Interactive visualization of business processes

3TU. School for Technological Design

STAN ACKERMANS INSTITUTE

Interactive visualization of business processes

Eindhoven University of Technology

Stan Ackermans Institute/ Software Technology

TU Delft
Delft University of Technology

TU/e Technische Universiteit Eindhoven University of Technology

UNIVERSITY OF TWENTE.

Interactive visualization of business processes

Eindhoven University of Technology

Stan Ackermans Institute/ Software Technology

Partners

UWV

Eindhoven University of Technology

Steering Group AlbertoVasconcellos

Marcus Dees

Massimiliano de Leoni

Date September 2015

Partnership This project was supported by Stan Ackermans School of Design and UWV.

Contact Eindhoven University of Technology

Address Department of Mathematics and Computer Science

MF 5.097b, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands

+31402474334

Published by Eindhoven University of Technology

Stan Ackermans Institute

Printed by Eindhoven University of Technology

UniversiteitsDrukkerij

ISBN A catalogue record is available from the Eindhoven University of Technology Library
ISBN: 978-90-444-1383-0
(Eindverslagen Stan Ackermans Instituut ; 2015/045)

Abstract Today's Process-Aware Information Systems (PAIS) logs huge amount of data. The data contains a set of activities that are actual executed with in a business process. For example place a request for unemployment claim or pay compensation. This is a starting point for doing the analysis of a business process using Process Mining techniques. In Process Mining, one technique named LogOnMapReplay, which is dynamically visualizing the executed business processes by producing a process movie. This tool is a prototype that's why it comes along with various limitations and missing functionalities. This report describes the steps taken to overcome the limitations of the tool. It also describes design and implementation of various functionalities developed on LogOnMapReplay tool. This report also describes an evaluation conducted on UWV unemployment business process to find the usefulness and intuitiveness of the tool.

Keywords Process mining, ProM framework, XES standard, Map, LogOnMapReplay plug-in, XQuery, Petri-net.

Preferred reference Neha Gupta, Interactive visualization of business processes.
Eindhoven University of Technology, SAI Technical Report, September, 2015
(ISBN: 978-90-444-1383-0)

Foreword

UWV is the governmental institute that executes the social security laws in the Netherlands. UWV is responsible for handling claims of employees that become unemployed, become ill or whose labor capacities change due to an illness or an accident. UWV provides benefits when an employee has an entitlement and helps with the reintegration of employees into a fitting job.

Customers of UWV are entitled to benefits. They are also obliged to fulfil certain responsibilities. When a customer is not compliant with the rules, then a reclamation can occur. A reclamation is the situation where the customer has received more benefits that he was entitled to.

A change in the number of reclamations was the trigger to form a project group to investigate what has been causing this change. Next to that the project should identify and quantify the root causes of the reclamations themselves.

Finally, when the root causes are identified, a monitoring system should be created. This system should act as an early warning system when the number of reclamations starts changing again or when there is a change in the distribution of the root causes.

The ProM plug-ins presented in this report are the result of these requirements. With the plug-ins UWV is able to execute the monitoring of the reclamations and perform high level analysis of the reclamations when changes are detected. The plug-ins are very generic. This means that they have a wide application area. The plug-ins represent a valuable extension of the Process Mining toolbox.

Marcus Dees

September 2015

Preface

This report summarizes the technical report of Interactive visualization of business process project. The project was carried out by the author as her final project of the Professional Doctorate in Engineering (PDEng) program in Software Technology provided by the Eindhoven University of Technology and the Stan Ackermans Institute.

The target audience of this report is a technical audience and a process analyst with a basic understanding of the software, business processes and an interest in Process Mining. Chapter 1 and 2 gives an overview of the project context and the stakeholders of the project. The core concepts of the project is described in Chapter 3 and 4. It is recommend for every reader because it explains the key concepts related to process mining and highlights the limitations of the existing system. The technical reader should read chapter 5, 6, 7, and 8 to understand how the system is designed and implemented. Chapter 9, summarizes of the evaluation of the system using UWV's business processes data. Chapter 10, concludes the project by summarizing the results objected and by identifying the possible future work. Chapter 11 describes the details related to the project management process.

Neha Gupta

September 2015

Acknowledgements

This project would not have been possible without the support of many people.

I wish to express my sincere gratitude to my Project Steering Group members for their guidance and advice during the course of the project. A special thank you goes to my supervisors AlbertoVasconcellos, Marcus Dees, and Massimiliano de Leoni. This project was successfully completed with their continuous support in every step of the project. I enjoyed our discussions that has always been in an encouraging and learning atmosphere for me. I would like to thank Marina Sereguina for the discussion and valuable contribution during the evaluation phase of the project. A special thanks to Flex Mannhardt for the discussions and valuable contribution to the project.

I want to thank my colleagues from OOTI generation 2013 for the great time we spent together. I would especially like to thank Ad Aerts, and Maggy de Wert for giving me the opportunity to join the OOTI program and for their guidance and support throughout the whole OOTI period.

No such project can be completed without family support. I am grateful for the continuous support from my family. Tarun, your unconditional love, support and unflagging encouragement gives me the strength and motivated me to move forward in life. No words can express, how lucky I am to have you in my life. I want to thank my parents, and my parent's in-law for their unconditional support and positive energy. My report is dedicated to my grandfather. He is a truly an inspiration for me.

Neha Gupta

September, 2015

Executive Summary

Today's information systems store huge amount of data. For example data logs during the process of buying an airplane ticket or of applying for a claim. The managers of a company want to know the steps of execution and potential scope of improvement in the business processes. In the context of business processes, the analysis of data requires Process Mining techniques to gather meaningful information.

Process mining allows extraction of knowledge about business processes from the event logs. In process mining, there is a visualization technique called "Turning event logs into a process movie". This technique enables a process/business analyst to replay and visualize the behavior of executed events as recorded in the logs. The LogOnMapReplay is a plug-in that provides this functionality. The plug-in is realized in the Process Mining open-source framework called ProM framework. UWV is interested in analyzing the event logs of its customers on LogOnMapReplay visualization plug-in. However, the plug-in has several constraints, which limits its usability:

- The plug-in stores processed data in-memory. The plug-in requires more than 6 GB memory space to process three hundred thousand events on a laptop equipped with an Intel Core i7 Processor at 2.20GHz.
- Basic functionalities such as filtering data are not available in the plug-in.
- Main input file called Map file is manually created. Manual creation requires technical guidance and knowhow.

The focus of this project is to improve the scalability and enhance the features of the LogOnMapReplay plug-in. The main challenge of the project was to improve the processing capability of the tool multifold. To achieve this goal, we analyzed several options and chose a NoSQL database called MapDB. MapDB stores the data on a local or network disk. Thus, the limitation of processing number of events moved from in-memory size to the amount of space available on-disk.

The features such as filtering events and attributes while visualizing a process movie are introduced. The *comparative analysis* concept is also introduced to detect unusual activities and show relativity of an event.

We also present the design and implementation of a new plug-in for automating the process of generating a map file. We conducted an evaluation of the plug-ins on UWV reclamation data, to analyze the usefulness and intuitiveness of both plug-ins. The evaluation conforms to the expectations of the analysts. We also propose few suggestions that a process/business analyst should consider while creating an event log and processing the data using the LogOnMapReplay plug-in.

Table of Contents

List of Figures

List of Tables

1. Introduction

This report describes the technical details for the "Interactive visualization of business processes" project performed at Uitvoeringsinstituut WerknemersVerzekeringen (UWV). This chapter provides an introduction to the project context and UWV organization structure.

1.1 Context

This project was conducted by Neha Gupta as a part of her Professional Doctorate in Engineering (PDEng) in Software Technology program at Eindhoven University of Technology (TU/e). The project is carried out in collaboration with UWV and Architecture of Information Systems group of the Mathematics and Computer Science department of TU/e.

Process-Aware Information Systems (PAIS) are increasingly used by organizations including UWV to support their business processes. Every PAIS records an execution of process instances in event logs. These logs capture information about activities performed in an organization, for example a customer applies for an unemployment benefit. Every event records the execution of an activity instance by a given resource. It also records the execution time along with the status of an event such as start, complete.

Process mining techniques allows extraction of knowledge about business processes from the PAIS. In process mining, there is a visualization technique called "Turning event logs into a process movie" with which a process analyst can visually replay the behavior of executed process instances as recorded in the event log. With minimum knowledge of process mining, process analysts can gain insights into potential problems of their business processes. UWV is interested in analyzing the event logs of its customers on LogOnMapReplay visualization tool to streamline its own business processes.

1.2 UWV

UWV is the Social Security Agency of the Netherlands. It is responsible for the implementation of employee insurance such as unemployment benefits and sickness benefits in The Netherlands. UWV also fulfills the important social task of helping people stay employed or find new employment.

As Figure 1 shows, UWV has expertise, knowledge and experience within five key tasks:

1. **Employment**: Helping the client remain employed or find employment, in close cooperation with the municipalities.
2. **Social Medical Affairs**: Evaluating illness and labor incapacity according to clear criteria.
3. **Benefits:** Ensuring that benefits are provided quickly and correctly if work is not possible or not immediately possible.
4. **Client & Service:** Ensuring that a customer's queries are answered quickly, clearly and with unambiguous answers. They are also the driving force behind improvements in UWV's other division services.
5. **Data management**: Ensuring that the client needs to provide the government with data on employment and benefits only once.

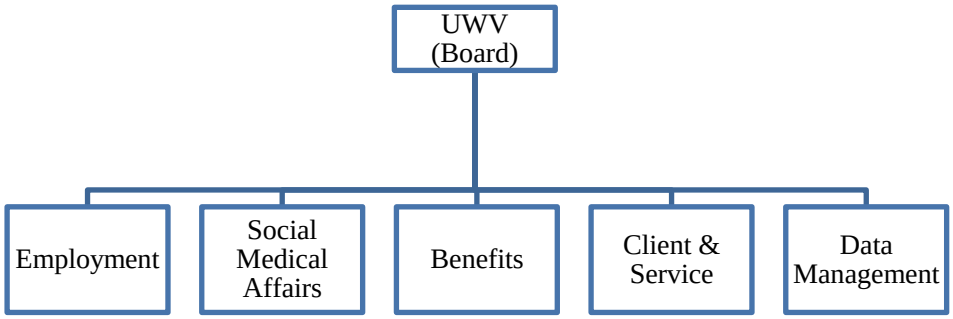

Figure 1: UWV core business chart

1.3 Research on Reclamations

This project "Interactive visualization of business processes" is a part of the main project "Research on Reclamations" as shown in Figure 2.

To understand the project context, we are going to use an example. Mr. Heezik became unemployed. He applies for an unemployment benefit at UWV. The company will evaluate the current condition of the Mr. Heezik and decide whether he is eligible for the benefit or not. If he met all the terms and conditions for receiving an unemployment benefit then UWV starts paying him.

Suppose, after three month Mr. Heezik found a job but he forgot to inform or intentionally did not inform UWV about his employment. Later, from internal or external data sources, UWV found that he got the job but in the meanwhile UWV has already paid excess benefits. This means he didn't comply with the UWV rules and regulations and, hence, he has to pay back the benefits which he was not entitled to receive. A fine will also be added of at least €150 and at most 100% of the benefit. Claiming unentitled benefits and fine from customer is called reclamation.

Since the number of reclamations has significantly increased in the last two years, the Client and Expertise department of UWV is doing research on reclamations that usually occur.

In this project, we are enhancing the capability of LogOnMapReplay plug-in, so that UWV's process/business analyst can easily find the business processes and customer's patterns that happens before reclamations occur.

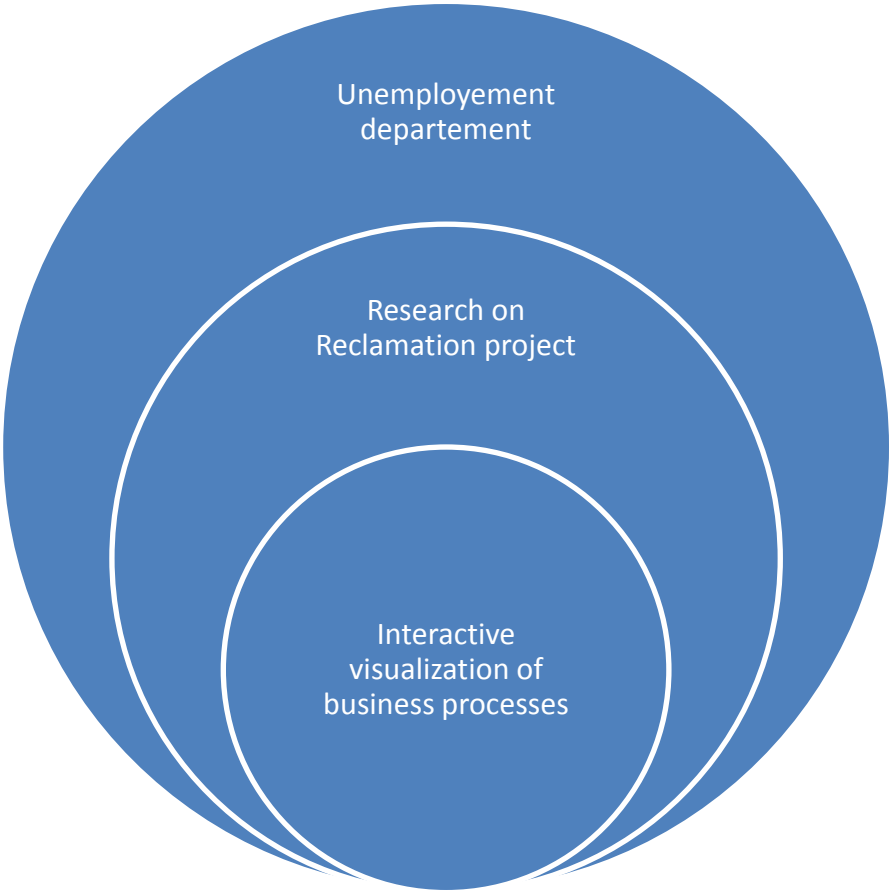

Figure 2: Project organization context

1.4 Limitations of existing implementation

After various experiments and discussions with stakeholder the following constraints were observed and listed, which limits its usability the tool:

- The capability of handling the processing of the big data. UWV's system produces and stores a huge amount of data, roughly three million events per month. This triggers to have an improved version of this tool which has capabilities to handle big data. The plug-in stores processed data in-memory. Due to memory limitations, the plug-in cannot process more than three hundred thousand events on a laptop equipped with an Intel Core i7 Processor at 2.20GHz.
- The basic features such as filter data, comparative analysis are not available in the tool. Due to these missing features the process analyst is unable to do the effective analysis of an event log.

- The main input file called Map file is manually created. Manual creation requires technical guidance and knowhow. It takes significant amount of time to create one map.
  To overcome the limitations of the tool became the primary goal of the project. The secondary goal is to provide the insights on how this tool can be used to analyses the UWV's business processes.

1.5 Experiments

Various experiments were conducted with UWV's business analyst to evaluate the usefulness, effectiveness, and intuitiveness of the LogOnMapReplay tool. We also evaluate whether a business analyst can gain the insights and analysis patterns from an event log. We use reclamation and unemployment business processes data from year 2013 and 2014 for analysis. During experiments, we identify patterns in the unemployment business process. We also identify the correlation between various attributes available in an event log.

1.6 Document outline

This section provides an overview of the chapters as follows:

Chapter 2 introduces the stakeholders and explains their interest and influence on the project.

Chapter 3 provides the necessary domain knowledge. It includes process mining concepts, event logs, petri-net model and an overview about "LogOnMapReplay" tool. This chapter helps reader to build an understanding for the later chapters.

Chapter 4 gives an overview of the problem analysis including the problem definition, the goal of the project, and also the problem of the existing plug-in "LogOnMapReplay" in the ProM framework and how the requirements has been derived from its limitations. This chapter also describes the functional and non-functional requirements of the project and the list of project deliverables.

Chapter 5 describes the system architecture.

Chapter 6 describes the details of the design and implementation of Automatic Map Generator (AMG) plug-in.

Chapter 7 describes the existing bottlenecks of the LogOnMapReplay plug-in. This chapter also focus on how the system is re-designed and implemented to improve the scalability of the plug-in.

Chapter 8 describes the new features that are designed and implemented of the LogOnMapReplay and MergeMapFile plug-ins.

Chapter 9 illustrates how the LogOnMapReplay and AMG plug-ins helped in proving the insights that the business analyst had discovered during the Research on Reclamation project about the UWV business processes and customer's profile.

Chapter 10 provides a summary of the project results, the conclusion and suggestion that were derived from the results and recommendation for future work.

Chapter 11 presents an overview of the project management process followed to achieve the desired results on time,

2. Stakeholder Analysis

2.1 Introduction

This chapter introduces the stakeholders for this project. The stakeholders are distinguished based on their interests and Table 1 presents an overview. The participation and support from all stakeholders was crucial to the project's success.

2.2 UWV

The Research on Reclamation project, which is mentioned earlier is of the Client and Services department of UWV. The project team is interested in a visualization tool. UWV's stakeholders, listed in Table 1, are people who will be the first users of the visualization tool.

Table 1: UWV stakeholders

Stakeholder Name	**Role**	**Description**	**Interest**
Mr. AlbertoVasconcellos	Project Manager	He is intermediator between the project team and UWV management. He helps in identifying the requirements and defining the problem of the project. He evaluates the performance and progress on monthly's basis.	He is interested in the value addition to the business processes as a result of the data analysis performed using the visualization tool.
Mr. Marcus Dees	Liaison	He is the connection between the software engineering and business analyst teams. He provides the knowledge of UWV's business processes. He helps in defining the requirements and business rules of the project. He also monitors and evaluates the performance on a weekly basis.	He is interested in a full-featured visualization tool with big data processing capabilities. He is also interested in a document that highlights the features and improvements in the visualization tool.
Miss Marchella Maria	Business analyst	She helps to define requirements for the visualization tool.	She is interested in features that allow quick data

		She also helps in creating the dataset with Marcus and Marina for the analysis of the dashboard.	analysis. Some process overviews require several computations steps, an explicit representation of the results is her key interest.
Mrs. Marina Sereguina	Business analyst	She helps in defining the requirements for the visualization tool.	She shares the same interests as Marchella.
Mr. Henry Anijs	Business analyst	He works in the enforcement department. He identifies the benefits which are not possible to give to the customer. He is the point of contact for any enforcement related query from their department.	He is primarily interested in the features of the visualization tool. He wants to know the capabilities of the visualization tool and how it can contribute to the process analysis.

2.3 TU/e

TU/e is responsible for the educational aspect of this project.

The primary stakeholder from the TU/e is Dr. Massimiliano de Leoni, Assistant Professor, who has the role of university supervisor and technical advisor. He provides the guidance concerning the design and implementation and also evaluates whether the proposed documentation meets the standards of a PDEng project. Meetings with the university supervisor took place every week in order to share and discuss the design and implementation of the project. He also has the role of helping with the academic aspects of the project.

Another TU/e stakeholder is Dr. Ad Aerts, the Program Director of the Software Technology PDEng program. His role is to ensure that the project meets the quality requirements of the PDEng program.

3. Process mining: State-of-the-Art

This chapter elaborates the Process Mining domain that is relevant to the project. In this chapter we provide the necessary information of this domain in order to help the reader for better understanding the architecture, design, and implementation.

3.1 Process Mining

Process mining (van der Aalst, 2011) is a process management technique that allows for the analysis of business processes based on event logs. The basic idea is to extract knowledge from event logs recorded by an information system. Process mining aims at improving this by providing techniques and tools for discovering process, control, data, organizational, and social structures from event logs. Figure 3 shows that process mining establishes links between the actual processes and their data on the one hand and process models on the other hand.

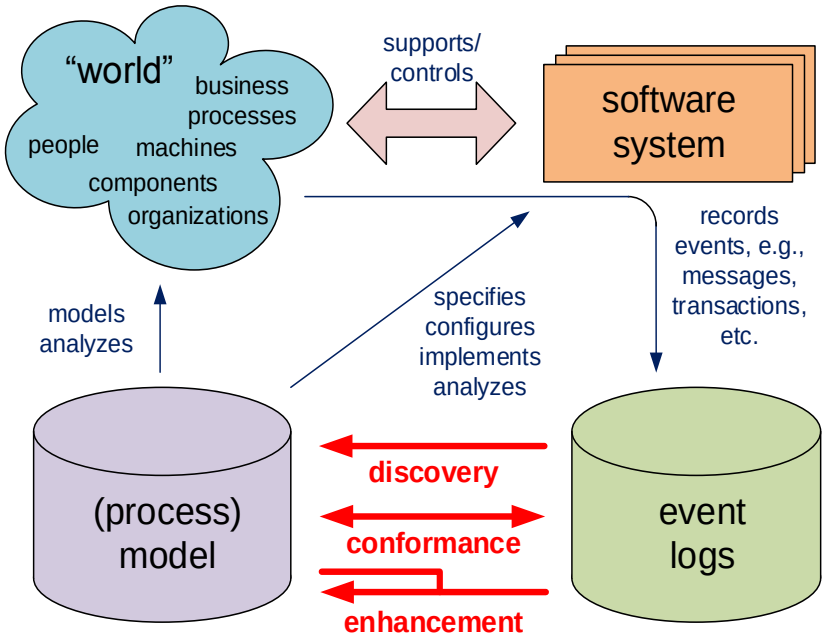

Figure 3: Three basic type of process mining: Discovery, Conformance, and Enhancement

Process mining is used when user wants to bridge the gap between data mining and business processes. In the context of reclamation analysis, UWV handles various customers' requests on daily basis i.e. the real world shown in Figure 3. UWV has a system that records all the customers' requests and transactions in various databases. For analysis, we extract reclamation data from the data bases and the process analyst can apply various process mining techniques such as conformance checking or bottle-neck analysis.

