BETTER CONFORMANCE CHECKING FROM PROCESS MODEL GENERATED BY PROCESS TREE

A THESIS SUBMITTED TO
THE FACULTY OF ARCHITECTURE AND ENGINEERING
OF
EPOKA UNIVERSITY

BY

HEGI GJOKA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

APRIL, 2021

Approval sheet of the Thesis

This is to certify that we have read this thesis entitled **"Better conformance checking from process model generated by process tree"** and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Assoc. Prof. Dr. Ali Osman Topal
Head of Department
Date: February, 20, 2021

Examining Committee Members:

Assoc. Prof. Dr. Name Surname (Computer Engineering) _____

Assist. Prof. Dr. Name Surname (Computer Engineering) _____

Dr. Name Surname (Computer Engineering) _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name Surname: Hegi Gjoka

Signature: _____

ABSTRACT

BETTER CONFORMANCE CHECKING FROM PROCESS MODEL GENERATED BY PROCESS TREE

Hegi Gjoka
M.Sc., Department of Computer Engineering
Supervisor: Dr. Igli Hakrama

Process mining field is massively used nowadays by medium and large size enterprises, in order to have a better understanding of the processes that occurs and it helps them in the decision-making aspect by presenting the workflow of the organization as a whole and evidencing bottlenecks and unknown procedures that before were unknown or irrelevant. This study aims to check conformance and to analyze bottleneck of these processes by means of process mining techniques. To check conformance and to do bottleneck analyzes, an Enterprise resource planning system (ERP) was taken into consideration. A custom technique of data gathering (system event logs) was used for this information system (IS), based on an aspect oriented programming (AOP) technique. Also there was the need for robotic process automation (RPA) technique application, due to COVID-19 pandemic, which made data gathering difficult due to the enterprise lockdown. The result showed a better conformance checking after usage of the inductive miner algorithm and Petri net representation of the process model. The result showed a better conformance checking by means of fitness and precision. As for bottleneck analyze, the only conclusion was due to operator delay during processes.

Keywords: *Process Mining, Process Models, Petri nets, Process tree, Inductive miner, Software logs, ERP, Bottleneck analysis*

ABSTRAKT

KONTROLL ME I MIRE KONFORMANCE SE PROCESS MODELS QE JANE GJENERUAR NGA PROCESS TREE

Hegi Gjoka

Master Shkencor, Departamenti i Inxhinierisë Kompjuterike

Udhëheqësi: Dr. Igli Hakrama

Fusha e process mining përdoret masivisht në ditët e sotme nga ndërmarrjet e përmasave të mesme dhe të mëdha, në mënyrë që të kenë një kuptim më të mirë të proceseve që ndodhin dhe i ndihmon ata në aspektin e vendimmarrjes duke paraqitur rrjedhën e punës së organizatës si një e tërë dhe duke evidentuar pengesat dhe procedurat e panjohura që më parë ishin të panjohura ose të parëndësishme. Ky studim synon të kontrollojë konformitetin dhe të analizojë vonesta e këtyre proceseve me anë të teknikave të process mining. Për të kontrolluar konformitetin dhe për të bërë analiza të vonesave, u mor në konsideratë një sistem i planifikimit të burimeve të ndërmarrjes (ERP). Një teknikë e personalizuar e mbledhjes së të dhënave (loget e sistemit) u përdor për këtë sistem informacioni (IS), bazuar në një teknikë të programimit të orientuar drejt aspekteve (AOP). Gjithashtu ishte nevoja e zbatimit të teknikës së automatizimit te proceseve me robote (RPA), për shkak të pandemisë COVID-19, e cila e bëri të vështirë mbledhjen e të dhënave për shkak të bllokimit të ndërmarrjes. Rezultati tregoi një kontroll më të mirë të konformitetit pas përdorimit të algoritmit inductive miner dhe paraqitjes me anë të Petri nets të modelit të procesit. Rezultati tregoi një kontroll më të mirë të konformitetit me anë të përshtatshmërisë dhe saktësisë. Sa i përket analizës së vonesave, përfundimi i vetëm ishte për shkak të vonesës së operatorit gjatë proceseve.

Fjalët kyçe: *Process Mining, Process Models, Petri nets, Process tree, Inductive miner, Software logs, ERP, Bottleneck analysis*

ACKNOWLEDGEMENTS

I would like to express my special thanks to my supervisor Prof. Dr. Igli Hakrama for his continuous guidance, encouragement, motivation and support during all the stages of my thesis. I sincerely appreciate the time and effort he has spent to improve my experience during the last year of my graduate studies. Also a special thanks to “Our Lady of Good Counsel Foundation” for letting me use their ERP system, that I also took part in the development, that it is used as a case study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Process mining is a field that is first created and developed by Professor Will van der Aalst in the Einthoven University of Technology in 1999 and the first book edition for this field was released after 14 years of research and development, in 2013 and the second edition in 2016. This field is present as a sub-field in two of the most dominant sciences that are present today, like Data Science and Process Science.

Data Science is an inter-disciplinary field that make use of algorithms and processes to extract knowledge and insights from structured and unstructured data [1]. Some of it sub fields are Data mining, Big data, Statistics, and among of them is Process mining.

Process Science in an inter-disciplinary field that it main purpose is to turn data into value. Value is then provided in different forms as automated decision making, predictions, data visualization, etc. [1] Also this science is decomposed in it sub fields that are Stochastics, Optimization, Process automatization, Business process management (BPMN) and among of them is also presented Process Mining. Since this field is presented in both sciences is also referred as the “bridge” between Data science and Process science, best described by the Figure 1 [1].

Process mining find usage in big and medium size enterprises/businesses software’s and departments, like Customer Relation Management (CRM), Supply Chain Management (SCM), and Enterprise Resource Planning (ERP). Process mining consists in representing the business processes by a process model. This process model is created by algorithms like (alpha algorithm, rapid miner, heuristic miner, petri nets, etc.) and tools like ProM (developed by Professor Aalst and its colleagues at Einthoven University) and Disco. This tools uses algorithms and event logs generated business processes (softwares). The flow of process model discovery is best represented by the Figure 2 [1].

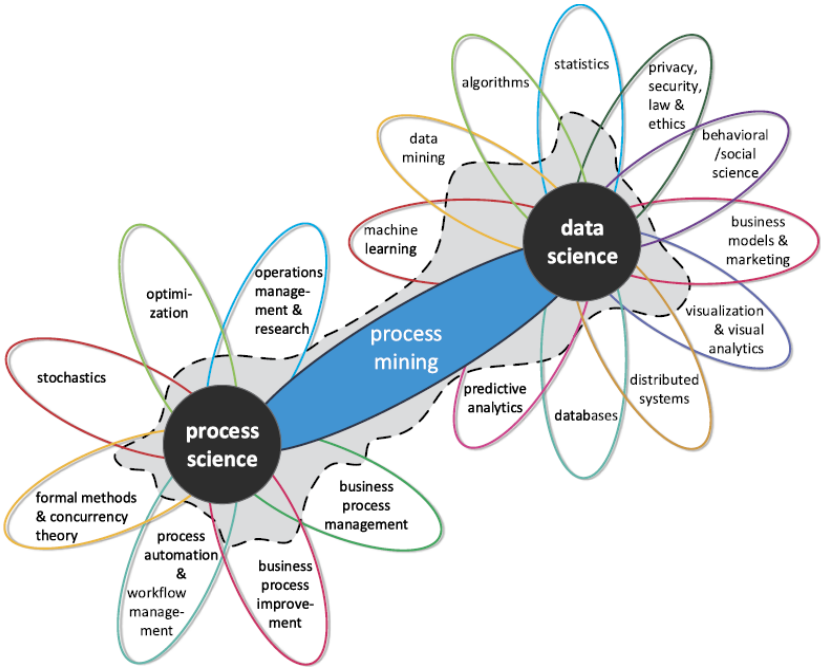

Figure 1: Process mining as the bridge between Data science and Process science [1]

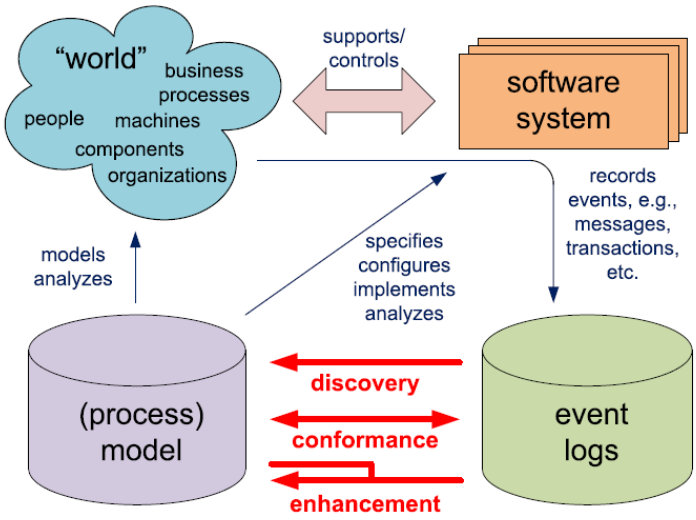

Figure 2: Process model discovery flow [1]

1.1 Thesis Objective

The primary objective in every study is to investigate the literature, in order to find related problems or lack of application in practice of your field of study. In this

thesis, will be presented an application of process mining in one business ERP system. By applying this technique to this system we will be able to discover a process model in order to compare it with the real one and to analyses bottlenecks and discover new processes that aren't present in the real planned process design. This is pretended to be achieved by means of the Disco and Prom tool and the logs will be generated in an asynchronous way by means of request interceptors since the ERP that is going to be taken in consideration is a web application.

1.2 Organization of the thesis

This thesis is divided in 7 chapters including this first one. The organization is done as follows:

The second chapter includes the literature that I have taken in consideration in order to state the objectives for this thesis and to achieve its purpose. It is going to be treated the main relevant works that are going to be crucial in the understanding the problems and the subject in order to applied it on a real software.

The third chapter consist on the methodological approach that is presented in this thesis in order to achieve the final purpose. Since this is the application of process mining techniques in a real ERP system, the method used in this thesis is an empirical methodology in order to analyze quantitative data gathered from a case study.

The fourth chapter consist in the description in detail of the ERP system that is taken into consideration that in our case is Odonto. This system has different usage in the company, since it serves as an inventory warehouse where all the transaction, purchases and inventory movements are registered in this system. Also it serves as virtual register for patients and their medical cartel. The reception operators use it to book dates for patients.

The fifth chapter describes all the work that it is done for gathering data in order to turn them into event logs. After wards this event logs will be screened and preprocessed in order to be then consumed by the process mining tools available. Also in this chapter is going to be treated an uncompleted work.

The sixth chapter will represent the part of the analysis made in this thesis. Some of them are bottleneck analysis of the software, process model discovery of unknown paths and conformance checking using different algorithms.

The seventh chapter will present the conclusion of the application of the methodology used in this thesis, a brief summary of the analysis section. In this session is mention also a future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The literature chapter is the most valuable chapter of this thesis as at this section will be defined the title and the reason of it. This chapter will briefly describe the background I have in terms of existing and gained knowledge during the research of papers and scientific articles to define the main motivation of the study and the work that these materials have left unsolved. This chapter will be divided into four parts which will be, the background, the works that are solved in the materials that have been considered, and the works that these materials have left unsolved to reach the definition of this thesis by defining the last section of this chapter that is going to be the research gap.

3.2 Background

In this section are going to be treated the four main elements of this thesis, which are process mining, alpha algorithm, petri nets, and inductive miner. These materials are taken from papers and scientific articles, as well as from the second volume of Professor Aalst's book "Process Mining".

