UNIVERSITÄT KOBLENZ · LANDAU

Fachbereich 4: Informatik

Compliance analysis of an ERP system in an SME based on process mining and business rules

Masterarbeit

zur Erlangung des Grades eines Master of Science
im Studiengang Web Science

vorgelegt von

Stela Nebiaj
Mat.-Nr. 216100836

Erstgutachter:	Prof.Dr. Patrick Delfmann
	Institut für Wirtschafts- und die Verwaltungsinformatik
Zweitgutachter:	Msc. Carl Corea
	Institut für Wirtschafts- und die Verwaltungsinformatik

Koblenz, im März 2019

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	☐	☐

. . . . . .

(Ort, Datum) (Unterschrift)

Abstract

Recently the workflow control as well as compliance analysis of the Enterprise Resource Planning systems are of a high demand. In this direction, this thesis presents the potential of developing a Workflow Management System upon a large Enterprise Resource Planning system by involving business rule extraction, business process discovery, design of the process, integration and compliance analysis of the system. Towards this, usability, limitations and challenges of every applied approach are deeply explained in the case of an existing system named SHD ECORO.

Zusammenfassung

In letzter Zeit sind die Workflow-Kontrolle sowie die Compliance-Analyse der Warenwirtschaft Systeme sehr gefragt. In dieser Richtung präsentiert diese Arbeit das Potenzial der Entwicklung eines Workflowmanagement-System auf einem grossen Warenwirtschaft-System, durch die Extraktion von Geschäftsregeln, die Ermittlung von Geschäftsprozessen, das Design des Prozesses, die Integration und die Compliance-Analyse des Systems. In diesem Zusammenhang werden Benutzerfreundlichkeit, Einschränkungen und Herausforderungen jedes angewandten Ansatzes im Fall eines bestehenden Systems mit dem Namen SHD ECORO eingehend erläutert.

Acknowledgement

I would first like to thank my supervisor **Prof. Dr. Patrick Delfmann** and his PhD student **Carl Corea**. They consistently allowed this paper to be my own work, but they unconditionally assisted whenever I ran into a trouble spot or had a question about my research or writing.

Second, I would also like to thank the experts who were involved in this research: **Dr. Daniel Bildhauer** and **Dr. Tassilo Horn**. Their passionate participation and input was very valuable for me.

Finally, I must express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Business processes are the most crucial part of an organizations success (Cardoso et al. 2004). Therefore, according to Cardoso et al. (2004), information technologies that focus on process management and process improvement have been good candidates to help organizations to fulfill their corporate visions and to improve their competitive positions. Two well known systems in the recent decades are ERP (Enterprise Resource Planning) and WfMS (Workflow Management Systems). They focus differently on business processes. On the one hand, ERP systems are data-centric and must be configured by setting various application parameters to control the workflow. On the other hand, WfMS are process-centric and support workflows involving humans and IT applications (Szirbik 2005). Considering the WfMS features, the ERP vendors have the tendency to integrate WfMS with their systems to control the workflow based on the business rules as well as increase the transparency of the company's internal procedures (Cardoso et al. 2004).
Moreover, as an integral part of a software adaptation project, it is necessary and common to discover the business rules (Ross 2003). But business rules are usually not implemented as an identifiable component of a system. There are times when business rules are implicitly embedded into the source code of business applications and due to the complexity of the code, it is a herculean task to manually search for them. In this case, a semi or totally automatic approach is needed for the business rule extraction from such business applica-

tion (Paknikar Shantanu 2014). This thesis presents three possible approaches followed by their usability and limitations based on the extracted results. Furthermore, automated techniques; such as process mining, help in discovering business processes in process aware systems like WfMS (van der Aalst et al. 2007, Mans et al. 2008, Perez-Castillo et al. 2011). But, in this research is investigated process mining technique upon a non process aware system called SHD ECORO. Additionally to the automated technique also a manual technique based on Porters value chain is used, which supports the development of a big but generalized picture of all the activities that are involved in the overall process and how they are interrelated (Dekker 2003, Porter 2001).
The process of adapting a WfMS is generally based on design, analysis, configuration, enactment and evaluation (Weske 2012). This thesis walks through each of the steps on SHD ECORO as a large ERP system in the furniture market.

1.1 Context of this work

This thesis shows the possibilities of discovering business rules and business processes from a large ERP system like SHD ECORO, with the goal to integrate a Workflow Management System (WfMS). SHD ECORO is a large ERP software specialized in the furniture industry. A process control is not possible for SHD ECORO today, but is increasingly demanded by the customers. Furthermore, since rules provide modeling of business logic in a declarative manner (Herbert Gomez Tobon 2010, Huang et al. 1996), it is important to ensure that business processes are aligned towards company goals and regulations. Documentation of business rules is not offered by SHD ECORO, therefore a process of extraction them from the code should be first undertaken to then integrate them in the compliance analysis of the system. In this case, it is necessary to understand the structure and dynamics of SHD ECORO, raising the current problems, identify improvements and promote common understanding about the desired situation. The main goals of this research are as follows:

- *Showing the feasibility of business rule extraction in large ERP systems:* Busi-

ness rule extraction is considered a challenging task for large systems, due to their complexity and size (Paknikar Shantanu 2014). Hence, it would be an overload of work and time to manually check for the business logic in the code. Few automatic approaches that deal with COBOL source code software are designed in some research (Huang et al. 1996, Sneed & Erdos 1996, Sneed 2001) to solve this issue. But, there are less alternatives on how to automatically find out business rules from Java source code. Therefore, in this research some of existing methods are implemented on SHD ECORO source code. Their results are discussed and a novel approach is proposed, based on the specifications of the system.

- *Investigating the performance of business process discovery techniques in large ERP systems:* Business process mining is an automated technique which permits the discovery and preservation of all meaningful embedded business knowledge by using event logs, which represent the business activities executed by an information system. Event logs can be easily obtained through the execution of process-aware information systems (Perez-Castillo et al. 2011, van der Aalst et al. 2007, Mans et al. 2008). However, in this case SHD ECORO as an ERP software even being non-process-aware information system, it implicitly supports organisations' business processes (Cardoso et al. 2004, Szirbik 2005). Apart from the automatic technique, this research presents a manual technique and its results as well.

- *Exploring the possibilities of WfMS (Workflow Management System) integration in ERP systems:* Throughout the last decades, Business Process Management (BPM) gained importance in the business community. Although BPM methods were first used in manufacturing, now they are widely applied in service industry as well (Harmon & Wolf 2008). Recently, the ERP vendors tend to integrate a WfMS into ERP systems. For instance, some vendors integrate workflow components, such as activities, transitions, actions, splits, joins and sub-processes, to make their systems workflow-driven. This helps to facilitate the customization and deployment of ERP systems (Cardoso et al. 2004). The potential of developing and managing a WfMS

upon an ERP system will be will be covered in this research.

1.2 Research methodology

This thesis is an analytic and applied research where the facts or information already available about the research topic are used to make a critical evaluation of the materials. Additionally, they are applied to find a solution for an immediate problem facing an industrial/business organization (Kothari 2004). In the line of the previous argumentation, this thesis explores the approaches already available about business rule extraction and business process discovery techniques and evaluates by applying them to integrate workflow-driven systems to ERP systems in order to facilitate the customization and deployment of them.

1.3 Structure of this work

The interconnection of the chapters as well as their research goals is shown in figure 1.1. Whereas the introduction of the chapters is summarized in the following section.

1. **Introduction.** *Chapter 1* provides an overview of this research by announcing the research topic, the existing problem, the scope of the research as well as the research goals.

2. **Basic concepts.** *Chapter 2* introduces the foundations of this research by providing; an introduction to ECORO as one of the systems facing the problem raised in the precious chapter, a presentation of the main concepts of Business Process Management field such as business processes and business rules, as well as a review of WfMSs and their alternatives in the market.

3. **Business Rule Extraction from the source code.** *Chapter 3* describes methods used to generate business rules from the code written in Java. It covers the methodology, the results and limitations of each of the approaches.

4. **Business Process Discovery.** *Chapter 4* first discusses the techniques used to discover business processes, which are then implemented step by step in SHD ECORO.

5. **Modeling and executing a business process.** *Chapter 5* provides one approach of integrating Camunda as a WfMS upon SHD ECORO. First the architecture of Camunda BPM Platform is explained. Afterwards it is integrated and some results of business process execution in compliance to the business rules are presented.