There are three basic type of process mining techniques: discovery, conformance, and enhancement.

Discovery, deals with the creation models from event logs without any a priori models. An example of this technique is applied in alpha miner algorithm (van der Aalst et al. 2004), which creates a Petri-net model describing the behavior observed in the event log.

Conformance, deals with the comparison of an a priori model with event logs. The aim is to detect the deviation and inconsistencies between an event log and process model. For example, there is a process model indicating that purchase orders of more than one million Euro require two checks. The analysis of the process model and event log will show whether this check is followed or not.

Enhancement, aims to extend or to improve the existing business process model from an event log. There is a-priori model. This model is extended with a new aspect or perspective. An example is the extension of a process model with performance data, i.e., some a-prior process model is used to project the bottlenecks of the business processes.

3.2 Event log

An event log is the main enabler to apply process mining techniques. It is extracted from different data sources for example databases, transaction logs. It assumes that it is possible to record an activity such that each activity refer to a trace and an activity. An event log is the main enabler for the LogOnMapReplay plug-in.
Figure 4 shows the standard model of an event log. It consists of several entities: a log, a trace and an activity

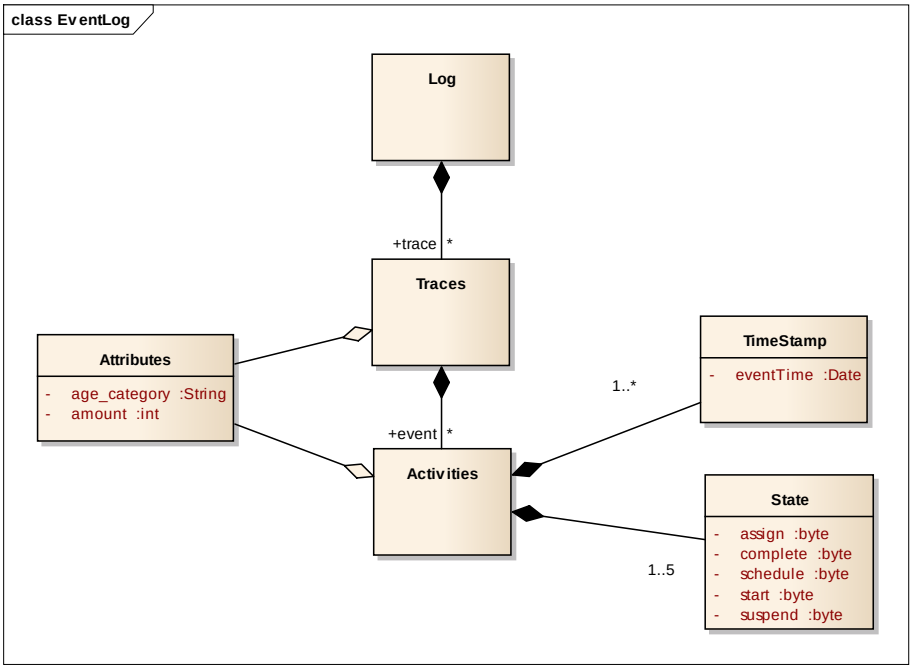

Figure 4: A standard model of an event log

1. **Log**: The logs contains sequences of events or task, which refers to the execution of certain business processes such as unemployment business process. These events are listed in chronological order. A log contains a set of traces. For example, UWV's unemployment business process, has an arbitrary number of customers who ask for unemployment benefits.
2. **Trace**: A trace consists of several events such that each event instance is related to exactly one case. Here a customer represents one case. A customer can have multiple traces. For instance, a customer has the following set of events in a trace:
    a. Customer applies for an unemployment claim.
    b. UWV evaluates the customer present situation.
    c. UWV accepts or rejects the claim application.
3. **Activity**: An activity or task that was executed for a particular case. Figure 5 shows the life-cycle of an activity. An activity must have at least one of the following States :
    - ***Schedule***: An activity was created but was not yet assigned to a resource.
    - ***Assign***: An activity was assigned to a resource but not yet started.
    - ***Start***: An activity had commenced.
    - ***Suspend***: An activity was temporarily halted with the possibility of continuing the execution later.
    - ***Complete***: An activity was completed.

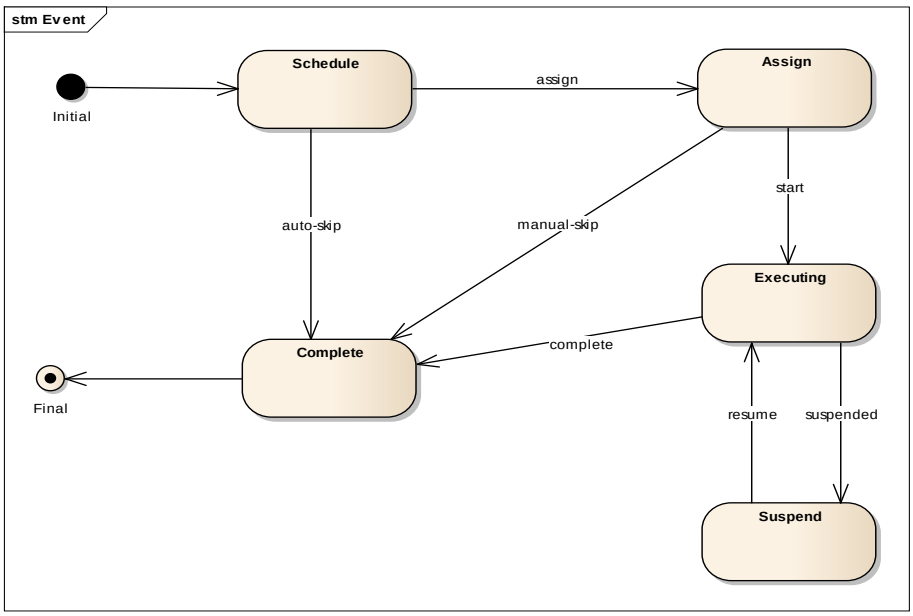

Figure 5: State diagram for life-cycle of an activity

Each activity is associated with the timestamp when they occurred in the information system. An activity life-span is determine when an activity has at least start/assign/schedule state and a complete state. The duration between these states defines the life-span of an activity in the LogOnMapReplay plug-in. In a trace, events are sorted based on timestamp. An activity can also contains attributes of a trace or an attribute such as age, amount, and office location. Table 2 shows an example of an event log that can be possibly produced by Process-Aware Information System.

Table 2 shows a fragment of an event log corresponding to the handling claim request. Each row represents one activity that has occurred at specific time. The activities are grouped per customer. The Table 2 has two traces Customer 1 and 2. Customer 1 has ten associated activities. The first two activities for Customer 1 are the execution of a *Claim* activity which has started on 6th April, 2014 and completed on 7th April 2014. An activity may or not have an attribute value for example *Claim* event name does not have amount but *Payment* and *Start reclamation* has amount.

Table 2: An example of an event log

Customer ID	Event name	Time stamp	State	Age	Amount
1	Claim	06-04-2014	Start	20	-
	Claim	07-04-2014	Complete	20	-
	Claim decision	07-04-2014	Start	20	-
	Claim decision	10-04-2014	Complete	20	-
	Payment	10-04-2014	Start	20	2000
	Payment	10-04-2014	Complete	20	2000
	Payment	10-05-2014	Start	20	2000
	Payment	10-05-2014	Complete	20	2000
	Start reclamation	11-06-2014	Start	20	2500
	Start reclamation	11-06-2014	Complete	20	2500
2	Claim	07-04-2014	Start	45	-
	Claim	07-04-2014	Complete	45	-
	Claim decision	08-04-2014	Start	45	-
	Claim decision	09-04-2014	Complete	45	-
	Payment	10-04-2014	Start	45	1000
	Payment	10-04-2014	Complete	45	1000
	Start reclamation	10-05-2014	Start	45	1300
	Start reclamation	10-05-2014	Complete	45	1300

3.3 XES standard

Extensible Event Stream (XES) (Christian W. Günther, 2014) standard was chosen for representing an event log. This standard was adopted as a standard by the *IEEE Task Force on Process mining*. XES is a generic XML-based standard for event logs. This standard is widely used in various process mining techniques such as LogOnMapReplay, and Alpha miner. The format is supported by various tools such as ProM framework, Disco.

Figure 6 shows an example of an event log in XES standard. An XES file contains an event log, which consists of any number of traces. Each trace has a sequential list of events. The traces, events can have any number of attributes. XES does not have a fixed set of mandatory attributes for each element (trace and event). However, to provide semantics for such attributes, the log refers to extension. An extension defines the semantics for attribute.

For example, the *time:timestamp extension* defines a timestamp attribute of *dateTime*.

The following is a subset of standard attributes defined by the extension of XES standard (Christian W. Günther, 2014) are used in this project.

- The *concept:name* extension defines the name attribute for traces and events. For traces, the attribute represents a unique identifier for a case. This extension is mandatory for traces. For events, the attribute represents an event name.
- The *lifecycle:transition* extension defines the State of an event. The possible values of this attribute are Start, Complete, Schedule, Assign, and Suspend.
- The *time:timestamp* extension defines the timestamp attribute of an event.

Figure 6 shows a fragment of the XES XML file of an event log of mentioned in the Table 2. In this example, three extensions are declared *concept, time, and lifecycle*. These three extensions are the mandatory for the LogOnMapReplay tool. The usefulness of an event is discussed in Section 3.5.

```
<log xes.version="1.0" xes.features="nested-attributes" openxes.version="1.0RC7">
    <string key="concept:name" value="PMDS_WW_VFV_Sample_ReclOnly_Weight_005_C01.zip"/>
    <trace>
        <string key="concept:name" value="1"/>
        <event>
            <string key="lifecycle:transition" value="start"/>
            <string key="concept:name" value="Claim"/>
            <date key="time:timestamp" value="2014-04-06T00:00:00.000+01:00"/>
            <int key="Age" value="20"/>
        </event>
        <event>
            <string key="concept:name" value="Payment"/>
            <string key="lifecycle:transition" value="start"/>
            <date key="time:timestamp" value="2014-04-10T00:00:00.000+01:00"/>
            <int key="Age" value="20"/>
            <int key="Amount" value="2000"/>
        </event>
        <event>
            <string key="concept:name" value="Payment"/>
            <string key="lifecycle:transition" value="Complete"/>
            <date key="time:timestamp" value="2014-04-10T00:01:00.000+01:00"/>
            <int key="Age" value="20"/>
            <int key="Amount" value="2000"/>
        </event>
    <trace>
</log>
```

Figure 6: An example of an event log in XES standard

3.4 Petri-net model

Petri-net (van der Aalst, 2011)is a formalism which is used to define the control-flow semantics of process models. The petri-net models are the optional input for the LogOnMapReplay tool (see section 3.5.2). Figure 7 is an example a process model represented in a Petri-net model. Figure 7 model describes the handling request of a claim business process within UWV. A customer may request for claim for various reasons such as unemployment or sickness.

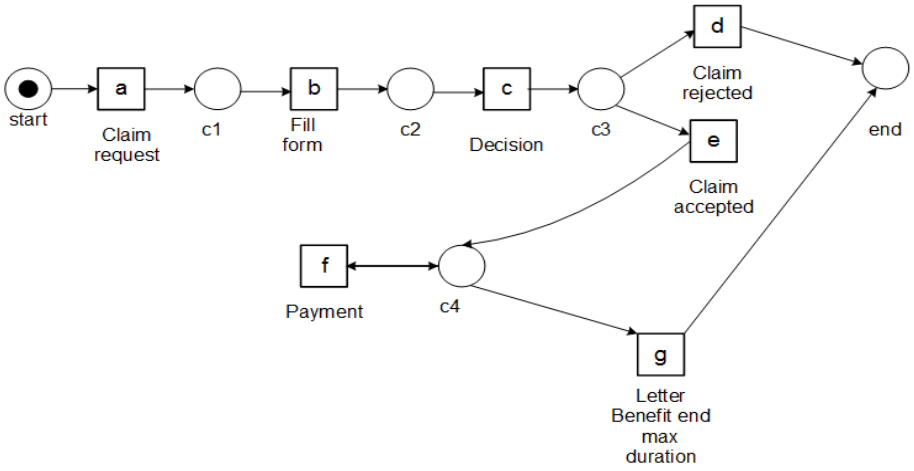

Figure 7: An example of a Petri-net model

A petri-net is a bipartite graph consisting of transitions, places, and arcs. A transition is represented by a square. A place is represented by a circle. Transitions are connected through places. An arcs run from a place to a transition or vice-versa; never between places or between transitions. A token is the enabler for the transitions. In Figure 7 a [start] place has a token. The behaviour of a petri-net is defined by the firing rule, tokens can flow through the model. The firing rule for a transition is characterises by subtracting the number of token present at an input place equal to the number of arcs connected to a transition and adding the number of token already available at the transition output.

A transition is enabled i.e., the corresponding transition can occur, if all input places hold a token. This means the business process starts with Claim request only. An enabled transition can fire thereby consuming one token from each input place and producing one token for each output place. Hence, transition [a] is enabled at place [start]. Firing *a* results at the place [c1]. Note that one token is consumed by the *Claim request* transition and one token is also produced at place [c1]. Now, the transition [b] become enabled and place [start] is no longer enabled. Similarly firing rules are followed for other places such as c2 and c3.

3.5 ProM

ProM[1] is a generic Open Source framework for applying process mining algorithms. It is most common and popular tool for applying process mining techniques. It follows a plug-in based architecture. This frameworks allows developers to add new process mining techniques by adding new plug-ins. The ProM framework has over 300 plug-ins for process mining, analysis,

[1] http://www.promtools.org

monitoring, and conversion. Figure 8 shows an overview of the ProM framework.

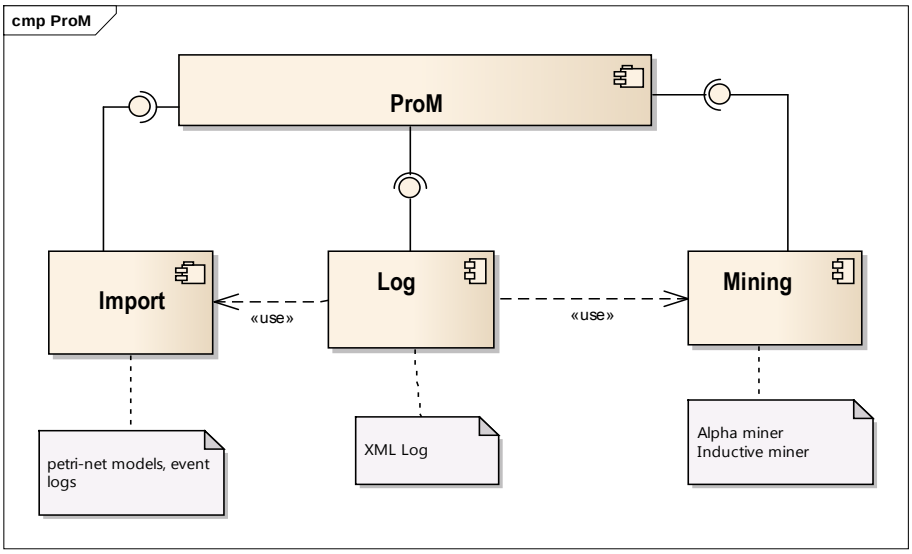

Figure 8: Overview of the ProM framework

Each component depicts different plug-ins that are available in the ProM framework. The import plug-in, allows to import petri-net model, log, and visualization data into ProM framework. The log plug-in, allows the user to filter events from an event log. It reads data file in the XML format. The miner plug-in, allows the user to extract petri-net model using an event log. Various techniques such as alpha miner and inductive miner are available in the ProM framework, which allows the user to extract petri-net models from an event log.

3.6 LogOnMapReplay plug-in

LogOnMapReplay plug-in (Massimiliano de Leoni, 2014) allows event logs to be visualized and replayed on maps. The plug-in processes an event log to produce a time-based process movie. The main goal of this plug-in is to visualize the execution of activities as recorded in an event log. The visual analytics techniques is the basis for developing the framework of this plug-in. it is an approach which *combines automated analysis techniques with interactive visualisations for an effective understanding, reasoning, and decision making on the basis of very large and complex data sets.* (Thomas, 2005)

The tool combines the visual analytics techniques to visualize the incomplete process instances of an event log. This visualization technique helps the process analyst in gaining insights on potential problems or useful insights their business processes. The result can later be confirmed or refute by the analyst. The process analyst may also use other process mining techniques for instance bottleneck analysis or conformance checking to verify the results of the visualization plug-in. The plug-in is realized in the ProM framework.

As shown in Figure 9, an event log and a map file are the mandatory input files for this plug-in. A map encodes a different viewpoint on the execution of an event for example a Cartesian map, a geographical map, and a process map. A map file contains an arbitrary number of maps. In Section 3.5.2, we will further discuss about maps and introduces interesting examples.

The plug-in produces a process movie for each map. The executed data can be analyzed from different point of view. Each movie enables a process analyst to gain insights and do further analysis based on the preselected criteria. As discussed in Section 3.2, an activity have different states. Each activity is presented as a dot, which is projected on a map at a meaningful position. If one or more activities are at the same coordinates then the framework merges the dots and present as a pie-chart. A merged dot shows one or more activities of the same type that belongs to the same or different traces. To project a dot, an activity must have at least start/assign/schedule state because the tool visualize the life-span of an activity.

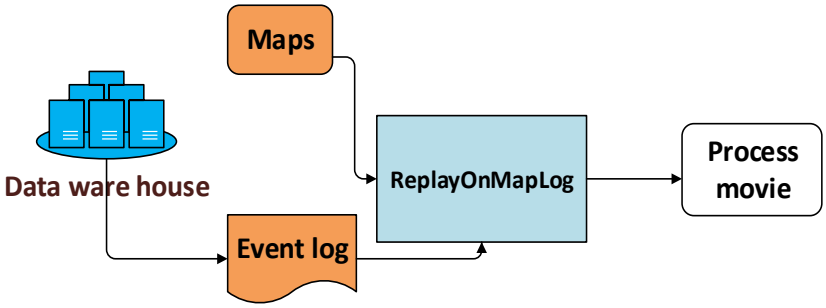

Figure 9: Overview of the LogOnMapReplay plug-in

In order to define how states can be represented on a map, a process analyst needs to choose an image for a map and define the positions for activities as pie charts on that image. For example a process analyst wants to create a process movie for a process model map. The plug-in projects an executed activities such as Claim, Claim decision, which are available in an event log. An activity is positioned on the transitions of a process model corresponding to the executed event.
The occurrence of an activity makes the system enter a particular state. Hence, by replaying all activities of the event log in chronological order, it is possible to rebuild a process history, i.e. the sequence of states the system went through.
Such an annotated map can be seen as a photograph, and thus, a process history can be visualised as a sequence of photographs, played together to form a movie. A process movie is a sequence of photographs. A photograph is composed of three type of dots; valid, invalid, and hidden.

- **Valid**: It has a valid and meaningful position for a map. The valid dot will be projected onto a map.

- **Invalid**: It has an invalid position (means x and y axis values are negative) for a map. The plug-in will list the event name and trace id on the Invalid sidebar of GUI.
- **Hidden**: A map file does not contains an encoded viewpoint for an event. The plug-in will list the event name and trace id on the hidden sidebar of GUI.

Each activity is represented as a set of pie charts projected onto a map. Each pie-chart captures one or more activities of the same type belonging to different process instances. The diameter of a dot depends of the number of activity instance that need to project on the map at a particular time. The dot size can increase or decrease depending on number of activity instances that are active at same time.

The LogOnMapReplay plug-in has a Graphical User Interface (GUI) for visualizing a process movie. The set of movies produced, one per map, is visualized and played using GUI (note that movies has to produces first before they can be played, i.e. they are not generated on the fly). The interface has various functionalities such as play, pause, and speed. Figure 10 shows a screenshot of the GUI and a process movie produced by LogOnMapReplay plug-in. The user can press the play button to see the visualizing of the executed activities of an event log. The activities are sorted in a chronological order and are subsequently played in the same order. By playing a movie for each map, the user can gain different insights on the execution of activities. By pressing the play button the user can see the continuous behaviour of the activities. The user can go back and forth in the movie by applying different speed or choosing the timeframe from the timeline.

When multiple activities are projected on a process model map as dot at the same coordinates then all dots are aggregated and creates a pie-chart with slices. Each slice is coloured based on the chosen colour scheme. The plug-in has three types of the colour schemes, which are as follows:

1. **State schema**: An activity is coloured based on the current State of an event.
2. **Age schema**: An activity is coloured based on the age of an activity which means the moment that activity occurred in the movie and the current time of the movie. If the activity has just started then the colour of the dot is projected white and as longer it remain on the visualization panel, it starts progressing toward black colour.
3. **Attribute value schema:** An activity is coloured based on the end-user selected attribute and a value of an event.