Process Mining, is the field of analyzing data to recreate (discover) the processes or the workflow of that the data may contain. This field as is presented in the book of Process mining [1] as the gap between two field of science, which are Data science and Process science best described by the Figure 1 in the introduction chapter. In both of them process mining appears, but in different approaches. In the field of Data Science, its subfield process mining has a process agnostic approach whereas in the field of Process Science, its subfield process mining has a more workflow approach

omitting its other data that it might contain [1]. Process mining has firstly started in 1999 in Eindhoven University of Technology and its first mining algorithm that has used was alpha algorithm that was responsible of discovering concurrency in event logs which was and is the most important source of the process of discovering process models. When the data in the entire environment of the internet increases exponentially, process mining reached the boom of its development and usage in academic and commercial systems e.g. CRMs, ERPs, SCMs, etc. Process mining open-source tool ProM it is used in many companies now day's in order those to analyze its processes, behaviors of workflow and customers, bottlenecks, and hidden processes. This tool helped them to optimized their enterprise and be more productive and more low-cost in resources. Process mining works like in the Figure 2 in the first chapter, were the software that it is used in the company by the employees generated logs for each event of the company environment that the user inputs to it. This logs are stored into a database of event logs and from this the logs are formatted into XES, XML, MXML, CSV, etc. files in order to run algorithms upon this files and discover, conform, and enhance the process model derived by the event logs.

Alpha algorithm, is the first algorithm that is responsible to discover concurrency in the event logs that are faded in it. It works by recreating a footprint table of the event logs that have special relations between processes e.g. L = [<a,b,c,d>^{3}, <a,c,b,d>^{2}, <a,e,d>], where the sequence a,b,c,d is shown 3 times in the event logs and so the others as in the Figure 3.

	a	b	c	d	e
a	$\#_{L_1}$	$\rightarrow_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$
b	$\leftarrow_{L_1}$	$\#_{L_1}$	$\|\|_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
c	$\leftarrow_{L_1}$	$\|\|_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
d	$\#_{L_1}$	$\leftarrow_{L_1}$	$\leftarrow_{L_1}$	$\#_{L_1}$	$\leftarrow_{L_1}$
e	$\leftarrow_{L_1}$	$\#_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$

Figure 3: Alpha algorithm footprint table [1]

Petri Nets, are an UML representation of any kind of processes, represented by a group of placeholders, arcs, and tokens that moves into these placeholders. It consist of places, transitions, and arcs. Arcs connects places and transitions together to form the petri net. The phase from which the place runs to a transition is called input place transition and the phase from which the transition runs to a place is called output place of the transition. In petri nets we also see the usage of tokens which is the state that follow the path among places and transitions from the start of the petri net to the end. If a transition placeholder has one input arc it require a token consumption from the previous place to create one token of its own, in the case it has two or more input arcs it needs that the previous places to consume their tokens in order to create its own. In the other hand, if the transition has one output or more output arcs, it is necessary to consume its token in order that the process to flow. This workflow representation technique is best represented by an example diagram that is shown in Figure 4.

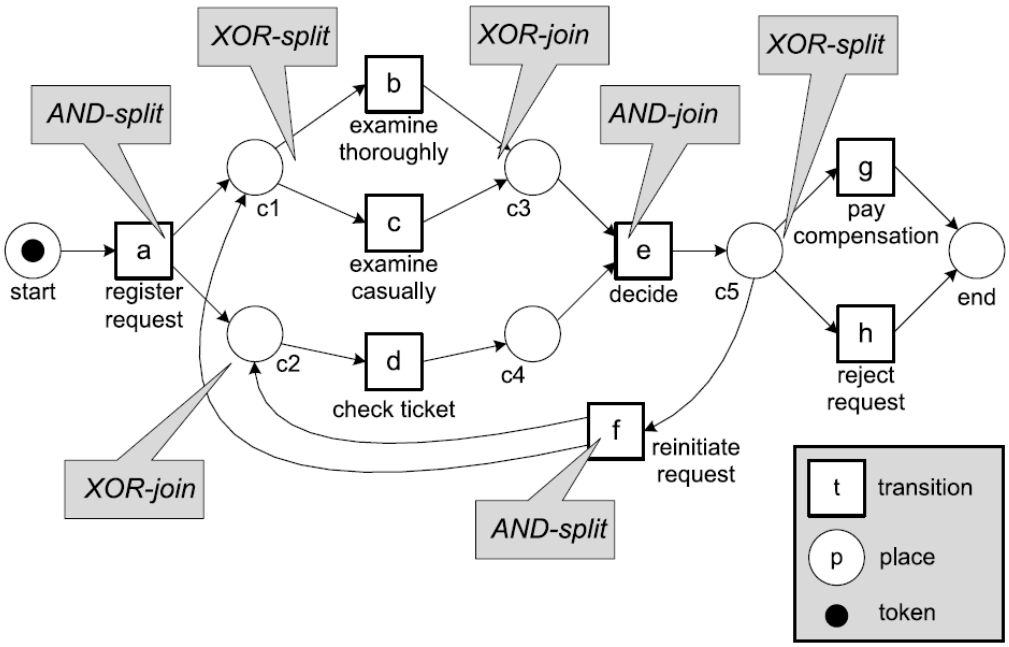

Figure 4: Petri net diagram [1]

Inductive Miner, is the technique based on alpha algorithm process model. In additional to the alpha algorithm that creates the model based on the footprint table, the inductive miner technique for the first time introduces the tree representation of the model by divide into branches the processes based on their relation shown in the

footprint table. Afterward, the process model can be represented in any view as Petri Net, BPMN, etc. derived from the tree, Figure 5.

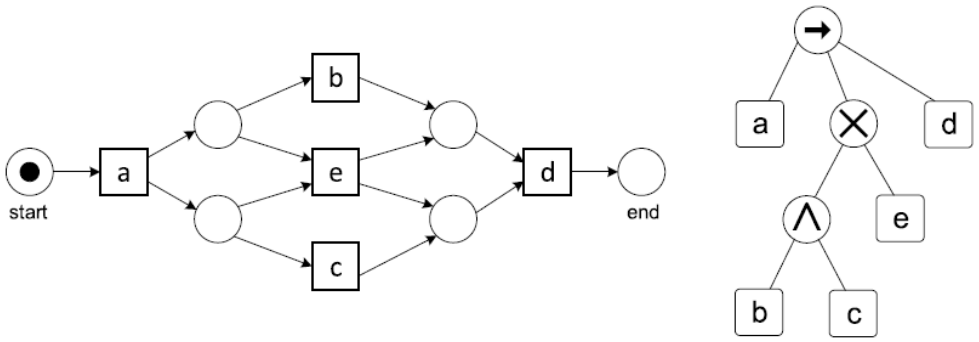

Figure 5: Petri net and the tree representation by Inductive miner [1]

Robotic process automation, is like business process automation softwares and techniques, which is main difference and advantage is the metaphorical word robot. This technique or software is used to automate the manual processes it performs in businesses. This technique consists of instructions given by the user to perform a certain task. RPA now have a variety of tools that are equipped with various commands which can be translated into instructions. RPAs are used in both the academic and the industrial part. In one of the papers considered for this thesis is used RPA's with Form-to-Rule F2R which creates a continuous cycle of changing instructions which are improved by the logs of the RPA system [2]. In the Figure X below is described this technique in use.

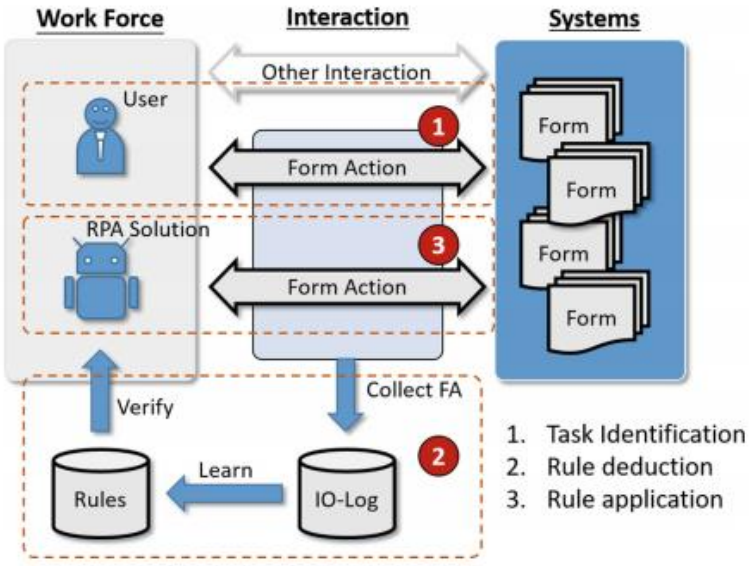

Figure 6: F2R approach in RPA [2]

Aspect-Oriented Programming (AOP) is a programming paradigm that aim to increase modularity by allowing the separation of cross-cutting concerns [3]. AOP add additional behavior to classes, methods, and functions without touching (modifying) their code, by pointcut specification e.g. log method called with a numeric parameter and a currency parameter, as "Transaction made $1000". This king of programming doesn't affect the business logic of the code and it can be added at the very end or in an existing code without mutating it. Cases that I have observed with this kind of programming is in java spring framework were the instance of an object is created by a factory code that allow the usage of aspect functions. Aspect functions are mainly logging function that uses pointcut and cross-cutting methodology without affecting the code and its business logic. AOP is best described by the Figure 7 where are shown 3 Objects A, B, and C with their on functions (business logic) and Aspect Configuration, Logging Aspect, and Transaction Aspect Functions.

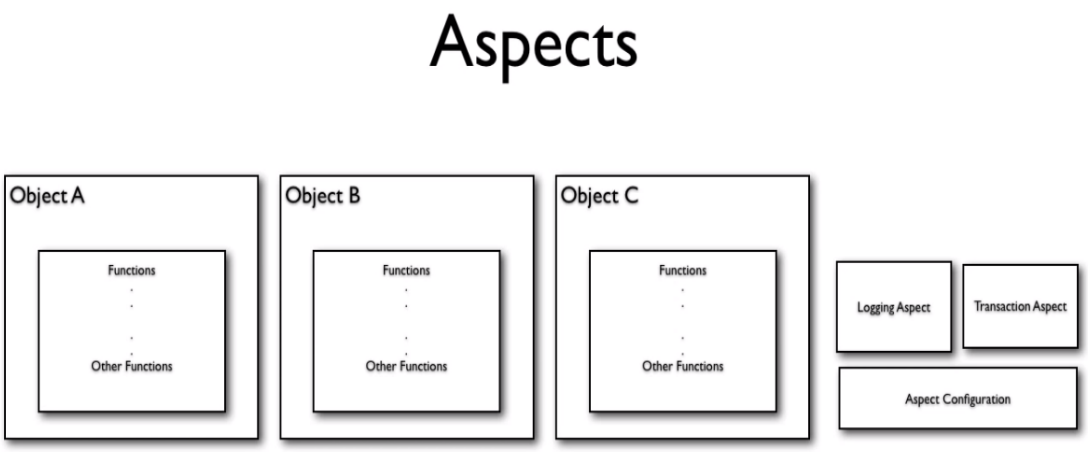

Figure 7: Classes & Aspects

In java spring framework code of aspect is like shown in the Figure 8. @Before is used to log aspect before the function or method is called, and there are many annotations and parameters catchers like @After, @AfterReturning, @AfterThrowing, @Around, @Pointcut, pointcut, JointPoints, returning, and throwing.

```
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.AfterThrowing;

@Aspect
public class AfterThrowingExample {

    @AfterThrowing("com.xyz.myapp.SystemArchitecture.dataAccessOperation()")
    public void doRecoveryActions() {
        // ...
    }

}
```

Figure 8: Aspect class

3.3 Summary of the work that has been done

In this section are shown all the problems that are resolved in the papers/journal/article that we found from different conferences and also in Research Gate. Metamodeling of reusable templates for process modelling in some platforms like BPM IE to improve company productivity [4]. Applying process mining into different software like Microsoft Power BI in order to use multidimensional data (event logs) [5]. Filtering logs without any trade of in the 4 forces that are applied in the process model, by filtering infrequent logs or paths [6] or divide big event logs into small subset with sampling strategies to gain time [7]. Dealing with flat Object Centric event logs [8] [9] and to use that by turning it into the right format (e.g. XES and CSV) in order to be recognized by the tools that are in this moment being. Fast conformance checking by abstracting event logs and turning them into tables (footprint) to reduce the time of evaluating the process model discovered [10]. Designed framework (tool) for process mining like ProM, Disco, Rapid Miner, and others.

