency(metric is shown inside the activity) of more than 50.000 are included. The paths slider(called connections in Celonis) is set to the lowest possible value 75,7% to show as few connections as possible. The resulting process model can be seen below in figure 4.2.

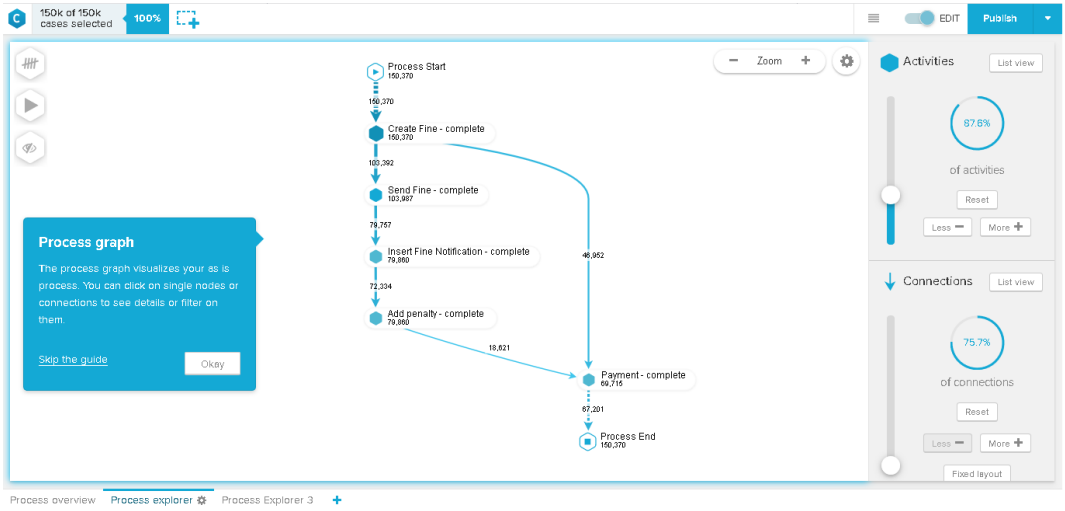

Figure 4.2.: The discovered process model in Celonis, adjusted to show 5 activities

The process model created by Celonis when all activities and paths are included is shown in figure A.2 on page 55

4.2.3. ProM

When launching ProM the user will always start with an empty workspace as sessions are not stored when closing ProM. After importing the event log the user can click on it and use it in a plug-in by pressing the 'use resource' button with the play symbol. The user is offered a list of plug-ins available in ProM. All plug-ins have specified in- and outputs and only actions matching the selected in- and outputs are shown. As the user initiated this step by using the event log it will already be added as an input and only actions using an event log as input will be shown. To find the right plug-in the user can either search for a plug-in by entering the name into the search bar, or by selecting it from the list of plug-ins. ProM also offers filtering the plug-ins by their technique. To discover a process model the user can click on the lightbulb symbol in the top left to show only discovery plug-ins.

In the following all plug-ins available in ProM Lite 1.2, that require only an event log as input and directly produce a Petri net, are used to discover a process model. The 'Mine with Inductive visual Miner' plug-in will also be used, even though the output is not a Petri net, because it offers functionalities similar to Disco and Celonis. All of the plug-ins applied in this section use the 'Road Traffic Fine Management Process' event log as input to discover a process model.

Petri nets can be visualized in different way in ProM. Petri nets in this section will be visualized using the 'GraphViz Petri net visualization'.

Alpha Miner

After selecting the 'Alpha Miner' plug-in the user will click start, choose the 'Event Name' as the 'Event Classifier', and 'Alpha' as the version of the algorithm to be applied. For this log this will be the default options. After clicking finish the Alpha algorithm will produce the Petri net shown in figure 4.3.

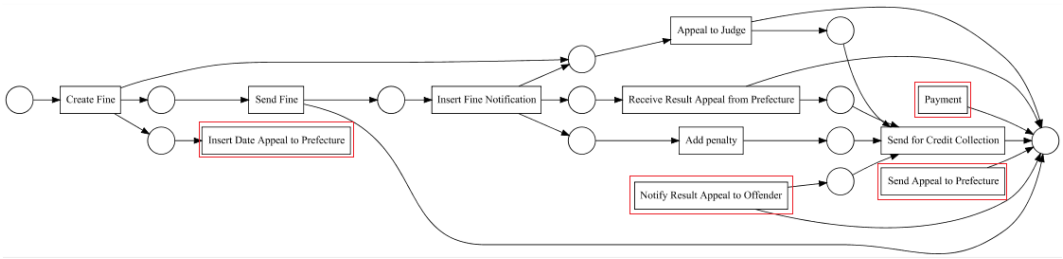

Figure 4.3.: Petri net discovered by applying the 'Alpha Miner' to the 'Road Traffic Fine Management Process' event log

ILP-Based Process Discovery

ILP stands for Integer Liner Programming. The plug-in is available as a normal and an express version. The regular version has three configuration levels(express, basic, advanced). The express variant only requires the user to select the desired event classifier. The most important step for the basic configuration is choosing a miner. A preview of the model is shown next to the selection of miners. Choosing a different miner will update the process model after some time. It took more than three minutes with the 'Road Traffic Fine Management Process' as input. This event log has over 150.000 cases the plug-in will likely perform significantly better with less cases. To mine a process model the midi miner was chosen with the default configuration. The resulting process model is shown in figure 4.4.

Figure 4.4.: Petri net mined by applying the the 'ILP-Based Process Discovery' plug-in

Mine Petri Net with Inductive Miner

To configure this plug-in the user has to select the event classifier, an inductive miner variant and choose a noise threshold. If the Noise threshold is set to 0.00 perfect log fitness is guaranteed , meaning all behaviour observed in the event log will be present in the process model. The Petri net shown in figure 4.5 was mined with a noise threshold of 0.20.

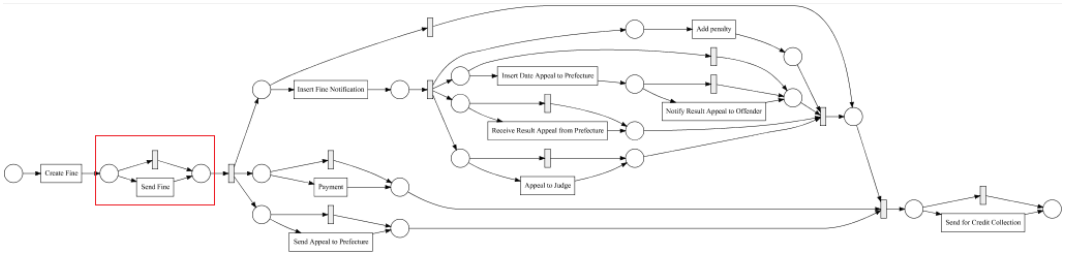

Figure 4.5.: Petri net mined by the 'Mine Petri Net with Inductive Miner' plug-in

Note: The 'Mine Petri Net with Inductive Miner' is not shown in the list of discovery techniques when clicking the lightbulb symbol, despite being a discovery plug-in

Mine with Inductive visual Miner

The 'Mine with Inductive visual Miner' plug-in does not produce a Petri net as the output. It will instead produce an 'Inductive visual Miner' resource. Similarly to Disco and Celonis this miner allows for the adjustment of the shown process model by raising or lowering the activities and paths sliders. The process model can be exported as a Petri net or process tree. Both this and the 'Mine Petri Net with Inductive Miner' are working with process trees. In his dissertation *Robust process mining with guarantees* S.J.J. Leemans describes the advantages of process trees in great detail in his PhD-thesis(Leemans, 2017).

Figure 4.6.: Inductive visual miner created using the 'Road Traffic Fine Management Process' event log as input

The key features of this plug-in will be described in the following list:

A process model mined with the current configuration. When sliders are adjusted a new process model is mined.

B Sliders to adjust the frequency threshold for activities and paths appearing in the model.

C Selection of filters to apply to the event log.

D Button to export the process model 'A' as a process tree or process model

E Individual cases running through the process model can be visualized and will be displayed as yellow dots.

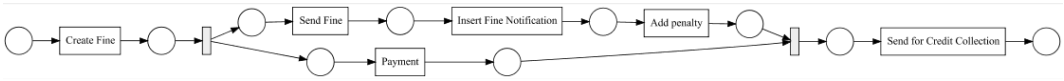

Figure 4.7.: Petri net mined by applying the 'Mine with Inductive visual Miner' plug-in

conclusion -ProM discovery plug-ins

A valid process model should fulfill a few basic requirements. One of these is that it should have exactly one starting place (a place with no incoming connections) and one final place/ sink (a place that has no outgoing connections). Disco and Celonis fulfill these requirements by creating an artificial starting place and sink. All other activities should have at least one input and output place. An activity without an input place could be executed at any time in the process, and also any numbers of times. An activity without an output place would not effect the rest of the process, an exception would be the consumption of a token that is needed for another activity.

As can be seen in figure 4.3 there are several activities that don't have any places as inputs before them. It becomes obvious that the discovered process model is not very accurate when considering the implications. The activity 'Payment' in 4.3 does not have any incoming connections from places. This means the activity can be executed without requiring a token from an input place. The activity could be executed at any time, and any number of times. The payment activity in the Petri net refers to incoming payments of traffic fines. Assuming this process model to be accurate, the acceptance of payments would not require the creation and sending of a fine. The acceptance of payments without any corresponding fines to justify it would be a perfectly fine execution of the process. While this sounds pleasant, it might lead to some legal complications. The other process models that were discovered fulfill these basic requirements.

4.2.4. Conclusion

Disco and Celonis meet the expectations. Both tools allow the user to created process models quickly and adjust it to the desired detail level. ProM meets the expectations, but only the 'Mine with Inductive visual Miner' was adjust the mined process model to the preferred detail level. In the application of these tools Disco and Celonis were able to mine an adjusted level significantly faster(mostly in less than a second). The Inductive visual Miner in Prom took more than a minute.

Comparing the models that Disco(figures 4.1 and A.1) and Celonis (figures 4.2 and A.2) were able to discover highlights the benefit of being able to discover process models with different detail levels. Process models with very few activities can be discovered first and detail can gradually be introduced back into the process. This can make it significantly easier when being faced with new complex processes.

In the following of this thesis will avoid using event logs that produce process models as complex as the one shown in figure 4.1 to improve readability. It would not be possible to perceive any activity names, let alone details or metrics. However, it is important to point out that is exactly the complexity of the processes that makes the application of process mining techniques so effective!

4.3. Performance analysis using process discovery with additional performance information

Use case: Application of process mining techniques to mine frequency and performance information of activities in a process.

Expectations: The tools should be able mine a process model. It should be possible to annotate this process model with frequency and performance information to quickly see things like

- Which activities take longest to complete
- Which activities are completed fastest
- Wether any bottlenecks are impacting performance

This annotated process model should provide a starting point for further analysis and the tools should be able to analyze case variants in more detail.

This use case will show the application of the tools to mine and display performance information of processes. The event log in this section was chosen because the aim of the performance analysis in this chapter was to apply performance analysis techniques of the tools to the same process model. The aim was also to consider all frequent activities in this analysis. the real life event logs available on data.4tu.nl [3] mostly resulted in process models had too many activities and paths to retain decent readability. Other process models were straightforward but the performance analysis showed no unexpected behaviour that gave grounds for an analysis. Another requirement for the event log was the inclusion of start and end timestamps. The event log analyzed in this use case is about a repair process for telephones in a company. The event log is used in the ProM tutorial ProM [4] where some background information regarding the process is provided. The process starts by registering the telephone a customer sent in. In the next step the phone is analyzed and the problem is categorized. The customer is informed about the fault and the repair can be handled by one of two teams. One of the teams will repair simple, and the other one complex defects. Some faults can be repaired by both of the teams. After the repair is finished the device is sent to the QA department where it will be analyzed wether the defect is fixed or not. If the telephone was not successfully

[3] Real life event logs at data.4tu.nl
[4] ProM tutorial at promtools.org

repaired it will be sent to the repair department again. If the repair was successful, it will be archived and the phone will be sent back to the customer. To save time the company will only try to repair a problem a limited number of times. The process model can be seen below in 4.8.