Each slice of a pie-chart is coloured based on the chosen colour scheme. Figure 10 screenshot shows the active activities on 29 January 2014: 12:22:30. The screenshot shows a photograph of a movie which is of a geographical nature. At UWV, the reclamations are occurring at various geographical work location of UWV. Here, each activity is projected on the map of The Netherlands as a pie-chart. The position to project a dot or a pie-chart is chosen based on the e geographical work locations of UWV. At any given point in time, the size of the pie-charts are associated with a particular office location of UWV. It represents the number of activities that refer to the reclamations of the office location for example Eindhoven, Amsterdam, and Alkmaar.

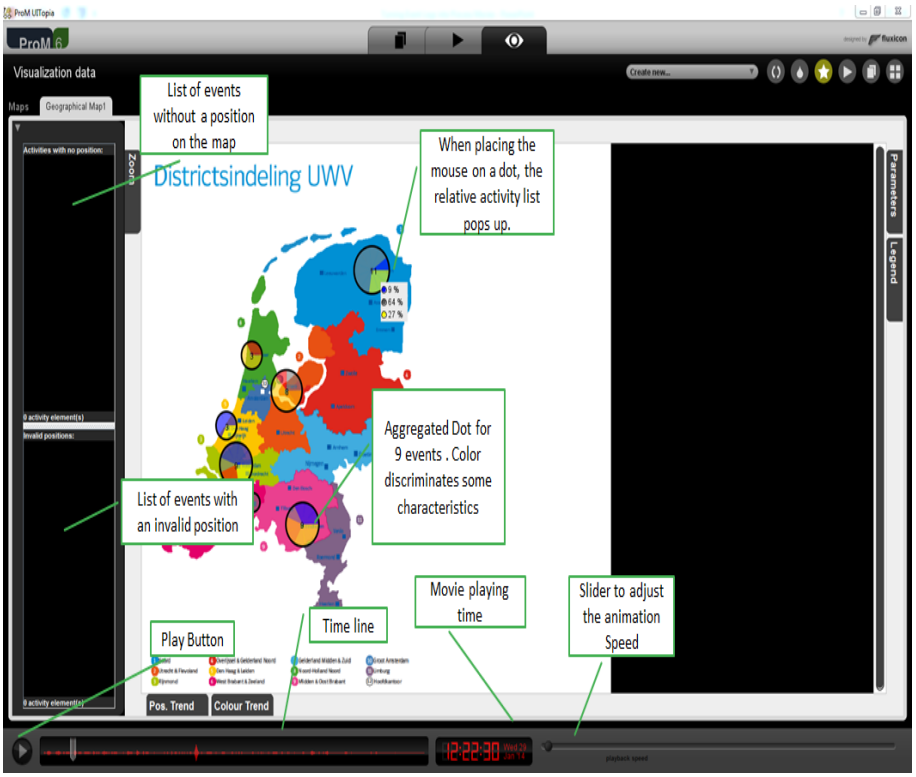

Figure 10: A screenshot of a process movie produces by LogOnMapReplay plug-in

The activities has different attributes for example age, or reclamation amount. Therefore, the pie-charts can be sliced according to the percentage of activities with given attribute. Each attribute value is assigned with a different colour. Here, the pie-charts are sliced according to age. By clicking the play button, the user starts visualizing the process movie. Alternatively, the user can click anywhere on the time line to view the corresponding photograph of the movie.

The plug-in determines the life-span of an activity when an activity has at least start/assign/schedule state and a complete state. As shown in Table 3, Activity A started on 8 June 2014 and completed on 11 June 2014. The life-span on Activity A is three days. This tool will visualize an Activity A for three days and the process analyst can observe how an activity has progressed with time. Activity B has a complete state so the tool does not visualize it because it does not have a start state.

Table 3: A set of events

Activity	State	Time stamp
A	Started	2014-06-08T04:00:12.007+02:00
B	Complete	2014-06-08T04:00:12.007+02:00
A	Complete	2014-06-11T04:00:12.007+02:00

3.7 Map

A map (Massimiliano de Leoni, 2014) encodes a different viewpoint for the executed activities. Each activity is recorded in event logs can be projected on a map as a dot at the specified meaningful position. A map file has an XML-based standard. A map can be of five types namely: Cartesian, process (petri-net model), Timeline, Geographical, or Organization chart map. A process analyst can choose or handcraft an image of a map.

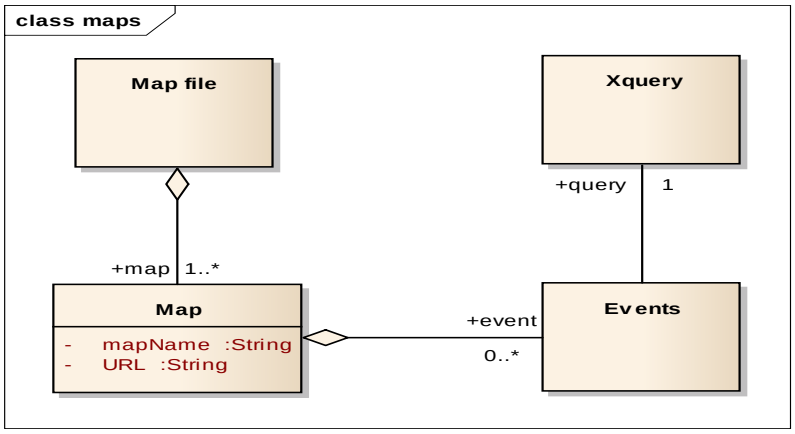

Figure 11: Map file structure

Figure 11 depicts the concept of a map file. Each map contains a set of events and each event contains an XQuery. An XQuery[2] calculates the meaningful position for a background image. An XQuery (W3C, 2014)is a query and functional programming language that queries and transforms collections of structured and unstructured data, usually in the form of XML, text and with vendor-specific extensions for other data formats (binary).

There are two type of XQuery: Static and Dynamic XQuery and they are as follows:

1. **Static**: A static XQuery is one whose x, y and z are coordinates are defined.
2. **Dynamic**: A dynamic XQuery expression has a position function and the coordinates are determined by given attribute values.

Figure 12 is an example of both static and dynamic XQueries

[2] http://www.w3.org/TR/xquery/

```
<![CDATA[
    <coordinate>
        <x>{257 + (number(//Age) div   18) *257}</x>
        <y>{1200- (171 + ((number(//Amount)) div 6100)*171)}</y>
        <z>0</z>
    </coordinate>]]>
```

Dynamic

```
<![CDATA[
    <coordinate>
        <x>1028</x>
        <y>450</y>
        <z>0</z>
    </coordinate>]]>
```

Static XQuery

Figure 12: An example of static and dynamic XQuery

3.7.1. Cartesian map:

An inspiration for a Cartesian map is taken from two- dimensional Cartesian coordinate system. The x and y axis of a Cartesian map is represented by the attribute values for an event log such as amount and age. Figure 13 shows an example of a Cartesian map. This map helps in finding out the co-correlation between two attribute values. Based on the position defined for the x and y coordinates, the visualization tool can visualize a dot at that pixel point of a map.

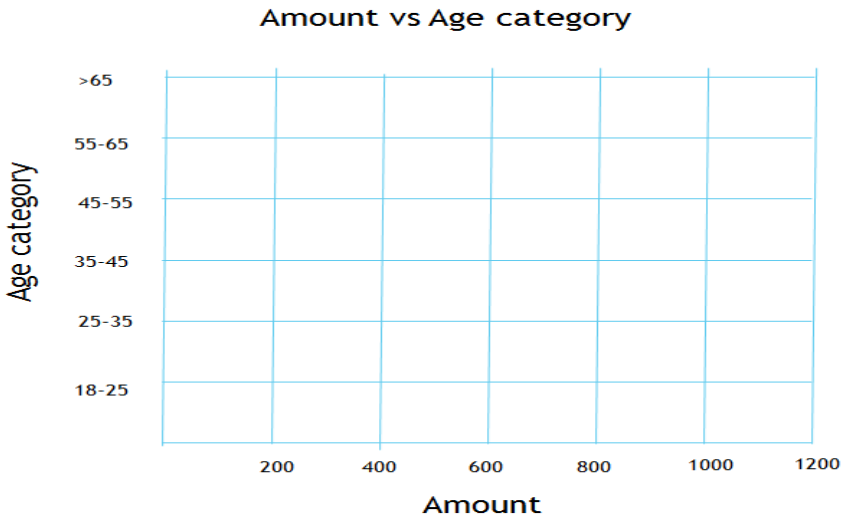

Figure 13: An example of a Cartesian map

3.7.2. Process map

An inspiration of a process map is taken from a petri-net model. This map helps in analyzing the correlations between activities with-in a business process. An example of a process map is shown in Figure 7. An activity is projected where the transition name of a petri-net model and activity name from an event log has the same name.

3.7.3. Geographical map

An inspiration of this map is taken from a country map. Figure 14 shows the example of The Netherlands map. This map has been modified and colored based on UWV office locations. This map is useful for analyzing events based on the office location. The LogOnMapReplay plug-in projects dots based on the work distribution of a company.

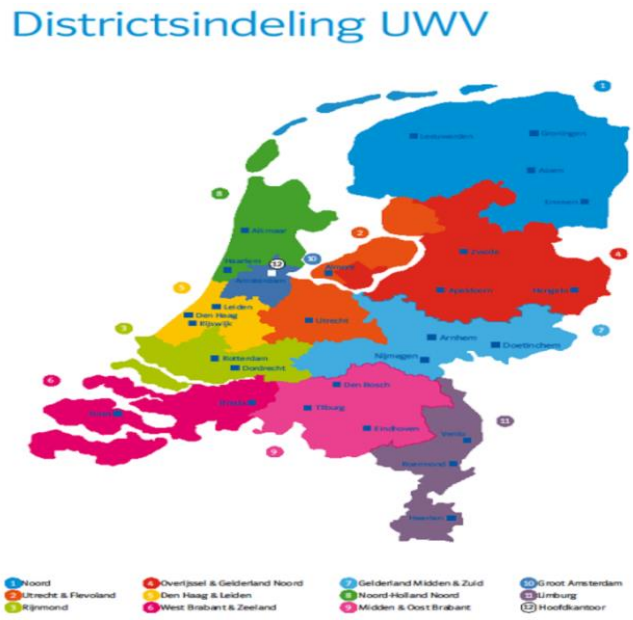

Figure 14: An example of a geographical map

4. Problem and Requirement analysis

This chapter describes the problem definition and requirements of the project. The limitations of the LogOnMapReplay plug-in is the main requirement in this project. The chapter continues explaining the most important functional and non-functional requirements of the project.

4.1 Problem definition

UWV is committed to continuously improving their business processes for the customers. With a growing number of reclamations, UWV has to make a lot of effort to recover the money from customers and not all the money is recovered. The company does not have control over the inflow of new reclamations insights. They do not have insight into the causes of these reclamations.

They have identified reclamation data by connecting different data sources. An event log is available but there are no operational support for business analysts and managers. It is difficult to analyse an event log and find answers for the following questions:

- How the reclamations events have progressed with time?
- Is there a specific category/cluster of customers, who often create reclamation?
- Is there any co-relation between reclamation, specific group of

In process mining various techniques are available such as conformance checking, and bottleneck analysis. But these techniques are less effective if process/ business analyst don't know which aspects of the data are worthy to analysis. The tool called LogOnMapReplay plug-in, dynamically shows behavior of process instances. To use this plug-in, a process analyst does not require guidance.

In Chapter 3, we have briefly discussed the functionalities of the LogOnMapReplay plug-in. This plug-in is a prototype, it shows the flow and timeline of events as they are recorded in execution. The first challenge of this project is to enhance the features of this plug-in. Another challenge is to quickly process huge event logs. Currently, the plug-in supports the processing of two hundred thousand events on a laptop equipped with an Intel Core i7 Processor at 2.20GHz. The processing of a process movie takes more than 7 hours and it used 5 GB of memory. The details of these limitations are discussed in Section 4.3. To overcome these limitations, we need a solution that offers an optimized way of processing an event log.

4.2 Project goal

The stakeholders require an enriched monitoring tool for analyzing their reclamation data. The primary goal of this project is to enhance the features and improve the scalability of the LogOnMapReplay plug-in. The plug-in

must be capable of quickly processing at least five hundred thousand events in less than eight hours. The secondary goal is to provide first insights for the following questions.

- How to identify patterns in the UWV reclamation data?
- How to identify any specific cluster of customers, who often creates reclamation?

4.3 Limitation of LogOnMapReplay plug-in

At the beginning of the project, various experiments were conducted with UWV process/business analysts to find out the limitations and missing features of the plug-in. The following use case scenarios and corresponding maps were created to show the existing features of the plug-in

- Show business processes on process model map.
- Show reclamation on the geographical map.
- Show co-relation between age and reclamation amount on the Cartesian map.

During experiments, the following limitations were identified in the plug-in.

1. **Manual map file creation**: A map file is the main input file for the LogOnMapReplay plug-in. As we have discussed in Chapter 3, the user can create various types of maps (for example a Cartesian map, or process model map). Currently, the user has to manually create these map files and it includes:
    - **Choose or create an image**: User has to either hand-craft or choose a background image for a map. For example, to create a Cartesian image, the user must be aware of attribute name and their corresponding unique value from an attribute. The manual file creation is a time consuming and tedious process.
    - **Create an XQuery**: To create an XQuery requires technical expertise or need assistance from IT personnel. The user must know and pays attention to events that would be worthy of analysis.

The main user of LogOnMapReplay plug-in is process/business analyst and they would like to avoid this tedious process of creating a map. Due to these reasons, they are interested in a tool that will automatically create a map file which includes an image and specified event XQueries. A process analyst selects their area of interest events and attributes from an event log and the tool creates a map file.

2. **Quickly process an event log:** Every month UWV handles 1.5 million customer requests and approximately 500 thousand customers have reclamations. This means the tool must be capable of quickly processing at least 1/3 size of an event log.
   In the current implementation, it takes more than seven hours and it used five GB of memory to process two hundred thousand events on a laptop equipped with an Intel Core i7 Processor at 2.20GHz. Due to the following reason the processing events are slow in the plug-in.
    - **Storing objects in-memory**: The reason for using large amount of memory is because each processed photograph object (related to movies) is kept in-memory. There is a

linear dependency between numbers of active activities vs number of photographs.

- **XQuery expression computation**: Another reason for the slow computation is the processing of an XQuery expressions of the fly. The LogOnMapReplay plug-in takes significant amount of time to compute the meaningful positions of an activity on a map. This expression runs all maps available in a map file and all activities that are present in an event log.
- **Sequential processing of an event log:** The existing implementation is single threaded. As such, it unable to take advantages of system equipped with multiple CPU or a single CPU with multiple core.

3. **Prototype:** The plug-in has basic functionalities with which a process analyst can get insights. The following missing functionalities were identified:
    a. **Filter of events and attributes**: Once a process movie is produced, the user does not have freedom to filter point-of-interest events and attributes. For instance, while watching a process movie if the user would like to choose the visualization of dots only for two events such as Claim, payment, and in addition those event must have age between 30 to 40 years then the plug-in does not have an option or an alternative to filter those values. That's why there is a strong need to add filter functionality in the plug-in.
    b. **Export/show data**: The plug-in does not allow the user to export the sub event log of a dot, which is projected on the visualization panel of the plug-in. This feature allows a process analyst to narrow down the scope of analysing an event log.
    c. **Show pie-chart details**: The plug-in does not allow the user to see details of the pie-charts. Suppose when the user saw some usual flow of events in a process movie then the user might be interested to see the details of those pie-chart. This feature might help the process analyst in identifying and narrow down the scope of potential problematic area in their business processes.
    d. **Highlight the problem**: In the current implementation, while plotting dots onto the map, each dot has an equal weight of one. In a process movie, it is difficult to detect events when something is not ordinary.
    Table 4 shows an example to calculate the weight of the dots for the number of dots that are plotted on a geographical map at a certain time. Currently, five, ten, and fifty dots are plotting at Amsterdam, Groningen, and Eindhoven office

location respectively. Amsterdam population is five, one hundred at Groningen, and two hundred customers at Eindhoven. The weight is determined by below formula

$$W = D/A$$

Where D is number of dots of any location onto a map, A is selected attribute value, and W is the weight.
If a region has dense population then it is expected to have more customers within a Claim business process. When this situation is vice-versa then something unusual is happening in an event log. That's why it would be interesting to visualize relative corresponding to the point-of-interest attributes. As shown in Table 4, the entire population of Amsterdam is active for certain business process. And ¼ the population of Eindhoven is active. Based on this calculation, a process analyst can conclude that need to focus more on their Amsterdam office then Groningen and Eindhoven office.

Table 4: Absolute weight

Office	**Number of dots (D)**	**Population (A)**	**Weight (W)**
Amsterdam	5	5	1.00
Groningen	10	100	0.10
Eindhoven	50	200	.25

e. **Indexation:** To determine if events has drifted means (increase, decreased, and same) from a certain point in time. It would useful for a process analyst to be able to select a point in time and show the difference between the selected timeframe and rest timeframes of a process movie. The following types cases are considered :
   - **Increase**: The number of events are increasing from the selected time frame. This is projected with black colour.
   - **Decrease**: The number of events are decreasing from the selected time frame. This is projected with yellow colour
   - **Same**: There is no change. The number of events are same on both timeframes

   The table 5 shows the calculations that is expected from the LogOnMapReplay plug-in. In Table 5, the T1 timeframe is chosen to calculating indexation for the T2, and T3 timeframe. The result expected for each coordinate of the T2 and T3 timeframes is also shown in Table 5.

Table 5: An example of indexation

Coordinates	T1	T2	T3
100,100	10	12-10 =2	9-10 =-1
100,200	12	1-12 =-11	9-12=-3
100,300	0	12-0 =12	9-0 =9

4.4 Use case scenario

The following use case scenarios are identified, based on the meetings with the stakeholders and experiments conducted on the tool.

1. Create map file
2. Quickly process an event log
3. Create filter
4. Show/Export data of pie-charts
5. Show relative
6. Show drift

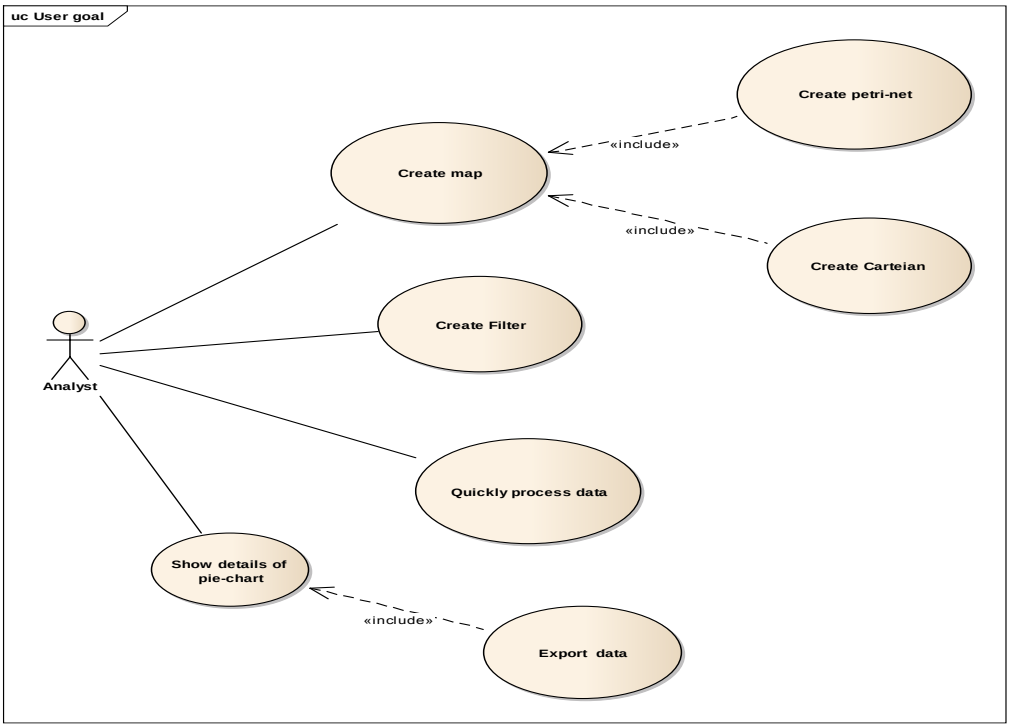

Figure 15: User goa

4.5 Functional requirements

Based on the meetings with the stakeholders and experiments, a list of functional requirements was created.

4.5.1. FR1 – Automatic Map Generator (AMG)

One of the requirements is to design and implement a new plug-in with Graphical User Interface (GUI) which will automatically generate maps. Irrespective of the map type, a user must specify the events for which the map must be generated. Each map consists of an image and XQuery related to the selected events. A process analyst can create a map file for visualizing the event log. For this project, we are focusing on two types of maps namely Cartesian map and Petri-net map.

This requirement was subdivided into three smaller requirements based on their functional aspects. Requirement FR1.1 focuses on the graphical representation of the user interface. Requirements FR1.2 and FR1.3 focus on generating Cartesian and Petri-net map respectively. Details of the sub-requirements are presented in Table 6.