3.4 Summary of unsolved problems and future work

Here, in this section are mentioned to you the problems that aren't solved yet and future works. Some of the problems are dealing with multi sources event logs that are generated by different software [11] where it is unable to tell if the log of the other software is the other part remaining of the previews log in the other software even if they are consecutive in the chronological time space. Dealing with a single case notation in the event logs with only one column "flattened event data". Extracting meaningful and valuable event logs from all kind of software in the right format in order to be processed by the existing process mining tools [8] and also from different sources (software) [12].

Integrating process modelling methodology with business process modelling methodology with the requirements for enterprise development in Web APIs [4]. Integrating process mining analysis, conformance checking, and bottleneck analysis into Power BI [5]. Handling problematic event logs, e.g. event logs with missing logs,

isolating behavior event logs [6], finding out the best subset when we are dealing with big event logs [7], filtering event logs that do not contain business activities e.g. logging into the software or changing password [12], detecting high level service with correlated event logs [9].

3.5 Research gap

In this section of this chapter we is the part that is define as missing or that is shown in the future work part in the other works that we have read in the literature review part. Some possible research gaps which are in the future work part and another one that I, with the advice of my supervisor of my master thesis, have formulated:

- Dealing with multi source event logs.
- Extracting meaningful and valuable event logs from software.
- Filtering out non business logs from event logs.
- Optimizing business workflow using process mining techniques.

CHAPTER 3

METHODOLOGY

In this chapter, will be described the main methodology approach of the thesis based on the most relevant methodology techniques and the methodology to approach the applied case that is going to be taken into consideration.

3.6 Methodology approach

The methodology that will be used in this thesis is a “Case Study Method”, because it will be taken into consideration an ERP system of a dental clinic. This ERP manage a big size clinic with 10 armchair and 12 dentists, 4 receptionists, and 3 accountants. This dental clinic is also the main internship provider for dental students. There are more than 100 students every year that finish their internship in order to take their diploma. The technique that is applied in this system for gathering data is going to be categorized as a “Quantitative Method”, because process mining in order to generate process model needs to be feed with event logs.

The first step was to choose a very big software, which in this thesis is chosen an ERP that is used to in managing the resources of the business. This system will be described in detail in the next chapter of this thesis, so you can skip the rest of the methodology description if you are interested only in the case that it is taken into account.

The second step of this work was to gather data in a quantitative method, because event logs are the main resource of process mining and process model discovery. In this part there will be a parallelism between the comprehended technique in the above chapter, which is AOP and the technique applied in this case study. AOP is mainly used in some frameworks like Java Spring, Groovy, PHP, Python, .Net, etc.

Since none of the frameworks used in this case study belong to the above list, a new way of implementing this technique will be needed. This new way will be combined with interceptors in the front part of the application which is written in Angular, the framework which has as core feature a wide range of concepts and among them are the interceptors. Interceptors as they serve to monitor or add metadata to the request, which is done by the front in terms of backend and response, which is done vice versa. This method is similar to the AOP technique as it does not interfere with the business logic of the application. For this step, is taken in account asynchronous event log generation from this interceptor, which is mentioned above for performance reason, because the system is very big and find 90% usage in all departments of the dental clinic. Also in this part will be mentioned in detail the use of RPA in the part of generating event logs, because in the period that this thesis has been completed, all humanity has passed the COVOD-19 pandemic, which brought many loss of life, closure of businesses , and economic downturn. Business closures led to this thesis event logs to be generated in an automated way by RPA, as the business in which this ERP operated was closed for a period of 4-5 months.

And last but not least is the third step, which will be the essence of this thesis because will be analyses of process mining techniques that result in a more conformant model of the process made in this ERP as whole. This will be presented in Chapter 6 of this thesis. Three techniques are considered in it which are Petri Net, BPMN, and Inductive miner. Petri Net and BPMN are two techniques used in process mining to generate process models from event logs generated by system usage. While Inductive miner performs the same process as in the two techniques above, but the only difference between this technique and the above-mentioned techniques is the generation of a process tree. This process tree can then be turned into a process model by combining it with event logs. This leads to the creation of a better process model, as the process tree is flexible and its connections can be interpreted in several ways depending on the event logs.

CHAPTER 4

Analyses of the ERP

In this chapter we will explain in details the ERP that is taken into consideration in this thesis. The ERP system is property of the “Dental Clinic Our Lady of Good Counsel” and is referred by the name of “Odonto” since in Italian the dentistry clinic is called “Clinica Odontoiatrica”.

A brief explanation of the ERP system. This system is built by the IT Office of the “Our Lady of Good Counsel Foundation” in 2016 and is still in development (in 2020 I also took part in the development). The frontend part is built in Angular 5 framework and uses PrimeNg and Ultima 5.2.0 template. The backend in the other hand is built in Java, and uses Rest Jax and Postgres Database.

The application that has been considered is divided into several menus and sub-menus. The first panel we encounter is the dashboard where all the stock items are listed. At the top of this panel we have cards which show us an overview of inventory status, services, patients, doctors, departments and products near the expiration date. Also at the top we notice 4 buttons which are the purchase button, material consumption button, movements’ button and inventory regulation button as shown at Figure 9 below.

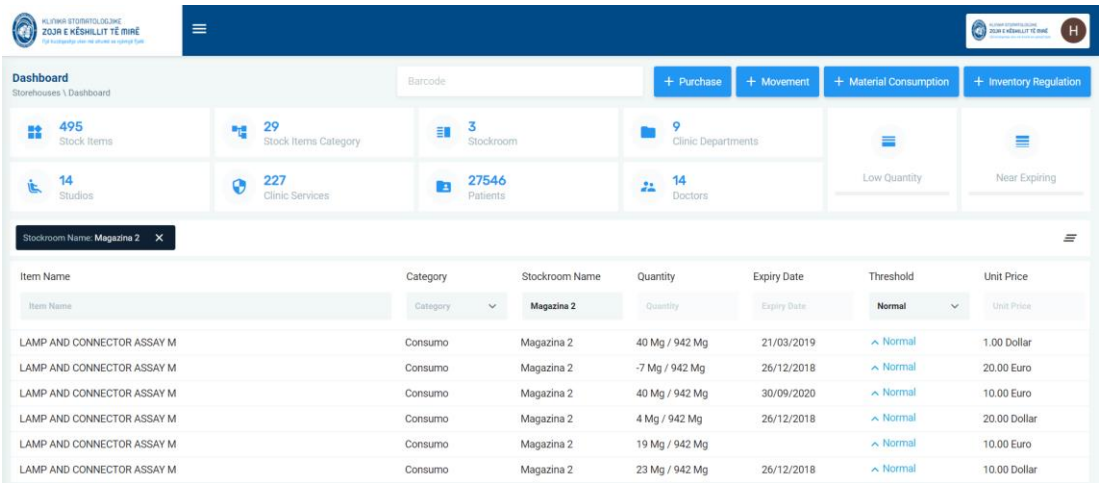

Figure 9: Odonto web app dashboard

Top menus are the warehouse, booking, clinic, and armchair section. In the warehouse menu we notice the respective sub-menus which are transactions, stock items, suppliers, and flow.

In the transactions section we can look at any transaction that we can, which can be purchase, material consumption, inventory adjustment, or transfer of stock items from one warehouse to another. We also look at the date of the transaction, the inventory that participated in this transaction along with all its settings such as expiration date, quantity purchased or consumed, unit of measurement and unit price. If we click on a row the application will open a side panel in which I will show every detail of the transaction. All transactions are in chronological order from the latest to the earliest. This panel also has the relevant filters. In the appendix section there is Figure 25.

In the warehouse menu we also find the stock item sub-menu. This panel contains all the products which are registered in the application. Also as in the previous panel we have the ability to filter the inventory. In this panel we will be able to see the name of the product, its category, stock status, minimum stock, final price and supplier. In the appendix section there is Figure 26.

Also in the warehouse menu is the sub-menu of the supplier which is composed of filters and supplier settings as shown at the Figure 27 at the appendix section. In this panel, if we click on the row, a side panel opens where there is the possibility to

modify the supplier which is selected or a new panel opens in which appears in chronological order all the tax invoices that have been made by this supplier.

In the booking menu is the calendar of the clinic where all the doctors who are staff of the clinic are displayed. In this panel the reception operator can book a calendar for a patient by clicking on the date on which the patient will appear at the clinic as shown at Figure 10 below.

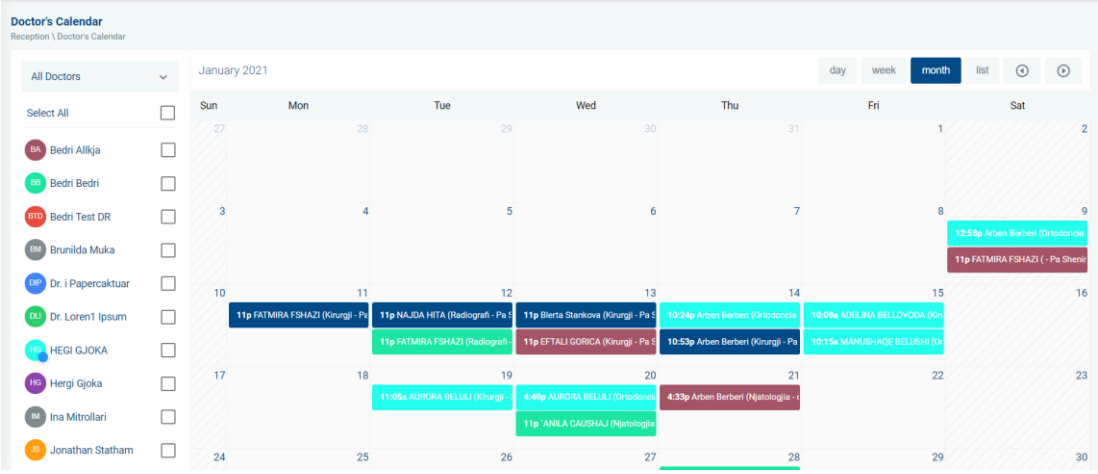

Figure 10: Odonto booking calendar

After clicking on the date in the calendar, a side panel opens where the booking data must be filled in, such as the clinic department, the doctor who will perform the medical examination, the patient, the booking schedule and a comment. Also in this side panel the operator can add the services that the patient requires such as teeth whitening as shown in the appendix section Figure 28.

The clinic menu contains sub-menus such as services, offers, patients and doctors. The panel of patients and doctors is almost identical as the relevant filters and data of each patient / doctor are displayed. If we click on a row we have the opportunity to modify the person, delete him, and look at his medical file in the case of the patient and his card with services in the case of the doctor respectively described by Figure 29 and Figure 30 at the appendix section. In this side panel the doctor also has the opportunity to see his booking calendar, the materials he has consumed, the services he has performed in chronological order, he can also access the armchair panel which will be explained below.

In the menu of the clinic is also located the sub-menu of services. The service panel has the relevant filters and service settings such as service name, service department, service cost, and recent performance of this service.