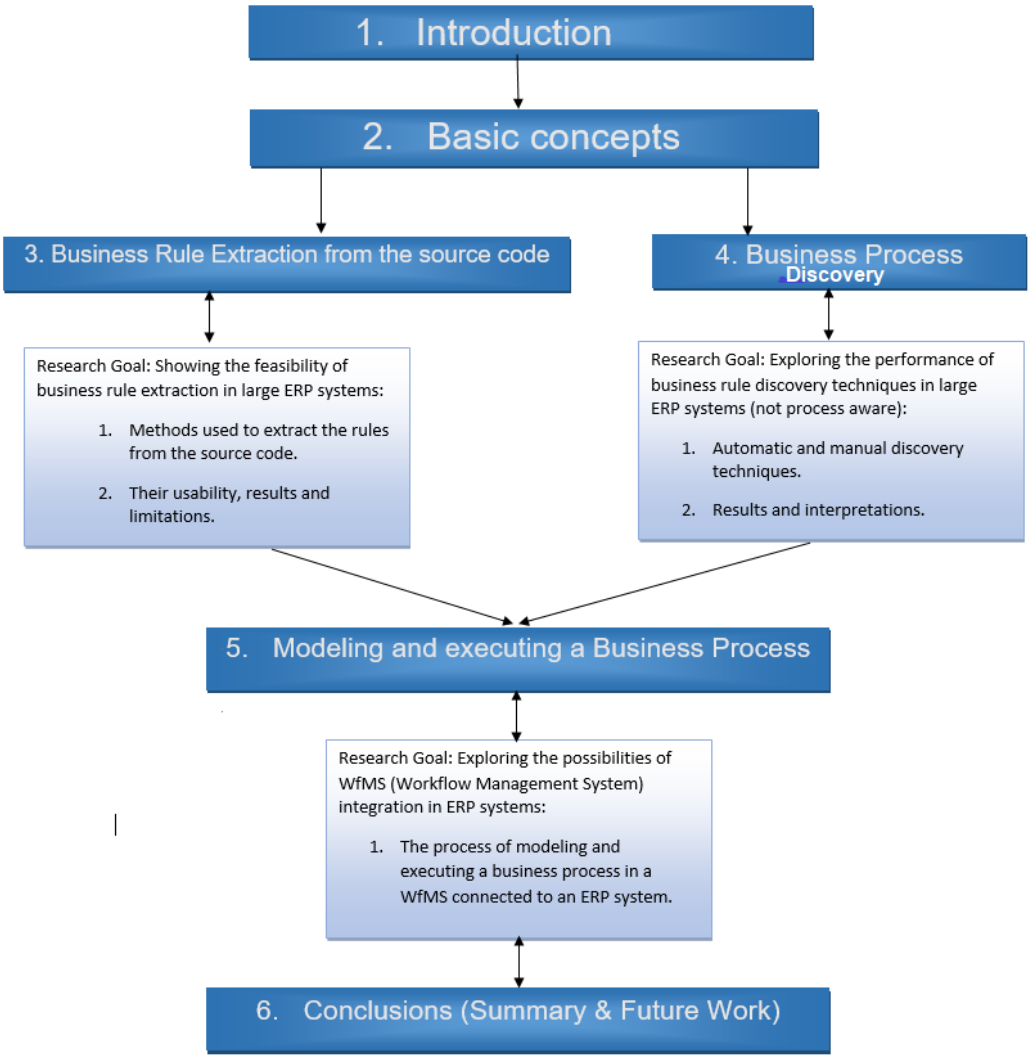


Figure 1.1: Interconnection of the chapters

Chapter 2

Basic concepts

This chapter introduces in general terms SHD ECORO as the Enterprise Resource Planning (ERP) software used in this thesis. Furthermore, it lays the basic concepts of Business Process Management. Moreover, another section of this chapter will cover the requirements of a WfMS and most used alternatives in the market, where just one of them will be selected for further work in this research.

2.1 Introduction to SHD ECORO

SHD ECORO, as one of several products delivered in the market by SHD AG, is an ERP (Enterprise-Resource-Planning) software, specialized in the furniture trade. Its development has started since '90s and now SHD ECORO counts around 100 worldwide customer, where more than 50% of them are located in Germany and the rest is from Netherlands, China, Switzerland etc...

SHD ECORO is implemented in Java and it has around 3.2 million lines of code. The most important modules also shown in the figure 2.1 are: purchase (einkauf), sale (verkauf), logistik (logistik) and requirement (vorgaben). It has several internally hard-coded processes, some of which are controllable via parameters. It is an example of many software systems developed in recent decades, in which the process logic in the code is hardwired

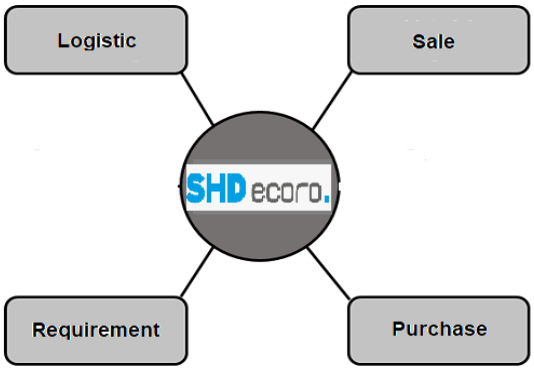


Figure 2.1: SHD ECORO main modules (source:self-made)

and thus the approaches, opportunities and risks from this work are largely transferred to other legacy applications.

Process management tools based on formal process description languages such as BPEL and BPM allow the graphic but formal description of processes in an executable form by the respective tool. They promise thereby an increase of the transparency of the company-internal procedures and lastly a saving by unification, simplification and flexible configuration as well as automation of these. Such process control is not possible for SHD ECORO today, but is increasingly demanded by the customers. According to Cardoso et al. (2004), integration of a WfMS in an ERP system is applicable. Possible approaches will be deeply discussed in the following chapters.

The basic concepts of BPM as well as a review of process engines and their alternatives in the market are explained in the upcoming subsections.

2.2 Business Process Management concepts

Since industrialization, work processes (defined in section subsection 2.2.1) have existed as products of continues search to optimize business efficiency by focusing more on the routine aspects of work activities. With the introduction of information technology, all processes that were before carried out entirely by humans, are partially or totally automated. According to Huijsman & Noordveld (2017), Business Process Management

(BPM) is:

> "*supporting business processes using methods, techniques and software to design, enact, control and analyze operational processes involving humans, organizations, applications, documents and other sources of information.*"

Moreover, Huijsman & Noordveld (2017) and Gillot (2008) do not consider BPM as a technology, but as a process-oriented management discipline, used as an umbrella term for strategic approaches that focus on dealing with business processes, process change and business rules.

Recently, the interest on BPM has been increasing, due to previously expressed advantages it brings to a business (Szirbik 2005). A growing interest of organizations has been noticed, with the aim to improve their business processes in order to be more competitive in a globalized economy that passes nowadays through a severe financial crisis with restrictive market conditions and limited profit margins (Harmon & Wolf 2008). Regarding Rudden (2007), general benefits expected from analyzing and modelling business processes using BPM are as follows:

1. *Efficiency* meaning elimination of manually given data and reduction in manual analyses and process cycle time.

2. *Effectiveness*, is provided from handling exceptions better and faster, making better decisions and from a consistent execution.

3. *Agility* is strongly related to flexibility, the ability to change quicker and support new business models.

Benefits expressed above seem to be the same as in implementing any other information system. Thus, Ko (2009) expresses them in more detail. According to him, some of the prominent benefits of analyzing and modelling business processes consist on an increased visibility and knowledge of companys activities, more ability to identify bottlenecks, easier identification of potential areas of optimization, reduction of lead-times, a

better definition of duties and roles in company and also modeling helps in fraud prevention, auditing, and assessment of regulation compliance.

Implementation of business process management is an iterative and incremental process defined into some stages that follow each other created so known BPM life-cycle (Weske 2012).

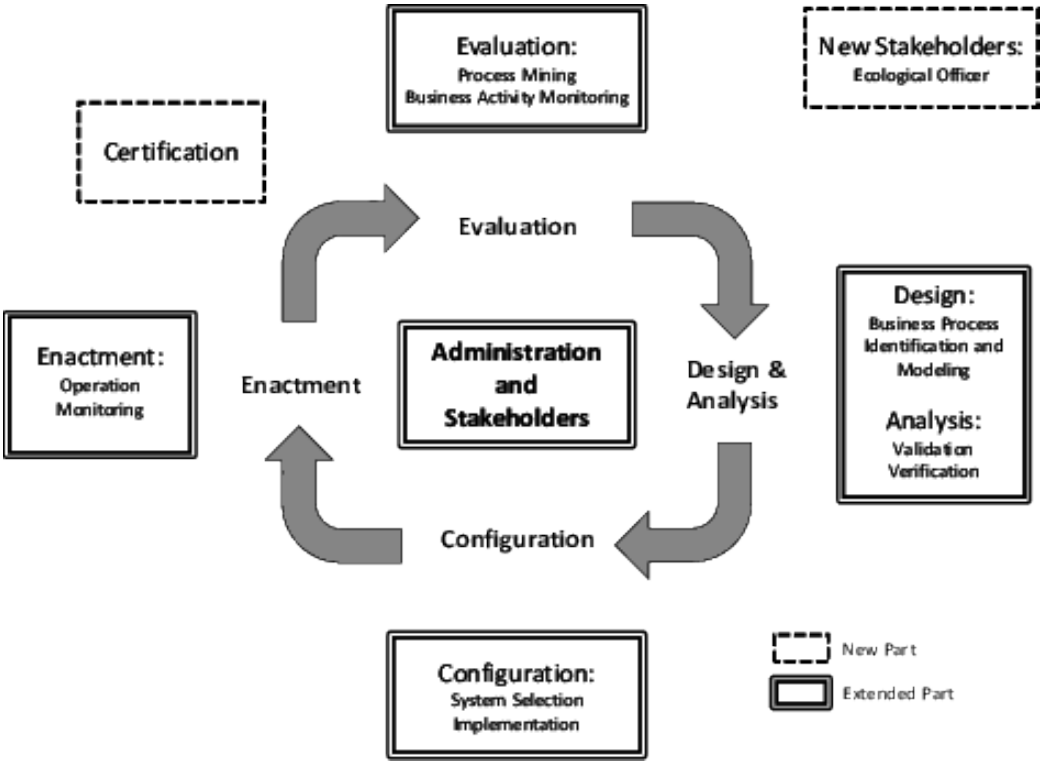


Figure 2.2: BPM Lifecycle (source:(Weske 2012)

According to Weske (2012), BPM lifecycle is a continuous process where the most crucial phases consist of:

1. *Design & Analysis:* Processes are identified, electronically modeled into Workflow Management Software (WfMS) and then analyzed by going into validation, simulation and verification. Graphical standards are dominant in this stage.