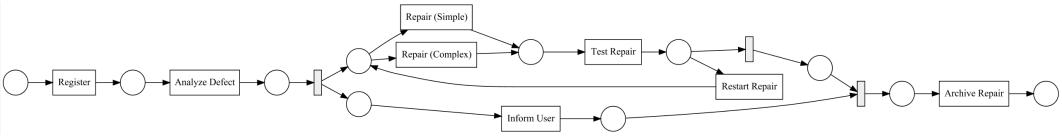

Figure 4.8.: Petri net of the phone repair process

The event log contains additional performance information. For the activities 'Analyze Defect', 'Repair(Simple)', 'Repair(Complex)' and 'Test Repair' a start and end timestamp is recorded. However, this was not realized in the event log by two timestamp columns. Instead the start and end of the activity are present as separate activities in the event log. For example the 'Test Repair' activity is split into two activities: 'Test Repair+start' and 'Test Repair+complete'. All traces in the process should end with the repair being archived. The event log contains some traces that have a different end activity. So the first step in all tools will be to filter the event log to remove all cases that end with a different activity.

In the following subsections performance analysis techniques are applied and hypotheses about the process are proposed to showcase potential insights these techniques can provide.

4.3.1. Performance analysis in Disco

The event log is imported the same way as in the process discovery use case. In Disco a filter can be applied by pressing the button in the bottom left. Next an endpoint filter is added by clicking the 'click to filter' button and selecting 'Endpoints' as the filtering option in the 'Discard cases' mode. All end event values that are not 'Archive Repair' are deselected by clicking on them. In this case 'Inform User', 'Repair (Complex)' and 'Test Repair'. The filter is applied and all cases in which the last event is not 'archive repair' will be discarded.

Detail sliders can be adjusted as described previously. To get a first impression of the process and the performance the activity slider is set to 100%, including all activities in the process map. The paths slider is set to 0%, including only the most frequent connections between activities The default metric on the process map in disco is the frequency metric. It can be switched to the performance metric by clicking in the bottom right on the button 'Performance' with a small stopwatch. The process map can show different performance metrics. The total, median, mean, maximal or minimal duration of each activity and connection can be displayed. It is also

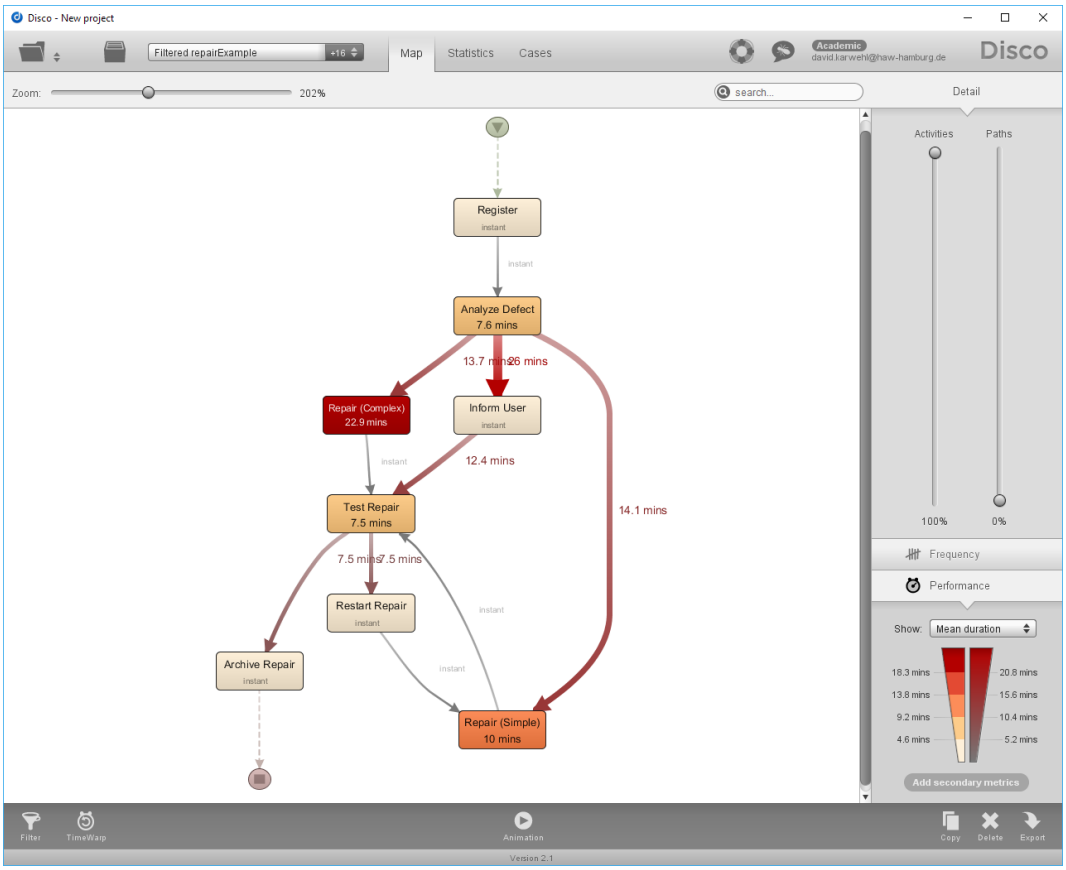

Figure 4.9.: The discovered process model of the repair example in Disco with the activity slider at 100%, the path slider at 0%

possible to add one of the frequency or performance metrics as a secondary metrics. To gain information about the average performance of one instance in the process the mean duration is selected. Figure 4.9 shows the process map mined with these settings.

The first thing that can be noticed is that the activities 'Analyze Defect', 'Repair(Simple)', 'Repair(Complex)' and 'Test Repair' are not split into a start and an end activity in the process map of Disco. Instead Disco is showing only a single activity for each of them and the duration between start and completion is shown in it. Knowing the flow of the process the position of the 'Inform User activity' is not expected. The outgoing connection to the 'Test Repair' activity seems to imply a more central role in the process.

Focusing on the average performance of the process all activities and connections between activities are shorter than 30 minutes. So in general, one instance of the process might not take much longer than one hour. Switching to the statistics tab in Disco and switching to the case

duration in the center confirms this hypothesis. The average case duration is 66.6 minutes. Knowing other activities are not relying on the 'Inform User' activity, connections from and to this activity can be ignored.

Switching back to the map, the average time for the complex repair is about 23 minutes while the simple repair takes about 10 minutes. As expected, the complex repair does take significantly longer than a simple one. However, the time between the analysis of a defect and the start of the repair activities is almost the same. Between the analysis of a defect and the start of a complex repair 13.7 minutes pass on average. On average 14.1 minutes pass before a simple repair starts after the defect was analyzed. Judging by the short duration of the process this is unlikely to be artificial delay but rather caused by a waiting time before resources for the repair are available. The fact that the two waiting times are so even could indicate that resources are allocated very evenly between them. Furthermore, the circumstance that some repairs can be completed by each of the teams should help to distribute workload more evenly just as it seems to be displayed in the process map.

In this example there are no apparent flaws in the performance of the process. An example of such a flaw could be a simple repair taking longer than a complex repair.

On the process map, with the path slider being set to 0%, the activity 'Restart Repair' only leads to the 'Repair(Simple)' activity. It is very much expected that a telephone could still require a complex repair if it was not repaired properly the first time. To visualize this behaviour in the process map the path slider must be raised. Raising the path slider affects the 'Inform User' activity the most, adding additional connections. As stated before these connections to the activity do not provide useful information.

Setting the path slider to a value between 77 and 95 % produces the process model that includes the connection between the ''Restart Repair' and 'Repair (Complex)' activities. Adding a frequency metric as a secondary metric helps to better analyze the performance of the process. The connections between the 'Restart Repair' activity and the two repair activities are annotated 'instant', indicating that a repeated repair is handled faster than a new repair.

Switching to the frequency metric and choosing 'Show: Absolute frequency' and 'Case frequency' provides more performance information. The process model mined with these settings is shown in figure A.3 on page 56. This shows that repairs were restarted 307 times in total, and in 231 out of 1.000 cases. By applying a 'Follower' filter it is possible to find out wether the requirement of the restart of the repair is more likely after a complex repair or a simple repair. A 'Follower' filter is applied, keeping all cases in which the 'Analyze Defect' or 'Inform User' activities are directly followed by the 'Repair(Simple)' activity. It is important to include the cases in which 'Inform User' is directly followed by 'Repair(Simple) because the

activity 'Inform User' could happen between the analysis of the defect and the repair being executed.

Out of 1000 cases, restarting the repair at least once was necessary in 231. The filtered process model is shown in figure A.4 on page 57 shows that the initial repair was a simple repair in 427 cases, and restarting the repair after an initial simple repair was necessary in 189 of these cases. So an initial simple repair is far more likely to require a follow up repair. On the other hand the success rate in complex repair seems high, judging from the low amount of cases that require the repair to be restarted after a complex repair. The high amount of cases where simple repairs requiring the restart of the repair process at least could provide a valuable starting point for further analysis, but would be out of the scope of a demonstration of applying the performance analysis tools of disco.

4.3.2. Performance analysis in Celonis

The first step is to import the event log as a new data model as described in process discovery use case 4.2 on page 22 . The creation of a new data model from an event log has some optional steps. One of these steps is the selection of an end timestamp column. Unfortunately, the end timestamp is logged in a different activity as stated before. After creating the data model a new analysis is created as described in the previous chapter. It is useful to create the default analysis, because the process explorer is included in the default analysis. The first step is to apply a filter to remove the cases that do not end in the 'Archive Repair' activity. This is done by clicking on the 'Add new selection' button in the top left and selecting the 'Activity selection' type. In the bottom right the 'Archive Repair' activity is added to 'CASE ENDS WITH' section. This selection again leaves 1.000 valid cases in the data model. First the activity slider is set to 100 % to display all activities in the process. Inspecting the process model shown in figure A.5 on page 58 shows that Celonis does not merge the start and end activities into a single activity as Disco did. This results in a process model with four more activities, reducing readability. To ensure more readability of the process model in the analysis it is advisable to group these activities. Activities can be grouped by pressing the gear symbol in the top right and clicking the 'New group' button. In the next step a name for the group and the activities that should be included in the group are chosen. The four new groups are 'Analyze Defect', 'Repair(Simple)', 'Repair(Complex)' and 'Test Repair' groups, all including their respective start and completion activity. The metric is also changed to the average throughput time by clicking the tally-symbol in the very top left and selecting the 'Throughput Time(AVG) option. By hovering the mouse over the stopwatch button the time unit for the process model can be changed to units to better fit the process.

Adding connections to the process model will add connections starting and ending in these new subgroups. The times annotating these connections are the times it takes for the activity to complete. Most added connections are again linked to the 'Inform User' activity, making the process model seem bloated and reducing readability. Celonis offers the option to hide activities. Because the 'Inform User' activity can appear at a number of points in the process without affecting the process it makes sense to hide this activity. The third of the three icons in the top left, the crossed through eye icon, is selected and the 'Inform User' activity is deselected, resulting in the follwing process model.