The newest menu added to this application is the armchair menu in which a panel opens which can be called otherwise as a standalone application as it is adapted to be used on a 10 inch tablet. This panel is divided into two sub-panels where one of them contains the list in chronological order of all bookings of the day and the other panel contains the details of the booking, which means that it contains the patient data, the booking data and the list of services required by the patient as shown in the Figure 11 below. The doctor from here can manage all his daily schedules and performing the necessary services for each patient. According to this way of work that the doctor performs, shows transparency with patients and working hours.

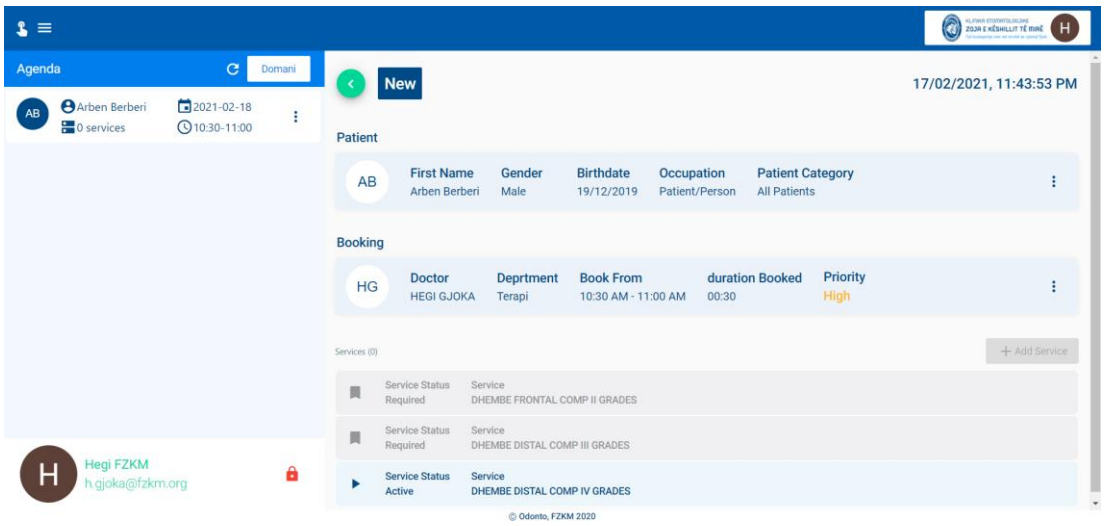

Figure 11: Odonto poltron standalone web app 10”

This ERP system is best described by the Figure 12 below that shows the whole workflow of the application. This information system manage the whole clinic departments as the warehouse, reception, therapy, orthodontics, surgery, etc. In the warehouse department it manages the stock items, the individual magazines of dentists, the suppliers, this system performs CRUD operations for purchases, inventory regulations, consumptions, and others. In the reception department it manages the dentists, their calendar for appointments, and they register patients. This ERP manages the medical cartel of each patients, which consist of their services that they have done,

the dentist list that have performed their services, and others. The second most important department, after the warehouse, is the armchair of the dentists. With a 10 inch tablet the dentist manages his patients and the services that the patients needs.

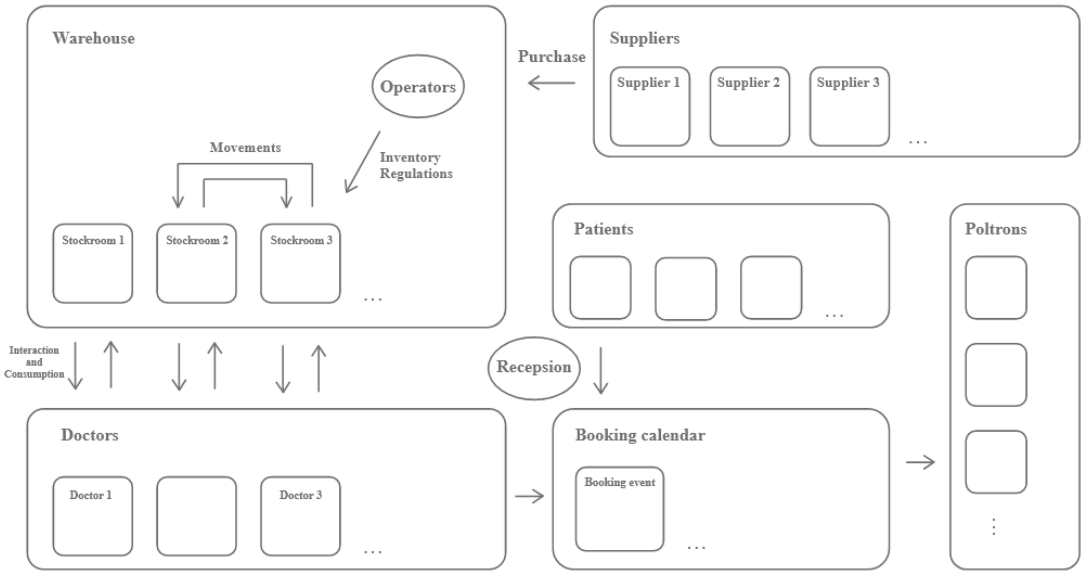

Figure 12: Odonto workflow diagram

The following chapter will explain the implementation of interceptors on the frontend to perform asynchronous requests in the direction of a backend application which is able to store logs in the database to be then analyzed, in order to achieve the purpose of the thesis.

CHAPTER 5

DATA GATHERING

5.1 Introduction

In this chapter is going to be explained in detail all the work that is done in order to gather meaningful data from the software that is taken into consideration. To achieve this goal in order that the application is not affected by the performance and actions, it is intended to create an interceptor in the frontend. This interceptor is able to intercept request and response. Since the request and the response are all that the application performs and displays, there is no need to intervene more and in other parts such as the backend. In order to achieve the purpose of the thesis this interceptor must process this data and store it in chronological and tabular manner. This is supported by a backend application created in the newest NestJs framework, whose task is to process the data coming in the form of an asynchronous request from the interceptor located on the frontend. After processing the data this application will save the log in the database which is created in Postgres.

5.2 Interceptor

In the Angular (framework in which the frontend part of the application is created) the concept of interceptor is widely used to manage the part of errors such as 401 unauthorized status code, 403 forbidden, 404 not found, etc. This example shows that the interceptor can capture the response of a request and can view its details. The interceptor is also used in the composition of a request, for example if an action needs to have a authorization token as its header, in order for the server to verify the authenticity of the action and the operator performing it. According to the above sentence it is understood that the interceptor can mutate the request. Interceptors in angular are best described by Figure 13 below.

Figure 13: Interceptor diagram

5.3 NestJs framework and Postgres database

NestJs is a new framework similar with Angular, which is built on top of node.js and uses http server framework like express (which is default). Nest it is used for backend application, micro services, or both. Since I as the writer of this thesis have experience as a frontend developer with Angular framework, it was easy to create a backend in this technology. This framework uses typescript and combines Object Oriented Programming (OOP), Functional Programming (FP), and Functional Reactive Programming (FRP). In order to achieve the goal of creating a backend in this framework, it is needed a database that communicate with backend. Since this framework uses OOP it is divided into modules and classes. The structure is same as the structure of Angular with modules, pipe, services, and components, but instead of components NestJs uses controllers. Controllers are represented by a class with an annotation of @Controller as shown at Figure 14. The class is a single tone that means that it is initialized in the beginning of the application start. Inside this class contains methods that may or may not been annotated with @Get, @Post, @Put, @Delete, etc. for intercepting request that uses this methods. The rest work the same as in Angular.

NestJs framework has lots of tools when it comes to database connection. This tools are used for relational and no relational databases, even GraphQL.

cats.controller.ts

```ts
import { Controller, Get } from '@nestjs/common';

@Controller('cats')
export class CatsController {
  @Get()
  findAll(): string {
    return 'This action returns all cats';
  }
}
```

Figure 14: Controller class example

Postgres is an open source relational database that in 2020 was on top three databases used. Since in this thesis the backend application uses Object Relational Mapping (ORM), the database knowledge is not needed.

5.4 Implementation of interceptor and Log service

In this part will be presented step by step the implementation of the interceptor in the frontend application. Apart of the interceptor it is needed also a service in order to send request in the backend.

Step one is to create an interceptor class that with intercept every single request made by the interaction with the frontend application and the responses that the server will return. The interceptor consist of intercepting the request and intercepting the response from the server without making any changes to it to not affect the application consistency. First the Figure 15 below with show a visualization of the code that it is used in the interceptor and is going to be explained in details.

```
@Injectable()
export class LoggerInterceptor implements HttpInterceptor {
  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
    const loggerService = new LoggerService();

    return next
      .handle(loggerService.setRequest(req))
      .pipe(
        tap( next: (event: HttpEvent<any>) => {
          const tokenOrUser: string = req.headers.get('Authorization');
          loggerService.setTokenOrUser(tokenOrUser);

          if (event instanceof HttpResponse) {
            loggerService.pushLog(event);
          }
        })
      );
  }
}
```

Figure 15: Interceptor implementation

The interceptor class is decorated with the @Injectable decorator which turn the class into a singleton class that is injected into any request that is made. The first argument “req” of type HttpRequest is the request composed by angular that is passed through the network to the backend server. The argument “next” of type HttpHandler helps to pass the request and intercept its response. LoggerService class is the service that will make the request to our backend in NestJs as form of an event log. As we see the request that passes through the handle function and the event (response) is not mutated. The pipe function accepts functions as a real pipeline and passes the argument to each one, for example tap function which is main idea is to tap into a stream and only read the information without mutating it.

Step two is to create a service class that will serve to make asynchronous request to our backend in order to generate event logs. This class will not be a singleton as all service class are because in the development state, I as a researcher encountered problems as the interceptor implicitly uses httpClient and this argument injected into the interceptor cannot been passed to the service constructor. So I pass the httpClient argument to the service in the app module via a static method and store it in a static parameter in the service class. Having the advantage that an angular project creates a new instance in each browser that the client opens, the static method does not create

performance and functionality issues. This service has two main methods, which are setRequest and pushLog. The setRequest method gets the request object as an argument from the interceptor and store it in a parameter called req of type IRequest tha is a siple object that contains the url of the request, the method which can be GET, POST, PUT, and DELETE, and the request body in case it is a POST or PUT request. The setRequest method after storing the request to the parameter, it then returns the original request back to the interceptor. The other method pushLog check if the request is defined and make an asynchronous POST request to our backend. This class also contains the initLoggerService static method that its main purpose is to assign the httpClient argument to the static paramenter, in order to make available to the pushLog method the possibility to make an asynchronous request. The Figure 16 will be shown the implementation of setRequest method and Figure 17 will be shown the implementation of the pushLog method.

```
private _req: IRequest;
private _logModel: LoggerModel;

public static initLoggerService(httpClient: HttpClient, url?: string): ILoggerService {...}

public setRequest<T = any>(req: HttpRequest<T>): HttpRequest<T> {
  this._req = req;
  this._logModel = new LoggerModel(this._req);

  return req;
}
```

Figure 16: LoggerService class setRequest method

Figure 17: LoggerService class pushLog method

The whole respective code for this class will be shown into the appendix section of this thesis.

5.5 Reasoning for this implementation

In this section will be described in details the reasons that this technique is used and its advantages and disadvantages. This methodology technique for extracting request and responses in the frontend was due to performance reasons. Since the interceptors in angular intercepts the request toward the server and the server response back to the client without affecting or mutating them. An angular project after it has been built and deploy to a specific server and host. The application for each request from the client creates a new instance in the browser where it is requested. While the application in the backend after it is built and deploy to the server, an instance is created once and only once in that server to handle the client requests which in our case is the client's interaction with the frontend application. If the interceptor used was implemented in the backend application, it would create issues in terms of performance, since this application as a whole is an ERP and has millions of interactions and communications with the database. For this reason this

implementation is done in the frontend, as the angular supports the usage of interceptors.

5.6 Creation of a new backend in nestjs

In this section will be explain in detail the creation and implementation of a log handler backend application in order to receive request from the interceptor and the log service that is previously implemented into the frontend application.