Table 6: List of sub-requirements for Automatic Map Generator

Requirement ID	**Requirement name**	**Description**
FR1.1	Create GUI	Design and implement a GUI in ProM framework for a map file creation.
FR1.2	Create a Cartesian map	User selects x-axis, y -axis attributes for example, Age and Gender from an event log file. There can be four type of Cartesian map: • Literal - x and y axis attributes have String/ Boolean values • Continuous – x and y axis attributes have integer or float values • XLiteral – x axis has a string value attribute and y axis has an integer or float value attribute • YLiteral – x axis has an integer or float value attribute and y axis has a string value attribute.
FR1.3	Create a petri-net map	An event log and petri-net models are the input files for the AMG plug-in. The GUI must have the following functionalities: • Give the most appropriate name match between an event of an event log file and transition of the petri-net model. • Support one-to-many mapping between events and the petri-net model transitions, for example (Claim, Change form out1, and Change form2) should be allowed to map with Claim transition.

FR1.4	Merge map file	A tool which allows user to merge two or more map files

4.5.2. FR2 – Quickly process an event log

UWV's system has a huge number of events that are logged on a daily basis. Another requirement, important for all stakeholders of the project, is scalability in terms of processing these event logs. The visualization of the event logs is generated by a plug-in called the LogOnMapReplay plug-in. The LogOnMapReplay plugin uses the map file and the event logs to generate the visualization as intended by the user. The plugin must be capable of processing at least five hundred thousand events. Table 7 lists the sub-requirements that have been derived from this requirement.

Table 7: List of sub-requirement of processing an event log

Requirement ID	Requirement name	Description
FR2.1	Faster computation of events	The tool must be capable of processing 500 thousand events in maximum 8 hours
FR2.2	Less memory consumption	The tool must be memory efficient to support processing of at least 500 thousand events at a time.

4.5.3. FR3 – Create filter

In LogOnMapReplay plug-in, while playing a process movie, the user must be able to choose which events and attributes with its corresponding values. The plug-in will visualize only selected events and attribute values.

Based on the functional aspect, this requirement is divided into three sub-requirements. FR3.1 and FR 3.2 focus on different filter levels and FR3.3 focus on the GUI of the filter functionality. Details of the sub-requirements are shown in Table 8.

Table 8: List of sub-requirements for creating a filter on LogOnMapReplay plug-in

Requirement ID	Requirement name	Description
FR3.1	Event-based filter	User can select any number of events for example, ten events are available in an event log and user would like to visualize only three events such as Claim, Start reclamation, Payment.
FR3.2	Attribute-based filter	User can select any number of attributes and their corresponding values. For example, show only female whose age is between 40 to 60 years from the event logs
FR3.3	Create GUI	Design and implement a filter panel for selected activity instance and attributes in LogOnMapReplay plug-in.

There is an AND relationship between both event-based and attribute-based filter. For example, if user has selected Claim, payment events, and Female from gender attribute then LogOnMapReplay plug-in will visualize dots corresponding to claim and payment which has female only.

4.5.4. FR4 – Show/Export data

While playing a process movie, user must be able to see the detail of selected pie charts data in a table. A user should have flexibility to have different chart representations of the same selected pie chart, for example bar chart. Details of requirements are shown in Table 9.

Table 9: List of sub-requirements for export /show functionality of LogOnMapReplay plug-in

Requirement ID	Requirement name	Description
FR4.1	Export data	Export a sub event log file for a selected a pie-chart on the visualization panel of a LogOnMapReplay
FR4.2	Export image	Export image of a visualization panel or selected pie-chart.
FR4.3.1	Show data	Show detail of selected pie chart in a table form
FR 4.3.2	Show a configuration panel	Show a configuration panel which lists all the maps for a map file. The user can select the maps and the LogOnMapReplay plug-in will produce a process movie only for the selected maps.

4.5.5. FR5 – Show deviation

The objective of this requirement is to detect events in the process that are different from what is expected. The plug-in detects incidents, trends, and concept drifts from playing a process movie. The plug-in must be capable of detecting if some value is out of the ordinary. Table 10 lists the sub-requirements that have been derived from this requirement.

Table 10 : List of sub-requirement for highlighting the problem on LogOnMapReplay plug-in

Requirement ID	Requirement name	Description
FR5.1	Show relative	The user can select attribute for which weight of the dots needs to be calculated. Before plotting dots, the plug-in will calculate the combined weight. The new diameter of a dot is decided based on the calculated combined weight for all dots that are present at same coordinate of a map.
FR5.2	Show drift	The user can select a specific time from a process movie. The plug-in will visualize the difference between from the selected point in time and all other points in time. The dot size is calculated based on the difference between those two time frames. The difference can be positive, negative, and equal. **Positive**: The number of dots are increasing from the selected point on time. The dot will plotted with black colour. **Negative:** The number of dots are decreasing from the selected point on time. The dot will be plotted with yellow colour. **Equal**: if the number of dots from the selected point on time and any other time are equal then nothing is no dot visualization.

4.6 Non-Functional requirements

We identified the following non-functional requirements, as relevant to the project. The criteria can be used to assess the quality of the proposed solution.

4.6.1. NFR 1- Extensibility

The design and implementation of the AMG plug-in must consider future growth for creating other maps such as timeline, geographical map.

4.6.2. NFR 2- Usability

This is one of the major concerns of the UWV stakeholders. The main user of this visualization tool is a process analyst. Both AMG and LogOnMapReplay plug-ins should be intuitive and a process analyst should quickly be able to understand and analyses expected/unexpected behavior of events. Both plug-ins should be easy to learn. UWV's process analyst should not spend more than a day to get familiar with both plug-ins features and functionalities.

4.6.3. NFR 3- Configurability

Both plug-ins can be used by any organization to do the analysis for their business processes. That's why design and implementation of all features should not have any external source dependency.

4.6.4. NFR 4- Scalability

UWV wants to process at least five hundred thousand events on the LogOnMapReplay plug-in. The current implementation does not support the processing of these many events. There is strong need to enhance the scalability of the tool.

4.6.5. NFR 5- Backward compatibility

All new features and performance enhancements done on LogOnMapReplay plug-in must be backward compatible with the existing implementation of the plug-in.

4.7 Design criteria

After the requirements were identified the following design criteria were chosen to be used in the design and implementation phase of the project. These criteria helped in evaluating the functional and non-functional requirements of the project.

- **Ease of use**: This is the main concern for the stakeholders. The end-user of the tool are business/ process analysts of the UWV. The tool should features which helps in the quick analyse of the data from the tool.
- **Performance**: UWV wants to process at least five hundred thousand events on the LogOnMapReplay plug-in. The current implementation does not support the processing of these many events. There is strong need to improve the memory and computation capabilities of the tool.

4.8 Project deliverables

The main deliverable excepted as an outcome of this project are:

1. An implementation for next version of the "LogOnMapReplay" plug-in based on the requirements.
2. A design and implementation of two new plug-ins named Automatic Map Generator (AMG) and MergeMapFile based on the company's requirements.
3. A project report that explains the project goal, scope, requirements and design decision to fulfil those requirements, and the results of the project.
4. Evaluation (as a part of the report) to show the capabilities of the plug-ins
5. A presentation that summarizes the project achievements.

5. System Architecture

This chapter describes the proposed architecture to meet the functional and non-functional requirements described in Chapter 4.

5.1 System Design

Since the ProM framework is easily extensible and enforces a plug-in based architecture we decided to implement two new plug-ins named AMG and MergeMapFile for creating and merging map files respectively. Figure 16 shows the system architecture that has been designed for this project. In the ProM framework, the data stores in-memory. This allows the user to import data once into the framework and later the same data can be used to apply various process mining techniques to do the further analysis with an event log.

Both AMG and LogOnMapReplay should use the same event log. The reason of using same file is, the AMG plug-in generates XQueries for a map based on the event names that are available in an event log. The LogOnMapReplay plug-in processes an event log based on the available XQueries in a map file. This reduces the risk of losing the events projection on the visualization panel of the LogOnMapReplay plug-in. The following section provides an overview of all plug-ins that are designed, implemented and used in this project.

- **Automatic Map Generator (AMG):** It is a new plug-in design and implemented in ProM framework. It aims for an automatic maps creation to support LogOnMapReplay plug-in. The AMG plug-in satisfies functional requirement FR1.1, FR1.2, and FR 1.3 requirement and also satisfies non-functional requirement NFR1 and NFR 2. The interface of this plugin requires an event log and optionally petri-net models which can be created or discovered using miner algorithms. The AMG plug-in supports PNML standard format petri-net models.
- **MergeMapFile**: This is a new plug-in which has been design and implemented into ProM framework. It allows the user to merge two or more map files. This plug-in satisfies functional requirement FR1.4. The interface of this plug-in requires minimum two and maximum five map files.
- **LogOnMapReplay**: This is an existing plug-in in the ProM framework. The plug-in design has been modified to improve the performance and scalability. This design change satisfies functional requirement FR 2 and non-functional requirement NFR 3, 4. The new features have been implemented to satisfy functional requirement FR 3, 4, 5 and non-functional requirement NFR1, 2.
- **Map file**: It contains maps such as Cartesian, Process map created by AMG plug-in or merged by MergeMapFile plug-ins. A map file is the main input file for LogOnMapReplay and MergeMapFile plug-ins.
- **Log:** The user can visualization and filter events/traces with various existing plug-in in ProM framework such as Extraction Sample of Traces plug-in. An event log supports various formats which as XES, MXML formats. The AMG and LogOnMapReplay plug-ins support XES format. There is one plug-in called XESLite which supports

XES format. We are using XESLite[3] plug-in because, it allows user to import more than three million event log in the ProM framework.

- **Miners**: Various miners' algorithms such as inductive miner and alpha miner are available in the ProM framework. Using these miner's techniques, the user can discovery a petri-net model from an event log.

[3] https://svn.win.tue.nl/trac/prom/browser/Documentation/Meetings/XESLite.pdf

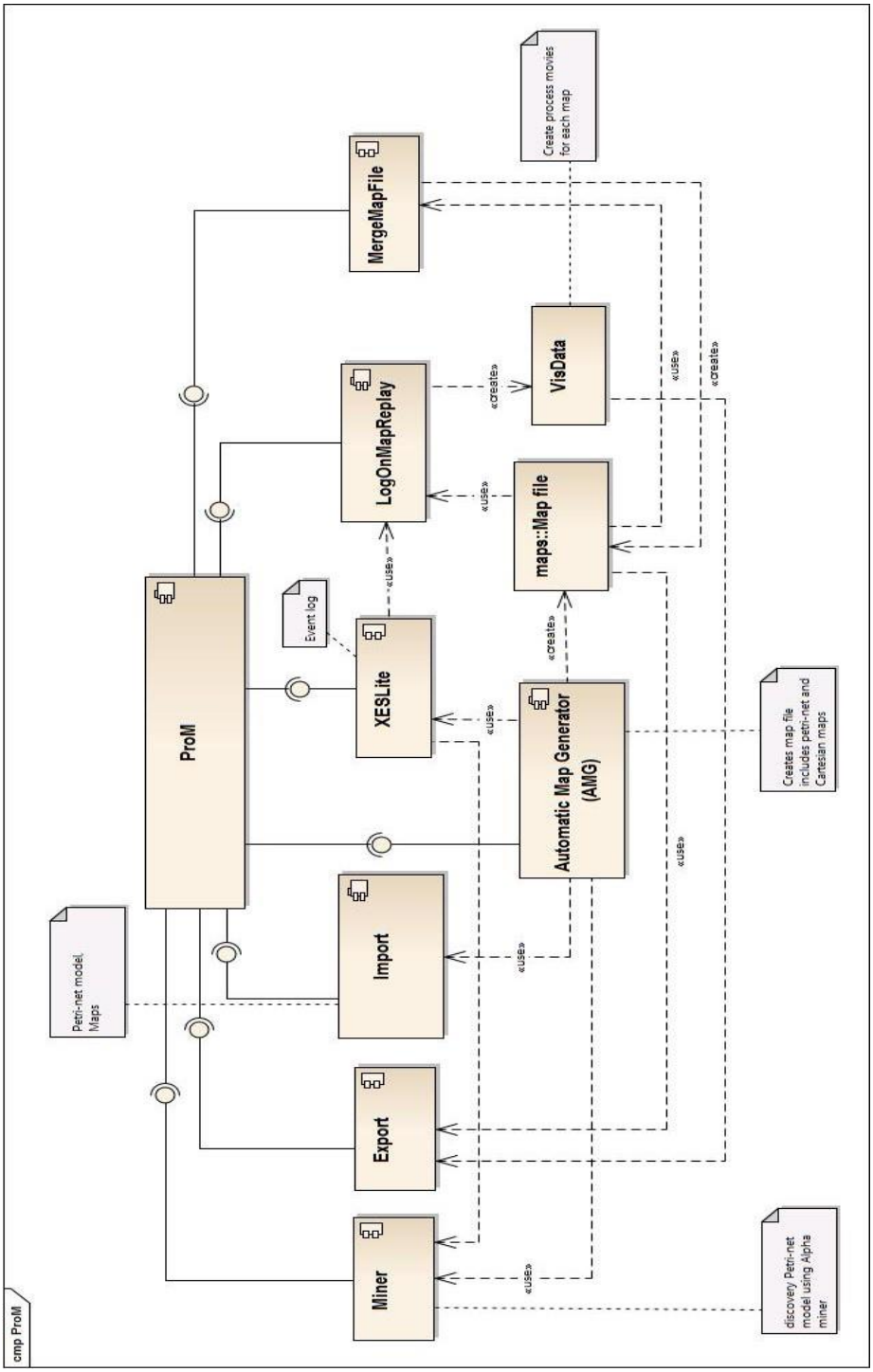

Figure 16: Overall system design

6. Automatic Map Generator plug-in

This chapter describes the Automatic Map Generator (AMG) plug-in requirements. This chapter also focus on the development process, description, and results of the plug-in. The main criterion for designing AMG plug-in is to generate a map configuration file and background images.

6.1 Plug-in description

The AMG plug-in generates a XML map file. A map file can have arbitrary number of maps. Each map contains the location of the generated image and XQueries which contains meaningful position function for their corresponding event. The GUI editor has been designed and developed for this plug-in. It allows the user to make the selection for the types of maps he/she would like to generate. A map file is one of the main input file for the LogOnMapReplay plug-in. The plug-in satisfies FR1requirement and its sub-requirements which are listed in the Table 7 and the NFR2 non-functional requirement. It will also satisfies the NFR2 non-functional requirement.

The high-level design of an AMG plug-in is shown in Figure 17. The plug-in takes an event log and petri-net models as input files. An event log is a mandatory input and petri-net models are only required when the user wants to generate a process model map. The plug-in allows the user make a selection for the number of Cartesian and process model map that needs to be generated. The selection of the process model maps depends on the number of petri-net models that are imported into the ProM framework. The plug-in supports the creation of nine process model maps and fifteen Cartesian maps in one go. Each map is generating an image and XQueries.

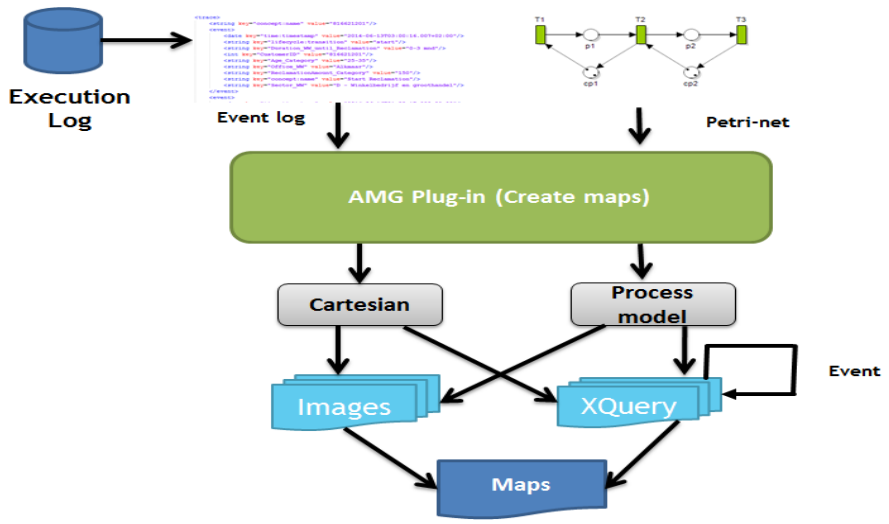

Figure 17: High level design of AMG tool

An image is a background image for a process movie. The XQueries are used while preprocessing an event log on the LogOnMapReplay plug-in to produce a process movie. The XQuery has the following tasks:

- It helps in identifying the type of dot such as valid, invalid, and hidden.

- It helps in defining the position (x and y coordinates) where the dot will be projected.

The requirements for designing the GUI and a generating a map file are different for both maps. That's why the design and implementation for both maps has been treated differently. For example, the creation of a Cartesian image is based on selecting x and y axis attributes and the creation of a process model map is based on the given petri-net model. The plug-in provides the mapping between the events of an event log and given petri-net model. Figure 18 shows the initial screen of the map selection. Here the user can choose the number of maps he/she wants to generate and the directory location where the plug-in saves the generated images.

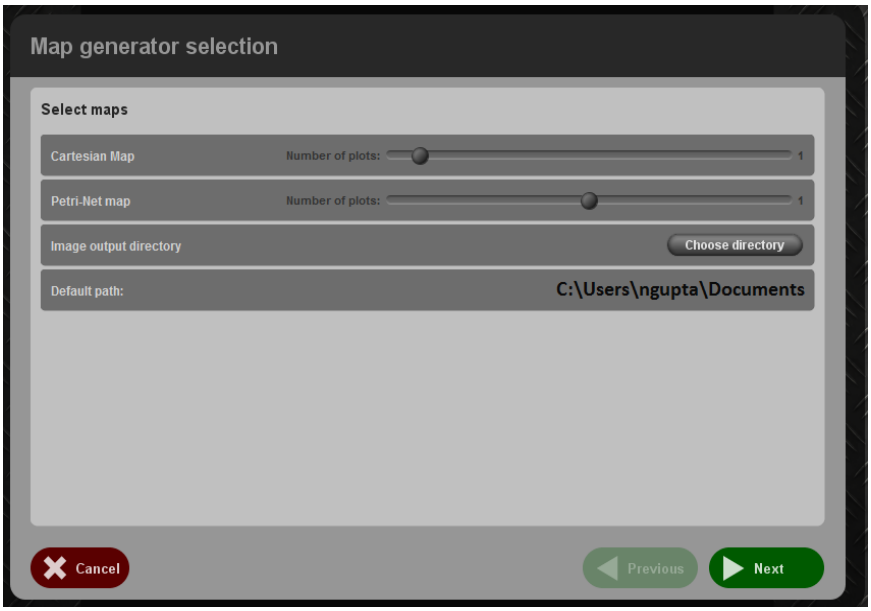

Figure 18: The map selection

6.2 Cartesian map

As mentioned before, a Cartesian map is inspired on the Cartesian coordinate system. It has x and y axis coordinates. An axis can be of a literal, continuous, or datetime type. For this project, we have focused on the literal (which is of a String or Boolean type) and continuous (which is an Integer or Float type).

Therefore, the plug-in allows the user to create four types of Cartesian maps namely Literal, Continuous, XLiteral, YLiteral. The definition for each map has been described in Section 4.5.1. Based on the selected x and y attributes, the plug-in decides the type of map, its needs to generate.

Figure 19 shows the initial screen for the Cartesian map, which is designed for the making attributes and events selection. The X and Y attributes combobox are populated based on the attributes available in an event log. The events list is also populated based on the unique events present in an event log. The user can also define the desired width and height for an image. The X and Y boundaries and their divisions are active when the user has selected a continuous attribute. For example age is a continuous attribute and office location such as Eindhoven, Amsterdam values is a literal attribute.

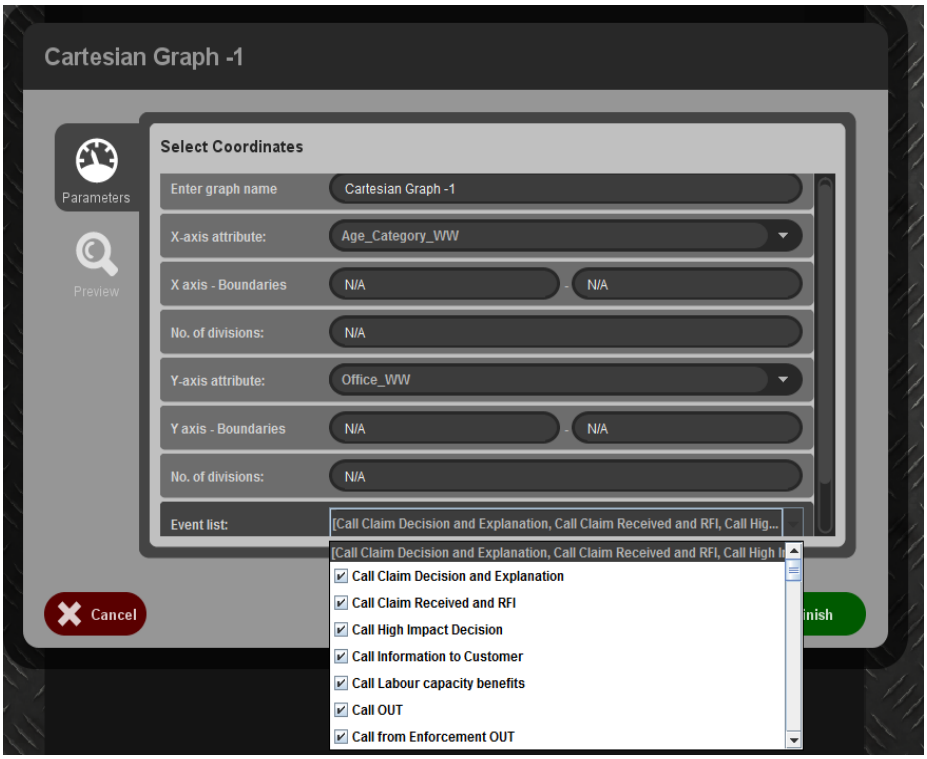

Figure 19: Selection coordinate for Cartesian map

6.2.1. Creation of an image

There are two factors need to be considered while creating an image: a line and a label.