2. *Configuration:* In this stage, WfMS is selected, implemented tested and deployed. Its standardization is difficult because of different existing IT architectures of di-

verse enterprises.

3. *Enactment:* Electronically modeled business processes are deployed in WfMS engines. Operation standards dominate this stage, but processes are also monitored and maintained.

4. *Evaluation:* with the help of appropriate analysis and monitoring tools, bottlenecks and potential fraudulent loopholes in business processes can be first identified and then improved by the BPM analyst.

The first stage is a fundamental one, due to the fact that the quality of business processes resulted in the business process modeling phase is decisive to the success of business process management of an organization. The steps for this thesis will go through the lifecycle phases depicted previously in this chapter.
The following subsections will introduce two of the main concepts of BPM (Gayialis et al. 2016), very important for this thesis; business rules and business processes.

2.2.1 Business Processes

Business processes are found in business organizations and across organizations. A business process is commonly defined as:

> " *a collection of activities that takes one or more kinds of input and creates an output that is of value to the customer. A business process has a goal and is affected by events occurring in the external world or in other processes.(Ko 2009)*"

Generally, in an organization, processes are categorized into two main sets, namely; material processes, which scope is to gather physical components and deliver physical products and information processes, which relate to automated tasks and partially automated tasks. There is also a composed set, called business processes which are the information processes and/or material processes describing the market center activities of an organization (Georgakopoulos et al. 1995).

Based on Ko (2009) classification, business processes can be divided into private or public ones. Private business processes are internal to the enterprise and can be at the strategic, management, or operational level. Whereas, public business processes involve external organizations, e.g., delivery of goods, ordering of materials, etc. Those are also usually known as Collaborative Business Processes (cBPs).

2.2.2 Business Rules

Business rules are defined as being a set of operations, definitions and constraints applied in a company, which represent relevant actions or procedures aiming at defining or constraining some precise aspects of a business (Cosentino et al. 2012). Based on the works by Chaparro et al. (2012), Bajwa et al. (2011), business rules consist of two types, namely: behavioral also known as operative rules, and definitional which can be also found as structural rules.
The first mentioned set govern the behavioral or business operations in an optimal and suitable way. They always hold the sense of obligation or prohibition, can be directly violated and not all of them are automatable.

- **Not automatable:** E.g. A Nurse must visit a Patient at least every 1 hour
- **Automatable:** E.g. An Order over $ X must be accepted on Credit without a Credit Check

Whereas definitional rules structure and organize basic business knowledge. They hold the sense of necessity or impossibility and cannot be directly violated. All of them are automatable but not all of them can be considered as business rules. Usually there are two possibilities attached to this type of rules; classifications and computations ones.

- **Classification:** E.g. A Customer is always a Premium Customer if the Customer has placed Orders of more than $X
- **Computation:** E.g. The Total Price of an Order is always computed as the Sum of Order Item Prices

This research will have as a focus both types of business rules explained previously.

2.3 WfMS requirements

Before going into implementation steps for SHD ECORO, it is essential to specify some requirements and general need of the actual WfMS (Workflow Management System). From previous research (Quesada & Gazo 2007), a company management system should focus on success factors, as a key for their development, and must be tailored to the specific industry in which the company operates. For diverse industries also the CSF (Critical Success Factor) differ based on the structure, competitive strategy, industry position, geographic location, environment factors, time factors etc. However, in most of them, there should be three to six CSF that determine success and where the information systems should be derived from. Furthermore, getting to know about KPIs (Key Performance Indicators) helps in revealing the requirements of a management system to assist in the optimization of manufacturing processes, in the case of furniture industry (Rico 2017). Those factors and indicators together with business goals, act as basic criteria and requirements for information system selection and implementation process. Each of the goals and CSFs should be related to BPM tool requirements to be fulfilled.

The requirements of a management system could be functional, which depicts what the systems should do to achieve the business goals, business rules and interface requirements, or non-functional ones, which are qualities or attributes that the system should fulfill in addition to the functions it preforms and include performance, reliability, scalability, security and interoperability requirements (Tammy 2017). Both of two categories are needful for a satisfying selection of information software. In an overall view, a management system must include a workflow, must be accessible, must include an easy to use process mapping tool, must allow for customization and it also must allow collection and report on metadata (Financesonline 2018).

In order to find out requirements, firstly the goals of the business should be revealed. Regarding furniture industry, as depict in Lourens & Jonker (2013), the identified CSFs

through diverse literature are value, speed, innovation and flexibility. The value is related to costs, providing value for money, quality and reliability of the product. It is strongly connected to the KPIs relevant to C-Level Employees like CEOs, COOs, and everyone else with a C in their title. According to Rico (2017), C-Level Employees aim to identify opportunities to save money, improve efficiency, and increase production capacity.
Speed is defined as volume flexibility, fast and secure delivery. The right information delivered to the production staff, according again to Rico (2017), helps manage expectations, prevent downtime, and ensure everyone does their part. Some of the KPIs that assist in it are: work order, due date, inventory levels and expected receipt date. To quality, the main goal is to increase advantage by trying to be at the same league as competitors regarding the quality of the product and by statically controlling the supplies and production, the quality and formalization and standardization of processes. This CSF is linked to the crucial KPIs of floor managers, being: safety stock reporting, primary item inventory levels, capacity, cycle time and manufacturing throughput.
The last CSF mentioned by Quesada & Gazo (2007) is innovation. It focuses on presenting new products, design quality, and improving production and management techniques, as well as providing products of premium value for the customer. In this case, innovation is associated to new management software implemented on the existing ERP system. The previous CSFs are important and the initial phase is thinking of how and what features should have the management system to support them.
All the information implicitly described in this section is shortly depicted in the table 2.1.

2.4 WfMS alternatives and selection

As also discussed above, business process models help the economic organization management in taking adequate decisions in important problems of the organization life, with impact in generating the income in order to be in accordance with stakeholders expectations. The first step in achieving this goal is to use an adequate business process modeling language and management system to represent their business processes.

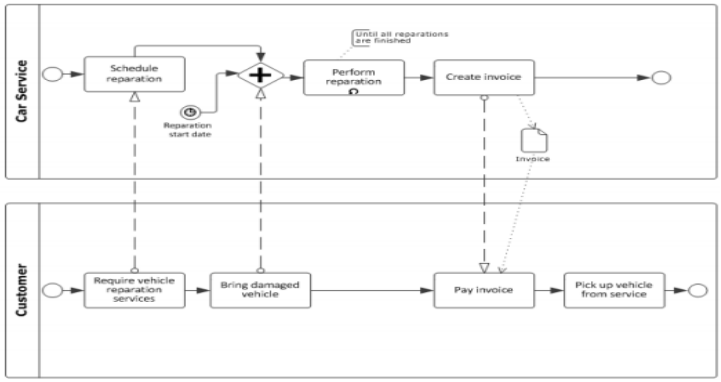
CFS and Goals	Management system requirements
Decrease project costs	Convenient price and implementation costs, Support and reliability of the tool
Decrease project duration	Same as above
Get rid of inefficiencies (flexibility, fast and secure processes	Easy to understand models (Visual Process Diagramming Tool), easy to design models (drag and drop from designer), dynamic analysis of the business processes, ability for transition, ability to represent all the components of an ERP System
Standardize the processes	Easy to understand models (Visual Process Diagramming Tool), easy to design models (drag and drop form designer), dynamic analysis of the business processes
Get an insight into business processes	Same as above
Increase of the culture of the processes	Ability for transition, ability to represent all the components of an ERP system, dynamic analysis of the business processes
Extended IT support	Ability for transition
Maximize process consistency	Chain Forms support for modeling
Get an insight into the business rules	Support business rule management

Table 2.1: BPM tool Requirements based on Business CFS and Goals

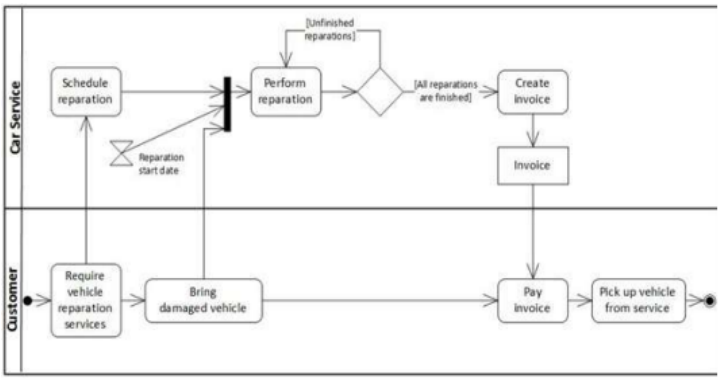
2.4.1 Modeling Language

During the last decade, a new generation of languages and tools for process modeling have been developed. Existing languages have been also enhanced, such as Unified Modeling Language (UML) 2.0 Activity Diagram, whereas new ones like Business Process Modeling Notation (BPMN) 2.0 and Business Process Execution Language (BPEL) have experienced rapid taken-up (Geambaşu 2012). All of three mentioned languages have a common focus on providing an understandable and integrated notion for business process modelling. But, they operate at different levels:

" *UML AD and BPMN are graphical and informal notations targeted at analysts while BPEL is a textual and executable language targeted at application developers (Wohed et al. 2006)*"



(a) BPMN



(b) UML.