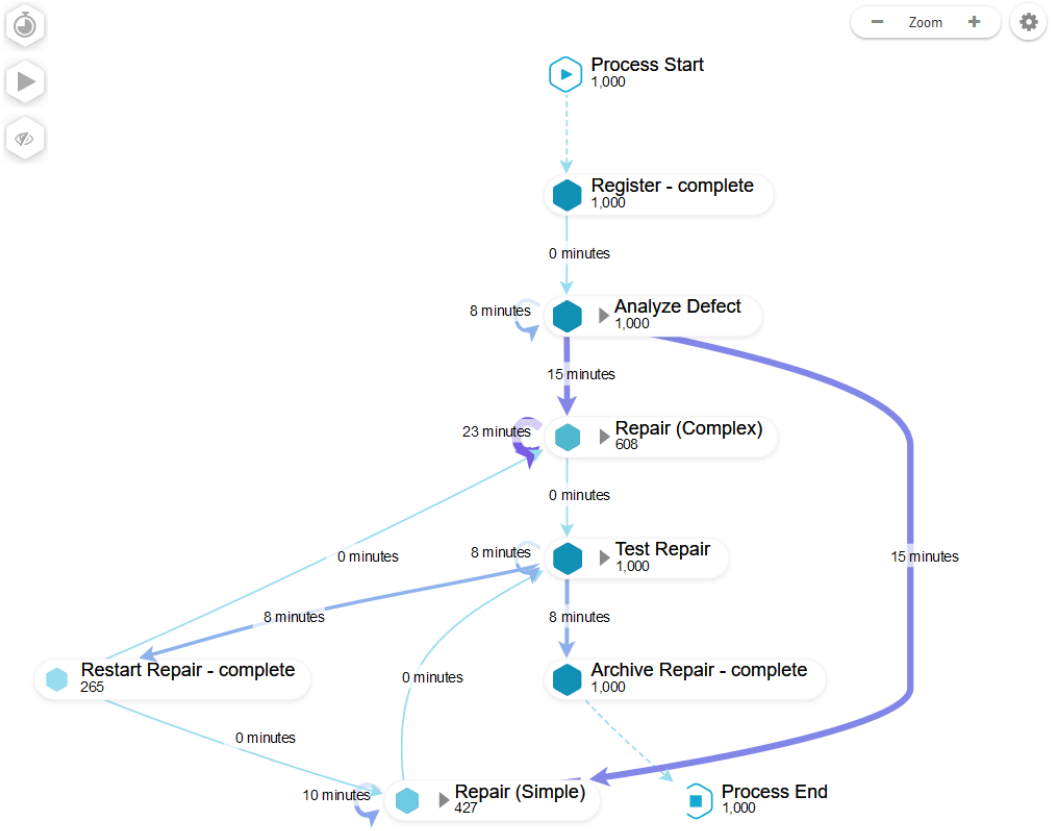

Figure 4.10.: The discovered process model in Celonis after grouping the activities with start and end activities and hiding the 'Inform User' activity

When analyzing the model it becomes apparent that the waiting time for both repairs is about equal, hinting at good allocation of resources in the company. Times throughout the process are also short, resulting in a process that takes place in the timeframe of hours rather

than days. Switching to the 'Process overview' and selecting 'Show results in' to minutes, by clicking on the gear in the top right of the 'Process overview' shows that the average case duration is 66 minutes as can be seen in figure A.6 on page 59. The average time for a complex repair is 23 minutes. Taking more than twice as long as a simple repair takes on average(10 minutes).
Note: Hiding the 'Inform User' activity has a significant impact on the performance calculation of the process model. When the activity is included in the process model the throughput time for the complex repair is shown as 18 minutes, while the connections to between the 'Inform User' activity and 'Repair(Complex)' group are taking 13 minutes each on average. This could be a coincidence, or it could mean that the 'Inform User' activity is not irrelevant for the performance of the process, but that the customer has to be informed and asked for permission for some complex repairs.

The model in figure A.5 shows that it was necessary to restart the repair at least once in 265 cases. Switching the KPI to 'Activity Frequency' shows that 'Restart Repair' was executed 369 times. Finding out if a group of repairs is more likely to need additional repairs could lead to a significant improvement in the process. *In the following the repairs are only split into two groups. They are split on the type of repair that was executed first. The event log also contains a 'defectType' column. Splitting the repair by their event type would allow for more accurate analysis but will be out of the scope of this thesis.*

If simple repairs often need additional repairs, this would be crucial to know when analyzing performance. Focusing on the average throughput time would not be enough. Analyzing wether additional repairs are more likely to be needed after a simple or complex repair is not feasible directly with full accuracy. This is because 'Analyze Defect' is followed by 'Repair(Simple)', Repair(Complex)' and *'Inform User'*. So when a filter is applied that removes all cases in which 'Analyze Defect' is directly followed by 'Inform User', to remove all cases in which a simple repair was attempted first, is not enough.

It is important to state that some of these cases will remain in the event log because the order of events in the event log can be: *Analyze Defect -> Inform User -> Repair (Simple).* This filter (called selection in Celonis) can be applied in Celonis by clicking on 'Add new selection' in the top again and choosing 'Process flow selection', then selecting that 'Analyze Defect - complete' activity should not be directly followed by 'Repair (Simple) -start'.

The resulting process model can be seen in figure A.7 on page 59. This figure shows that out of 573 cases in which a complex repair was likely executed first only 108 of theses cases require a new repair. So by applying

4.3.3. Performance analysis in ProM

The analysis starts by importing the event log into ProM. The first step is to filter the event log before any further analysis. This is done by selecting the event log and clicking on the 'Use resource' button, with the play symbol. Next, the word 'Filter' is entered into the search box to find the plug-ins that contain the word filter. An alternative would be to select the filter button in the top left of the 'Actions' box. The plug-in 'Filter Log using Simple Heuristics' by H.M.W. Verbeek is selected next. In the initial step the plug-in is configured to keep, remove, or discard instances that the filter applies to. For this use case only the case instances the filter will apply to will be kept in the event log. In the next step the start events do not require any filtering because the only start activity in all case variants is 'Register+complete'. In the filtering of the end events, the plug-in shows these five events: 'Archive Repair+complete', 'Inform User+complete', 'Repair(Complex)+complete', 'Repair(Complex)+start' and 'Test Repair+complete'. They appear as end events in the event log. In further analysis only case variants that correctly end in 'Archive Repair+complete' will be included. In the final steps events can be removed from the log, but this is not necessary for this analysis. The filtered log contains 1.000 cases and 10.845 events in total. To get a first idea about performance of the event log, it is selected and the eye symbol on the right is clicked to inspect the resources. A 'Dashboard', showing 6 key metrics and two graphs, an 'Inspector' (enabling inspection of individual instances, and a 'Summary' tab, showing more information about events and resources in the case) are available. The 'Summary' tab includes a table that shows the absolute and relative occurences of events in a table, that can be seen below in figure 4.3.3.

Inspecting the event table of the 'Summary' tab shows that the 'Register+complete', 'Inform User', 'Analyze Defect' and 'Archive Repair' events are occuring 1.000 times in 1.000 cases. This indicates that these events are executed exactly once every case. A simple repair is occuring 697 times, and a complex one 672 times. A repair was restarted 369 times in total.

This filtered log is used to mine a Petri net of the process. The 'Mine Petri net with Inductive Miner' plug-in will be used with a noise threshold of 0.20. This ensures that the process model will represent the vase majority of case variants; but as filtering the log previously was necessary, choosing a noise threshold > 0.0 is advisable. The resulting Petri net 4.8 at the beginning of this section on 32. To analyze the performance of the process the filtered event log and the discovered Petri net are selected, and the 'Replay a Log on Petri Net for

All events

Total number of classes: **12**

Class	Occurrences (absolute)	Occurrences (relative)
Test Repair+complete	1369	12.623%
Test Repair+start	1369	12.623%
Inform User+complete	1000	9.221%
Archive Repair+complete	1000	9.221%
Register+complete	1000	9.221%
Analyze Defect+complete	1000	9.221%
Analyze Defect+start	1000	9.221%
Repair (Simple)+complete	697	6.427%
Repair (Simple)+start	697	6.427%
Repair (Complex)+start	672	6.196%
Repair (Complex)+complete	672	6.196%
Restart Repair+complete	369	3.402%

Figure 4.11.: Table showing the occurencies of events for the repair process in ProM.

Performance/Conformance Analysis' plug-in is selected. The plug-in detects all necessary sequence patterns and for the activities 'Analyze Defect', 'Repair(Simple)', 'Repair(Complex)' and 'Test Repair' a pattern exists that includes both the start and the complete event. However, the patterns also still contain those that only contain the start or completion event. Each of the four activities 'Analyze Defect', 'Repair(Simple)', 'Repair(Complex)' and 'Test Repair' has four different patterns, so patterns that only contain a start or completion event must be removed. The correct selection of patterns is shown in figure A.8 on page 60. When clicking next the whole activity name has to be considered not just the name before the first bracket. The patterns will now be correctly selected and the plug-in requires no further configuration. Costs for movements do not need to be set or changed in the last step. since this is only relevant for conformance checking. In the last step in the dialog 'Would you only consider performance between synchronous moves?' needs to be answered by the option'No, move on models are assumed to be firing transitions as soon as they're enabled'.

A visualization is generated showing the performance projection onto the model. Performance information is projected onto the events. In the 'Inspector'(in the top right of the Petri net) all element statistics for each element of the model can be viewed. What performance information should be projected onto the model can be selected in the 'Display' tab of the 'Inspector' . The place color is set to 'None', because the places do not add information that could not be more conveniently projected onto the events. The transition color is now set to 'throughput time (avg)' to provide an idea of how long each of the activities take. The model

with the average throughput time projected onto it can be seen in figure A.9 can be seen on page 61. This model shows that the average throughput time for activities ranges from 0.00 ms (instant) to 22.93 minutes for a complex repair. The instant activities only have a start timestamp, so no duration can be recorded for them. Only collecting additional timestamps for four activities could have been a choice because the other activities can be expected to be performing well. Out of the four activities that do have average throughput times shown a complex repair takes the longest, requiring a little more than twice as long as a simple repair(9.93 minutes) to complete Both the analysis of a defect(7.58 minutes) and the testing of a repair(7.5) require less time to complete than a simple repair does on average. Relatively to each other the activities seem to perform well with the repairs requiring the longest to be completed and a simple repair taking longer than a simple one.

Projecting the average waiting time onto the model, as can be seen in figure A.10 on page 62, shows that there seems to be a significant waiting time before activities take place, when compared to the throughput times of activities. For example the average waiting time before 'Repair(Simple)' was executed was 9.46 minutes, with the activity taking averagely 9.97 minutes to complete. Availability of more resources might help to reduce the waiting time.
Projecting the transition frequency, as can be seen in figure A.11 on page 63 shows that the times both repairs were executed are very close. A complex repair was executed 672 times, a simple repair 697 times. Restarting a repair was necessary 369 times.

4.3.4. Conclusion

All three tools were able to meet the requirements and were able to provide the user with useful performance information. All tools offered a process model that could be annotated with different frequency and performance information. As expected the commercial tools Disco and Celonis offered some key features the ProM plug-ins was missing:

- Adjustment of process model with the frequency sliders for activities and paths during the analysis
- Addition and removal of filters during the analysis

It would be possible to apply filters and adjust the model in ProM. However, this would require the user to navigate back to the workspace, adjust the model and event log with other plug-ins and restart the analysis. This is especially inconvenient as the model and filters might need to adjusted very often during an analysis.

A very convenient feature of Celonis in this analysis was the option to hide any activity in the model. While Disco and Celonis both offer activity sliders, these remove activities based on frequency. So the option to hide any activity allows the user to hide activities that are very frequent, but not relevant in the analysis.

The analysis of this process will be continued in more detail, from a resource perspective, in the next section.

4.4. Performance and process flow analysis focusing on resources

Use case: Application of process mining techniques to provide performance information for resources and insights into how resources are working together.

Expectations:

- The tools should be able to mine and visualize statistics about the performance of the individual resources.
- The tools should be able to mine and visualize which resource completes which activities.
- The tools should be able to mine differences in the performance of a case, depending on which resources are involved.
- The tools should allow the user to find out more about how individual resources are working together.

The log of the repair process in the previous chapter also contains resource information, showing which employee or system executed each event. So this chapter will not introduce a new process but provide further analysis of the repair process from a resource perspective. This chapter will also use the filtered event log that only includes cases ending in 'Archive Repair' as the basis for analysis.
In the previous chapters Disco, Celonis and ProM were able to provide the same, or very similar, insights into a process. However, these tools approach resource based analysis differently so a direct comparison between the tools is more complex. This chapter will serve to give an introduction to analyses on resources in a process.

The following analysis is based on the background knowledge provided at the start of the last chapter, and also knowledge about the process gained in the performance analysis.

4.4.1. Resource based analysis in Disco

Disco offers useful information in the statistics tab under the resource view. The view in figure shows a graph and a table. The graph will visualize one of five available metrics about the resources (metrics can be chosen by clicking on them next to the graph).