1. **Line**: The number of lines needed to draw depends on the range specified for both x and y axis. The lines for a literal attribute are decided based on the unique values present in an event log for a chosen attribute. The lines for continuous map is based on the value specified by the user. The default value for number of division for a continuous attribute is five.

2. **Label**: The value for each line differs respective to its attribute type. The label for literal attributes is decided according to identified unique values for a selected attribute. For example, the user chooses a gender attribute which has two values such as Man, Women. The plug-in generates two labels for these respective lines.

   The label for continuous attributes is calculated based on the defined range (minimum and maximum values) such as [0, 1000] and the given number of divisions. Let $xMin, xMax$ defines the minimum and maximum value of an attribute, $xDiv$ the number of divisions,

and $lNum$ defines the line number. This is the formula to calculate the value for a label.

$$Label = xMin + (lNum - 1) * ((xMax - xMin) / xDiv)$$

For example, suppose the user chooses `age` attribute with $xMin = 0$, $xMax = 100$, $xDiv = 5$ and $lNum = 2$. The plug-in thus specifics

$$Label = 0 + 1 * ((100 - 0) / 5) = 20$$

Figure 20 shows a continuous Cartesian map which is generated by the AMG plug-in.

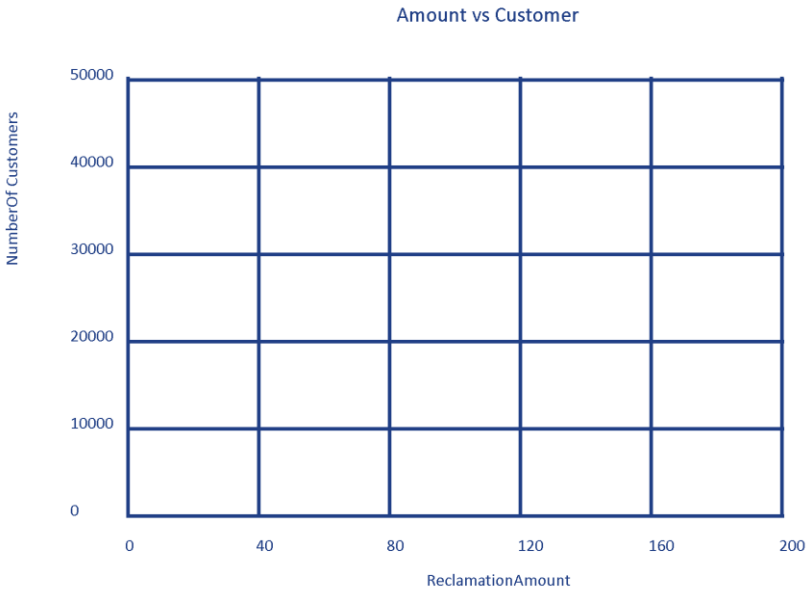

Figure 20: A continuous Cartesian map image generated by the AMG plug-in

6.2.2. Create XQuery

The selected attribute triggers the decision for generating either static or dynamic type of XQuery for an event. The user selects events for a map for which the plug-in should generates an XQuery. The plug-in identifies the type of query for an event based on the selected attribute type for x and y coordinates of a map. The attribute can be of a literal and continuous type.

Literal: The intersection points of all line of both axis are calculated because each intersection point has a specific values for x and y coordinates. The plug-in creates if-else based query. While processing data in the LogOnMapReplay identifies the exact x and y coordinates based on the attribute values present for an event and XQuery generated by AMG plug-in. Figure 21 depicts an example of dynamic XQuery generated by the AMG plug-in. Here `Age_Category_WW` attribute is chosen for y axis. The attribute has five unique values in an event log. For each unique value a static y coordinate has been decided.

```
<y>{if(string(//Age_Category_WW)= "18-25 jaar") then 1000
    else if(string(//Age_Category_WW)= "26-35 jaar") then 800
    else if(string(//Age_Category_WW)= "36-45 jaar") then 600
    else if(string(//Age_Category_WW)= "46-55 jaar") then 400
    else if(string(//Age_Category_WW)= "55 jaar of ouder") then 0
    else -1 }
</y>
```

Figure 21: An example of dynamic XQuery for a literal attribute

Continuous: The continuous attribute has a range of values for example an amount has a range of 0 to1000. Due to this reason, the plug-in generates a generic dynamic XQuery. The LogOnMapReplay plug-in runs a query processor which takes an event attribute value and a dynamic XQuery as input and the processor process the query and returns x and y coordinate. Based on the coordinate's values, the LogOnMapReplay plug-in decides the type of a dot

Consider px and py as the x and y axis coordinates, $xrow$ and $ycol$ are the number of rows and columns, lx and ly are the first labels of x and y axis which are generated while drawing an image, $xVal$, $yVal$ are the event attribute value for each axis and w and h are an image width and height. These are the formulas to calculate the x and y axis position.

$$px = (w / (xrow + 2)) + (xVal / lx) * (w / (xrow + 2))$$
$$py = h - (h / (ycol + 2) + (yVal / ly)) * h/ (ycol + 2))$$

For example, suppose the image size is (1800 x 1200) pixel (w= 1800 and h =1200). The amount and age attributes as $xVal$ and $yVal$. Let's consider $xVal$ = 200 and $yVal$ =30 and label values are lx = 20 and ly = 20. And map has $xrow=10$ and $ycol=5$.

$$px = (1800/(10+2)) + (200/20) * (1800/(10+2)) = 1650$$
$$py = 1200 - (1200/(5+2) + (30/20) * 1200/(5+2)) = 771$$

Figure 22 shows the detail design for creating a Cartesian map. Since the type of Cartesian map creation decision was on the fly that's a factory pattern is applied. The factory pattern[4] is a creational pattern which uses factory methods to deal with the problem of creating objects without specifying the exact class of object that will be created.

By applying this design pattern, we solved the creation problem for the Cartesian map and it also allows future extensibility for adding the

[4] http://www.oodesign.com/factory-pattern.html

functionalities to create more Cartesian maps such as datetime Cartesian map. This design helps in satisfying NFR1 non-functional requirement. The *CartesianFactory* class decides the type of map. The *Cartesian* class has four types of Cartesian maps. Each child class is responsible to create an image and its XQueries.

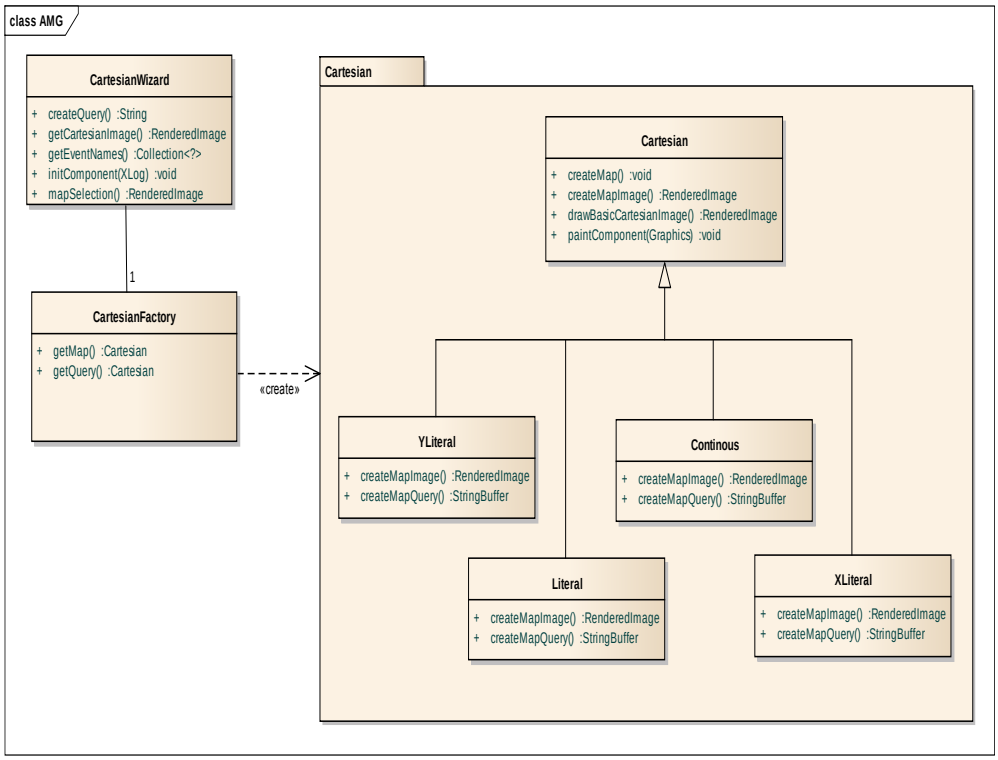

Figure 22: Detail design for creating a Cartesian map

6.2.3. Process model map

As mentioned earlier, the process model map visualizes the events onto the corresponding transition in the given model. In order to activate the plug-in for the creating a process model map, the user has to provide the Petri-net formalism and an event log. In this project, we focused only on the Petri-net formalism, but we believe that it is relatively easy to extend the implementation to support other process modelling formalisms. There was a different set of requirements for designing the GUI for process model map as compare to the Cartesian map. The main requirements of GUI are as follows:

- Allow user to map an event with any transition of a petri-net model.
- Provide suggestions for the most suitable match between process instances and transitions of a petri-net model.
- Preview screen of petri-net model.
- Allow user to map a transition of a petri-net model to one, many or no mapping corresponding to an events of an event log.

The mapping schema is designed to satisfy the FR 1.3 functional requirement. The plug-in should be allowed to do three types of mappings which are one-to-one, one-to-Many and no mapping and the plug-in should not be allowed to do many-to-many mapping . The Table 11 defines the mapping schema in detail. Based on the select mapping, the plug-in will be generating a position function which is the XQuery.

Table 11: The mapping schema for creating position function

Mapping scheme	Possibility	Description
No Mapping	Yes	No position function
One-to-one	Yes	One transition of petri-net model is mapped with one event. It will not share the position function.
One-to-Many	Yes	One transition has multiple events. The events which are selected to be visualize onto the same transition will be sharing the position function.
Many-to-Many	No	It is not possible to have a mapping of one event to multiple transition. Because an event such as Claim, payment can have only one active process instance for a customer. For example Customer A can only have one active Claim for a certain duration

Figure 23 shows, an example of the mapping schema that has been designed for creating a position function. An event log has five unique events and a petri-net model has three transition. The Claim and Change form out events are mapped to the Claim transition. The Payment event is only mapped with Payment1 transition. And there is no mapping for a start reclamation event.

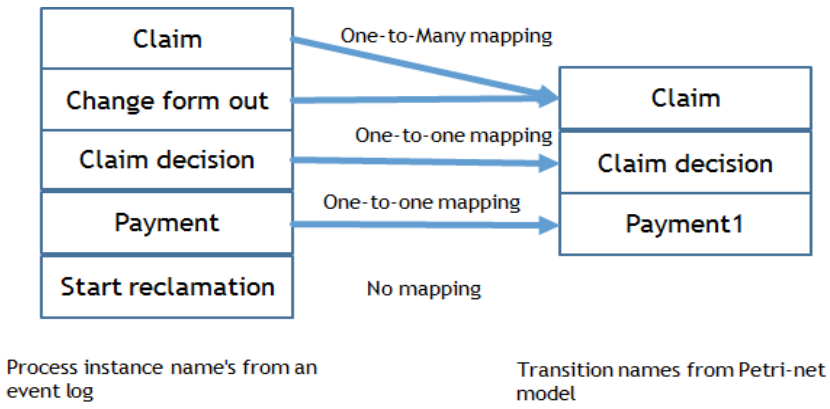

Figure 23: An example of process model mapping schema

Figure 24 shows the GUI that has been designed for creating a process model map. On the left side, the plug-in list unique events that are available in an event log. The transitions of the petri-net model are shown on the right side

of the GUI panel. Based on the mapping schema, each event has a list of all the transitions of a petri-net model. Based on the given input files, the GUI has to find the best match between an event and a transition. The best match means give either same name or the closest name. The *Levenshtein distance* algorithm is used to calculate the best match .If GUI found a match of both then the combo box of transition will be of grey color otherwise show the closest match but with yellow color. The different colors on the combo boxes helps the user in identifying events which are matched or not matched with the provided petri-net model. The GUI provides suggestions only, but the user is allowed to mapping an event with any transition of the petri-net model.

To satisfy the no mapping requirement, a NO MAPPING string has been added to the petri-net list. The user can choose to put all events on no mapping which do not have a match by select a default NO MAPPING check box. Based on the user selection, the plug-in will generate a static XQuery for all events except those events that have s no mapping option selected.

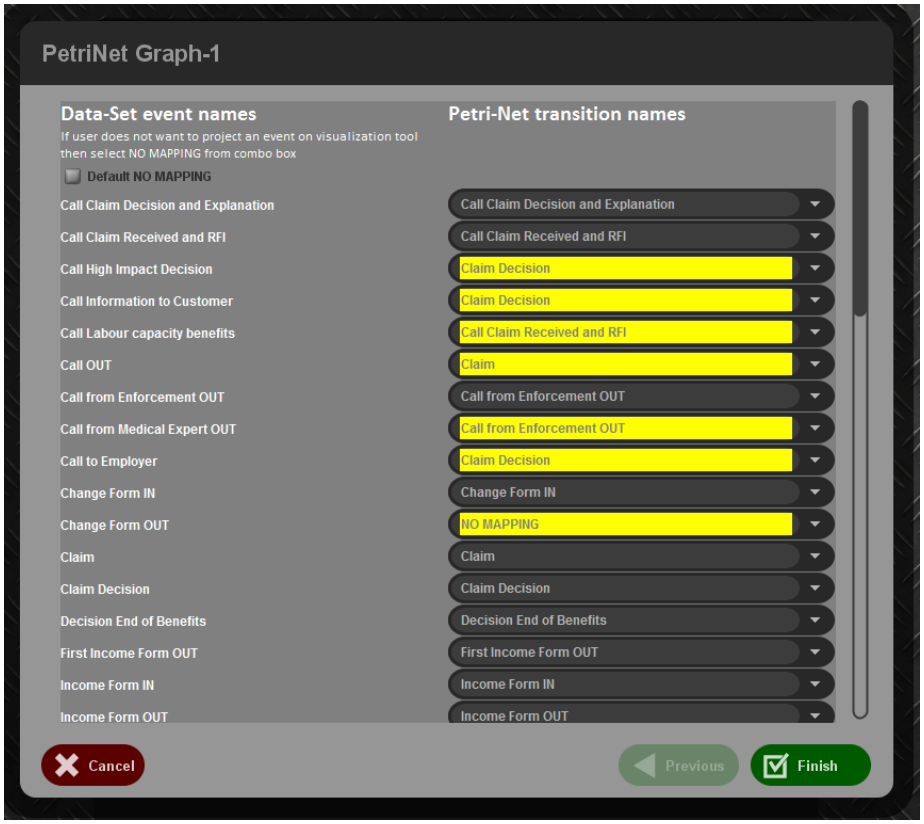

Figure 24: The GUI of the process model map

Figure 25 shows the detail design for creating a process model map. The PetrinetGraph and ProMJGraph API are used to read and render the coordinates of all the transitions that are available in a given petri-net model. XLog API is used to read an event log. The petriNetWizard class is handling the GUI design, which includes showing event that are available in an event log and also find and display the best match for the event. Since for this project, we have considered only petri-net models that why no pattern has been applied. But we believe that it is relatively easy to extend the design and implementation to support other process modelling formalisms.

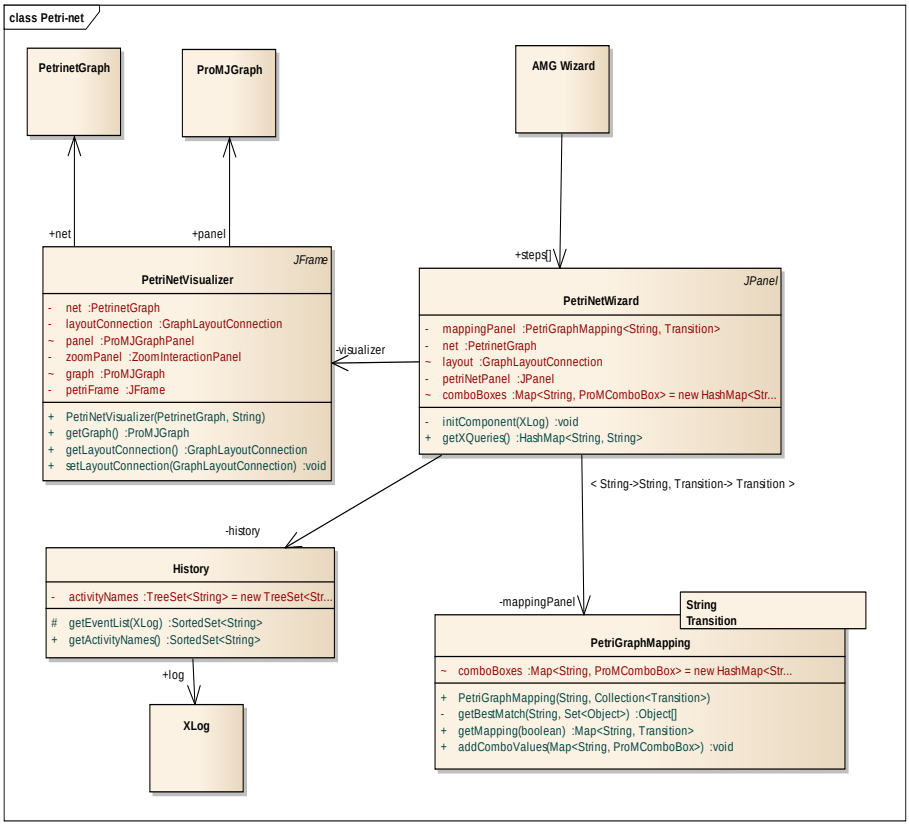

Figure 25: Detail design of a process model

7. Optimization of LogOnMapReplay plug-in

This chapter describes the limitations of the existing design of the LogOnMapReplay plug-in. then we focus on how the system is re-designed and implemented to improve the scalability of the plug-in.

7.1 Bottlenecks

After code analysis and several debugging rounds with various event log files, we identified the following limitations of the plug-in:

- Storing objects in-memory: In the existing implementation, the movie was stored in the heap memory.
- Sequential process: An event log was processed for each movie in a sequential order.
- Generic XQuery processing for maps: The processing of an XQuery takes significant amount of time.
- Code optimization: Identified the scope of doing general code optimization.
  Further in this chapter, we discuss how we handled each of the limitation to improve the processing capacity of the plug-in.

7.2 Storing object in-memory

One of biggest bottlenecks for processing a big event log is that the plug-in stores the processed movie object data in-memory. As the number of events grows, the plug-in requires more memory space. At certain point-in-time, the plug-in goes out-of-memory and it stops processing the data. Figure 26 depicts this problem of the plug-in.

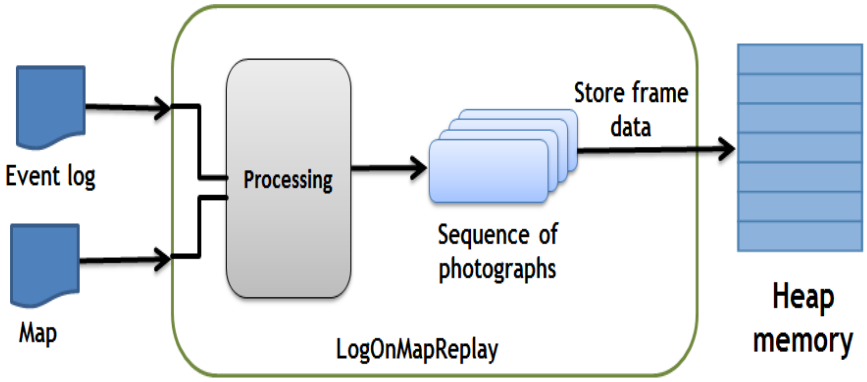

Figure 26: Problem in the current design

The memory consumption mainly depends on two factors. First, the number of active events and their life span (which is defined by the duration of an event). Second, the numbers of active events stored in each photograph. A photograph is composed of three sets, which respectively, include pie-chart with valid, invalid, and hidden. The photographs data is stored in a Map. A map is an interface that maps records key value pair. In our case, key is a timeframe and value is a list of active events. The list is stored in a serialized manner. Serialization is the process of turning a Java object into byte array and then back into object again with its preserved state. It is useful for various things such as sending objects over network or caching things to disk.