The first step was to create a new NestJs project using the command “nest new <project-name> (in our case log-generator)” from the command line. After creating the project the other step is to create a new log module, since nest has the same architecture as angular.

The second step is to create a log module with the command “nest generate module <module-name> (in our case logger)” and the following commands are for creating the controller and the service of this module “nest generate controller logger” and “nest generate service logger”. After this set of commands the next step is to install from node package manager (npm) the ORM that will be used to create table and handle insert, update, and delete request toward the database. In the command line we type “npm install @nestjs/typeorm typeorm pg” and “pg” stands for the type of the database that tis ORM will handle which in our case is a Postgres database.

The third step is to create an entity which is a class annotated with the @Entity annotation and will extend BaseEntity class provided by NestJs ORM, in order to enhance the functionalities of our class in order to handle insert row to the database, update row, and delete row. The entity stands for the table in the database, which in our case will be “log” table. The entity will have an auto increment column that serve also as a primary key with the name of “processId” of type number. The other columns will be “caseAction”, “casePatient”, and “caseStockItem” of type string, “action”, “timestamp” of type timestamp, “requestUrl”, “requestMethod”, “requestBody”,

“responseBody”, and “tokenOrUser” of type string. The Figure 18 below will best describe this entity.

```
@Entity( name: 'log')
export class LogEntity extends BaseEntity {
  @PrimaryGeneratedColumn()
  private processId: number;

  @Column()
  private caseAction: string;

  @Column( options: {nullable: true})
  private casePatient: string;

  @Column( options: {nullable: true})
  private caseStockItem: string;

  @Column()
  private action: string;

  @Column( options: 'timestamp')
  private timestamp: string;

  @Column()
  private requestUrl: string;

  @Column()
  private requestMethod: RequestMethodType;

  @Column( options: {nullable: true})
  private requestBody: string;

  @Column( options: 'text')
  private responseBody: string;

  @Column( options: {nullable: true})
  private tokenOrUser: string;
```

Figure 18: LogEntity columns that will be persist into the Postgres database

This entity also contain the main method for persisting into the database the event log. This method is defined as setData which accept as arguments data, save, and throwHandler. The data argument is of type ILogSetter which is an interface that has required 5 parameters that are “requestUrl”, “requestMethod”, “requestBody”, “responseBody”, and “tokenOrUser”. The save argument is of type Boolean and if is

true will persist the changes into the database, and last is the argument throwHandler which take a class exception and throws in case of issue occurring in the persisting step. The setData method fills the properties of this entity from the data argument and makes the preprocessing of the raw data into an understandable event log.

The first step for the preprocessing is to convert the requestUrl and requestMethod into an action for example GET "odonto/clinic/person/PERS00000000001/financial" to "getClinicPersonFinancial". With this set we have a self-descriptive action, which refers the process of getting the financial record of a specific person.

The next preprocessing occurs in the moment of defining a case for the log which can be a person id or a stock item id. To do that we can take into consideration the requestUrl, and in case of it has an id that starts with PERS that refers to a person and with PATB that refers to a patient booking event, will be inserted into casePatient column, if this id will not be present into the requestUrl we are going to parse the requestBody and search there if it has a key of personId. Also for the column caseStockItem we take into account the requestUrl in case it has an id that starts with SITM that refers to stock item, otherwise we parse the requestBody and then find if it has any matching key with itemId.

5.7 Reason for new backend implementation

This section will describe in detail the reason for implementing a new backend application. As mentioned above, this backend application consists of preprocessing the data coming from the interceptor which is implemented in the frontend, and stores them in a single table in a postgres database. The only reason this is a standalone application is for performance reasons. If we had implemented this logic, even if simple in the existing backend, we would have doubled the connections to the database as for each action a request is made, and this reduces performance by 50% or even worse we could hit the connection limits with the database and have a major issue. Even if this implementation were done in a more efficient way in the existing backend,

we would still have performance losses of up to 20%, as resources would be consumed. The other reason is, to make this technique as general as possible, so that it can be used in other systems as well. The only problem lies in the part of changing some parameters and the way of preprocessing some of the data coming to us from the interceptor, but anyway this is a shorter way than to be implemented from scratch in a new system. The Figure 19 best describes the flow of gathering data.

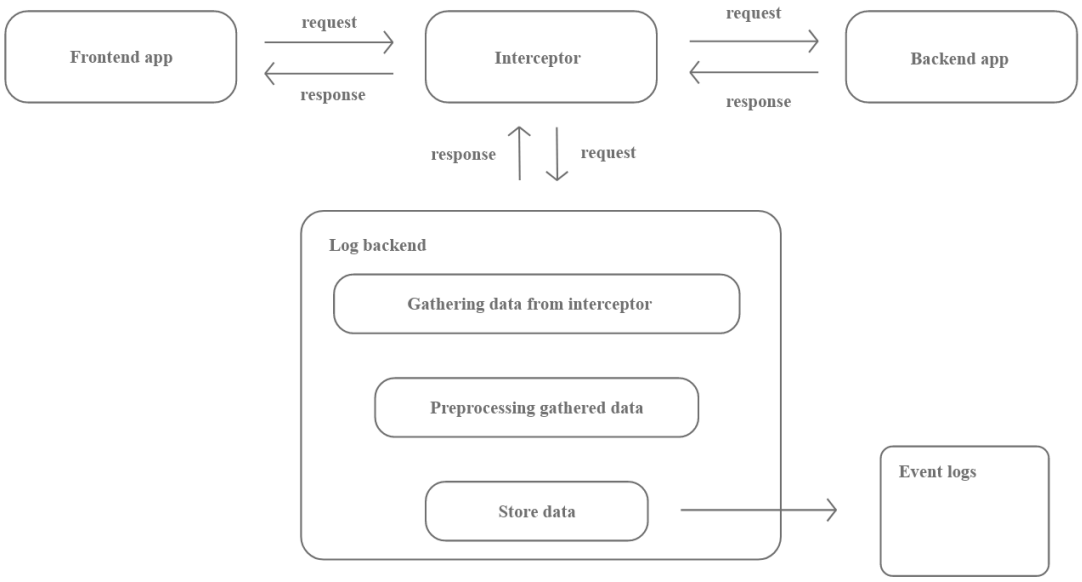

Figure 19: Data gathering technique workflow

Here we look at what the workflow of this data collection technique is like. The frontend application performs its functionality as normal, and the only thing is that it performs an extra request in our backend for every functionality it performs, but since this request is performed asynchronously it does not affect the performance part of the frontend application. We also see that the existing backend has nothing added, so its performance does not change. Connected to the interceptor is the new backend which performs preprocessing of the data coming from it.

In conclusion for this section we can say that we have found a very good alternative to get the logs we want and the way we want them from any system. This generalized technique can be suggested in other studies that will be conducted in this field such as process mining.

5.8 Future work

In this last section of this chapter is possible to be proposed as a future work for us or for other researchers in this field. Due to lack of time and unwillingness to overload this thesis with other techniques that are not directly related to this thesis. We propose the use of robotic process automation (RPA) to test the functionality of the system in development stage. Since RPA are somewhat related to the topic of process mining, you can use our technique to analyze not only the system processes in the production stage but also in the development phase, also using RPA you can propose the elimination of repetitive testing for any new changes made at this stage.

CHAPTER 6

EXPERIMENT AND ALANYSIS

This chapter will explain step by step the methodology followed during the development of the technique applied in this thesis. This chapter like any other chapter of this thesis will be divided into several sections. A chronological explanation will be given from the moment after data gathering, formatting them in the required format, log analysis with some of the process mining techniques, analysis of processes that are used frequently and those which cause major delays. The process mining techniques that will be applied in this phase are the techniques of Fuzzy Miner, Inductive Miner, Conformance checking.

6.1 Formatting event logs

This section presents the formatting part of event logs generated and collected in our database. From our postgraduate database we can download the table in csv format. To achieve the purpose of development and research, in this thesis is used for the analysis part ProM tool. ProM is a process mining tool which contains a variety of plugins which use some of the algorithms mentioned in a section above also in the literature chapter. To use this tool and all its techniques, event logs that have been collected and downloaded in csv format (e.g. Figure 20) must be converted to XES format. XES stands for extensive event stream and is a format based on the extended manipulation language XML format. This format is very simplistic and contains all event logs in a simple way to be later manipulated by ProM plugin algorithms. This format also contains meta-tags such as the timestamp format, who are the departments, the actions performed, the people involved in these actions, and other resources located in the event logs. This part is easily performed by ProM tool as shown in Figure 21 as it consumes logs in csv format and transforms them in XES format, requiring the tool user to specify a meta-tag for each column when converting the file.