Figure 2.3: Representation of a business process using *BPMN 2.0* and *UML AD 2.1.4* (source:(Geambaşu 2012)

Based on the capacity of being readily understandable and adequacy of the graphical elements to present the real business processes, both business process modeling languages, BPMN and UML AD, provide similar solutions for most of the patterns. The graphical symbols used for the representation of most parts of the process are similar in BPMN and UML AD, as can be also noticed by the graphical symbols used for the representation of the business process described in the figure 2.3.

BPMN current normative document includes a mapping of a subset of BPMN to WS-BPEL, while UML AD normative document does not define mapping to any BPEL(Geambaşu 2012). BPMN is capable in modelling private processes, public processes and collaboration processes at different levels of granularity. Most BPMN models can be mapped to execution code like BPEL which is its main strength over UML AD (Ko et al. 2009). Considering also that a large variety of WfMS providers support BPMN, it is a better fit for the aim of this thesis.

2.4.2 Workflow Management System

Some of used variants in the big market of WfMS (*BPMN Tool Matrix* 2018) are Camunda[1], Signavio[2] and Bizagi[3]. I picked them up, focusing on the latest release year, as a first parameter. To choose one of them is difficult without looking into their features and then select the most adequate one. In the following part I will provide a short introduction to the main characteristics of each of the previously named tools.

To get a general description of each of the tools, the fist source to check was their website. On Camunda official website, it is defined as being a free desktop application for modeling BPMN workflows and DMN decisions. Camunda Modeler supports BPMN 2.0, CMMN 1.1 (Case Management Model and Notation) and DMN 1 (Decision Model and Notation). Bizagi is free and described on its website as a cloud collaboration, speed and powerful drag-and-drop design tools bridge the gap between business and IT, engaging everyone in the process discussion from the start. Whereas on Signavio website, Signavio Process Manager is a proprietary tool called as the first web-based tool for modeling business processes. Considering the previously shown features of each of the tools, Camunda has an advantage because it is an open source tool, meaning that there are no worries about feature customization issues. And differently from Bizagi, which is also open source, it does not really have performance issues (*IT Central Station* 2018).

[1] `https://camunda.com/`
[2] `https://www.signavio.com/`
[3] `https://www.bizagi.com/`

Chapter 3

Business Rule Extraction from the source code

This chapter will deal with some approaches on how to identify and extract the business rules from the source code of any application written in Java programming language and specifically for SHD ECORO as an ERP system. The methodology, the results from the practical application in SHD ECORO and the limitations of the approaches are covered in this section.

3.1 General overview of Business Rule Extraction approaches

According to Ross (2003), business rule documentation is very important for at least two reasons. First, business rules make the business more flexible in front of challenges. They play a crucial role in strategizing, that is, ease the rethinking of the business idea and the finding of full and optimal solutions for the problems. Second, business rules keep track of business logic during redevelopment efforts. Thereby, as an integral part of a software modernization project, it is necessary and common to discover the business

rules. There are some possibilities to proceed in this step, either use the process model discovery, which is based on process mining from log data (Jadhav 2011, Vercruysse 2018, De Weerdt 2012) or identify and extract the implemented business rules into the code. In this part of the thesis, the source code of SHD ECORO is the main input information to find out some business rules.

Business rules are usually not implemented as an identifiable component of a system. There are times when business rules are embedded into source code of business applications and due to the complexity of the code, it is a herculean task to manually search for them (Paknikar Shantanu 2014). In this case a semi or totally automatic approach is needed for the business rule extraction from such business applications. It becomes very difficult when the whole software either is developed upon a legacy code or is linked to other legacy applications. A legacy code is a code that has had a long life cycle and often is built with outdated technologies and paradigms. It consists of large number of lines of code (Bisbal et al. 1999). Since SHD ECORO source code is very large, it requires an overload of work and time to manually check for the business logic in the code. Therefore, semi or totally automatic extraction frameworks would be useful in this step. Previous works (Huang et al. 1996, Sneed & Erdos 1996, Sneed 2001, LLC 2017) focus more on code written in sequential programming languages like COBOL or D*Code using code slicing, but few research deal with Java code (Paknikar Shantanu 2014, Cosentino et al. 2012). In the following sections will be presented some approaches that could be used for extracting the business rules from Java code. Firstly, it is shown the implementation, results and limitations of a tool named JBrex. Afterwards, based on the JBrex output and also the conditional nature of business rules, it is applied a new approach in which all the if-statements are extracted. Considering the limitations of this approach as well as the characteristics of SHD ECORO, the research is extended to another approach. It has the same baseline as the second one in which the rules are searched into if statements, but is applied using another logic that considers the place in the software where the business rules can be implemented. Lastly, usability and limitations of the approaches are summarized in a table where the differences are also emphasized.

3.2 First Approach: JBREX Tool

There is a model driven framework that deals with Java applications, called JBrex (Java Business rule extraction). It allows working on an abstract homogeneous representation of the system that avoids technological problems, meaning that the code is taken as a written sequence and it is not needed to have a running program, e.g. existing errors regarding importing of packages etc. (Cosentino et al. 2012). JBrex is based on three activities.
First is the variable classification that aims to reduce the number of variables to analyze and also finding variables that represent domain/business concepts and hint at business rules. Second comes the business rule identification that aims to find the chunks of code that are relevant to the domain variables, by slicing the code. According to this step, a set of chunks related to the same variable conform to a business rule. Last is executed the business rule representation that consists of presenting the extracted business rules through artifacts (graph, text, etc.) to enable human comprehension.

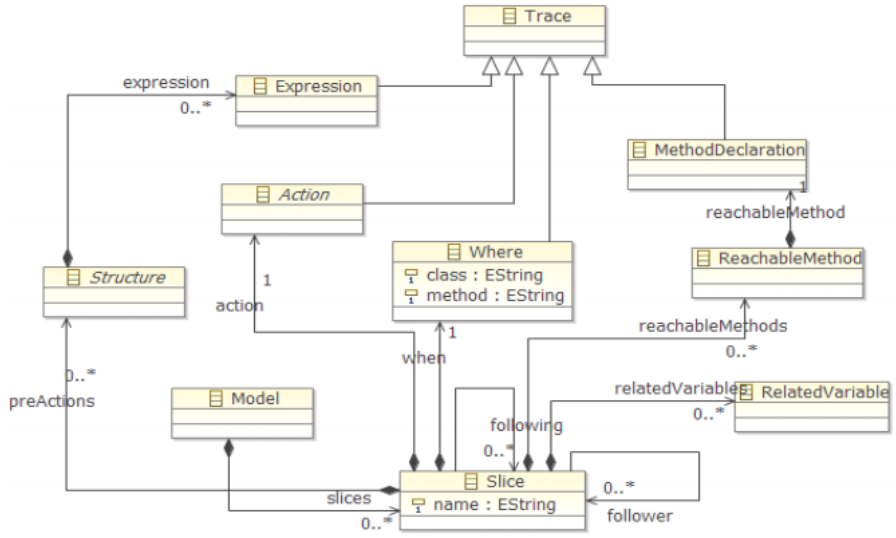


Figure 3.1: JBrex Business Rule metamodel (source:(Cosentino et al. 2012)

Figure 3.1 shows the output model of business rule extraction results of JBrex tool. The *slice* represents the chunks of the code and it has an *action* which is the rule statement be-

ing either a method invocation, an assignment, an object creation statement or a variable declaration. In the list of *pre-actions* are stored the structures of the rules which can be a loop statement, a variable declaration or an if statement and can have zero or more expressions. There is also the *where* entity in which is saved the class and method names from which the Slice has been extracted. For each *slice* it is added the *related variables* and *reachable methods* that are respectively the variables and methods invoked in *structures* and *actions* (Cosentino et al. 2012).

At first sight it seems to be an easy and effortless tool to be used, but it is necessary to first create an impression of the results it gives and then check closely whether it works for SHD ECORO or not. For this purpose, the tool was used on a small project and the results are shown for further analysis. The chosen projects is developed by the trainees in SHD for learning reasons and manages the learning objectives that are created from the training framework plan for each trainee in several departments. A learning target sheet per department is used to record what learning objectives one trainee has learned. The project is based on UI logic but is well structured and could give understandable output.