Since the RAM memory is limited on any system, to the processed objects and save them in-memory, a solution is needed which can save the serialized map objects on a disk. To support the processing of big data, a NoSQL database is needed. Because it is useful, when an application needs to access and analyze huge amount of unstructured data. There are various type of NoSQL databases such as Key-value Store, and Column-Oriented Store. A key-value store, allows the user to store data in a similar way as stored in a Map data structure. The Key-value stores are designed to store and scale up to millions or billions of key-value records. A Column-Oriented Store, allows to process and disturbed stores huge amount of data on different machines. The LogOnMapReplay plug-in stores serialized object in map on a single machine. Due to this reason, a Key-value Store NoSQL database was chosen to overcome the limitation of the plug-in. The main criteria for selection of a NoSQL database is following:

A. It should be written in Java because the LogOnMapReplay plug-in is written in Java. We wanted to minimize the overhead reading and writing the data into the database.
B. It should not add any dependencies in the existing plug-in. This is also one of the non-functional requirement (Configurability) of the project.
C. It should allow to write data on-disk.
D. It should allow to concurrent read/write the data.
E. It should be open-source.

7.2.1. Approach for choosing database

The approach followed during the NoSQL database selection process is as follows:

1. A list of available key-value databases on the market was created. An initial evaluation resulting in a short list of key-value java databases was performed.
2. An evaluation of the short list of key-value database was carried out. It resulted in the below comparison table of the key-value tools of the short list.

A list of available databases was created by searching online and discussions with peers.

Table 12: Comparison of key-value databases

	MapDB	BananaDB	Berkeley DB
Write in-disk space	Yes	Yes	Yes
Allow concurrently read/write	Yes	Yes	Yes
Good documentation	Yes	No	Yes
Ease of use	Yes	Unknown	Unknown
Active community	Yes	No	No
Licence	Open-source	Open-source	Commercial

After evaluating the available NoSQL database. The MapDB database is chosen for this project. The following are the reasons for choosing MapDB:

- It meets all the requirements that are mentioned above
- In ProM framework there is one plug-in called XESLite plug-in, which has already used MapDB database to read and write more than three million events of an event log and stored in-disk. This way, we minimized the risk of hitting the database selection boundaries at a later stage of the project.

7.2.2. MapDB

MapDB (kotek, 2015) is a java based database engine. It supports various storage modes, one of them is off-heap memory. It also supports hash based key-value pair stores on disk. The following features of MapDB that helped improve the scalability of the plug-in:

- **Low disk-space usage:** It serializes the objects and stores them on-disk, in a compressed binary format.
- **Concurrent:** It has record level locking and a concurrent engine. Its performance scales nearly linearly with the number of CPU cores. Therefore, data can be written by multiple parallel threads.
- **Fast:** The read and write of objects on-disk is fast. It can approximately write 1.7 billion objects in 30 minutes.
- **Flexible:** It can be used everywhere from in-memory cache to multi-terabyte database. This makes easy to configure MapDB to exactly fit the needs of the project.

Figure 27 shows the re-designed solution for enhancing the scalability of the plug-in. In the new design, the plug-in stores the sequence of photographs objects into the MapDB. When the plug-in starts processing an event log, it creates a new MapDB database file at the user-specified location.

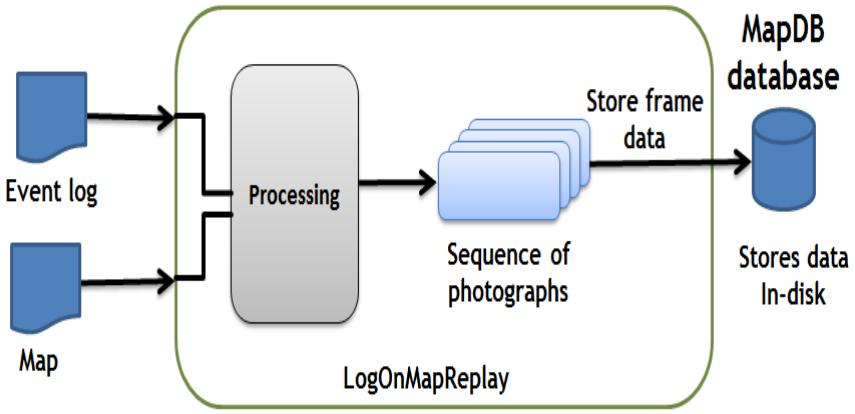

Figure 27: Re-designed solution for improving the performance using MapDB

7.2.3. NoSQL database structure

The structure of the database for storing the movies was derived from the existing schema of storing a process movie. The database structure created for the movies is shown in Figure 28. The *Maps* table is related to the type of maps available in the map file. Each map of the map file is described with a *Map name* and *queries*. An *ActivityKey* table describes the *trace_id* and *event_name* from an event log. An *AbstractWorkItem* table defines the type dots such as valid, invalid, and hidden. For each *AbstractWorkItem* table, the database stores its *ActivityKey*, *creation time*, *state*, *x* and *y* coordinate. A movie is generated for each map, and each map can be identified by its unique name. The *Movies* table, contains data corresponding to each movie. Each movie contains a Map which has timeframes and a list of *AbstractWorkItem* created for generating a process movie.

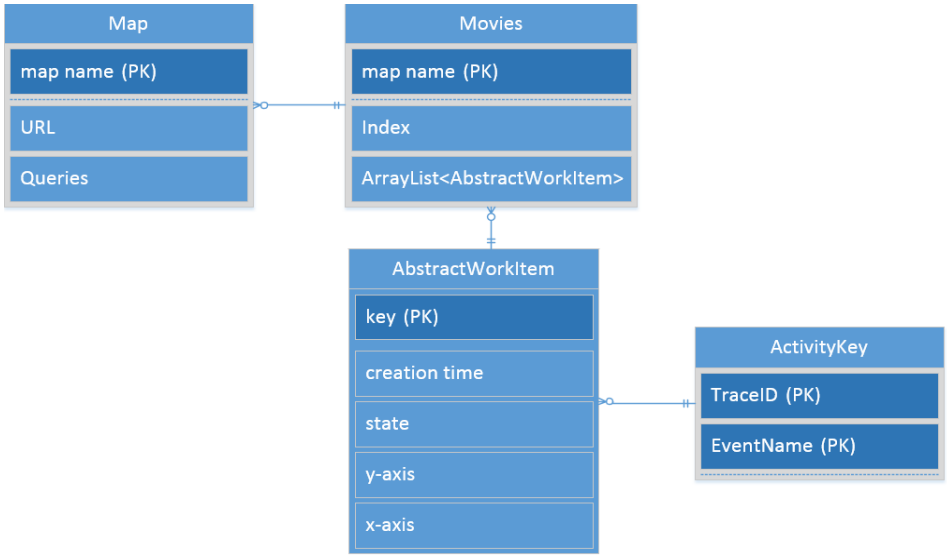

Figure 28: Database structure for storing movies

7.2.4. Re-design of LogOnMapReplay plug-in

Figure 29 is the class diagram that shows the details of the design structure after introducing MapDB into the plug-in.

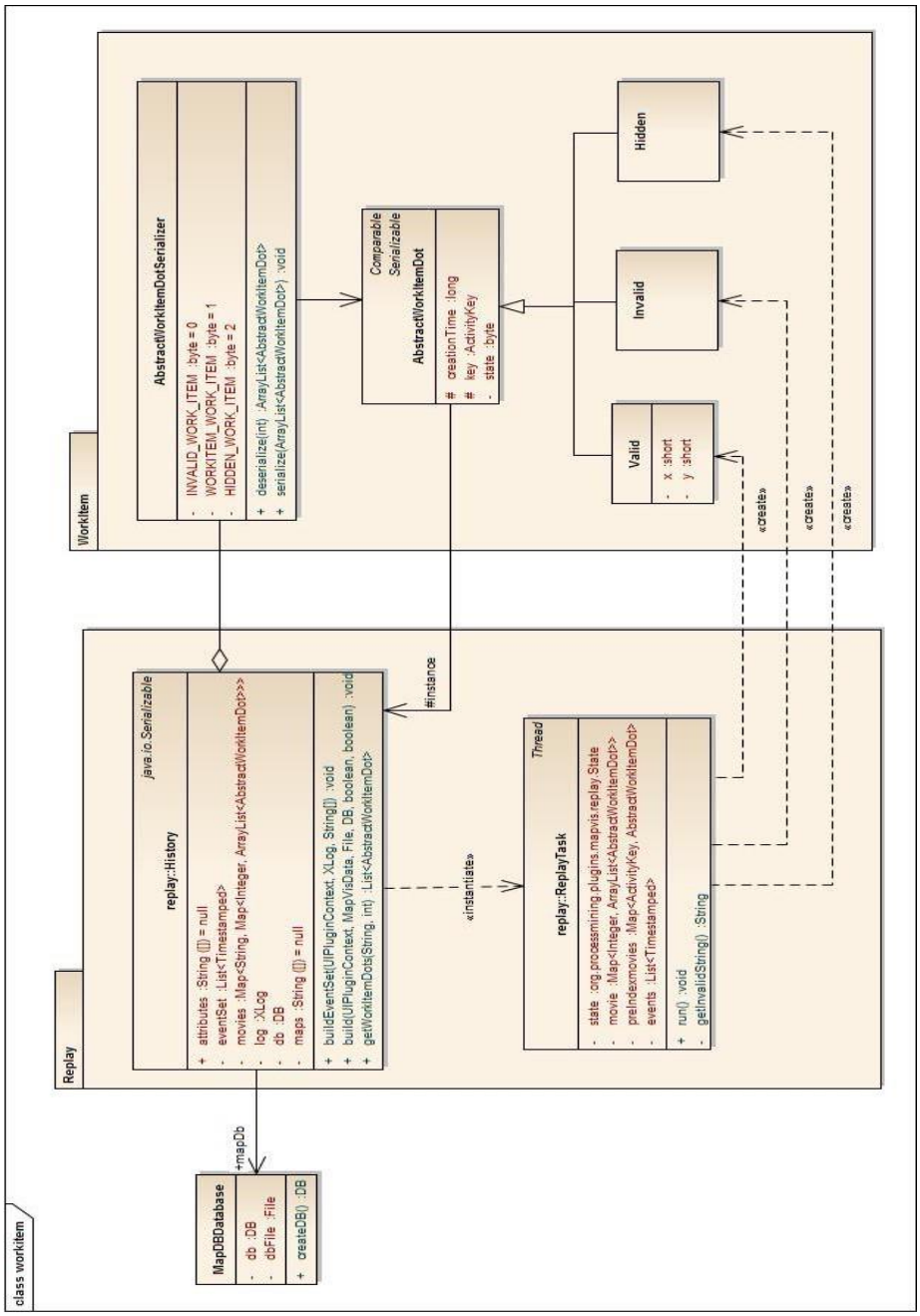

Figure 29: Detail design diagram after introducing MapDB

The *MapDBDataBase* class creates the MapDB database file on-disk. It has two main concepts:

1. ***DBMaker***: It is a builder style factory for configuring and opening a database.

2. **DB**: It represents storage and it has interfaces for accessing Maps and other java collections.

A *Serializer* is implemented to store the list of *AbstractWorkItemDot* objects for each timeframe. A builder pattern is used, to create different type of *AbstractWorkItemDot* step-by-step The *AbstractWorkItemDotSerializer* class takes care of the serialization and deserialization of objects which are created for a process movie. It is important to write a Serializer because this leads to better memory and computation performance because MapDB does not have to analyses the *Map* structure. The Serializer also helps in writing the data in a compatible format with standard Java serialization. To identify the type of dot, the following identifiers are created:

1. *INVALID_WORK_ITEM* refers to the invalid dot.
2. *WORKITEM_WORK_ITEM* refers to the valid dot.
3. *HIDDEN_WORK_ITEM* refers to the hidden dot.

The *ReplaytTask* class is a thread class that helps in concurrently processing events for each map. The class is responsible for running the XQueries for each event. The result after running an XQuery helps in identifying the type of dot. Table 13 shows the basis at which the framework identifies the type of dot. Here, the common factors are image height and image width which are *h* and *w* respectively and X and Y are values received after running XQuery.

Table 13: Identification of type of dot

X	Y	Type of dot
> =0 and < =w	> =0 and <= h	Valid
< 0	< 0	Invalid
> w	> h	Hidden

The *History* class, contains the references of movies produced by the plug-in. This is main class holds the implementation of building the process for producing movies.

By introducing MapDB we have removed the limitation of processing limited number of activities to produce a process movie. Using MapDB, we have partially satisfied FR2 functional requirement and NFR 3 and 4 non-functional requirement of the project. Since, the movie related objects are stored in-disk, the limitation of processing number of events has moved from available heap memory size to the amount of space available the on-disk.

7.3 Multithreading implementation

UWV is planning to run the LogOnMapReplay plug-in on a server, it is of CPU 16 cores processor. This leads us to explore the opportunity for applying parallelism in the plug-in. Multithreading is the ability of the CPU to execute multiple threads or processes concurrently. The aim to increase the resource utilization of a system. The system resources are shared among threads and processes.

After code analysis, we have evaluated that the movie of each map is built regardless of the other movies. Each movie produces a list of objects based on the available XQueries for a corresponding map. This gives us an

opportunity to apply multithreading for processing the data for each map of a map file.

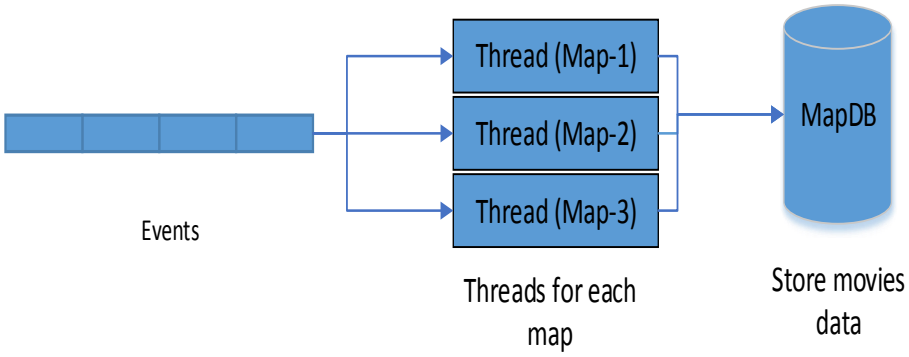

Figure 30: Re-designed structure for applying parallelism

Figure 30 depicts the re-design structure of the plug-in to apply parallelism. Each map has a list of events that are available in an event log. Each map is run on a separate thread, it is running on a separate core of the system. Each map is concurrently accessing MapDB to store the processed and produced data into the database.

The *ThreadPool*[5] design pattern is introduced in the plug-in. Now each movie runs on a separate thread which is submitted to the thread pool. A thread will be terminated when all the events are processed for a map. The data processing of a LogOnMapReplay plug-in will be completed, when all threads in a thread pool has competed it processing. A thread is concurrently writing data into the MapDB database. This helps in improving the computation time of the plug-in because the processing dependency between the maps is removed.

7.4 Improvement in a map file XML schema

Various experiments were conducted using the YourKit java profiling tool to find the methods that take signification amount of CPU time. The profiling tool indicated that the XQuery processor takes significant amount of time to run XQueries on the fly. After code analysis, we identified that if the plug-in is aware of the x and y coordinates for an event then the processing of XQuery step can be skipped. Map evaluation was conducted to identify the maps which for the plug-in can find x and y coordinates beforehand. After evaluation of all maps created by the AMG plug-in, we identified that a Cartesian Literal and a process model map are the ones for which plug-in is aware of their x and y coordinates beforehand.

[5] https://en.wikipedia.org/wiki/Thread_pool_pattern

Figure 31 shows an example of both static and dynamic XQuery. Here, a *Letter Reclamation* event has static XQuery. In this query, the plug-in is already aware of the exact x and y coordinate for an event. The plug-in can skip the XQuery processing step and the x and y position can be directly used to identify the type of dot.

The *Letter Reintegration during sickness* event has a dynamic XQuery. Here, the plug-in gets the value of age attribute at run-time and then it will find out the axis coordinates. That's why, the plug-in still requires to run a dynamic XQuery to find coordinates.

```
<query task="Letter Reclamation">
    <xquery>
        <![CDATA[ <coordinate> <x>451</x> <y>614</y> <z>0</z> </coordinate>]]>
    </xquery>
</query>


<query task="Letter Reintegration during sickness">
    <xquery>
        <![CDATA[<coordinate>
                    <x>{257 + (number(//age) div   12867.2) *257}</x>
                    <y>{1200- (171 + ((number(//age)) div 12867.2)*171)}</y>
                    <z>0</z>
                </coordinate>]]>
    </xquery>
</query>
```

Figure 31: Comparison of XQuery

To create static XQueries a Cartesian Literal, the map file schema has been modified. The new schema is shown in Appendix B. Now, all maps generated by AMG plug-in have been adapted to the new schema. To maintain the backward compatibility, the LogOnMapReplay plug-in supports processing with the old schema of the maps as well. Both schema are shown in Appendix A and B.

7.5 Code optimization

After code evaluation, we identified the opportunity for the code optimization. One of the code optimization is the search process for finding an *AbstractWorkItem* from previous timeframe. In existing algorithm, linear search was used to find an *AbstractWorkItem* from the previous timeframe. A trace id, event name, state, and creation time attributes of an activity defines whether it is a new activity or an existing activity. The algorithm scans and compares all objects in the loop, until it finds the same match between the attributes values of a new activity and existing *AbstractWorkItem* from the previous timeframe. The worst case computation time taken by this algorithm is: $O(n)$ because the algorithm is linearly comparing the objects.

A HashMap is introduced to a look-up map storing the results of the previous time frame of the movie. Here key is the combination of a trace *id* and *activity name* and value is an *AbstractWorkItem*. The HashMap implementation provides constant-time performance for the basic operations such as *get* and *put*. The *hashcode* for the key value are written for faster retrieval of the objects from a map. The worst case computation time taken by this algorithm is: $O(1)$

This code optimization helps in reducing the computation time of the plug-in because the previous timeframe data cached. Such code optimization techniques are applied at various places in the code to reduce the overall computation time of the plug-in.

7.6 Other factors impact the performance of the plug-in

The following external factors are listed, which also impacts overall the performance the plug-in.

1. **Import of an event log:** The XESLite plug-in provides two types of import options: Naive and MapDB disk buffered.
    a. Naïve: This options stores events in-memory. The advantage of this option is, the processing of read and write events is fast. The limitation of this option is, based on the available memory size, the number of events can be imported into the ProM framework.
    b. MapDB disk buffered: This will stores the event data into local or network disk. Using this option, the user can import more than three million events. The drawback of this option is that, the MapDB takes significant amount of time to read and write events.
2. **Location of reading/write data**: MapDB writes data into disk space and writing data on disk is a costly operation. The user can choose to let MapDB read and write data from local disk or network drive. The performance plug-in will be slow while reading and writing data over network drive as compare to local disk. One of the reason is write/read data over network adds the network latency. That's why it is important for the user to choose the appropriate location.
3. **System configuration**: The performance of plug-in is also dependent on the system configuration. The system should have at least 6 GB RAM size to process an event log otherwise the plug-in has to perform garage collection operations, which is an expensive operation in-terms of computations.
4. **General granularity of log**: If an event has multiple occurrences at the same timeframe with the same attribute values, then this event is not unique. It is an advice to filter these kind of events from an event log because it reduces the size of the log hence the computation and memory performance of the plug-in will be improved. Since the information is lost but this can be compensated by adding an attribute with the number of occurrences of the unique event. This attribute should be used to sum the events in a movie.

7.7 Results

An experiment was conducted using three event log to evaluate the performance of LogOnMapReplay plug-in after applying all optimization techniques. To process data, we used a laptop equipped with an Intel Core i7 Processor of 2.20GHz. For each experiment a different log was as described in Table 14. The Table 14 shows the results of before and applying the optimization techniques. It also reflects the total time gain achieved after improving the performance of the plug-in. With these results, we believe that now plug-in is capable of processing more than five hundred thousand events.

Table 14: Comparison of before and after applying optimization techniques

Run ID	**Dataset (number of activities)**	**Map number**	**Computation time (ms)**	**Computation time (ms)**	**Gain time (ms)**	**Gain time (%)**
			Before optimization	After optimization		
1	188797	3	2551462	136892	2414570	94.63
2	247468	1	16955623	2457597	14498026	85.50
3	495566	1	Out-of-memory	6848658	6848658	Cannot compare

8. Features of LogOnMapReplay plug-in

This chapter describes the new features that are designed and implemented of the LogOnMapReplay plug-in and MergeMapFile plug-in.

8.1 Filter of event and attribute

We introduced a filter mechanism which helps in narrowing down the scope of analyzing and deriving conclusion from the projecting dots on the map. It allows a process analyst to view only those events and attribute values which are of their interest. This mechanism will filter all type of dots such as valid, invalid and hidden dots. The dots that meet criteria will be projected on the visualization panel and rest of the dots will discarded. The user can filter events and attributes at any point in time while watching the visualizing of a process movie. This mechanism is satisfying FR 3 functional requirement including both sub-requirements.

As mentioned before, each dot contains a trace id, an event name. With this combination our framework can identify the uniqueness of the dot. The pie-charts dots and both sidebar lists will be populated based on the selected events and attributes. Each dot contains a trace id and its event name. The user can apply filter in three ways: Only events, only attributes, and both events and attributes.

- **Only events**: The user selects their area-of-interest only events. The framework checks whether the dot has selected event name or not. If the dot event name and selected event name match then visualization panel will project the dot onto the map otherwise it will discard that dot. Based on the selection of events, the diameter of a pie-charts will also changes. There is OR relationship between events. Figure 36, is a screenshot of filter panel which has listed all the events available in an event log.