processId	casePatient	caseStockItem	action	timestamp	requestUrl	requestM...	requestBody	responseBody	tokenOrUser
1	PERS00000300100	NULL	getClinicPerson	19/02/2021 20:05	/api/odonto/clinic/person/PERS000003...	GET	{}	{"status":{"code":"STATUS_OK"},"...	45952ef0-9be0-426a-9669-a78ed1f8947f
2	PERS00000000112	NULL	postClinicPatientBooking	19/02/2021 20:10	/api/odonto/clinic/patient/booking/new	POST	{"bookFrom":"2021...	{"status":{"code":"STATUS_OK"},"...	45952ef0-9be0-426a-9669-a78ed1f8947f
3	PERS00000300100	NULL	getClinicPerson	19/02/2021 20:11	/api/odonto/clinic/person/PERS000003...	GET	{}	{"status":{"code":"STATUS_OK"},"...	45952ef0-9be0-426a-9669-a78ed1f8947f
4	PERS00000300100	NULL	getClinicPerson	19/02/2021 20:11	/api/odonto/clinic/person/PERS000003...	GET	{}	{"status":{"code":"STATUS_OK"},"...	45952ef0-9be0-426a-9669-a78ed1f8947f
5	PERS00000000110	NULL	getClinicPerson	19/02/2021 20:11	/api/odonto/clinic/person/PERS000000...	GET	{}	{"status":{"code":"STATUS_OK"},"...	45952ef0-9be0-426a-9669-a78ed1f8947f
6	PERS00000000110	NULL	postClinicPatientBooking	19/02/2021 20:13	/api/odonto/clinic/patient/booking/new	POST	{"bookFrom":"2021...	{"status":{"code":"STATUS_OK"},"...	45952ef0-9be0-426a-9669-a78ed1f8947f
7	PERS00000300100	NULL	getClinicPerson	19/02/2021 20:13	/api/odonto/clinic/person/PERS000003...	GET	{}	{"status":{"code":"STATUS_OK"},"...	45952ef0-9be0-426a-9669-a78ed1f8947f
8	PERS00000010265	NULL	getClinicPerson	19/02/2021 20:13	/api/odonto/clinic/person/PERS000000...	GET	{}	{"status":{"code":"STATUS_OK"},"...	45952ef0-9be0-426a-9669-a78ed1f8947f
27	PERS00000000112	NULL	getClinicPerson	19/02/2021 20:18	/api/odonto/clinic/person/PERS000000...	GET	{}	{"status":{"code":"STATUS_OK"},"...	07f8b70c-fe5e-48a3-9c2d-9ca32b7fc3f0
9	PERS00000010265	NULL	postClinicPatientBooking	19/02/2021 20:15	/api/odonto/clinic/patient/booking/new	POST	{"bookFrom":"2021...	{"status":{"code":"STATUS_OK"},"...	45952ef0-9be0-426a-9669-a78ed1f8947f
10	PATB00000000566	NULL	getClinicPatientBooking	19/02/2021 20:15	/api/odonto/clinic/patient/booking/PAT...	GET	{}	{"status":{"code":"STATUS_OK"},"...	45952ef0-9be0-426a-9669-a78ed1f8947f
11	PATB00000000568	NULL	getClinicPatientBooking	19/02/2021 20:15	/api/odonto/clinic/patient/booking/PAT...	GET	{}	{"status":{"code":"STATUS_OK"},"...	45952ef0-9be0-426a-9669-a78ed1f8947f
12	PATB00000000568	NULL	getClinicPatientBooking	19/02/2021 20:17	/api/odonto/clinic/patient/booking/PAT...	GET	{}	{"status":{"code":"STATUS_OK"},"...	07f8b70c-fe5e-48a3-9c2d-9ca32b7fc3f0
13	PERS00000000110	NULL	getClinicPerson	19/02/2021 20:17	/api/odonto/clinic/person/PERS000000...	GET	{}	{"status":{"code":"STATUS_OK"},"...	07f8b70c-fe5e-48a3-9c2d-9ca32b7fc3f0
14	PATB00000000567	NULL	getClinicPatientBooking	19/02/2021 20:17	/api/odonto/clinic/patient/booking/PAT...	GET	{}	{"status":{"code":"STATUS_OK"},"...	07f8b70c-fe5e-48a3-9c2d-9ca32b7fc3f0
15	PERS00000000112	NULL	getClinicPerson	19/02/2021 20:17	/api/odonto/clinic/person/PERS000000...	GET	{}	{"status":{"code":"STATUS_OK"},"...	07f8b70c-fe5e-48a3-9c2d-9ca32b7fc3f0
16	PATB00000000567	NULL	getClinicPatientBooking	19/02/2021 20:18	/api/odonto/clinic/patient/booking/PAT...	GET	{}	{"status":{"code":"STATUS_OK"},"...	07f8b70c-fe5e-48a3-9c2d-9ca32b7fc3f0
17	PERS00000000112	NULL	postClinicPatientService	19/02/2021 20:18	/api/odonto/clinic/patient/service/new	POST	{"bookingId":"PATB...	{"status":{"code":"STATUS_OK"},"...	07f8b70c-fe5e-48a3-9c2d-9ca32b7fc3f0
18	PATB00000000567	NULL	getClinicPatientBooking	19/02/2021 20:18	/api/odonto/clinic/patient/booking/PAT...	GET	{}	{"status":{"code":"STATUS_OK"},"...	07f8b70c-fe5e-48a3-9c2d-9ca32b7fc3f0
19	PERS00000000112	NULL	getClinicPerson	19/02/2021 20:18	/api/odonto/clinic/person/PERS000000...	GET	{}	{"status":{"code":"STATUS_OK"},"...	07f8b70c-fe5e-48a3-9c2d-9ca32b7fc3f0
20	PATB00000000568	NULL	getClinicPatientBooking	19/02/2021 20:18	/api/odonto/clinic/patient/booking/PAT...	GET	{}	{"status":{"code":"STATUS_OK"},"...	07f8b70c-fe5e-48a3-9c2d-9ca32b7fc3f0
21	PERS00000000110	NULL	getClinicPerson	19/02/2021 20:18	/api/odonto/clinic/person/PERS000000...	GET	{}	{"status":{"code":"STATUS_OK"},"...	07f8b70c-fe5e-48a3-9c2d-9ca32b7fc3f0
28	PATB00000000568	NULL	getClinicPatientBooking	19/02/2021 20:19	/api/odonto/clinic/patient/booking/PAT...	GET	{}	{"status":{"code":"STATUS_OK"},"...	07f8b70c-fe5e-48a3-9c2d-9ca32b7fc3f0
22	PATB00000000567	NULL	getClinicPatientBooking	19/02/2021 20:18	/api/odonto/clinic/patient/booking/PAT...	GET	{}	{"status":{"code":"STATUS_OK"},"...	07f8b70c-fe5e-48a3-9c2d-9ca32b7fc3f0
23	PERS00000000112	NULL	getClinicPerson	19/02/2021 20:18	/api/odonto/clinic/person/PERS000000...	GET	{}	{"status":{"code":"STATUS_OK"},"...	07f8b70c-fe5e-48a3-9c2d-9ca32b7fc3f0
24	PATB00000000567	NULL	getClinicPatientBooking	19/02/2021 20:18	/api/odonto/clinic/patient/booking/PAT...	GET	{}	{"status":{"code":"STATUS_OK"},"...	07f8b70c-fe5e-48a3-9c2d-9ca32b7fc3f0
25	PERS00000000112	NULL	getClinicPerson	19/02/2021 20:18	/api/odonto/clinic/person/PERS000000...	GET	{}	{"status":{"code":"STATUS_OK"},"...	07f8b70c-fe5e-48a3-9c2d-9ca32b7fc3f0
26	PATB00000000567	NULL	getClinicPatientBooking	19/02/2021 20:18	/api/odonto/clinic/patient/booking/PAT...	GET	{}	{"status":{"code":"STATUS_OK"},"...	07f8b70c-fe5e-48a3-9c2d-9ca32b7fc3f0
29	PERS00000000110	NULL	getClinicPerson	19/02/2021 20:19	/api/odonto/clinic/person/PERS000000...	GET	{}	{"status":{"code":"STATUS_OK"},"...	07f8b70c-fe5e-48a3-9c2d-9ca32b7fc3f0
30	PATB00000000568	NULL	getClinicPatientBooking	19/02/2021 20:19	/api/odonto/clinic/patient/booking/PAT...	GET	{}	{"status":{"code":"STATUS_OK"},"...	07f8b70c-fe5e-48a3-9c2d-9ca32b7fc3f0
31	PATB00000000568	NULL	putClinicPatientBooking	19/02/2021 20:19	/api/odonto/clinic/patient/booking/PAT...	PUT	{"bookFrom":"2021...	{"status":{"code":"STATUS_OK"},"...	07f8b70c-fe5e-48a3-9c2d-9ca32b7fc3f0
32	PATB00000000568	NULL	getClinicPatientBooking	19/02/2021 20:19	/api/odonto/clinic/patient/booking/PAT...	GET	{}	{"status":{"code":"STATUS_OK"},"...	07f8b70c-fe5e-48a3-9c2d-9ca32b7fc3f0
33	PERS00000000110	NULL	getClinicPerson	19/02/2021 20:19	/api/odonto/clinic/person/PERS000000...	GET	{}	{"status":{"code":"STATUS_OK"},"...	07f8b70c-fe5e-48a3-9c2d-9ca32b7fc3f0
34	PATB00000000567	NULL	getClinicPatientBooking	19/02/2021 20:20	/api/odonto/clinic/patient/booking/PAT...	GET	{}	{"status":{"code":"STATUS_OK"},"...	5c45e426-025c-4d42-b45b-6de035d59dbd
35	PERS00000000112	NULL	getClinicPerson	19/02/2021 20:20	/api/odonto/clinic/person/PERS000000...	GET	{}	{"status":{"code":"STATUS_OK"},"...	5c45e426-025c-4d42-b45b-6de035d59dbd
36	PATB00000000569	NULL	getClinicPatientBooking	19/02/2021 20:20	/api/odonto/clinic/patient/booking/PAT...	GET	{}	{"status":{"code":"STATUS_OK"},"...	5c45e426-025c-4d42-b45b-6de035d59dbd

Figure 20: Event logs in csv format

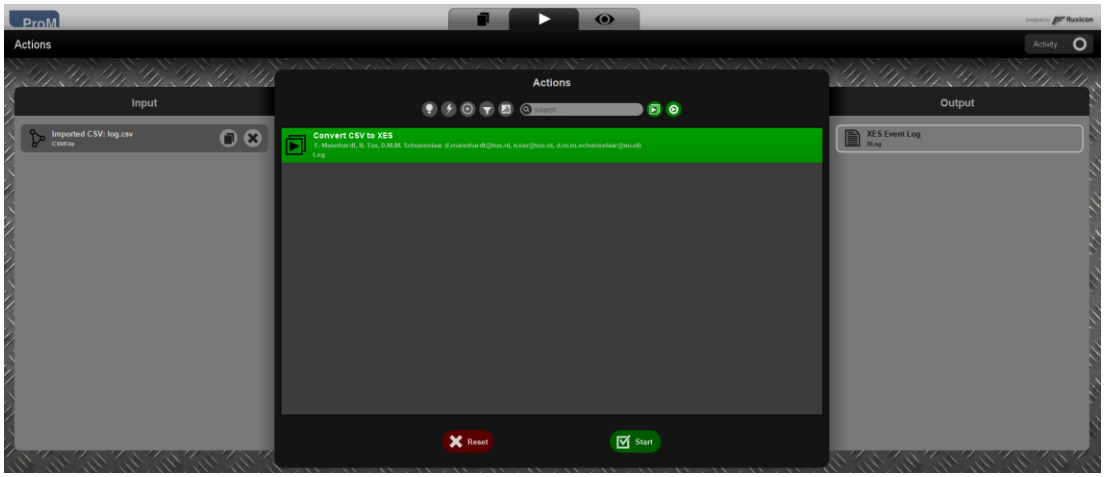

Figure 21: ProM tool XES format convert

The minimum required meta-tags that are needed to be specified are case, action, and timestamp. The meta-tag case specifies the case that is taken into consideration and the processes it goes through during its journey through the system. Action specifies the column where the action is described and the timestamp column that has the time when each log occurred. We can also specify other meta-tags such as start time and end time of an action, we can specify the people who participate in this action or departments, etc.

6.2 Event log analysis

In this session will be feature some of the process mining techniques, such as Inductive Miner, business process modeling notation BPMN, Petri Nets, and process

tree. After the phase of converting event logs to XES format we can use this new file to generate process models.

Since each process model discovery algorithm is part of a plugin. We will first use the Petri Net plugin. This plugin aims to create a representation of a simple process model like in the Figure 22. After this process we select the XES file and the newly created Petri net and pass it to a plugin called "Petri net instance replay", which aims to pass each event of the event logs to process model to make an analysis of the compliance or approximation of the process model with each case presented in the event logs.

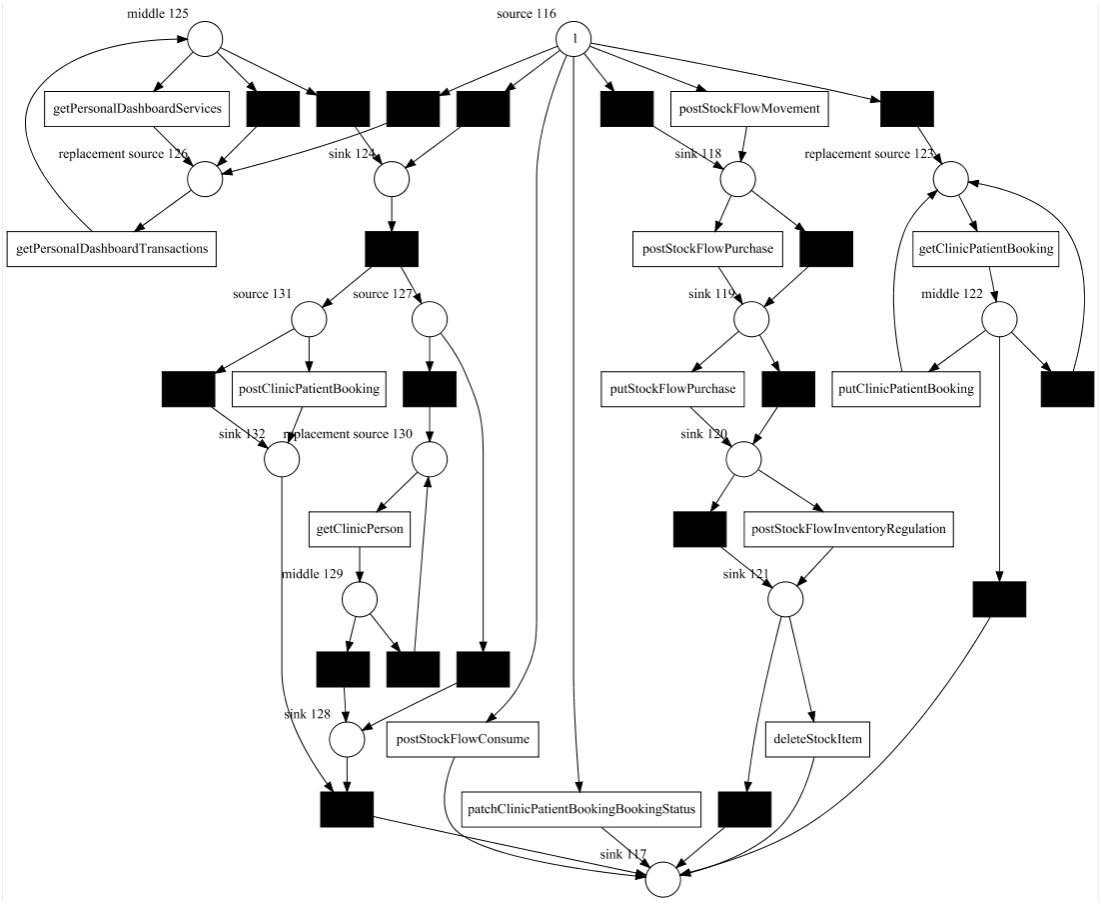

Figure 22: Petri Net representation of the event logs

BPMN representation is also a ProM tool plugin that is very similar to Petri net, but differs in connections. BPMNs have Boolean operators as 'and', 'or', and 'xor' operators. This presentation makes the process model as clear as possible for the

reader, having the legend of Boolean operators for what they serve. Figure 23 shows the BPMN generated by event logs.