To proceed with the tool, Eclipse is needed for modeling and some packages like MoDisco and ATL (Active Template Library) should be installed. It should also be mentioned that the last version of eclipse, Photon, does not support some previous packages of MoDisco. Therefore, to avoid errors into JBrex, releases for Luna version should be installed.

3.2.1 Implementation and Results

As a first step of this approach, some variables are to be previously selected from the project. For the chosen project, the variable **bereich** (department) was picked up by adding **SELECT-VARIABLE**, one line before the declared variable into the code. JBREX tool provides two types of rule presentation:

1. **A text file with all the rules, written like the following output.**

   Output:

   ruleId:1 - variable: `bereich`

———————-

(B) rule part - granularity:1

class: `BogenErfassungCtrl`

method: `onBereichChange()`

```
if(view.getcBoxBereich().getSelectedItem().isPresent())
model.setBereich(view.getcBoxBereich().getSelectedItem().get())
```

———————-

(A)rule part - granularity:0

class: `CoachingbogenModel`

method: `setBereich(BereichEntity bereich) notfound = bereich`

———————-

Granularity expresses the distance between the statements X that modify SV (sliced-variables) and the statements that allow to reach X. It orders the processes to happen in a logical sequence. In this case:

@related: `setBereich(BereichEntity bereich), onBereichChange()`

@sliced-variable: `bereich`

In one of related methods is the context and the business rule.

```
onBereichChange()
{if (view.getcBoxBereich().getSelectedItem().isPresent())
```

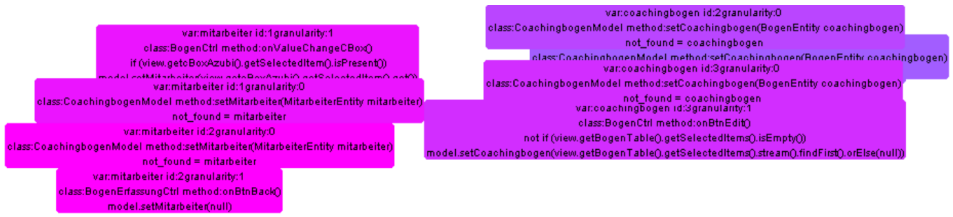
@context

```
{ model.setBereich(view.getcBoxBereich().getSelectedItem().get())
```

@rule }}

2. **Two graphml files.**

(a) ruleConnectionGraph



(b) rulePartConnectionGraph

Figure 3.2: Representation of rules extracted from JBREX (source:JBREX Output

- Figure 3.2a represents ruleConnectionGraph, where the nodes are the rules, the text inside is the code composing the rule and the edges are the connections between rules
- Figure 3.2b represents rulePartConnectionGraph, where the nodes are the chunks composing a rule and the chunks that belong to the same rule are colored with the same color.

As it is expressed above, the rule is hidden in one of the related methods. In general, this approach is based on method dependency of related business terms that in fact correspond to any of the variables in the code. Sliced variables, in this context are the domain variables which express the business terms. Identifying them is a significant step in extracting the business rules from the source code.
Afterwards, the tool was used also for SHD ECORO in an 8GB RAM System, where the pre-selected domain variable was ***Skonto*** into **Verkauf** module. The result was not shown, due to unsupported size of the project.

3.2.2 Limitations

The first challenge is selecting the variable to be sliced, known as domain variables that express the business terms. One of the solutions here can be identifying them from database tables. Column names in database tables can be a good proxy for the business terms (Chaparro et al. 2012). A short example of how the tables and column names can help in finding some business terms is depicted in the figure 3.3. Here the **Sale contract (kaufvertrag)** is related to **Discount (Rabbat), Addresses (Adresse)** and **Tours (Tour)**. All of them can be considered as relevant business terms. Moreover, in each of the respective tables can be extracted other business terms just by looking at the column names, such as *invoice (rechnung)*, *delivery note (lieferschein)* from **Tours (Tour)**, *pickup (abholung)* and *deposit (anzahlung)* from **Sale Contract (Kaufvertrag)**, *discount amount (nachlassBetrag)* and *currency (waehrung)* from **Discount (Rabatt)**, *advertising area (werbegebiet)* from **Address (Adresse)**.

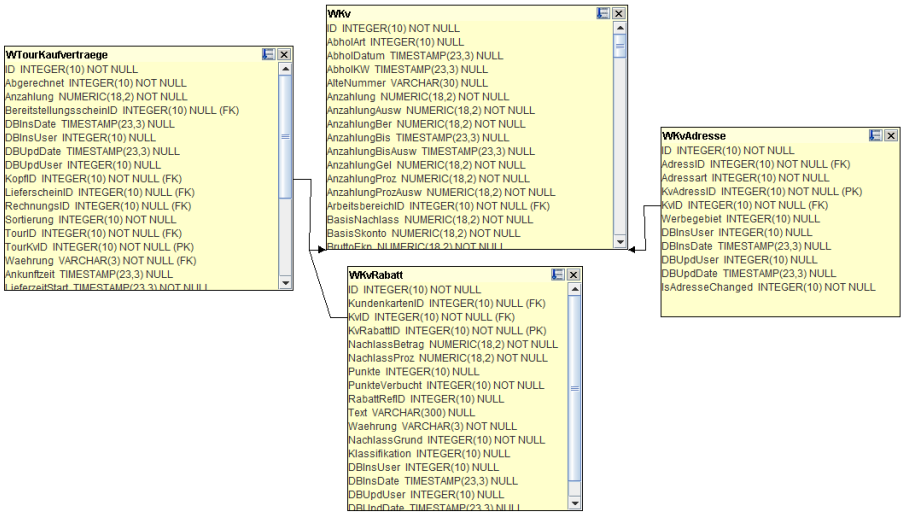


Figure 3.3: Database Tables and Entity relation to Business Terms

Secondly, considering SHD ECORO as an old system dating since 90s, the variable names could defer largely from each other and also from table entities or table names. However, to check whether a variable x can be a possible substitute for business vocabulary y, we can use the Levenshtein distance algorithm (Paknikar Shantanu 2014). Moreover, in the entity classes is the mapping that relates the names of the table fields with the used variables and it is a good clue for this issue. But, managing them is still a difficult problem, since there could be thousands of domain variables. Furthermore, the tool does not provide results for large projects like SHD ECORO, which has around 3.2 million lines of code. It works for small projects like the one used in this chapter. In the research conducted by Pizzoleto & do Prado (2016), Jbrex gave good results for a systems written in around 60K lines of code. But, for real world systems where the number of lines is extremely large it does not derive expected output. Lastly, for large projects, the representation of rules does not provide guidance in the understanding, because the size of the code is very large. Therefore, the results would need additional manual work to find out the rules among all the provided data.

3.2.3 Usability and Conclusions

JBrex is a tool suitable to extract the business rules from the source code of a project written in Java programming language. It is a model driven framework that works on abstract representation of a system. The main activities of the tool consist of slicing the code based on some pre-defined variables which represent the business terms and are referred as domain variables in this thesis and afterwards representing the business rules through artifacts such as text and graph for a better understanding. The text representation of the results contains the location of the rules in class and method level, the related variables and reachable methods that are respectively the variables and methods included in the actions. An action is a part of the code that contains the rule statement that is expressed into the context and the applied rule. Whereas, the graphs represent an overall view, where the sliced parts of the code are invoked in a separate boxes into the graph and arrows between them display the connections between rules. The parts can reflect either a complete rule or a part of it, where all the parts corresponding to the same rule have the same color. This approach has also some general limitations.

Firstly, it is not a fully automated solution, because the domain variables should be pre-selected manually. In this step come up two problems related to how to find the variables that represent the main aspects of the business and how to deal with the differences between variable names. For instance, a possible solution is using the database tables and entity names as a good proxy to business terms. Moreover mapping helps in relating the entity names and variables used for them in the code. But, the names of variables that represent the same concept can differ in an old system like SHD ECORO. In this case can be used the Levenshtein distance algorithm, which discovers strongly related variables. Nevertheless, in large projects it is hard to manage the domain variables.

Secondly, the project size affects extremely the performance of the JBrex tool. For very large projects that have millions of lines of code like SHD ECORO, it gives no results. But, for other projects that are relatively large and JBrex can perform on them, the output it provides cannot be managed or understood at fullest. Therefore in this case some additional work could be needed as well. As a conclusion, JBrex can be used for small

projects in small experiments, but it is not a good approach for big systems in the real world. In this case can be undertaken other approaches such as parsing the source code to find restrictions related to a possible business rule, which will be detailed in the following section.