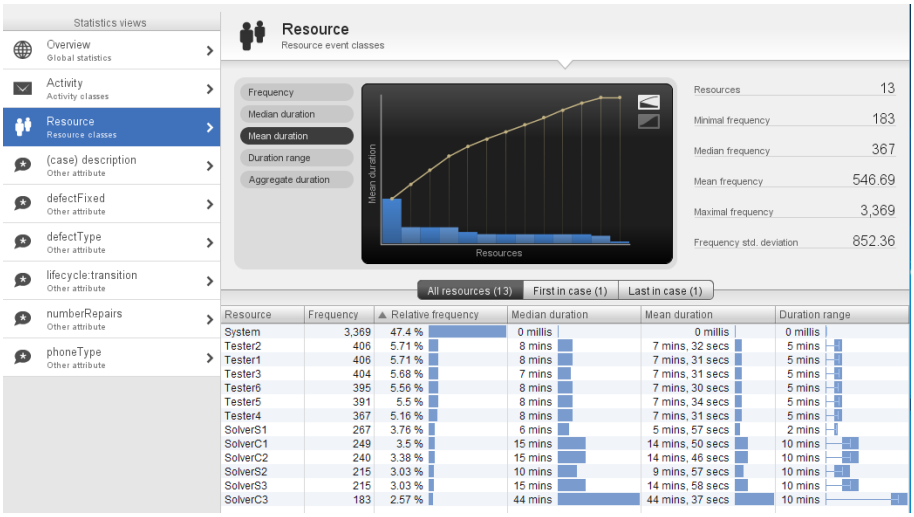

Figure 4.12.: The statistics tab in Disco showing the resource metrics

Starting the analysis by studying the 'Relatively Frequency' column, the resource that was responsible for handling the most events is the 'System'. The system handled 47,4% of all events and executed them instantly.

The next detail to notice is that the testing and repairs are not executed by a single team or individually named resources. Instead there seem to be three resources for each of the two repairs. These resources could be a single person or teams. A quick look at the frequency indicates that the first teams of the simple and complex repairs have a slightly higher frequency. There are also six 'Tester' resources for all of which the relative frequency is between 5-6 %. Sorting the table by 'Mean duration' shows that the mean duration for all testers is very close together ranging only from 7 minutes and 30 seconds to 7 minutes and 34 seconds. Based on their similar metrics it will be assumed that the 'Tester' resources perform the same tasks interchangeably which is why they will not be analyzed further.

The repair resources on the other hand show a difference in performance. Starting with the resources executing simple repairs, 'SolverS1' has the lowest mean duration with 5 minutes and 57 seconds, 'SolverS2' takes 9 minutes and 57 seconds on average, and 'SolverS3' takes the longest with 14 minutes and 58 seconds. This indicates that these three resources do not execute the repairs the same way. It could indicate that 'SolverS1' handles the simplest repairs, which might also be completed faster. It is also possible that 'SolverS1' attempts a fast and less thorough repair which might lead to additional repairs being necessary more often.

Likewise to the resources for the simple repairs,on average 'SolverC1' (14 minutes, 50 seconds) and 'SolverC2' (14 minutes 46 seconds) both take less time to attempt a complex

repair while 'SolverC3' (44 minutes 37 seconds). This might indicate that 'SolverC3' attempts a very thorough repair.

If it is true that 'SolverS3' and 'SolverC3' attempt the most thorough repairs, it would be expected that a repair that was attempted by these resources would not be handled by another resource executing the same repairs. In further analysis of the event log the following behavior would therefore be expected:

The resource 'SolverS3' is not followed by 'SolverS1' or 'SolverS2', and the resource 'SolverC3' is not followed by 'SolverC1' and 'SolverC2'.

Disco does not offer to visualize social networks like a handover of work network. Nevertheless it is possible test for this kind of behavior using different filter settings.

To test if the 'SolverS3' resource is ever followed by 'SolverS1' or 'SolverS2' a 'Follower' filter is created, specifying that only events are kept in which 'SolverS3' is eventually followed by 'SolverS2' or 'SolverS3'. Applying this filter leads to the message 'No cases passed your filter'. This confirms, that a repair that was attempted by 'SolverS3' will never be attempted by another resource for simple repairs afterwards. Following the same technique a filter is applied to test if 'SolverC3' is ever followed by 'SolverC1' or 'SolverC2'. Again, there are no resulting cases.

This implies that resources are specialized to either attempt faster or more thorough repairs, explaining the difference in their performance information. Applying another 'Follower' filter also reveals that 'SolverC1', 'SolverC2' and 'SolverC3' are never followed by 'SolverS1', 'SolverS2' or 'SolverS3'. So in this process a complex repair is never followed by a simple repair. Applying additional filters could provide additional insights about how these resources work together, but this will not be part of this thesis since it would not provide more information about the techniques Disco offers.

4.4.2. Resource based analysis in Celonis

In the first step the prebuilt 'Social' app is added to the analysis by clicking on the '+' in the bottom. If it was not done before the resource column needs to be declared. Resources are referred to as users in Celonis.

The PI Social app contains three tabs, 'Overview', 'Users' and 'Activities'. The 'Activities' tab in this app is the same as in the 'Process overview' tab. The top of the 'Overview' tab contains general metrics, the average amount of active users, average events per user per day, average cases per user, and average amount of users per case. At the bottom of the 'Overview tab' is a bubble plot, visualizing all resources as bubbles and coloring them either by their total event count or their average throughput time.

The throughput time calculated is not the actual throughput time for the events the user is executing but the average throughput time for cases the user is involved in. The 'User' tab also consists of this bubble plot. The unit of the throughput time is changed to minutes by clicking the gear symbol in the top right of the tab. When clicking bubbles in the bubble plot one of the upcoming options is to ignore the selected user. In this case it appears useful to ignore users responsible for testing and the system. All 'Tester' and the 'System' resources are selected and ignored.

Inspecting the bubble plot A.12 showcasing the total number of events on page 63 shows that 'SolverS1' and 'SolverC1' are completing the most repairs. The frequency numbers are doubled because the start and complete events are both counted towards this. Frequency and performance information for each user are only shown for the selected user instead of a table showing the information for all users. Visualizing the average throughput time produces the bubble plot shown below.

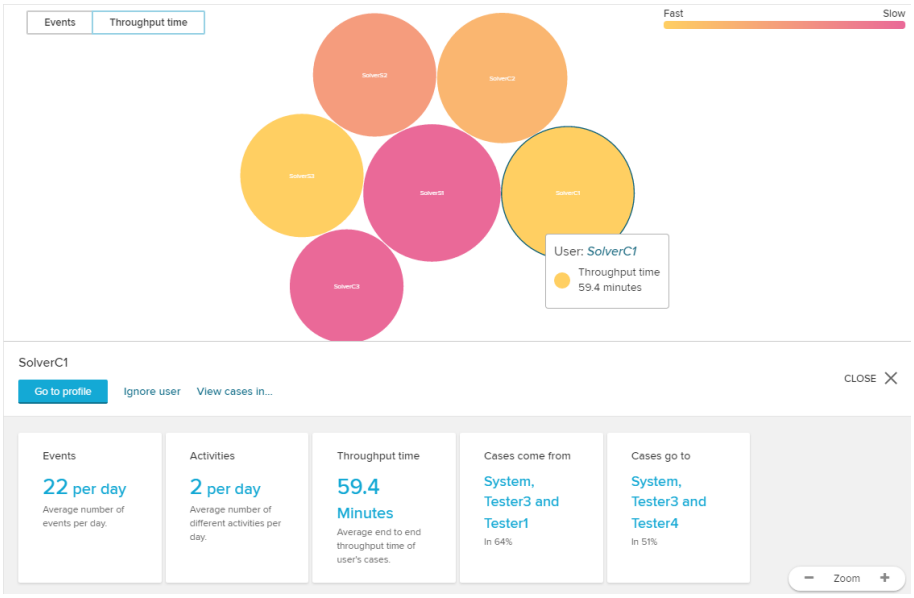

Figure 4.13.: Bubble plot from the 'Social' app in Celonis colored by the amount of total events resources completed

The values for the average throughput times are only shown when clicking on a resource and are not available to be viewed in a table in this app.

To give a better presentation of the throughput times they were extracted and are shown in the table below. The table is ordered by their average throughput time.

User	Throughput time
'SolverC1'	59.4 minutes
'SolverS3'	64.2 minutes
'SolverC2'	65.1 minutes
'SolverS2'	70.9 minutes
'SolverS1'	82.0 minutes
'SolverC3'	87.0 minutes

Looking at the average throughput time for the cases users are involved in shows that values range from 59.4 minutes for 'SolverS3', to 87.0 minutes for 'SolverC3'. This indicates that the user 'SolverC3' might either handle very complex repairs that might take long, or that the user is frequently involved in cases where complex repairs need to be repeated at least once. This can be tested by adding a 'Rework Selection' filter. The configuration for this filter is that cases will remain where the 'Repair(Complex) - start' activity is occuring at least two times. Applying this filter shows that a complex repair was executed more than once in only 49 cases. 'SolverC3' was only involved in 15 of these cases. So the longer throughput time for cases 'SolverC3' is most likely caused by repairs taking longer. It would be possible to analyze this further with the techniques Celonis offers. However, this will not be done in this thesis as it would not showcase additional techniques Celonis offers.

4.4.3. Resource based analysis in ProM

The 'Summary' tab available in ProM when inspecting a log serves well to get a first impression about the resources in a log. The tab shows a table showing the absolute and relative occurrences of resources. This table is shown in figure A.13 on page 64.

The table shows that there are three resources for each of the two repair variants and 6 resources that handle the testing of repairs. The testing resources all occur between 734 and 812 times. The first resource for simple repairs 'SolverS1' handles the most repairs, occurring 534 times, the other resources for simple repair both 430 times. For complex repairs the third resource 'SolverC3' occurs only 366 times.
Note These numbers are doubled again due to the start and event activities in the event log

The next step is to mine a social network in ProM. The filtered event log is used with the 'Mine for a Handover-of-Work Social Network' plug-in. Multiple transfers within one instance will be considered in the configuration to see all behaviour in the social network. For the same reason considering only direct succession is not enough because the resources handling the

repairs only work directly with the 'System' and the 'Tester#' resources. Instead the depth of the calculation will be set to twenty. This depth of calculation is chosen because the most events in any instance is 24 events(as the dashboard of the event log shows), and the first and last two events are always executed by the system and the testing resources. The resulting social network is hard to read initially because all 13 resources and their connections are shown, even though only the six resources responsible for the repairs are of interest. The other resources can not be hidden, but by switching the 'Mouse Mode' to 'PICKING' the 'Tester#' resources and the 'System' resource can be dragged on top of each other, reducing their impact on readability without losing out on information. The standard settings in the options to show the edges and vertex names in the network are kept. Additionally the option 'stretch by degree ration' is selected. Resources that hand more work to others than they are handed are compressed vertically. Resources that are handed more work by others than they hand out are stretched vertically. The social network created with this configuration can be seen below in figure 4.4.3.

Note: Resources can not be ignored in this network. Instead 'Tester#' and the 'System' resources were grouped manually because only the 'Solver' resources are relevant for the analysis.

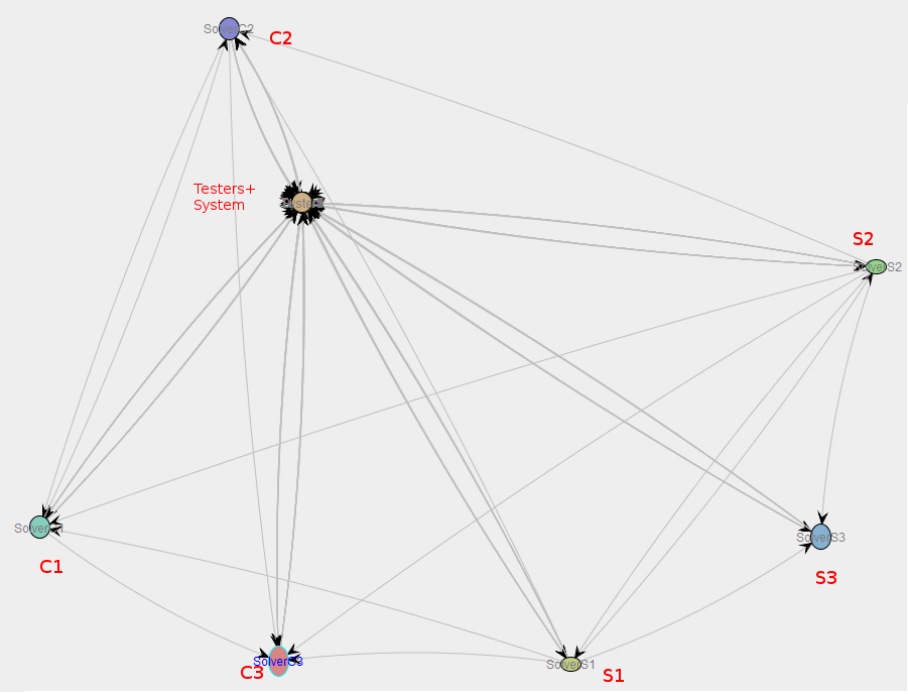

Figure 4.14.: Handover-of-work social network mined in ProM

Analysis of this network leads to the following conclusions.