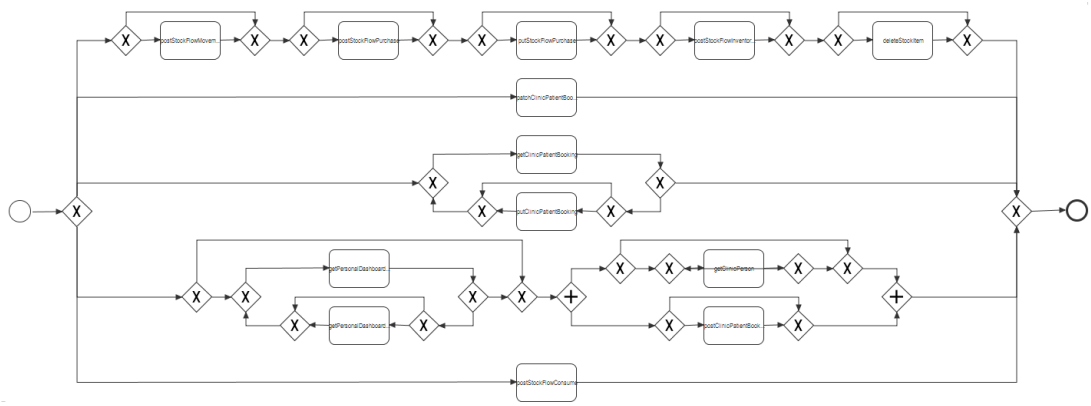

Figure 23: BPMN representation of event logs

And the last but most important plugin we will consider will be Inductive miner. Unlike the previous two plugins, this plugin does not generate a process model, but a process tree, and then this tree-like representation is converted to Petri net and BPMN. Also having a structure in the form of a tree, we have the advantage to be more conforming to event logs as a process tree can have some representations as process models. Also as the presentation of the BPMN process model that had represented the connections with the Boolean operator. As well as inductive miner process tree has its connections represented with Boolean operator with a slight change as there is also an arrow which indicates the sequence. The presentation in tree form is more complicated than all the other presentations, but the advantage of this kind of complicated presentation is that we should not display it this way for the simple user, but transform it into a BPMN which is very clear and self-explanatory.

6.3 Evaluations

This section will show the results and also the conformance checking three techniques used. After performing and repeating the part of the process model discovery by tweaking the data, we tested all the obtained models for conformance checking. In Figure 24 below you will see each of the selected cases to be tested for

conformance checking. The figure shows a scattered graph with two dimensions which are fitness and precision, where each point represents a model process and based on Pareto front [1] we consider the red points.

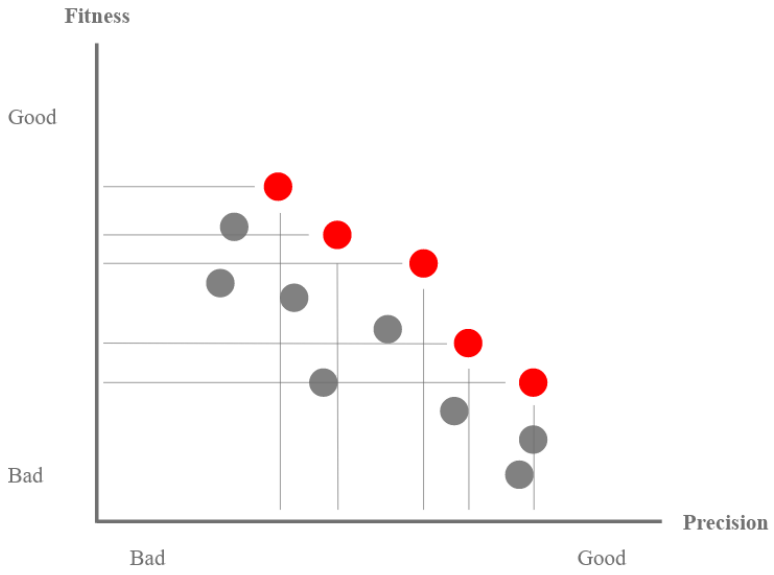

Figure 24: Pareto front for process models

Process models that passed for the second phase will be tested once again with these event logs and new event logs that even we in this thesis have not used or mentioned. Table 1 shows the records showing the tests performed for the Petri Nets process model generated by event logs. I notice that two out of 5 have had fitness and high precision in conformance checking with existing logs and new logs. In the figure above there is only a red dot located on the bottom right.

Table 1: Conformance checking for Petri nets model

	Conformance checking	
	Event logs	New event logs
Petri Net model 01	0.9947	0.9543
Petri Net model 02	0.9901	0.9484
Petri Net model 03	0.9857	-
Petri Net model 04	0.9742	-
Petri Net model 05	0.9107	-

Table 2 shows the records showing the tests performed for BPMN process models generated by event logs. In this case we have four models taken into consideration where one of them is part and the graph above in Figure 6.5 as a red dot on the top left. The red dot indicates that it has passed the tests with the new logs.

Table 2: Conformance checking for BPMN

	Conformance checking	
	Event logs	New event logs
BPMN 01	0.9273	0.9008
BPMN 02	0.8997	-
BPMN 03	0.8823	-
BPMN 04	0.8816	-

While Table 3 shows the tests performed for Inductive miner process tree which has been converted into two Petri nets process models and two in BPMN. The test was passed by 2 Petri nets process models and 1 BPMN.

Table 3: Conformance checking for inductive miner process tree

	Conformance checking	
Process tree converted:	Event Logs	New event logs
Petri net model 11	1.0	0.9867
Petri net model 12	0.9998	0.9802
BPMN 13	0.9641	0.9211
BPMN 14	0.9212	-

CHAPTER 7

CONCLUSION AND FUTURE WORK

In this last chapter are stated all the achievements of the thesis, as well as about the problems were encountered during the development and work that will be proposed to be performed in the future by the researcher.

The main achievement was the implementation of a very generalized and efficient technique, to extract logs from any kind of system and in the right way, containing the basic elements to be analyzed by process mining tools. As explained in the previous chapters, the ERP system consist of two parts, frontend in angular and backend in java. In the IS, very little was intervened in the frontend by introducing a single interceptor which performs asynchronous requests for each functionality of the system in a new backend in order to generate logs. The new backend which was implemented as standalone application for performance reasons, processes the data coming from the frontend, turning them into event logs which are being used by process mining tools. For the visual part and analyzes it is used the ProM tool. In this study are implemented several techniques to discover the process model of the system. Among a series of techniques, the inductive miner technique proved to be the most effective and the reasons why are explained below. The first is that the inductive miner creates a process tree which can then be represented visually by Petri nets or BPMNs, thus giving us the advantage of visualizing the process in several versions. The second advantage using process tree was that it can represent process models of different shapes. This process tree advantage helped us in the phase of bottleneck analyzing and conformance checking. In terms of fitness and precision process model generated by process tree, reached an accuracy of 98.67%, which means that the process model was visually simple and précised, and all the patterns occurring in event logs found themselves in the model. In terms of bottleneck analysis, the only inhibitory factor was the time it took the operator to complete the processes.

7.1 Limitation and future work

In this last section we will mention two limitations that we have encountered and tried to resolve. The first limitation occurred due to the COVID-19 pandemic which brought many people to their knees, as well as many businesses which were closed for more than 3 months. The business, in which was this operating system, unfortunately closed for a period of 4-5 months and made data collection impossible. This limitation turned into a possibility, as we thought that part of the data would be generated by a RPA. RPA generated 18% of the data that needed to be collected. Also in this regard we had our difficulties due to the complexity of the system. So our thesis leaves research space on the implementation of RPA intertwined with process mining to achieve automatic data generation and analysis. An example might be in the part of testing that takes place during software development.

REFERENCES

[1] W. M. P. van der Aalst, Process Mining, Data science in action, Springer, 2016.

[2] J. Gao, S. J. van Zelst, X. Lu and W. M. P. van der Aalst, "Automated Robotic Process Automation: A Self-Learning Approach," 2019.

[3] A. Restivo, A. Aguiar and A. Moreira, "An Incremental Approach to Testing AOP," *Communications in Computer and Information Science*, vol. 743, 2017.

[4] R. Yamamoto, K. Yamamoto, K. Ohashi, J. Inomata and M. Aoyama, "A Metamodel-Driven Business Process Modelling Methodology and Its Integrated Environment for Reusing Business Processes," *Journal of Software Engineering and Applications*, pp. 363-382, 2018.

[5] H. Nik, M. Reza and W. M. P. van der Aalst, "BIpm: Combining BI and Process Mining," in *Proceedings of the 8th International Conference on Data Science, Technology and Applications*, 2019.

[6] R. Conforti, M. La Rosa and A. Ter, "Filtering out Infrequent Behaviour from Business Process Event Logs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, pp. 300-314, 2017.

[7] M. F. Sani, S. van Zelst and W. M. P. van der Aalst, "The Impact of Event Log Subset Selection on the Performance of Process Discovery Algorithms," *Communications in Computer and Information Science*, vol. 1064, 2019.

[8] W. M. P. van der Aalst, "Object-Centric Process Mining: Dealing With Divergence and Convergence in Event Data," in *Software Engineering and Formal Methods*, Springer International Publishing, 2019, pp. 3-25.

[9] G. Li, R. M. de Carvalho and W. M. P. van der Aalst, "Configurable Event Correlation for Process Discovery from Object-Centric Event Data," in *2018 IEEE International Conference on Web Services (ICWS)*, 2018.

[10] P. Dixit, H. M. W. Verbeek and W. M. P. van der Aalst, "Fast Conformance Analysis based on Activity Log Abstraction," in *2018 IEEE 22nd International Enterprise Distributed Object Computing Conference (EDOC)*, 2018.

[11] A. Jalali, F. M. Maggi and H. A. Reijers, "A hybrid approach for aspect-oriented business process modelling," *Journal of Software: Evolution and Process*, vol. 125, p. 28, 2018.

[12] A. Jalali, "Weaving of Aspects in Business Process Management," 2018.

[13] M. Fernández-Ropero, H. A. Reijers13, R. Pérez-Castillo and M. Piattini, "Repairing Business Process Models as Retrieved from Source Code," vol. 147, pp. 94-108, 2013.