3.3 Second Approach: If-Statement extraction

From JBrex tool results, it is quite obvious that the business rule is just in a small part of any of the retrieved methods and its logic is enclosed in an if-statement structure. Based also on the fact that Business rules are identified in the form of action/consequence (if/then)(Herbert Gomez Tobon 2010, Sneed & Erdos 1996) from the syntactic and semantic information of every sentence, one working approach would be to retrieve all if-statements from the source code. Same idea was also conducted in the paper by Paknikar Shantanu (2014), where a step by step approach is presented in the figure 3.4 and explained as follows:

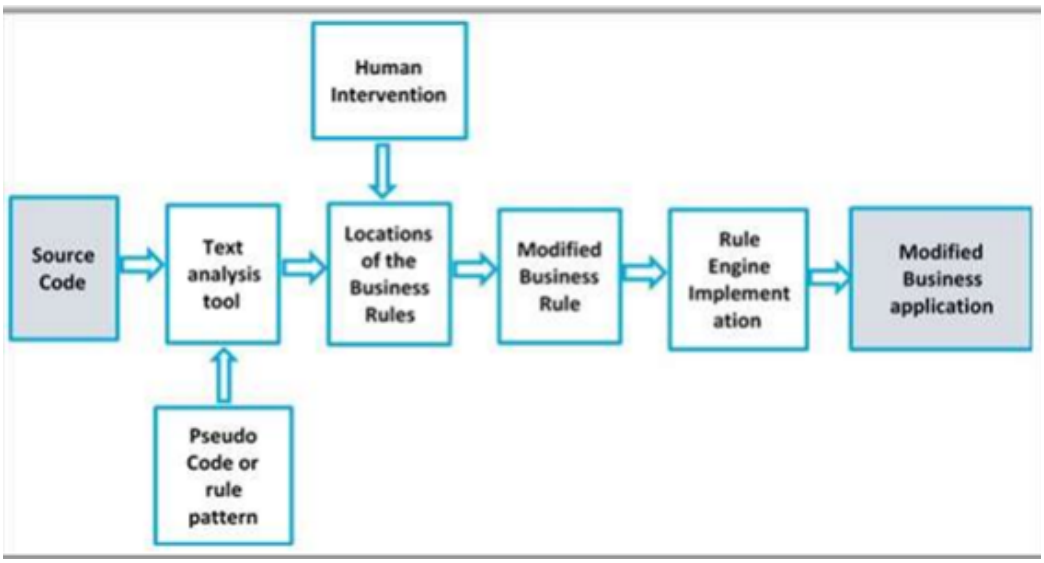


Figure 3.4: Business Rule Extraction Approach (source:(Paknikar Shantanu 2014)

1. BR (Business Rule) identification is carried out through the automatic search of the

if/then form in every sentence.

2. The code is formatted into semi-regular expressions and filtered to get rid of any contradictions or overlaps. Furthermore, if else block having similarity in logical operand will be grouped by applying cluster analysis among all the existing if else code blocks.

3. To find out whether the extracted code block represents the business correctly, business analyst should be consulted.

4. If the previous step is completed, then the code will be implemented in some rule engine for future changes control.

In SHD ECORO software source code, the amount of retrieved if-statements is very large. It consists in approximately 101159 ones. However, reducing by filtering, based on a logical and automatic way, would help to narrow the scope of the data.

3.3.1 Implementation and Results

The previous steps are a baseline for rule extracting experiment conducted on SHD ECORO. Actually, the specific undertaken actions are to retrieve and reduce the if-statements, to find overplappings, to find contradictions and finally to present the data to a business analyst.

First the source code is parsed using JavaParser to retrieve the if-statements. Often the applications have parts of the code that do not contain any business logic. In general, those chunks of code are related to how the application interacts with the users, what platform error handling it performs, what activities it keeps track of, etc. According to LLC (2017), some code artifacts that are not considered as business rules include:

- User interface design (e.g., the look and feel of the GUI)
- Database interfacing and error handling
- System and activity monitoring and logging

- Output formatting and printing
- Input data parsing
- Null constrains (Chaparro et al. 2012)

In the case of SHD ECORO, some other packages suggested from software developers as not containing any business logic, are added out manually. To reduce the scope, all the above code artifacts are taken into consideration and based on them is written a code to filter if-statements. Additionally, since some projects are very large, it is sufficient to look into their most important modules; such as ***Purchase, Sale, Settings, Logistics***.

Module name	Number of filtered if-statements	Percentage of total retrieved data
Purchase (Einkauf)	3660	37%
Sale (Verkauf)	3374	34%
Settings (Vorgaben)	861	8.7%
Logistic (Logistik)	1994	20.3%

Table 3.1: Total parsed if-statements for main modules in SHD ECORO

Based on the results shown in table 3.1, there are 9884 if statements retrieved, where the highest part of them comes from the Sale and Purchase modules. Whereas the number of excluded ones is 90544 and they are distributed as shown in table 3.2. It is noticed that the number of filtered out statements, based on the general conditions is considerably low, just 10% of total eliminated data, whereas manual exclusion takes the highest part of them. Thereby, we can say that an important input in this method is the help of software developers and software architects. The amount of the retrieved if statement is still very large to be afterwards processed manually. Hence, here will be added another constrain. For each main module of SHD ECORO just the if-statements that contain domain variables either in their condition or in their body will be considered for further analyses. To find out some of the domain variables, the tables as well as their relations into database, are consulted.

Classification	Nrumber of excluded data	Percentage of total excluded data
if statements having null checker conditions	5624	6.2%
if statements related to database and error handling	1156	1.2%
if statements related to output formatting, printing, input data parsing and translation	1894	2%
if statements related to system and activity monitoring and logging	1374	1.5%
packages that are added out manually because do not implement any business logic	80496	88.1%

Table 3.2: Excluded if-statements

Considering that Sale and Purchase modules are the ones that contain the largest number of retrieved if statements and the central item of both of them is the sales Contract (Kaufvertrag), it is searched into the KV (Kaufvertrag) table and its relations in the database to find the domain variables. In this case the domain variables are the table names that represent the business terms.

Some of the most dominant domain variables are as follows: ***nachlass, skonto, kv, vk, betrag, order, gutschein, saldo, abhol, material, filial, buch, zahl, proz, punkt, rabat, rechn, brutto, netto, steuer, angebot, chance, liefer.***

Module name	Before domain variable filtering	After domain variable filtering
Purchase (Einkauf)	2564	1488
Sale (Verkauf)	2335	1646
Requirement (Vorgaben)	763	392
Logistic (Logistik)	1585	1258

Table 3.3: Total domain variable filtered if-statements for each module in SHD ECORO

From the table 3.3, it is noticed that the filtering based on the pre-selected domain variables has narrowed considerably the amount of data for purchase, sale and setting module, but not for logistic one. It can be assumed that logistic module is mostly related to the

domain variables. However, even for other modules, the difference is not considered to be crucial, because the general goal is to go very near to the basis of the business logic using those business terms expressed by the domain variables. Other aspects of the business are of a minor importance.

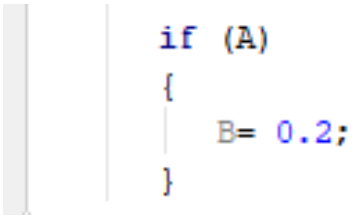
Second, the data generated after filtering the if-statements can be redundant due to overlappings, which are duplicated if-statements; meaning those which have the same condition. There exist duplicate conditions within a class that generally have the same implementation. But, in this case, it is interesting to see the statements that are implemented in more than one class, because the probability to find contradictions there is higher, due to diverse place, diverse logic or maybe diverse developer. To do so, all the if-statements that do not occur only in one class are extracted in a text file to be further examined. In the table 3.4 is displayed specific information about the duplicates that occur in more than one class. Respectively, the first column shows the total number of duplicated if statements, whereas the second column gives the duplicates that occur in more than one class. The duplicates can be implemented equally, which are situations when the block of if statement is exactly the same. But, we are not interested in those cases, because most probably they do not contain any contradiction. Therefore, in the third column are represented the number of those if-statements that even if they have the same condition, the implementation of the body is different. Here should be mentioned that because of the automatic extraction of same implemented body of the if statements, those having an equivalent implementation are considered being different as well. As can also been noticed from the values, the duplicates within a class compose more than 50% of the total number of duplicates. Nevertheless, due to their lack of information in this case, this large difference is considered to be minor. The so-called duplicates are deleted from the file that contains all the if statements, because they should be firstly filtered out of contradictions. After the examination, they will be added again into the total if-statements.

Third, apart from the redundancy of the results another problem that can be faced is related to the integrity of the data, more specifically is the existence of contradictions. In this context, contradictions are considered to be the implementations of the same if-

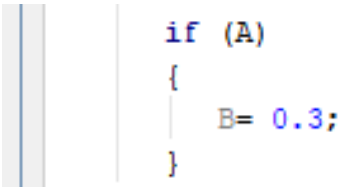
Module name	Total Duplicates	Multiple duplicates	Single implementation
Purchase (Einkauf)	1096	267	114
Sale (Verkauf)	1039	501	161
Requirement (Vorgaben)	98	5	3
Logistic (Logistik)	409	116	52

Table 3.4: Number of overlapping-s for each module in SHD ECORO

condition in different ways. For instance, the following cases conflict each other. But



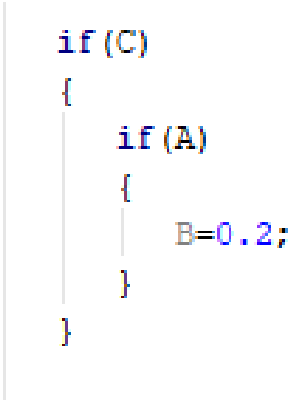
(a) Case 1



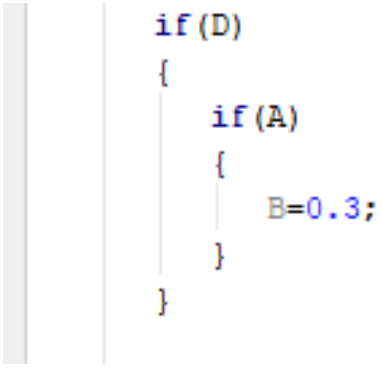
(b) Case 2

Figure 3.5: Example of a contradiction (source:self-made)

those kind of implementations are not always a contradiction. For blurry cases (e.g. an upper restriction exists, thus it controls the nested if statements like in the figure 3.5), the overall conditional block of code where the if-statement takes place, should be examined.