- The resources 'SolverS1' and 'SolverS2' hand over work to others the most. This means that repairs being completed by these resources are most likely to need additional repairs. Both hand over work to all other resources that are handling repairs.
- The resource 'SolverS3' does not hand over work to 'SolverS1' or 'SolverS2', but is handed over a significant amount of work by them(as is shown by the stretching of 'SolverS3'). This resource is also not connected to any of the resources for complex repairs. This suggests that 'SolverS3' attempts to complete repairs that 'SolverS1' and 'SolverS2' could not repair correctly.
- 'SolverC3' Does not hand over work to other resources, but a significant amount of work is handed over from 'SolverS1', 'SolverS2', 'SolverC1' and 'SolverC2'. This suggests that, similarly to 'SolverS3', 'SolverC3' is the most qualified resource to handle a complex repair.

4.4.4. Conclusion

Disco and Celonis mined and showed performance statistics about the user. The tools showed different performance statistics with Disco showing the average time resources took to complete an activity and Celonis showing the average throughput times for cases resources were involved in. No tool showed both these statistics. Disco also offered the statistics in a table. In the prebuilt Social app in Celonis the user had to click each resource to find out their performance information. This would not be practical for a process with more resources and would likely require the building of a custom app. ProM does not include plug-ins that offer a comparable view on the process.
All tools were able to provide more information about how resources work together. ProM offered the mining of a handover of work social network, that showed this very effectively. ProM also includes three more plug-in for the mining of social networks that cover different aspects of resources working together. Disco offered addition of 'Follower' filters. With this type of filter the event log could be searched for specific follower relations of resources. While this required more steps than analyzing the social network in ProM, it added a good amount of detail to the analysis. Celonis offered process flow filters for activities, so it was not possible to filter for resource follower relations.

4.5. Conformance checking of an event log to find violations

Use case: Conformance checking of an event log to the model of the process Expectations:

- The tools should provide insights to showcase wether all cases are following the guidelines of the model. If not all cases are conforming to the model the tools should allow the user to inspect these cases.
- The tools should also showcase where in the process most violation occur.

In this section a real life event log from 4TU.nl is used. The event log[5] is from the receipt phase of an environmental permit application process. The model is from the Coursera course 'Process Mining: Data science in Action' that is taught by Wil van der Aalst [6].

For the following use case the model will be considered as valid. The model is a Petri net file that is required in ProM for a conformance analysis. Because of this the application of conformance checking techniques is described in ProM first. Celonis requires a BPMN model so the model will transformed in ProM with the 'Convert Petri net to BPMN diagram' plug-in by A. Kalenkova. Disco does not offer conformance checking techniques and is therefore not listed as a subsection.

4.5.1. Replay of an event log on a process model

The ProM plug-in used applied this section is based on the replay technique so a quick explanation of the technique will be given.
When replaying an event log on a process model it is attempted to simulate the event sequence observed in the event log on the process model. If it's possible to execute all events in the sequence in which they appeared in the process model all moves will by synchronous. When a synchronous move is not possible it is necessary to do a move only on the model or the event log. *Three types of moves can be made during replay:*

- Synchronous: This move will always be made if both an activity in the process model as well as an event in the current case can be executed

When it is not possible to do a synchronous move on event log and process model they are in *misalignment*. To get an alignments between event log and model again moves will be done on either just the event log or just the model.

[5] The event log available at data.4tu.nl
[6] Coursera course Process Mining: Data science in Action

- Move model only: One or moves are done on the model only to bring the event log and model back in alignment
- Move log only: One or more moves are done on the event log only to bring the event log back in alignment

The algorithm will decide between moves only on the log or model based on which requires less moves to get them back in alignment.

4.5.2. Conformance checking in ProM

The Petri net and the process log are imported and used in the 'Replay a Log on Petri Net for Conformance Analysis' plug-in by Arya Adriansyah. Initially there is no final marking, so it needs to be created. The place 'sink' is the only place that is added as a 'Candidate Final Marking'. The 'Map Transitions to Event Classes' has to be reviewed in order to make sure that the correct activities in the event log and model are matched. All 'tau' activities in the model must not be matched with an activity from the log.
In the next step a notification is displayed because a few event classes of the event log are not mapped to any transition in the Petri net. When 'No, I've selected all necessary event classes' is selected, the event classes will be ignored in the replay. In the basic wizard the default options are confirmed. The purpose of the replay is selected to measure fitness, improper completion will be penalized. The algorithm applied is the 'A*Cost-based Fitness Express with ILP, assuming at most 32767 tokens in each place' algorithm. The next step allows the user to set the costs for moves during the replay.
The costs for the three types of moves can be set. The default configuration is a cost of 1 for all moves on either the log or the model alone. The costs can be set for the three types of moves as a whole. Individual transitions in the model and events in the log can also be given high or lower costs. This can be very useful to penalize non execution of crucial events.For this analysis all moves on model or log only are set to 1.

The full resulting Petri net annotated with conformance information is shown in figure A.15 on 66. The moves that were made with that transition are displayed inside the transitions. A close inspection shows that only a few transitions show a significant number of moves. These are shown in figure 4.15 below.

The two types of moves are shown the following ways in every transition (model and log, model only). The colored bar below this also shows a the ratio between the two types of moves. Green shows synchronous moves while purple shows moves that were done on the model

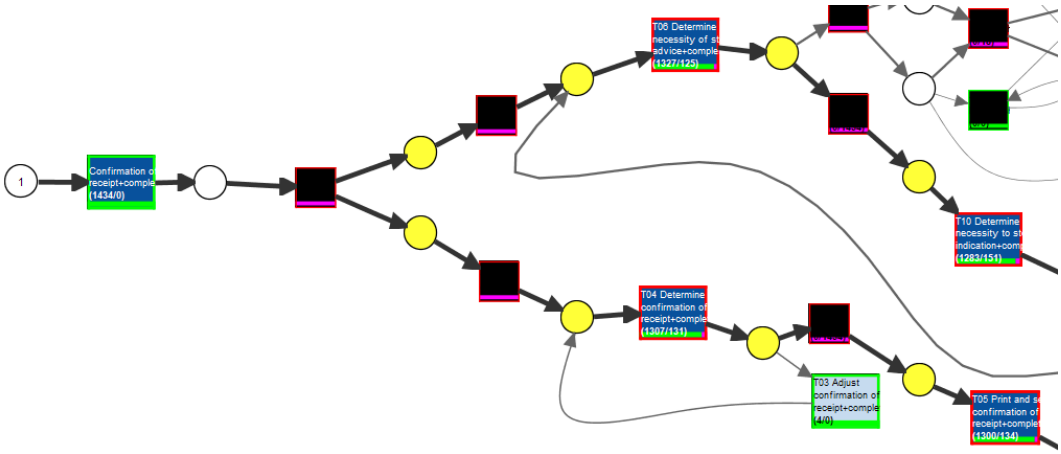

Figure 4.15.: Part of Petri net annotated with conformance information that includes most moves. Mined with the 'Replay a Log on Petri Net for Conformance Analysis' plug-in

only. Closer inspecting shows that for the transitions 'T04', 'T05', 'T06' and 'T10' moves on the model only were done 125 to 151 times in 1434 cases. From this it can be concluded that there are a significant number of cases that do not align with the model but no single transition stands out with especially much violations.

Switching to the 'Project Alignment to Log' visualization provides the visualization shown in figureA.16 on page 67.

Moves in the cases are colored according to the legend on the top right. Synchronous moves are shown as green, moves on the model only are shown purple and moves on the log only are shown yellow. Hovering over the events shows its name. There is no option to filter the cases by the amount of asynchronous moves that took place.

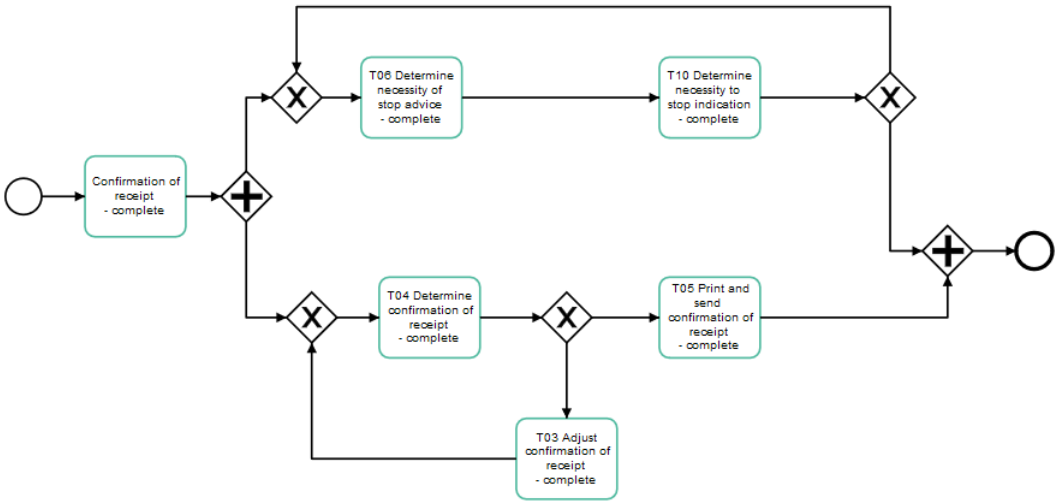

Figure 4.16.: Process model created in Celonis for conformance checking of the environmental permit application process log

4.5.3. Conformance analysis in Celonis

ProM offers a plug-in to convert a Petri net to a BPMN model. Unfortunately, the model produced by this plug-in does not describe the same behaviour and Celonis shows 100% violations when checking conformance. The BPMN model produced by the plug-in is shown in figure 4.16 on page 52. Instead of considering the whole process, the conformance analysis in this subsection will focus on a part of the process, consisting of the six activities that are also shown in figure4.15 (‘Confirmation of receipt’, ‘T04’, ‘T05’, ‘T06’ and ‘T10’).

The event log is filtered to include only these six activities by applying the ‘Filter Log using Simple Heuristics’ plug-in. The event log is imported into Celonis and a new default analysis is created. Next, a new ‘Conformance’ app is added the analysis. This app will prompt the user to either mine, upload, or create a process model. The following process model shown figure in 4.16 is created.

Launching the analysis with this process model produces the conformance overview shown in the figures A.17 and A.18 on page 68 and 69. This conformance overview shows that 94% of the cases are conforming to the model. 78 cases did not conform to the process model and caused at least one of the nine violations. The overview also provides a very insightful overview that shows that the conformance was relatively low, at 61.54%, at the start of the process history and improved over time with 100% of cases conforming to the model in the

last two months.
The key performance indicators (KPIs) in part two of the overview indicates that cases that caused violations took significantly longer to complete with an average throughput time of 13.8 days compared to 4.3 days for conforming cases. Below the KPIs is a list of the violations that occurred. The violations are ordered by their occurrence frequency and the KPIs that were shown for all violations are also shown for each violation, indicating which violations had the most impact on the performance of a case. Violations can be added to the 'whitelist', after which they will be ignored in the conformance analysis. In addition to this the user is able to view the cases that caused a violation in any of the apps of the analysis. This is very useful to find out why violations occurred and what impact they had.

4.5.4. Conclusion

Both tools were able so showcase that there is a significant number of cases(9%) that is not conforming to the model. ProM showed which activities are being skipped or executed out of order by annotating the Petri net with conformance information. ProM also provided a list of cases that caused conformance violations.
As might be expected of a commercial tool Celonis provided valuable performance information in addition to the conformance information. This indicated that conformance improved over time, but had a significant impact on the performance of the case.

5. conclusion

Each of the three tools is able to provide very valuable insights into processes and each of the uses cases ended with a short conclusion, highlighting some strengths and features in the tools. This chapter will give a short summary of these and draw a conclusion for each of the tools.

5.0.1. Disco

Disco is a powerful tool that is beginner friendly and applies process mining techniques like process discovery and performance analysis efficiently. The information mined in Disco is presented well as was shown in the use case on performance analysis 4.3. The large selection of filters profoundly enhanced the insights Disco provided and made up for the fact that Disco does not provide more advanced process mining techniques like conformance checking or the mining of a social network.