[14] D. DeMatteo, K. Heilbrun and E. A. Zillmer, Essentials of Research Design and Methodology, Hoboken, N.J. : John Wiley & Sons, 2005, p. 290.

[15] P. Wiśniewski, K. Kluza and A. Ligęza, "An Approach to Participatory Business Process Modelling: BPMN Model Generation Using Constraint Programming and Graph Composition," 2018.

APPENDIX

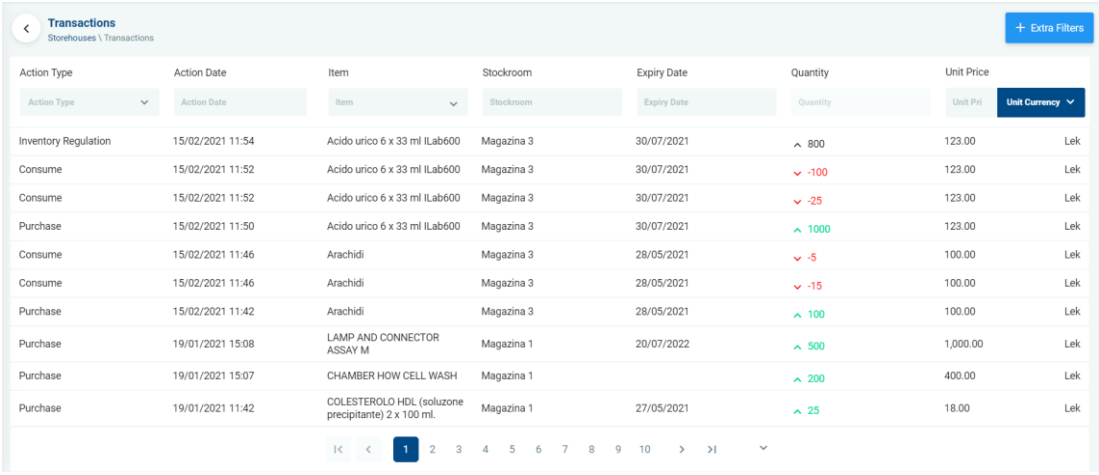

Figure 25: Transaction table

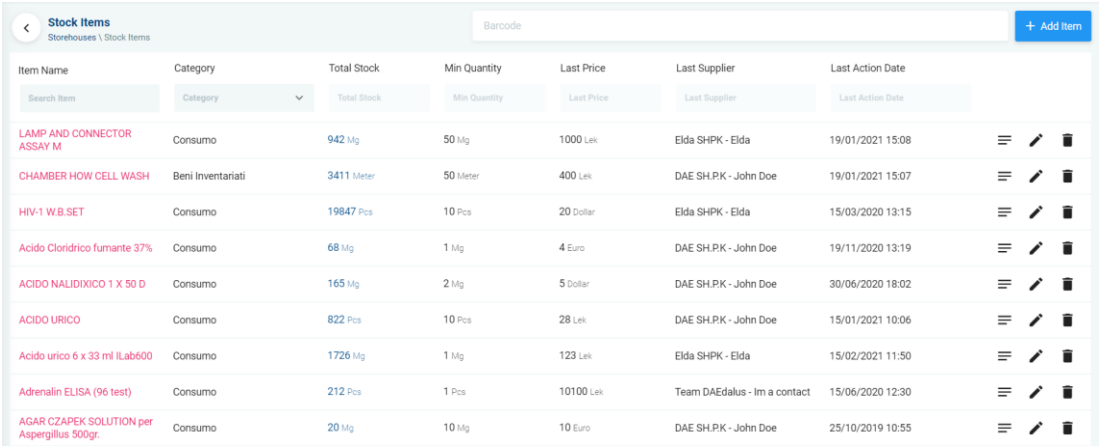

Figure 26: Stock item table

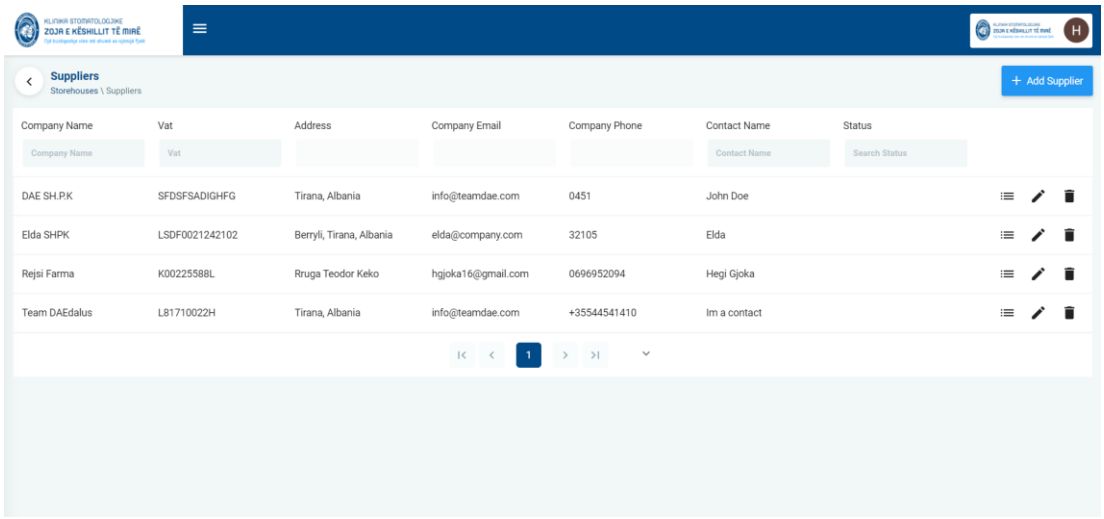

Figure 27: Supplier table

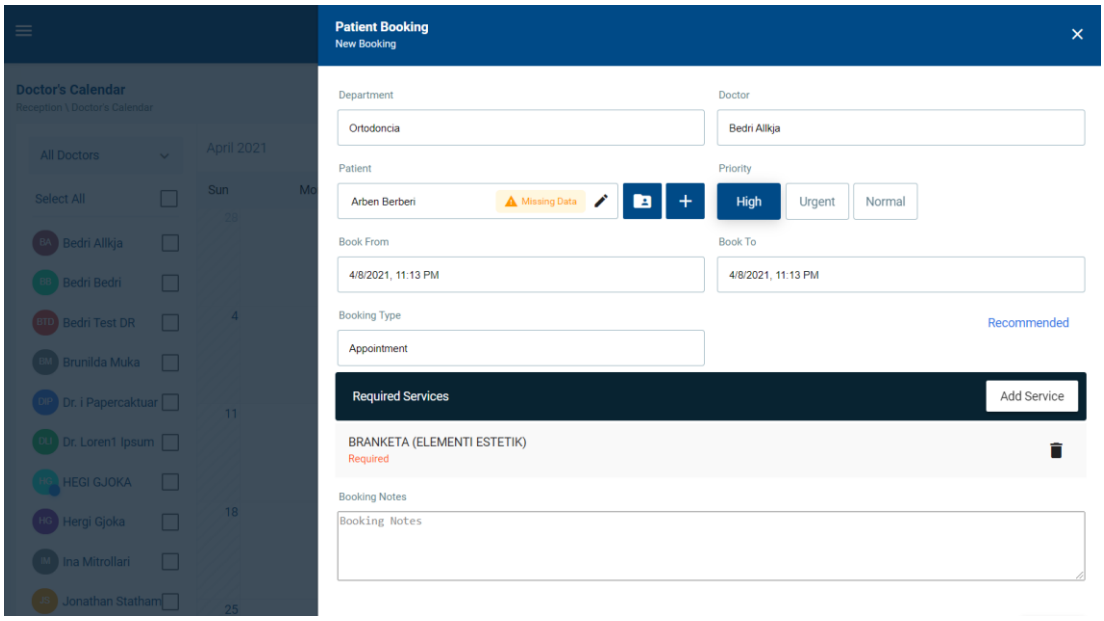

Figure 28: Adding booking appointment panel

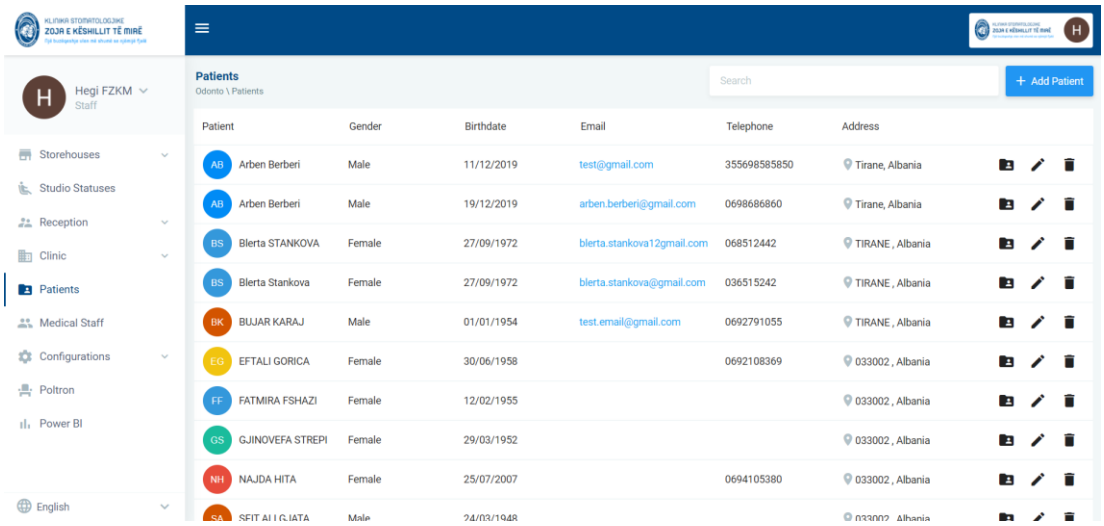

Figure 29: Patient table

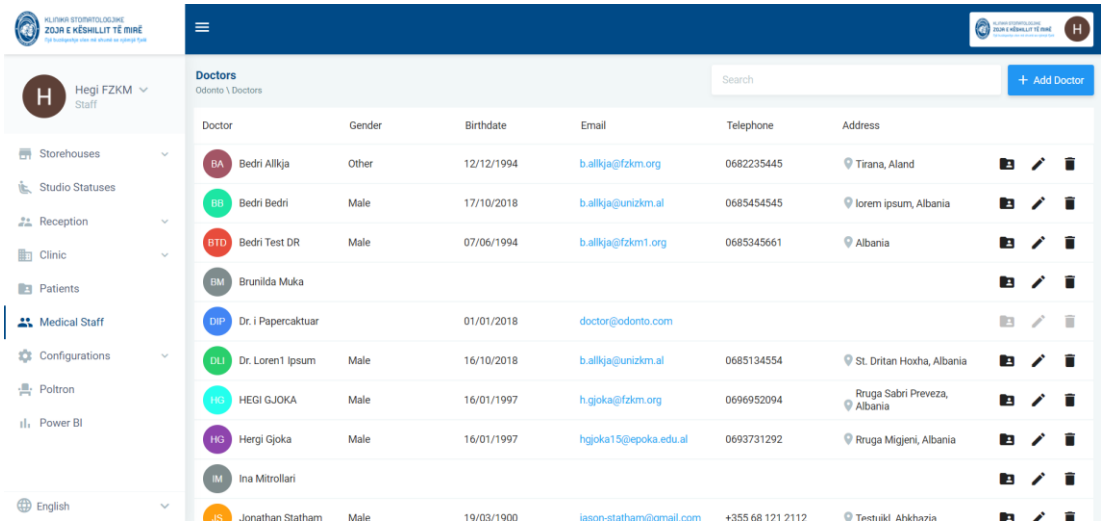

Figure 30: Doctor table