(a) Case 3



(b) Case 4

Figure 3.6: Example of a contradiction (source:self-made)

However, this is not a problem here, because the parser code gives every level of if-statement implementation. From the previous step, all the duplicates will be saved into a text file where those having the same condition will be grouped together automatically. This approach provides a better overview of them and makes it easier to manually analyze for contradictions. Here, the duplicates that have the same implementation block as well as those which do not contain any business logic will be deleted. After applying this manual control for SHD ECORO, 217 if-statements remained from all the modules, which will be examined later for contradictions. Lastly, the part of the code that does not contain any contraction will be added into the text file which has all the if-statement that were retrieved before. From the manual check resulted that there are no contradictions into the source code of SHD ECORO. Hence, all duplicate if-statements will be considered as additional data into the file that have all the data.

Fourth, after having all the extracted data, it is needed to consult any software or business analyst to check whether any of the found code block correctly represents the business (Paknikar Shantanu 2014) or contain business logic. The work that is done in this step is all manual, meaning that all the if statements are examined by going one by one into them. In around 1000 if-statements retrieved from the **sale** module of SHD ECORO, just around 20 of them resulted to have some business logic. Thus, from this approach concluded that just 2% of the total released data are valuable. Due to the large difference between the effort put on the implementation of this method and the amount of estimable output, this approach was not applied for other modules. Nevertheless, for a proof of concept, one module is enough to deliver results and limitations of the method. An example of how the result look like is shown in figure 3.7.

3.3.2 Limitations

In this approach is needed to have information about the way the software is developed, which is a crucial input for the step where the data is filtered. However, being a manual process, the probability of making any mistake is high. Thereby, the overall method results are questionable.

```
Class name: /ecoro-eh/wawi-ri/src/main/java/de/shd/wawi/ri/verkauf/model/Verkaufsposition.java
if (isAuslandsteuerRelevant()) {
    // wenn ein Defaultsteuersatz Ausland eingestellt wurde
    verkaufspreis.setMehrwertsteuersatz(VerkaufUtil.buildSteuersatzCont(verkaufskopf.mwStAusland));
} else {
    // Steuerpflichtig
    verkaufspreis.setMehrwertsteuersatz(vkpArtikel.getMehrwertsteuersatz());
}
```

Figure 3.7: Business Rule example from second approach.

In addition, the total retrieved data is very large in number, thus it could be expected to find some business rule candidates. However, the extracted business rules are significantly less than the total data.

A large part of the SHD ECORO business logic is written as SQL statements, e.g. searching based on logical restrictions. All this logic is not considered using this approach, since all the packages dealing with database are added out. Moreover, the constraints that are formulated either in the database or in the entity layer are not considered, even though they are important for SHD ECORO logic implementation.

The poor representation of variables brings problems in understanding fragments of code as well as difficulties in translating the code into valuable business rule.

Furthermore, the way of how a large part of the code in the considered packages is developed makes it difficult to find any business logic implementation in either the if-statement block, or in its else-block. For examples, there are methods that develop a logic, but the if statement is used just to check a restriction and its block contains either return or continue. Thereby, all the needed rule is out of the if-statement body.

3.3.3 Usability and Conclusions

The method of extracting business rules from the if statements invoked in the source code is based on the conditional nature of the rules. In general the restriction rules are im-

plemented as conditions that if some action happens then it is followed by some other operations. This approach is implemented by developing a script to parse the java source code. The first faced issue related to the size of the project is the large amount of if-statement retrieved from SHD ECORO. The solution to it is filtering based on a logical argumentation. According to some previous research, in a project is no business logic implemented in user interface design, database interfacing, error handling, system and activity monitoring and logging, output formatting and printing, input data parsing and null constraints. Thereby, all the packages dealing with them should be excluded. From SHD ECORO resulted that even after filtering the number of extracted statements was very large. Hence, some other artifacts are needed to reduce the scope even more. With the help of developers of the software, it is possible to add out some packages that according to them do not contain business logic. Furthermore, the data can be also filter from overlapping-s by adding out the statements having the same condition and implementation inside the body.

After filtering, a crucial part of the approach is checking for the compliance of the rules in a general level. In this step all the if statements that occur in different classes and have some condition but diverse implementation are grouped by the value of the condition and then are checked manually for existing contradictions. Afterwards, the filtered data are presented to either business analysts, software developers or software architects.

Since the approach requires manual work both in the filtering and analyzing phase, a common problem for every project is the high probability of the mistake which makes the results questionable. Moreover, for large and old projects like SHD ECORO, the poor representation of variables brings problems in understanding fragments of code as well as difficulties in translating the code into valuable business rule.

In the case of SHD ECORO, the filtering artifact that are used for this method do not work, because the logic written as SQL statements (e.g. searching based on logical restrictions) as well as the constraints that are formulated either in the database or in the entity layer are not considered even though they are crucial for SHD ECORO logic implementation.

This approach is not very effective for SHD ECORO and systems that are similarly im-

plemented. The success of this method strongly depends on the the development logic of software as well as on its size. Therefore, another approach will be presented in the next chapter, which will focus more on the specific features of SHD ECORO.

3.4 Third Approach: If-Statement extraction from database layer implementation.

This method of rule extraction from source code is more a derivation from the second approach discussed in this thesis. Considering its limitations, the following cases are extracted and used as baseline for this approach.

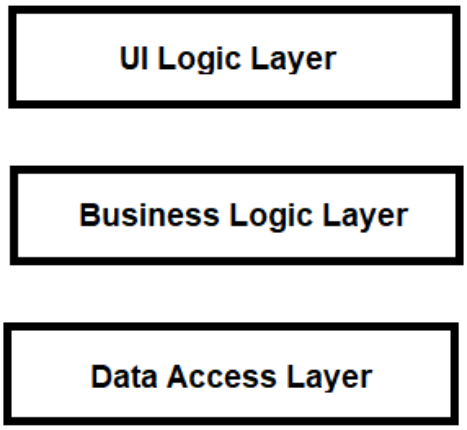


Figure 3.8: SHD ECORO Architecture

1. As an ERP system, the behavioral business rules are implemented using configurable variables. Therefore, the nature of this category of rules is not automable at all into the business layer.

2. As also specified in one of the limitations of the second approach, SHD ECORO business logic is written as SQL statements. According also to Szirbik (2005), business logic in ERP systems usually are formed as constraints on databases, with

additional computation, when appropriate. Thus, it is expected to find some business rules, either behavioral or structural ones, by searching into the data access layer implementation.

Moreover, in this rule extraction application will be included the if constraints that contain *throw* exception. Such an exception, occurs during the execution of a program, that disrupts the normal flow of the program's instructions. So, when it happens inside an if statement block of code, it notifies that either the implementation logic or application logic is affected. We refer to implementation logic as the programming logic, e.g. the type of data saved into a specific field of a table into database. Whereas the application logic, is the logic of the software, that is the business logic.

3.4.1 Implementation and Results

The implementation phases are similar as in the previous method of business rule extraction, but since the used filters as well as the results are not the same, all the steps will be performed again.

A script in Java is written to retrieve the if-statements from the code by adding out:

- User interface design (e.g., the look and feel of the GUI)
- System and activity monitoring and logging
- Output formatting and printing
- Input data parsing
- Null constrains

As also argued in the description of this method, different from the previous approach, the database interfacing and error handling parts of the code will be included. However, the same packages, suggested from software developers as not containing any business logic, are added out manually as well. The important modules remain the same, namely: sale, purchase, requirement and logistic. But, here will be parsed just the data access layer of

the application, which in SHD ECORO is included in the /bo/ packages. Additionally, in this method will be added also the if statements that contain *throw* exceptions in their body.

The approach continues with finding the overlappings. As it is also explicitly explained in the same step of the second approach, the overlaps are the if statements which have the same condition. But, it is the information extracted from those that happen in more than one class that can help to track contradictions. Hence, the parser is used to find duplicates in multiple classes and filter just the ones having different implementation. The results from parsing the code are shown in the table 3.5.
The trend of the data changes a bit from the results of the second method. Here, the ***Logistic*** and ***Purchase*** modules have the largest amount of if statements. It will be considered the Sale module again for further examination, because it is needed to work on the same environment for later comparisons between the approaches.