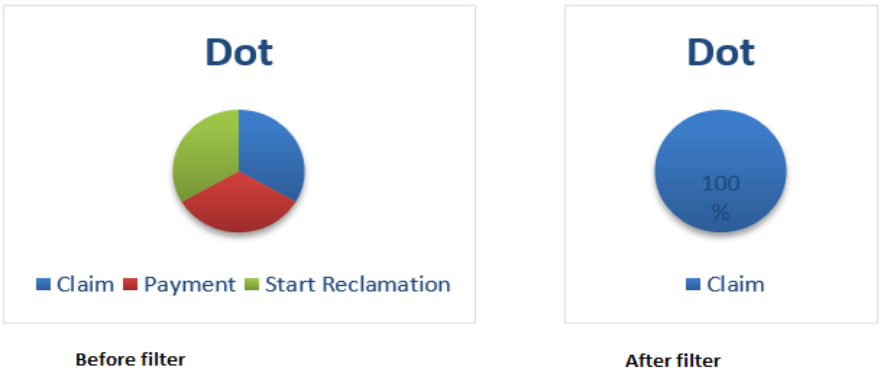

Figure 32: Filter before and after

Figure 32 shows how the dots are projected before and after applying filter mechanism For example, suppose a merged dot of a particular coordinates which consist of five claim, five payment, and five start reclamation events at a certain point in time. The diameter of this merged dot is fifteen.
The user wants to filter on claim event only. The framework will check whether all dots of a merged dot is of a claim event or not. Here the framework will project only claim event and now the dot diameter is five.

- **Only attributes**: The user selects their area-of-interest attributes. The framework checks the type of attribute: literal and continuous for the selected attributes.
    - If an attribute is of a continuous type, then the user has to specify the range then the framework evaluates whether the dot is in mentioned range or not. If the dot is the specified range then the framework will project the dot otherwise the framework will discard the dot.
    - If an attribute is of a literal type, then the framework checks whether the selected attribute value and an event value are same or not. If selected attribute value existing on the event then the framework will project the dot on the map otherwise the framework will discard the dot.

    There is *AND* relationship with other selected attributes, and *OR* relationship with in the same attribute. Figure 37, show the different attribute which are selected, the literal attribute has a combo box and a continuous attribute with text boxes.

- **Both events and attributes**: The user will select both events and attributes of their area-of-interest. There is an *AND* relationship between events and attributes. A dot must be from a selected events list and a dot should also have the selected values for the chosen attributes.

Figure 33 is a screenshot of a filter panel which is created for filtering events and attributes. After clicking the *Select attribute* button, the list of attribute available in an event log is shown. The end-user can chose all attributes of their interest.

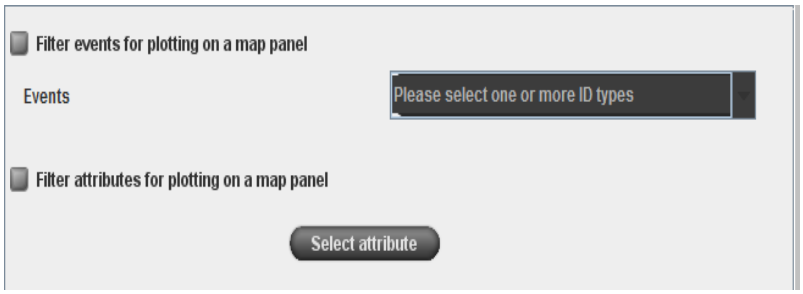

Figure 33: Screenshot of the filter panel

Figure 34 shows the screenshot of a filter panel with selected attributes. Few attribute such as *Age category WW*, *Gender* has combobox because these attributes are literal and they have pre-defined unique values. The attribute such as *NumberOf Customer*, *ReclamationAmount* are of continuous type. Their minimum and maximum values that is available in

the dataset are shown in textboxes. Here the user can define the range for which they are interested in applying filtration on the visualization panel.

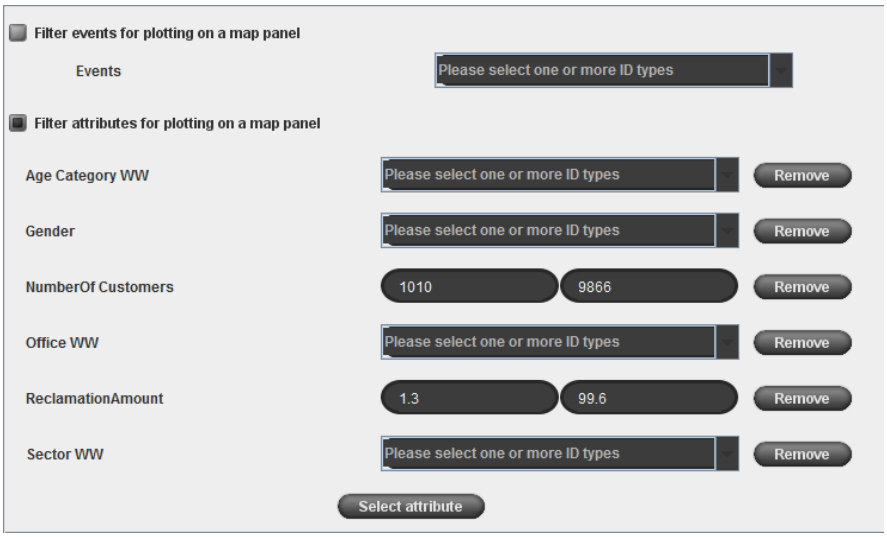

Figure 34: Screenshot of the filter panel with selected attributes

8.2 Show deviations

We introduced a comparative mechanism for detecting, when something unusually happens in an event log. We introduced two ways for analysing the deviation in an event log: indexing and relative. This mechanism is satisfying FR 5 functional requirement including both sub-requirements

8.2.1. Indexation

Indexation means whether the number of events has increased or decreased from a certain point-in-time. Instead of showing the number of running events, the process movie shows the difference in the number of running events

The visualization panel is projecting different events at different coordinates at a particular time. If number of events has increased from a selected timeframe then the plug-in will paint dots with black color. When the number of event has decreased from a certain time then the plug-in will paint dots with yellow color. The framework can encounter three type of following situations:

1. If the coordinates are available on both list (selected time and new time) then the framework will calculate the difference between the numbers of dots that are available on both timeframe. The framework will project the difference only. If the difference is positive then the framework will project the difference in black colour otherwise it

will project in yellow colour. For example the coordinates (200,100) is available in both t1 and t2 timeframes as mentioned below. The framework calculates the difference (which is18 shown in index) and the diameter of dot will the difference value and the framework will painting the difference at same coordinate.

2. If selected timeframe does not have dots at certain coordinate but the next or previous timeframe has dots at that coordinate. This means the number of events are increasing and the framework calculates the difference and it will paint the dots with black colour.
3. If selected timeframe does have dots at certain coordinates but the next or previous timeframe does not have dots at that coordinates. This means the number of events are decreasing and the framework calculates the difference and it will paint the dots with yellow colour. The two timeframes T1 and T2 with their the number of events at are available specified coordinates are as follows:

$$T1 = \begin{cases} (200,100)\ project\ 20\ events \\ (200,300) project\ 10\ events \\ (150,150) project\ 50\ events \end{cases}$$

$$T2 = \begin{cases} (200,100)\ project\ 2\ events \\ (150,150) project\ 10\ events \\ (300,300) project\ 10\ events \end{cases}$$

Table 15 shows the difference that has calculated for above T1 and T2 timeframe and the number of events that are available for each coordinate.

Table 15: Calculate value for index mechanism

Coordinate	T1	T2	Index
200,100	20	2	18
200,300	10	0	-10
150,150	50	10	-40
300,300	0	10	10

Figure 35, shows the screenshot of a Cartesian map. Here, at Jan 14, 2014 at 11:31:21 was chosen as the indexing reference point.

Figure 36, shows a screen shot of the actual visualization without applying indexing. A random time frame was chosen different from the selected time from where the indexing has been applied on the visualization panel.

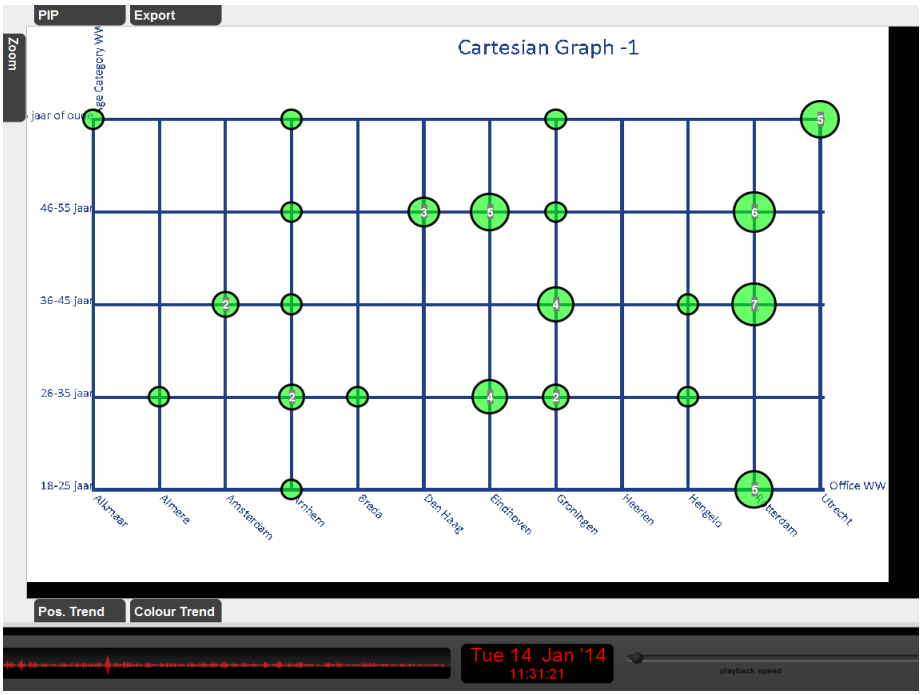

Figure 35: Screenshot of an index frame

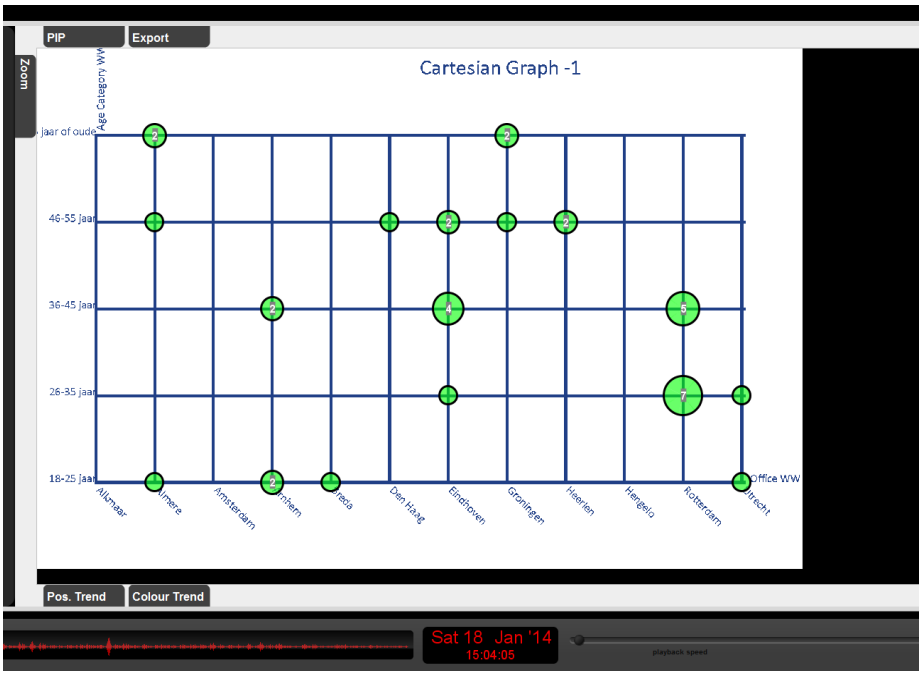

Figure 36: A screen shot without indexing

Figure 37. This timeframe compares the dots at each coordinates. Based on the algorithm that is discussed above. The plug-in projects dots in black and yellow color. This helps a process analyst, whether the number of business processes activity have increased or decreased from a certain time.

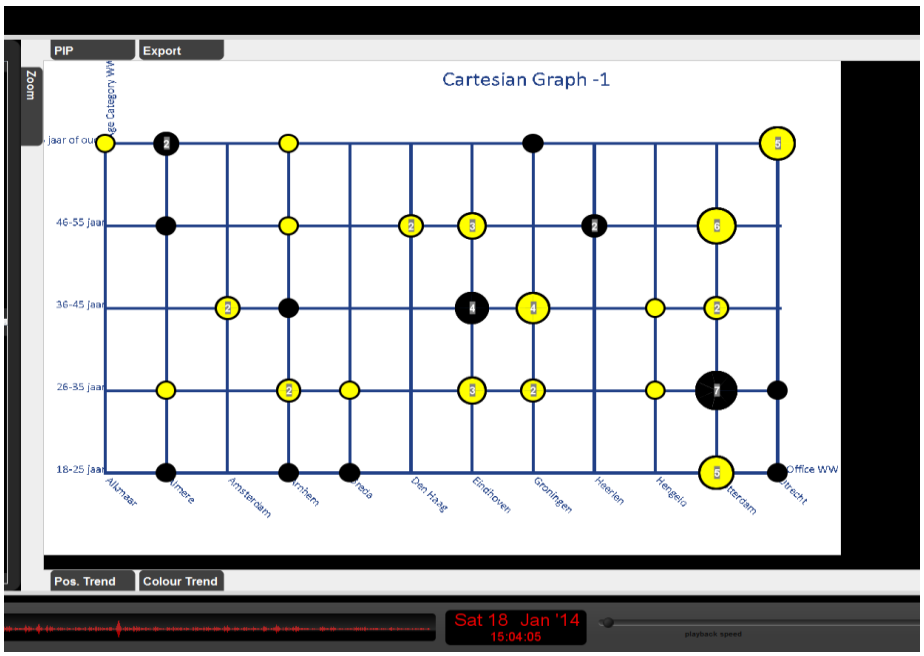

Figure 37 : Screenshot with applied indexation

8.2.2. Relative

In current implementation, the pie-chart size is determined by formula $n^{.04}$, where n is the number of events that are joined and merged at the same x and y coordinates. In new approach, the end-user will select a numeric attribute for which the drifts in the events needs to be detected and visualized. The drift corresponding to a particular attribute will be shown by changing (increasing or decreasing) the size of the pie-chart.

For each dot of any time frame

Calculate Relative number using R = $\frac{1}{D}$ where R is the new relative number and D is the value of selected attribute.

Save the value of R in a relative list

Find minimum relative number from the list. R_{min} = min (list)

For each dot of created list

Get R from the relative list and find the new dot size using

Dot_{Size} = R/R_{min} , where Dot_{Size} is the new size for a dot

Get total size of a dot using S = S +Dot_{Size}, where S is the new combined size of a merged dot.

The algorithm to find the new size of a dot

The formula for calculating the diameter of a dot is replaced with $D = S^{.04}$ formula, where s is the new combined size of a merged dot. Now the diameter of a dot is calculated with the newdotsize value. The angle for each dot is calculated by this formula:

$$angle = 360 * \frac{Dot_{Size}}{S}$$

Table 16 shows an example with the calculation for finding the relative number of each dot corresponding to the chosen attribute. Suppose, at random x and y coordinate there are five events and each event has a population attribute. The end-user chooses population corresponding to which relatively will be shown on the visualization panel.

Table 16: Calculate relative number for each event of a dot

Trace ID	**Event name**	**Office location**	**Population**	**Relative number**
1	Payment	Eindhoven	20	.05
2	Payment	Amsterdam	200	.005
3	Start reclamation	Eindhoven	20	.05
4	Payment	Eindhoven	20	.05
5	Start reclamation	Breda	10	.1

Form above table R_{min} =.005, which is of Trace ID 2 This value is chosen to find out the new dot size for each dot. Table 17 shows the calculation done to find the relative dot size for each dot. After summing the newdotsize for each dot, it is 41 whereas the original diameter of the dot is 5.

Table 17: Relative calculation to find the size of a dot

Trace ID	Relative number	Dot_{size}	newdotsize
1	.05	.05/.005	10
2	.005	.005/.005	1
3	.05	.05/.005	10
4	.05	.05/.005	10
5	.1	.1/.005	20

Figure 38, shows a screenshot of Cartesian map where relativity between the dots is applied. All dots which has two events are highlighted with red box. As we can see, there is the difference in the size of the dots. One dot which is at the interaction point of Rotterdam and 36-45 jaar has a bigger dot size as compared to the rest of the two or three events dots.

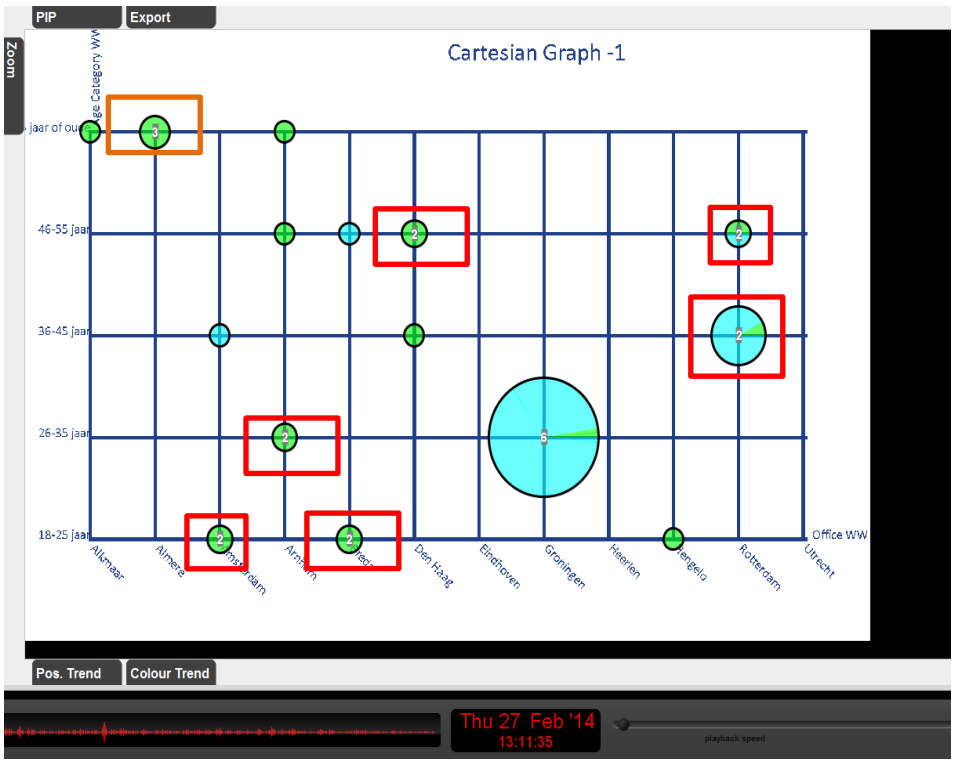

Figure 38: Screenshot to show relativity between dots

8.3 Show details of a valid dot

A visualization panel has projected valid dots at several x and y coordinates. When an end-user wants to see the details of the selected pie-chart. The

framework, identifies x and y coordinates and creates a list of dots that are present at those axis. It displays a panel, it includes a table, contains a list of events that are available at selected axis and chart which can be of two type's representation: a bar chart and a pie-chart. The end-user can choose any type of chart but by default the framework shows a pie-chart. The colour scheme of a chart is based on three type of colour scheme that is supported by the framework. The colour schemes are as follows:

1. **State schema**: An event is coloured based on the current State of an event.
2. **Age schema**: An event is coloured based on the age of an event which means the moment that event occurred in the movie and the current time of the movie. If the event has just started then the colour of the dot is projected white and as longer it remain on the visualization panel, it starts progressing toward black colour.
3. **Attribute value schema:** An event is coloured based on the end-user selected attribute and a value of an event.