5.0.2. Celonis

The full potential of the insights Celonis provides become more apparent when considering that only prebuilt apps were used in this thesis. These already offered useful insights into the process ranging from discovery the process model, with the desired level of detail, to the conformance analysis use case treated in section 4.5. A app customized app could provide even more detailed insights and once created, it can be used throughout the life of the process.

5.0.3. ProM

Throughout the thesis the plug-ins available in ProM were able to provide useful insights into the process. In some use cases this was not as convenient as it was in the commercial tools because adjusting filtering options or the configuration of the plug-in required the plug-in to be initiated again. However, the insights ProM were able to provide are put into perspective when considering how few plug-ins were applied in this thesis compared all the plug-ins available in ProM Lite 1.2 and so much more plug-ins in ProM 6-8.

A. Figures

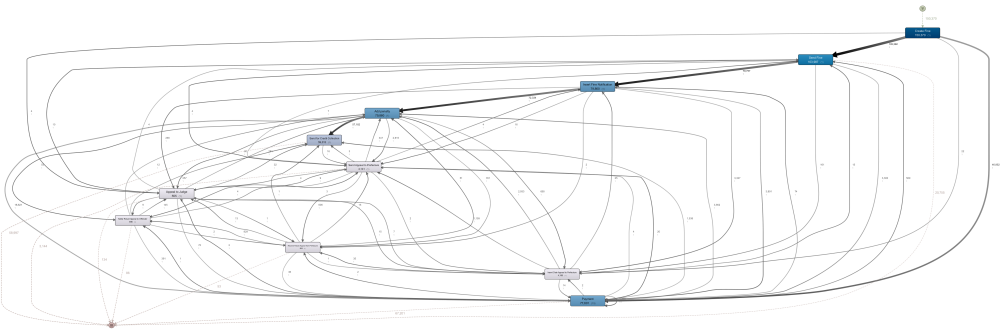

Figure A.1.: process model of the road traffic fine process discovered in Disco, showing all activities and connections

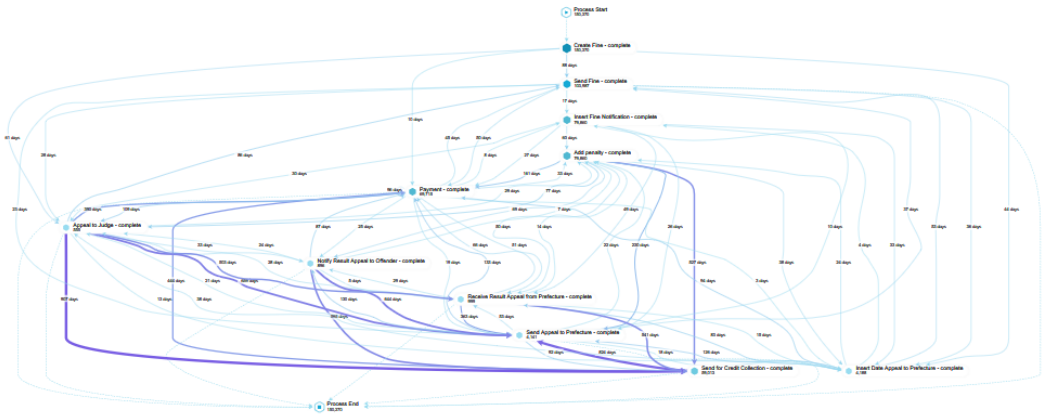

Figure A.2.: process model of the road traffic fine process discovered in Celonis, showing all activities and connections

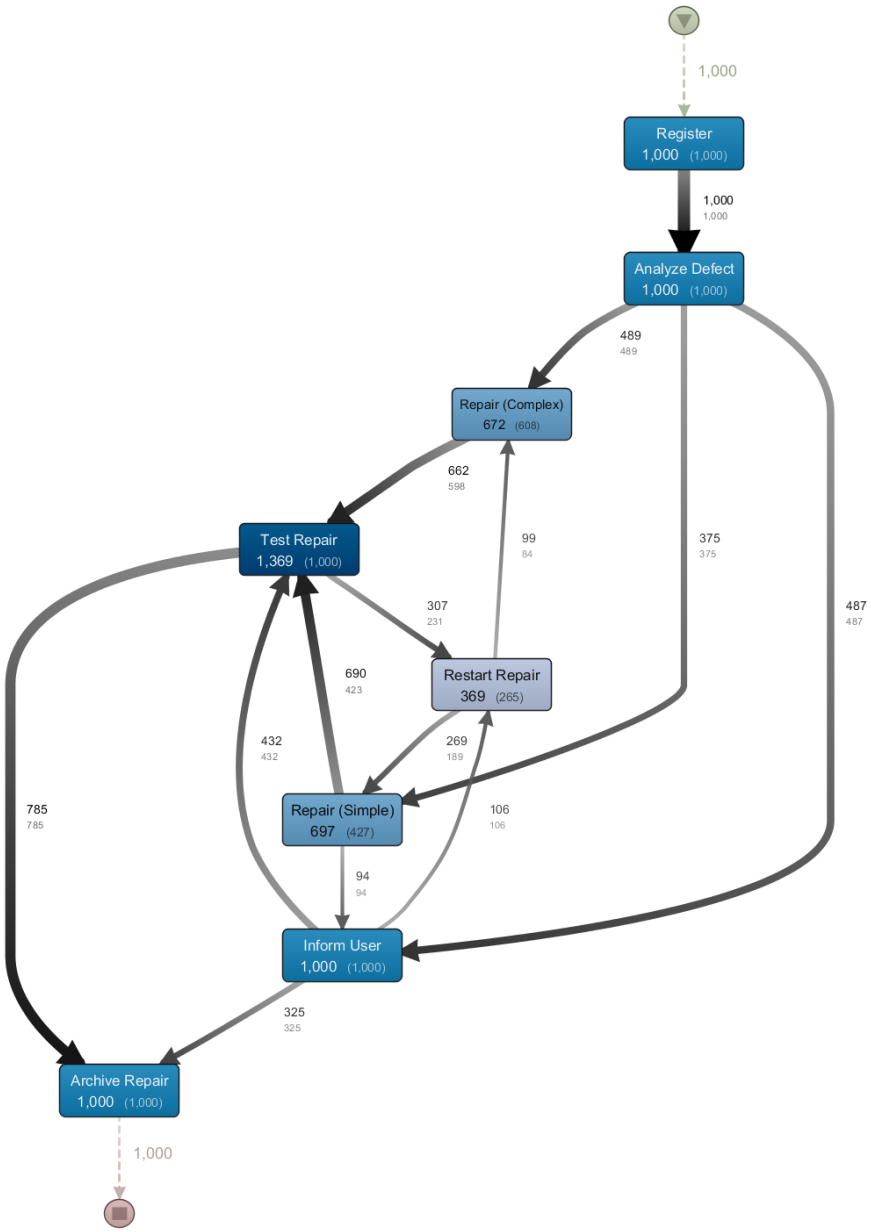

Figure A.3.: Process model in disco with the absolute frequency as the main metric and the case frequency as the secondary metric.

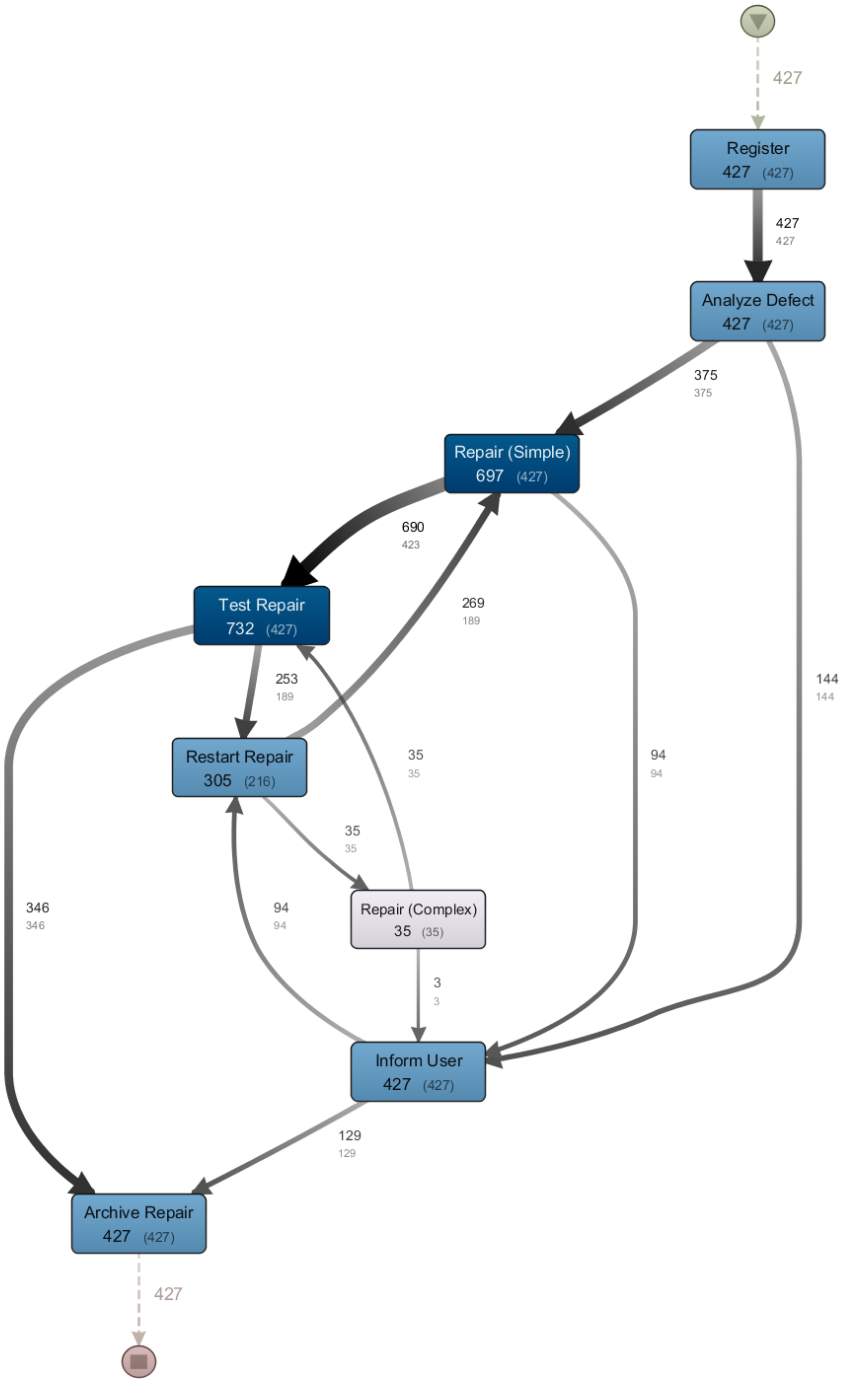

Figure A.4.: Process model in Disco only showing cases in wich 'Analyze Defect' was directly followed by 'Repair(Simple)'. Main metric-absoltue frequency, secondary metric-case frequency .