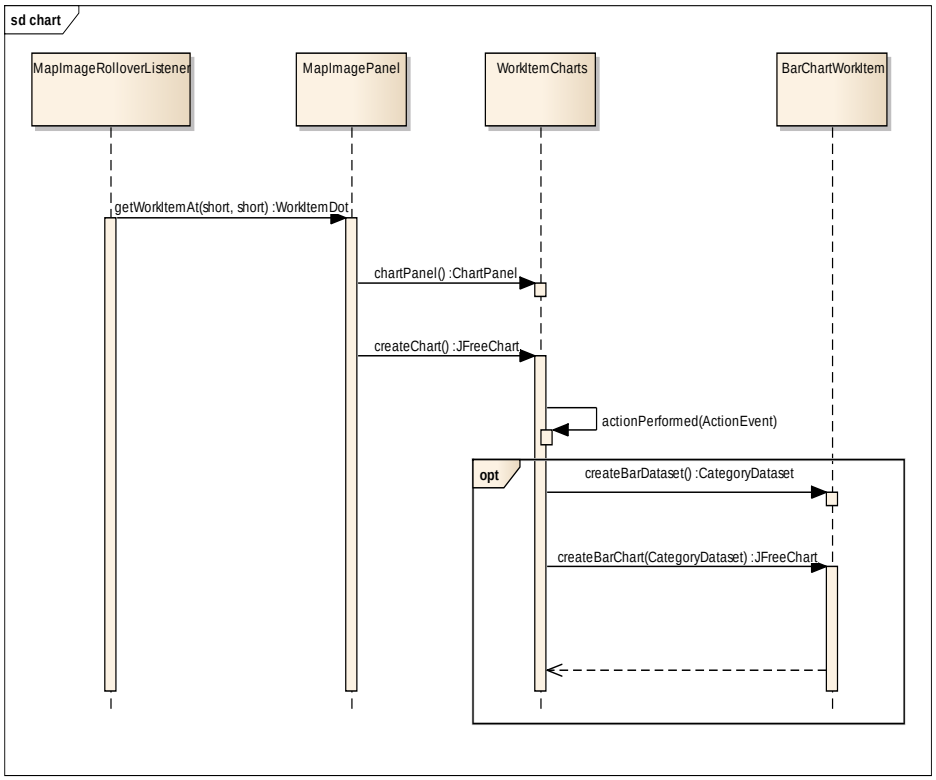

Figure 39: Sequence diagram to show details of a valid dot

This feature satisfies FR 4.3 functional requirement. The sequence diagram shown in Figure 39 depicts the sequence of steps take place to show the details of a pie-chart. The *MapImageRolloverListner* class finds the valid dots

that are project at (*x*, *y*) coordinate of a map. It finds the details and perform an action to create a chart based on the color scheme. The *createBarChart* method will return a *WorkItemChart* class, so that it can combine table and chart and display it on a pop-up panel. The diagram shows the steps take place for creating a panel a bar chart. Similar steps are followed to create a pie-chart for a panel.

A screen shot of a detail panel is shown in Figure 40. Here the *office location* attribute is chosen, which means the attribute value schema is applied. The colors for each office location automatically selected by the framework.

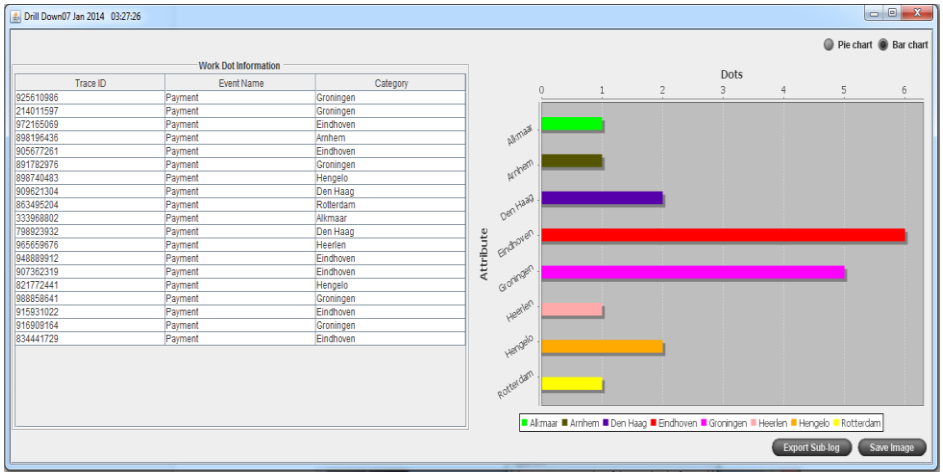

Figure 40: Screenshot of showing details of a valid dot

8.4 Show configuration panel

The framework displays a list of maps that are available on an input map file. The user can select for which maps are interesting to process for generating a process movie. This allows user to have flexibility to choosing which process movie need to be generated. A screen shot of a configuration panel is shown in Figure 41. The panel shows that, a map has three maps. The end-user can choose maps of their interest for producing a process movie. This feature satisfies FR 4.4 functional requirement.

Figure 41: Screenshot of a map configuration panel

8.5 Export sub-log

As mentioned earlier, a visualization panel projects valid dots at several x and y coordinates. The user can publish a sub-log for the selected coordinates on the ProM framework. The framework creates a sub event log which includes the traces, which are present on those coordinates. The sub-log file is created based on the main input event log for the LogOnMapReplay plug-in. Figure 42 shows a panel which has *Export Sub-Log* button with which, user can publish a sub-log on the ProM framework. This feature satisfies FR 4.1 functional requirement.

The sequence diagram shown in Figure 42 depicts the sequence of steps take place for exporting a sub-log of the selected the pie-chart from the visualization panel. The *MapImageRolloverListner* class is finds the valid dots at (*x, y*) coordinate of a map. The *WorkItemDot* class created a list which has all trace id available on a merged dot. An *ExportSubLog* class publish the all the events corresponding to those traces. The *publish* method filter all the traces and their events from the original event log.

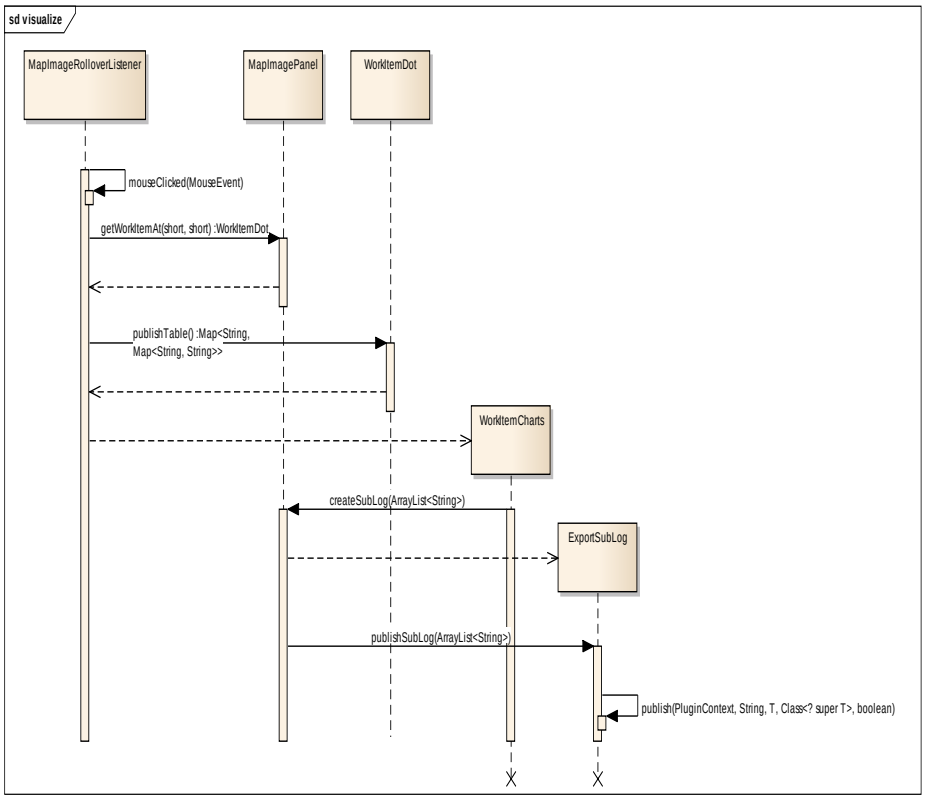

Figure 42: Sequence diagram for export sub-log

8.6 Merge map file plug-in

A new plug-in is created to support merging of two or more maps files. The plug-in supports merging of maximum five map file into one map file. After discussion, we decided the maximum limit for merging the map file. Although we believe it is easy to increase the maximum limit. Figure 43 depicts the logical view the design of the plug-in. The plug-in required minimum two map files. It supports merging of both old and new map file schema that are shown in Appendix in A and B. This plug-in satisfies FR 1.4 functional requirement.

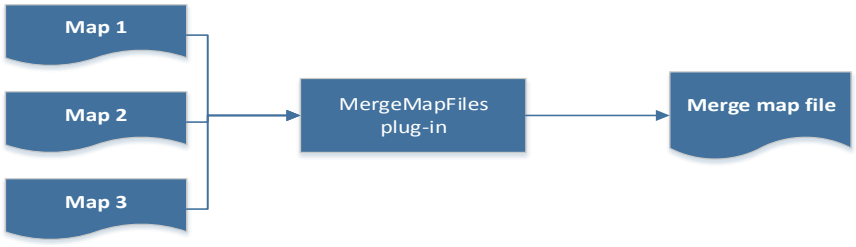

Figure 43: Logical view of Merge map file plug-in

9. Conclusions

This chapter summarizes the limitations of the existing implementation and the main results achieved during this project. It also lists the future work for LogOnMapReplay and AMG plug-ins.

9.1 Results

This project was initiated to enhance the functionalities and overcome the limitations of the existing version of the LogOnMapReplay plug-in. After discussion with the stakeholders, the main limitations were as follows

- Main input file called Map file is manually created. Manual creation requires technical guidance and knowhow.
- Improve the scalability which includes reduction in computation time and the memory usages of LogOnMapReplay plug-in on a laptop equipped with an Intel Core i7 Processor at 2.20GHz.
- Basic functionalities such as filtering data, comparative analysis were not available in the plug-in.

To overcome the limitations of the existing implementation, we developed a new plug-in and enhanced the features of the existing ones. Following describes the results of the two in brief:

1. The Automatic Map Generator (AMG) plug-in was developed to reduce the effort and time for creating different maps. In the AMG plug-in, we automated the process of creating a Cartesian and a process model map. This plug-in increases the usability of the LogOnMapReplay plug-in. The user does not have to go through the manual and monotonous process of creating a map file.
2. The MapDB database is introduced to store the processed data. This Key-value database adds no external dependency expect import one JAR file in the project. Before optimizations, the plug-in supported approximately two hundred thousand events and would process them in 6-7 hours on a laptop equipped with an Intel Core i7 Processor at 2.20GHz. In the same machine if the plug-in is processing more than three hundred thousand events, it goes out-of-memory. By introducing MapDB, more than one million event log can be processed in less than a day.
3. Various features such as filtering of events and attributes and showing drift are introduced to increase the usability of the tool. The evaluations (discussed in the previous chapter) show that the tool fulfils the requirements and meet the expectations of the stakeholders.

9.2 Future work

After the evaluation of the plug-ins, various observations have been pointed out which deserve attention for the future of these plug-ins.The following functionalities can be considered for the future work of the AMG plug-in.

- Automate the process of creating geographical, timeline, and organization maps.
- Create GUI in ProM framework for editing the map file.
- Create GUI in ProM framework for deleting maps from the map file.

- Improve the readability of Cartesian map by adding following functionalities.
    - Provide an option for selecting attribute values for a respective attribute in a Cartesian map. The plug-in draws line only for selected attribute values.
    - Define the position of a dot at the centre of each box of the attribute value instead of at the intersection point of the Cartesian Literal map.

The following functionalities can we considered for the future work of LogOnMapReplay plug-in.

- The plug-in should allow the user to aggregate several timeframes into a day, week, or month levels. This is to be able to group information based on time to perform a top-down analysis rather than bottom-up.
- The plug-in currently displays only the count of active activities in the centre of the dots. It is desirable that based on a numeric attribute, selected by the user, sum, mean etc. of the attribute-values is displayed instead. For example to show the total amount of the reclamations, instead of the number of reclamations.
- Make it possible to use an event logs with only compete life: cycle transition status. For example by adding an offset time for all events, and show events for the duration of this offset time.

10.Project Management

In this chapter, we discuss the management aspects of the project. It includes the project planning, communication, user acceptance testing, and work-break down structure.

10.1 Management process

We followed an agile approach with regular Scrum during this project. It is an iterative software development technique whose goal is to develop software through repeated iterations. The reason for choosing this approach was, we needed constant collaboration and feedback from UWV process analyst for following activities

- Prepare an event log.
- Evaluate and receive feedback on new or modifies features of LogOnMapReplay, AMG, and MergeMapFile plug-ins.

The project was divided into several phases. At the end of each phase, a demo meeting was conducted to show the new or improved features of each plug-in. Our tasks during the project can be roughly divided in five sub-tasks.

1. **Domain:** Study the process mining State-of-art domain, get familiar with UWV business processes, the stakeholder requirements.
2. **Experiments:** Conduct various experiments on existing plug-in.
3. **AMG plug-in**: Create and develop the design, and implementation of AMG plug-in.
4. **LogOnMapReplay plug-in:** Create and develop the design, and implementation of new features for this plug-in.
5. **Documentation:** Document, evaluation, and demo the plug-ins.

10.1.1. Planning and Scheduling

The planning meetings helped in dividing tasks into segments where each segment had measureable deliveries. Each segment contained a number of deliverables. The types of deliverables are: plug-ins versions, evaluation of plug-ins, and documentation. The detailed planning is documented in the work breakdown structure section 11.2. At the end of every month the plan for the coming deliverables was revised and updated.

10.1.2. Communicating with supervisors

In the beginning of the project, twice per week meetings with company supervisor and other company stakeholders were held to discuss the requirements and progress of the project. Once the requirements were clear, this was replaced with once a week or fortnight depending on the need for a meeting. The meetings with the university supervisor took place every week in order to share and discuss the design and implementation of the requirements of the project.

In order to align the expectation with two main stakeholders TU/e and UWV, a Project Steering Group (PSG) meeting was held. The PSG was composed of the university supervisor, the company supervisor, the project manager and I. During a PSG meeting the following tasks were performed:

- Update on the current status of the project.
- Discuss the further project planning.
- Discuss over the tasks for next month.
- Feedback of the progress of the project.

10.1.3. User Acceptance testing

After completion of each phase, plug-ins demo was conducted for all stakeholders in order to receive the feedback on developed features. The User Acceptance Testing (UAT) was conducted to determine if the requirements are met the expectation or not. This testing is done by the business/ process analyst of UWV. The process analyst wanted to check the ease-of-use the new features.

10.2 Work-Breakdown Structure

Figure 54 shows the timeline of the project with the major milestones. This project timeline was created in the beginning of this project. Pre-defining the milestones was helpful thorough out the project. However, in order to align the planning with the availabilities of the stakeholders, some activities changed over time. The milestones of the project are described in chronological order.

In the first phase, we got an overview of process mining techniques, and relevant information about UWV business processes.

In the second phase, we defined the plug-in requirements of the project based on the demo and discussions with stakeholders. We conducted a demo of existing LogOnMapReplay plug-in with defined use cases scenarios, discussed in Chapter 4. These demos helped us in understanding the existing features and functionalities of the plug-in and finding out the requirements of the project.

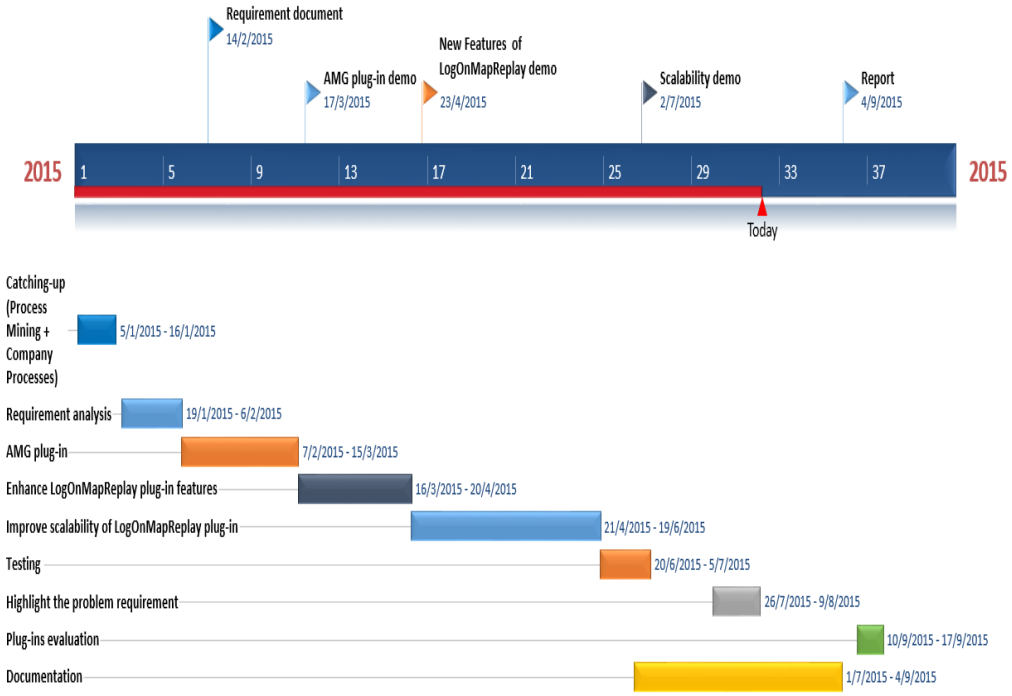

Figure 44: The project timeline

The design and implementation phase of all plug-ins required through understanding ProM framework. The ProM framework has Application Program Interface (API) for importing and using event logs in the plug-in. The developer has to follows certain guidelines for creating a new plug-in in ProM framework. During the design and implementation phase, it was important to understand the existing design, architecture, and implementation of LogOnMapReplay plug-in because we had to improve the processing time and adding/modifying the features in this plug-in.

The last phase included writing of the final report, evaluation of plug-ins, share and presentations for stakeholders with in UWV for sharing insights about the reclamation event log.

Appendix A New map file schema

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="specification">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:string" name="task_variables"/>
        <xs:element name="maps">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="map">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element type="xs:string" name="url"/>
                    <xs:element type="xs:string" name="resources"/>
                    <xs:element name="axis">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element type="xs:string" name="xAxis"/>
                          <xs:element type="xs:string" name="yAxis"/>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                    <xs:element name="queries">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="query" maxOccurs="unbounded" minOccurs="0">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element name="attributes">
                                  <xs:complexType>
                                    <xs:sequence>
                                      <xs:element name="attribute">
                                        <xs:complexType>
                                          <xs:sequence>
                                            <xs:element type="xs:string" name="x"/>
                                            <xs:element type="xs:string" name="y"/>
                                            <xs:element type="xs:string" name="xquery"/>
                                          </xs:sequence>
                                        </xs:complexType>
                                      </xs:element>
                                    </xs:sequence>
                                  </xs:complexType>
                                </xs:element>
                              </xs:sequence>
                              <xs:attribute type="xs:string" name="task" use="optional"/>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                  <xs:attribute type="xs:string" name="name"/>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Figure 45: New map file schema

Appendix B Old map file schema

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="specification">
    <xs:complexType>
      <xs:sequence>
        <xs:element type="xs:string" name="task_variables"/>
        <xs:element name="maps">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="map">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element type="xs:string" name="url"/>
                    <xs:element name="queries">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="query" maxOccurs="unbounded" minOccurs="0">
                            <xs:complexType>
                              <xs:sequence>
                                <xs:element type="xs:string" name="xquery"/>
                              </xs:sequence>
                              <xs:attribute type="xs:string" name="task" use="optional"/>
                            </xs:complexType>
                          </xs:element>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                  <xs:attribute type="xs:string" name="name"/>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Figure 46: Old map file schema

Glossary

Term	Description
PDEng	Professional Doctorate in Engineering
TU/e	Eindhoven University of Technology
XES	Extensible Event Stream
AMG	Automatic Map Generator
PAIS	Process-Aware Information Systems
ProM framework	Process mining framework
UWV	Uitvoeringsinstituut WerknemersVerzekeringen
XML	Extensible Markup Language
IEEE	Institute of Electrical and Electronics Engineers
GB	Gigabyte
MB	Megabyte
GHz	Gigahertz
FR	Functional Requirement
NFR	Non-Functional Requirement
GUI	Graphical User Interface
API	Application Program Interface
RAM	Random-access memory
UAT	User Acceptance Testing
PSG	Project Steering Group
CPU	Central Processing Unit

Bibliography

Christian W. Günther, E. V. (2014). *XES Standard Definition.*

kotek, J. (2015). *MapDB.* Retrieved from http://www.mapdb.org/

Massimiliano de Leoni, S. S. (2014). *Turning event logs into process movies: animating what has really happened.*

Petri net. (2015). Retrieved from https://en.wikipedia.org/wiki/Petri_net

Thomas, J. C. (2005). *Illuminating the Path.* Retrieved from http://www.visual-analytics.eu/faq/

van der Aalst, W. (2011). *Discovery, Conformance and Enhancement of Business Processes.*

W3C. (2014). *XQuery*. Retrieved from https://en.wikipedia.org/wiki/XQuery

XES. (n.d.). Retrieved from XES: http://www.xes-standard.org/

YourKit. (n.d.). Retrieved from https://www.yourkit.com/.

About the Authors

Neha Gupta received her Bachelor of Computer Science (2006) from MDU University, India and Master of Computer Science degree (2009) from Amity University, India. She carried out her final project in HCL Ltd. (Indian IT company) on designing database and Application development of "Retail management" project. After her Master's she worked for IT Services company, Steria India Ltd., for 2.9 years (2010-2013) as ETL Developer using Abinitio Tool in Banking domain (for Barclays Bank, UK).

In 2013 she joined the PDEng program in Software Technology at the Eindhoven University of Technology. Her final project as a part of the PDEng program is entitled "Interactive visualization of business processes": a monitoring tool for dynamically visualizing the executed business process. The project was carried out at UWV.

3TU.School for Technological Design, Stan Ackermans Institute offers two-year postgraduate technological designer programmes. This institute is a joint initiative of the three technological universities of the Netherlands: Delft University of Technology, Eindhoven University of Technology and University of Twente. For more information please visit: www.3tu.nl/sai.