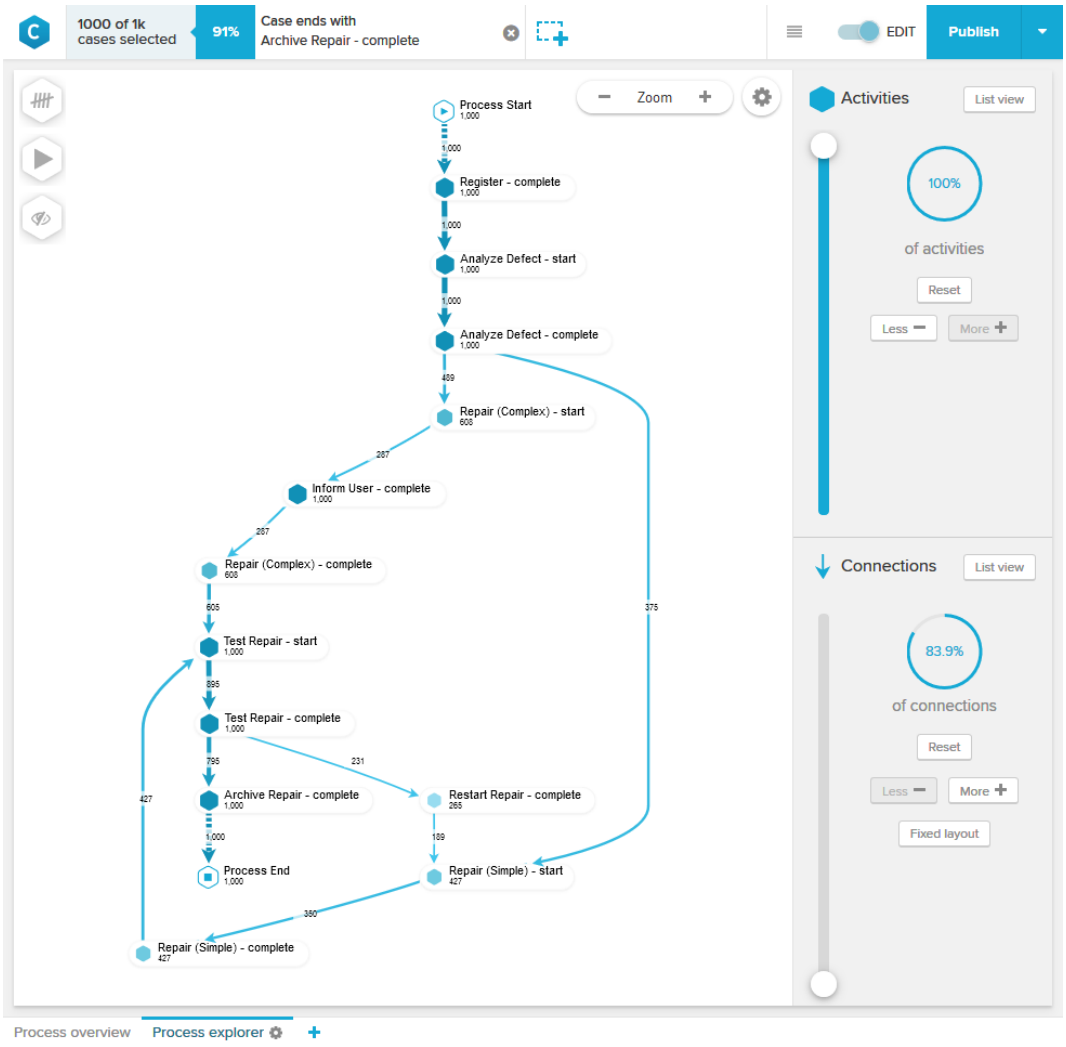

Figure A.5.: Process model of the repair process in Celonis with the activity slider at 100% and the connections slider at the lowest setting at 83.9 % .

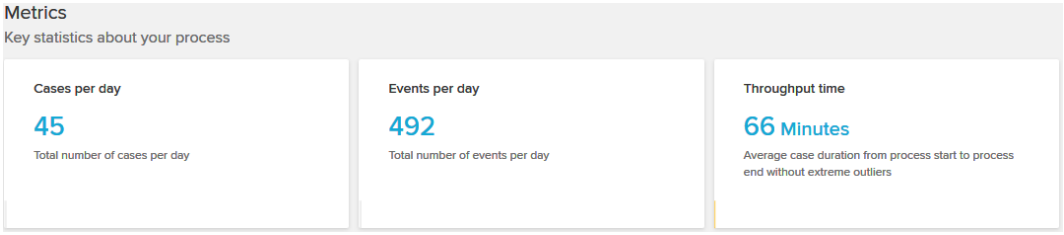

Figure A.6.: Part of the Process overview in Celonis showcasing the average case duration in the repair process is 66 minutes.

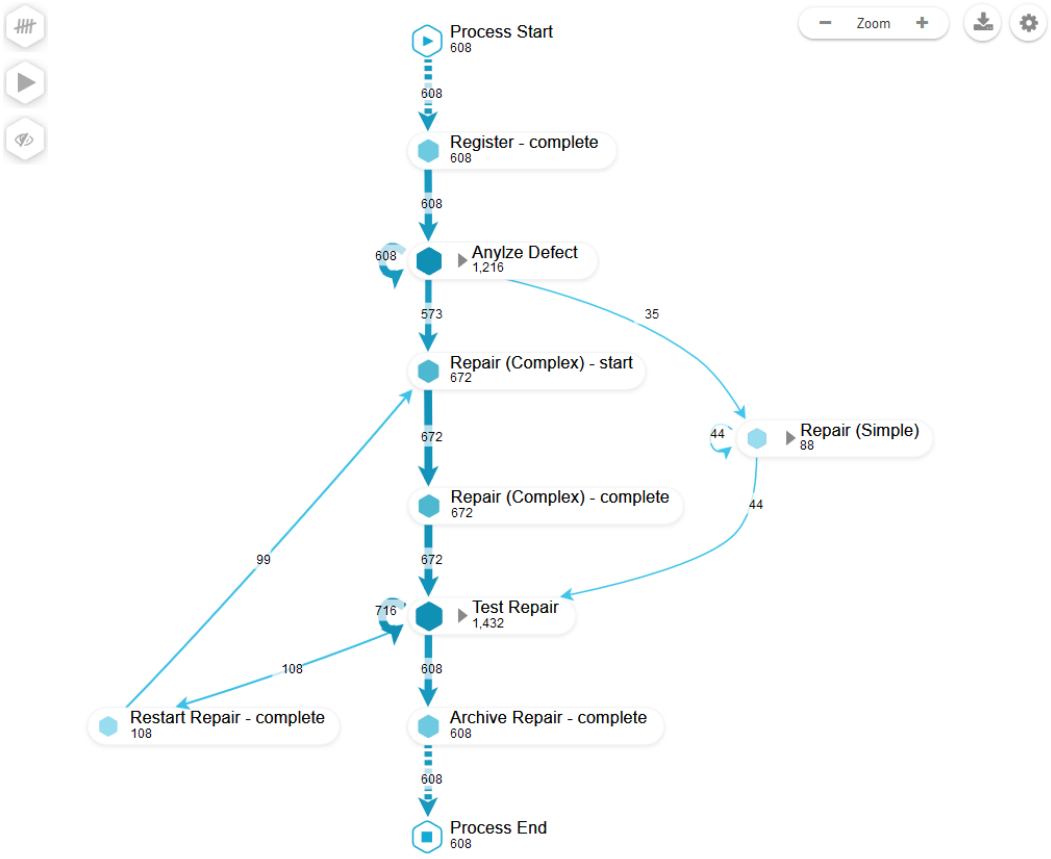

Figure A.7.: The process model shown in Celonis after removing all cases where the ‘Analyze Defect’ activity is directly followed by the ‘Repair (Simple)’ activity.

Figure A.8.: The correct selection of patterns in the 'Replay a Log on Petri Net for Performance/Conformance Analysis' plug-in in for the repair process.

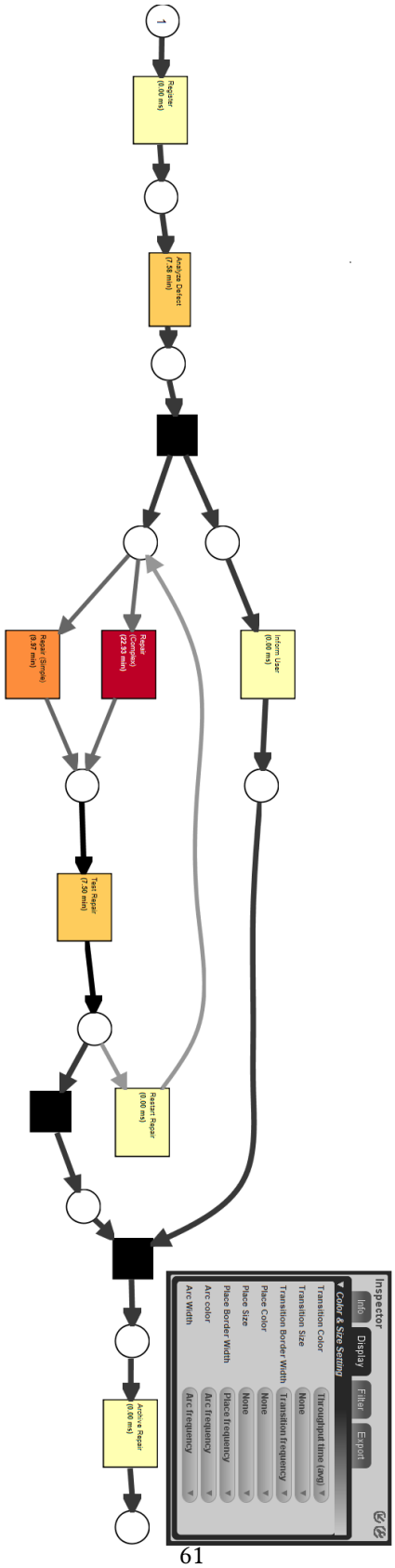

Figure A.9.: Model of the repair process with the average throughput time projected onto events. Created in ProM by the 'Replay a Log on Petri Net for Performance/Conformance Analysis' plug-in.

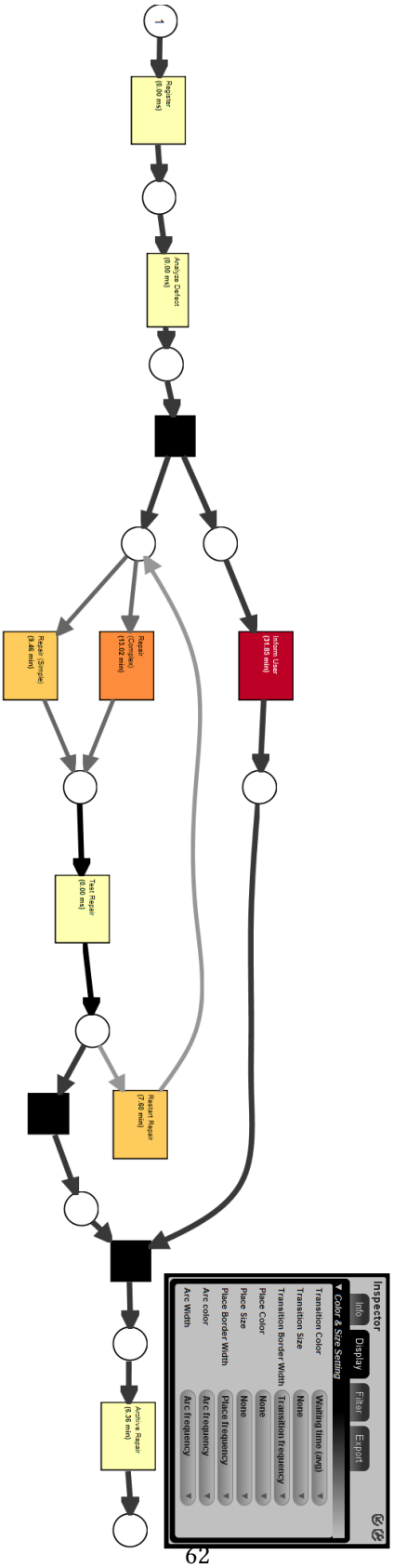

Figure A.10.: Model of the repair process with the average waiting time projected onto events.

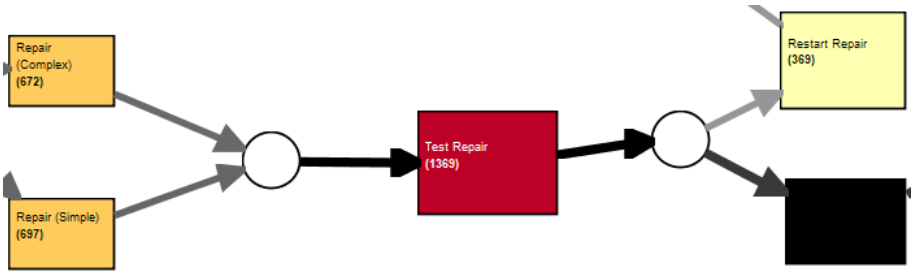

Figure A.11.: Part of the model of the repair process with the transition frequency projected onto events.

Performance and process flow analysis focusing on resources

Figure A.12.: Bubble plot from the 'Social' app in Celonis colored by the amount of total events resources completed

Resource

Event classes defined by Resource
All events

Total number of classes: **13**

Class	Occurrences (absolute)	Occurrences (relative)
System	3369	31.065%
Tester2	812	7.487%
Tester1	812	7.487%
Tester3	808	7.45%
Tester6	790	7.284%
Tester5	782	7.211%
Tester4	734	6.768%
SolverS1	534	4.924%
SolverC1	498	4.592%
SolverC2	480	4.426%
SolverS2	430	3.965%
SolverS3	430	3.965%
SolverC3	366	3.375%

Figure A.13.: Table showing the occurences of resources for the repair process in ProM

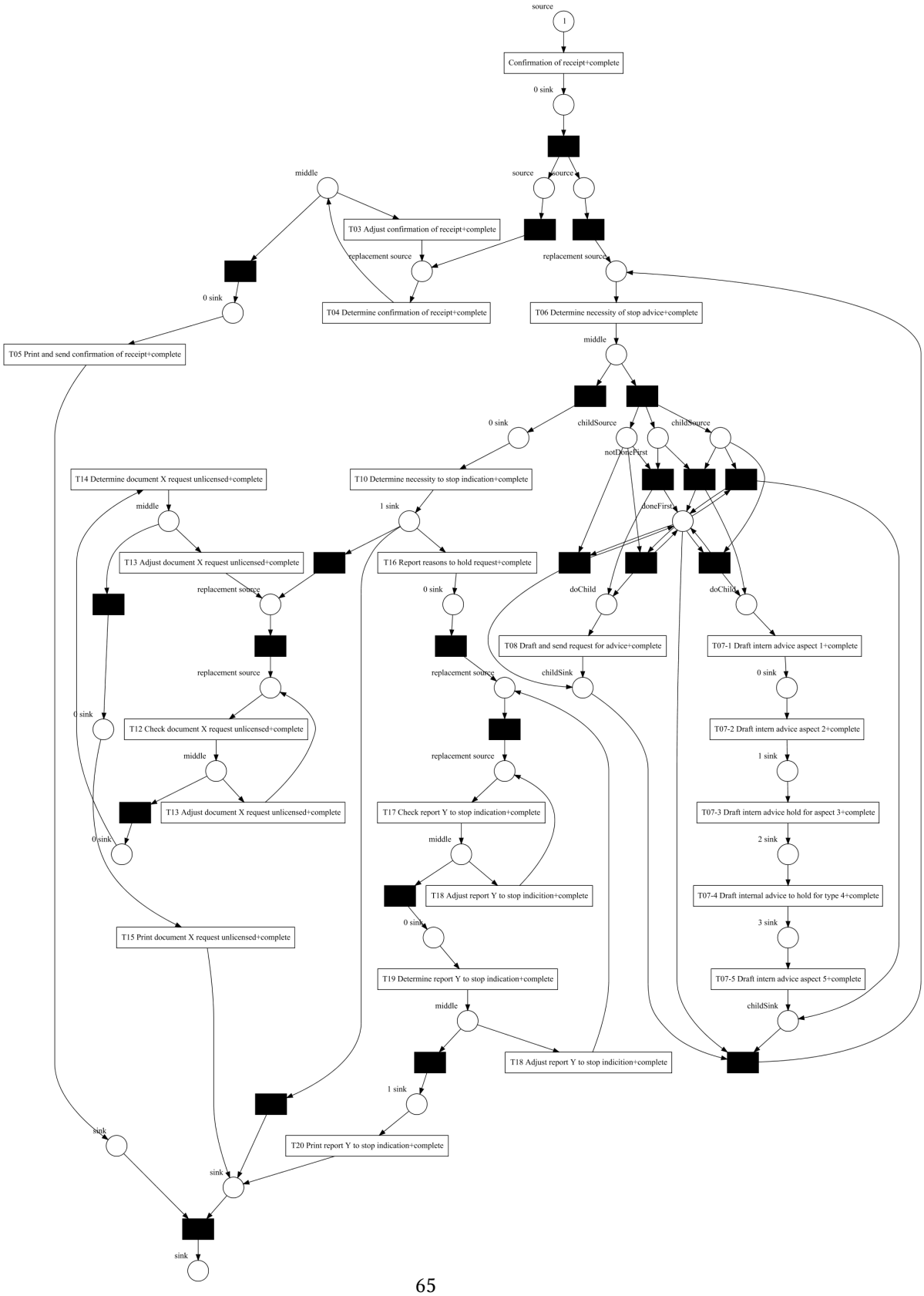

Figure A.14.: Petri net showing the receipt phase of an environmental permit process

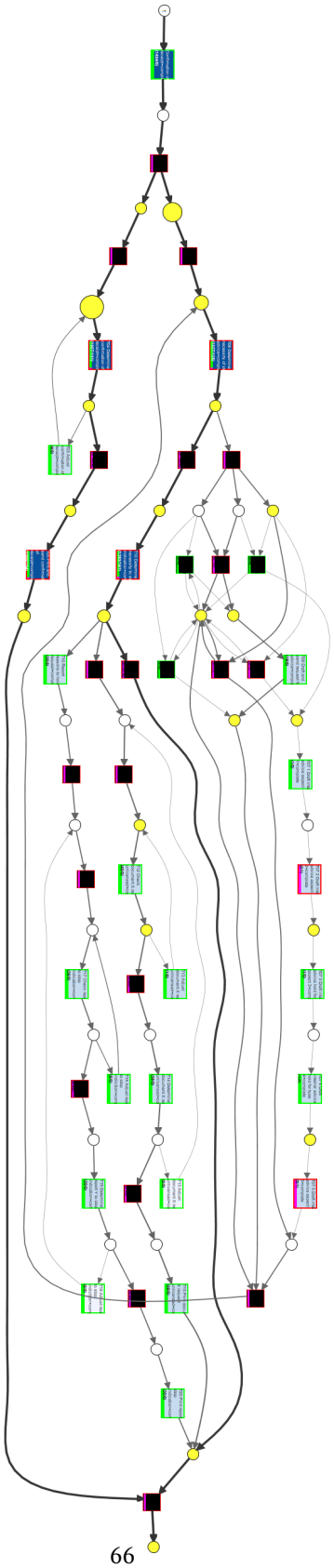

Figure A.15.: Petri net showing the receipt phase of an environmental permit process

Figure A.16.: 'Project Alignment to Log' of the 'Replay a Log on Petri Net for Conformance Analysis' plug-in applied to the environmental permit application process event log and model.

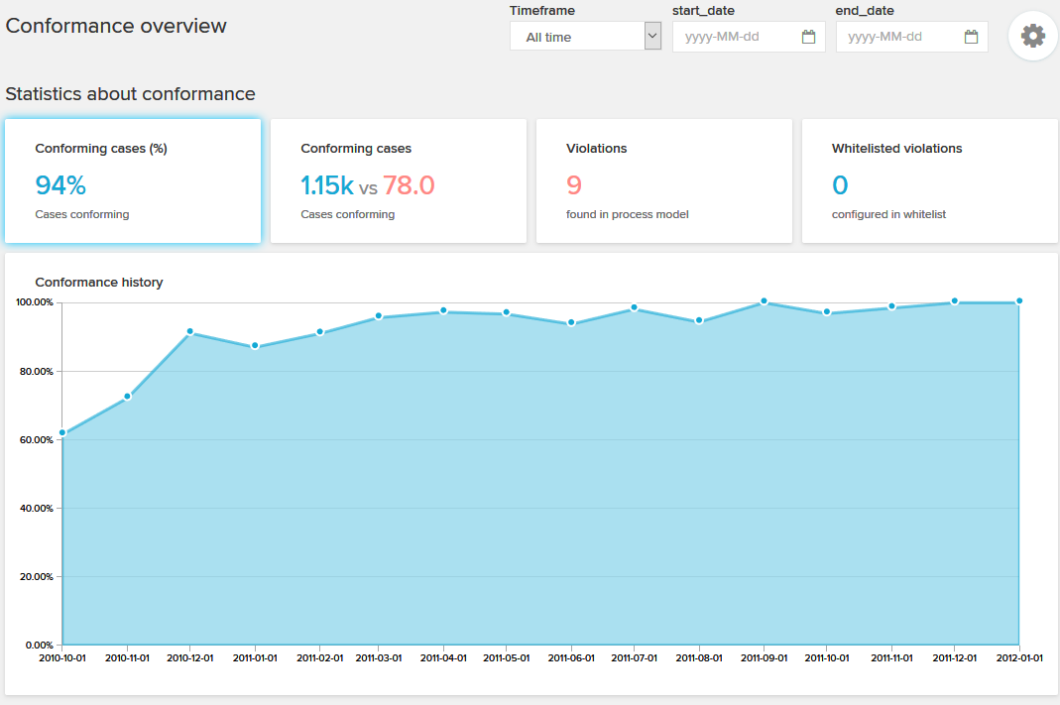

Figure A.17.: Part 1 of the conformance overview for the filtered environmental permit application process

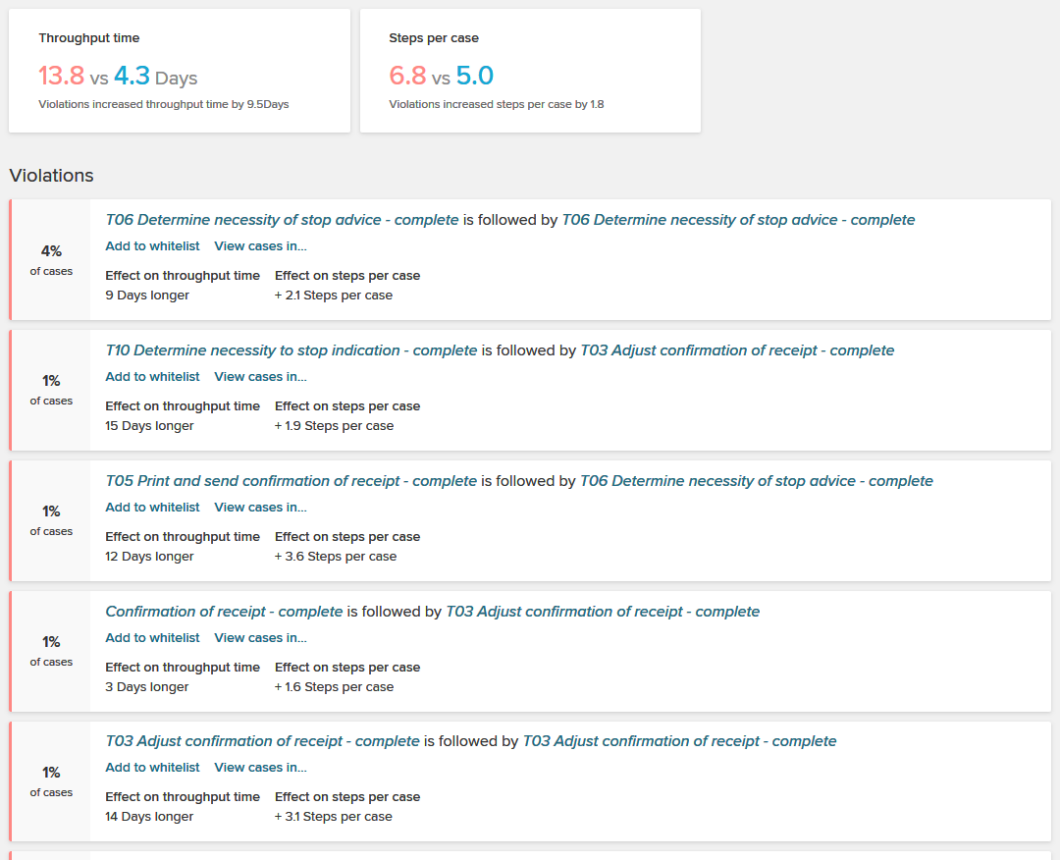

Figure A.18.: Part 2 of the conformance overview for the filtered the environmental permit application process

Bibliography

Cockburn, A. (2001). Writing effective use cases.

Leemans, S. (2017). *Robust process mining with guarantees*. PhD thesis.

Mannhardt, F. and Blinde, D. (2017). Analyzing the trajectories of patients with sepsis using process mining.

Mannhardt, F., de Leoni, M., Reijers, H. A., and van der Aalst, W. M. P. (2016). Balanced multi-perspective checking of process conformance. *Computing*, 98(4):407–437.

van der Aalst, W. M. P. (2014). Process mining in the large: A tutorial. 172:33–76.

van der Aalst, W. M. P. (2016). *Process Mining: Data Science in Action*. Springer, Heidelberg, 2 edition.

Veit, F., Geyer-Klingeberg, J., Madrzak, J., Haug, M., and Thomson, J. (2017). The proactive insights engine: Process mining meets machine learning and artificial intelligence.

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, July 11, 2018 David Karwehl