Module name	Total if statements	Total Duplicates	Multiple duplicates	Single implementation
Purchase (Einkauf)	1007	283	175	79
Sale (Verkauf)	677	142	90	40
Requirement (Vorgaben)	224	25	2	1
Logistic (Logistik)	1191	210	135	57

Table 3.5: Results for each module in SHD ECORO

In order to find contradictions (explained in previous section), there will be considered from the previous duplicate's samples only those that are implemented in different classes, have the same condition and have different implementations of the body. This corpus of data is then checked manually for duplicates. In this case the results showed no existence of contradictions in the source code of SHD ECORO. Hence, all duplicated if-statements will be considered as additional data into the file that have all the data.

Finally, around 700 if statements where delivered to be further analyzed manually. There resulted that around 90 constraints contain any business logic enclosed into their body implementation. Thereby, this method provides as valuable data just 12% of the

overall input ones. In the results dominate the structural rules, which can represent a classification or a computation. An example of discovered rules is depict in the following figures.

```
Class name: /ecoro-eh/wawi-bo/src/main/java/de/shd/wawi/bo/verkauf/kaufvertrag/bewprot/ChefInfoBewPflegeListener.java
// wenn KOMPLETTSTORNO, dann die Anzahlung auch negieren !!
if (isKomplettstorno(kv)) {
    beruecksichtigteAnzahlung = -(Double) updateEvent.getOldValue("anzahlung");
    insertSql.setDouble(7, beruecksichtigteAnzahlung * verkaeuferValue.getAnteil() / 100.00);
} else {
    beruecksichtigteAnzahlung = (Double) updateEvent.getNewValue("anzahlung") - (Double) updateEvent.getOldValue("anzahlung");
    insertSql.setDouble(7, beruecksichtigteAnzahlung * verkaeuferValue.getAnteil() / 100.00);
}
```

(a) Computation Rule

```
Class name: /ecoro-eh/wawi-bo/src/main/java/de/shd/wawi/bo/verkauf/kaufvertrag/KvSchnittstelleSessionImpl.java
if (isRueckbuchung) {
    return status == srvVorgabenModel.getTexttabelleIDFor(TextTabConstants.STATUS_KV, TextTabConstants.STATUS_KV_RUECKGEBUCHT)
     || status == srvVorgabenModel.getTexttabelleIDFor(TextTabConstants.STATUS_KV, TextTabConstants.STATUS_KV_LIEFERFAEHIG);
} else {
    return status == srvVorgabenModel.getTexttabelleIDFor(TextTabConstants.STATUS_KV, TextTabConstants.STATUS_KV_ABGERECHNET);
}
```

(b) Classification Rule

Figure 3.9: Example of Structural Rules (source:SHD ECORO parsed code)

```
Class name: /ecoro-eh/wawi-bo/src/main/java/de/shd/wawi/bo/verkauf/werbebrief/SchubigerWerbebriefSessionImpl.java
if (werbebriefKv.bruttoVkp > 1000 && werbebriefKv.isLieferKzOk) {
    kvValue.kvKZ4 = werbeBrief;
} else {
    kvValue.kvKZ4 = keinBrief;
}
```

Figure 3.10: Example of Behavioral Rule (source:SHD ECORO parsed code)

3.4.2 Limitations

There are found more structural rules from all the retrieved if-statements, which structure and organize basic business knowledge. Furthermore, into this set of rules dominate both the computational and classification ones. The problem here is that not all structural rules can be considered as business rules, even though they are automatable and easy to be found by automatic approaches. Thus, it is expected that the number of rules would be very low in comparison to all parsed data.

Most of the retrieved rules are implementation restrictions, that are programming logic constraints mostly related to allowed data format and data type, e.g. check before saving the data into database tables whether it is of the same required type.

Same as in the previous approach, there is a poor representation of variables. It makes the process of analyzing hard to undertake. Therefore it can bring problems in understanding fragments of code as well as difficulties in translating the code into valuable business rules.

In this approach is needed to have information about the way the software is developed. It is a crucial input for the step where the data is filtered. However, being a manual process, the probability of making any mistake is high. Thereby, the overall method results are questionable. (Also from the previous approach)

3.4.3 Usability and Conclusions

This third approach is an extension of the second one, because the basic idea is the same but it is applied including some other logic. In simple worlds, it is a method that focuses on the conditional format of the business rules and the filtering logic is modified based on the features of the system. Different from the second approach it includes if statements located inside the data access layer packages that contain SQL expressions and throw exceptions apart from user interface design, system and activity monitoring and logging, output formatting and printing, input data parsing and null constraints. Even after filtering the data for SHD ECORO, it resulted that the remain ones involve a large amount of if statements. Same as in the second approach, some other steps are undertaken in order to narrow the scope of the result. First, packages that according to developers do not contain any business logic are added out and are parsed if-statements that include some predefined domain variables either in their condition or their body. Moreover, the duplicates are deleted and the ones that have same condition but different implementation and also occur in diverse classes are further examined for possible contradictions. Afterwards, the results are manually checked for business rules by either business analysts, software developers or software architects.

Same as in the second approach, being a manual process, the probability of making any mistake is high. Thereby, the overall method results are questionable. Additionally, the poor representation of variables brings problems in understanding fragments of code as well as difficulties in translating the code into valuable business rules. In this method the results involve more structural rules in form of computational and structural ones, which structure and organize basic business knowledge. The problem that is raised in this method is that the number of rules could be very low in comparison to all parsed data because not all structural rules can be considered as business rules. Furthermore, most of the retrieved rules are implementation restrictions, that are programming logic constraints mostly related to allowed data format and data type, e.g. check before saving the data into database tables whether it is of the same required type. For SHD ECORO this approach was more effective than the second one.

3.5 Conclusion

A summary of the approaches set up, results and limitations is presented in the table 3.6.

	First Approach	**Second Approach**	**Third Approach**
Method	It is a model driven framework that deals with Java applications and based on some pre selected domain variables generates the chunks of code related to those variables which represent the rules. The output is in text and graph form.	It is an approach of parsing the source code and extracting business rules from the if statements invoked in the business layer of the software based on the conditional nature of the rules.	Same as the second approach but here are parsed also the if statements located inside the data access layer packages that contain SQL expressions.
Results	No results for SHD ECORO	From 1000 extracted if-statements just around 20 (2 %) of them resulted to have some business logic.	From 700 extracted if-statements just around 90 (12%) of them contain any business logic into their body implementation.
Limitations	Difficulties in identifying the domain variables. It does not provide results for large projects like SHD ECORO, which has around 3.2 million lines of code.	Very large scope that resulted in a large amount of if statements. A considerable part of SHD ECORO business logic is written as SQL statements, which is excluded from the analyzing process. There exists a poor representation of the variables, which increases the probability of making any mistake during the required manual check.	There are retrieved more structural rules (computational and classification), most of them are implementation restrictions characterized by a poor representation of variables. It is not a fully automatic approach and during the manual process the probability of making any mistake is high.

Table 3.6: Rule extraction approaches

Chapter 4

Business process discovery

This chapter contains an extended information about the Business Process Discovery techniques and their implementation. It starts with introducing the discovery process. Afterwards, the automated technique is further described as well as its results from the implementation on ECORO are shown and interpreted. Lastly, other manual techniques are represented.

Business Process Discovery is knows as a discipline of business process management, that scientifically extracts the implemented business processes and aims to create a AS-IS baseline of them. Moreover, it helps in identifying the process hierarchy, process owners, process entities, business rules, process operations, and statuses (Jadhav 2011). According to Jadhav (2011), Vercruysse (2018), in general there exist two main techniques to proceed with business process discovery:

1. Manual technique, which includes interviews, organizational structure and generic value chain analysis, etc.

2. Automatic technique, which uses the ABPD (Automatic Business Process Discovery) tools for Knowledge Discovery through databases; transaction and log data (De Weerdt 2012).

Due to the complexity of the organization structure, the automated processes as well as the

mature enough processes, the discovery task becomes very tricky. Therefore, either one or both of the previous techniques should be used (Jadhav 2011). In the case of ECORO, being an ERP system covering some processes in the furniture industry, the goal is to generally find out some basic fully automated business processes.

4.1 Process Mining

Process mining is an approach of learning from the previous behavior. It aims to extract aggregate, high-level information about several aspects of the process from some so-known event-logs (Günther 2009), which may originate from all kinds of systems (Mans et al. 2008), using diverse mining tools and mining algorithms. Process mining is applicable to all kind of Information Systems during Process Model Discovery, since the only requirement is that the system should record the actual behavior. For instance, based on (van der Aalst et al. 2007):

1. Workflow Management Systems (WfMSs) mostly register the start and completion of activities

2. ERP systems like SAP log all transactions, e.g., users filling out forms, changing documents, etc.

3. Business-to-business (B2B) systems log that save tracks of exchange of messages with other parties

Different from other analysis techniques, process mining has the advantage to provide accurate and factual information about the process, without relying on an idealized model of reality (Günther 2009). The base of Process Mining is the set of log data, which should contain some needed information about the overall business behavior. The figure 4.1 describes the general process of discovery based on process mining. There are processes running in some system and the events happening in every step are saved in the log data. Afterwards the log data is used as an input to inference some models that should represent the behavior of the system. The following subsections will represent a detailed description

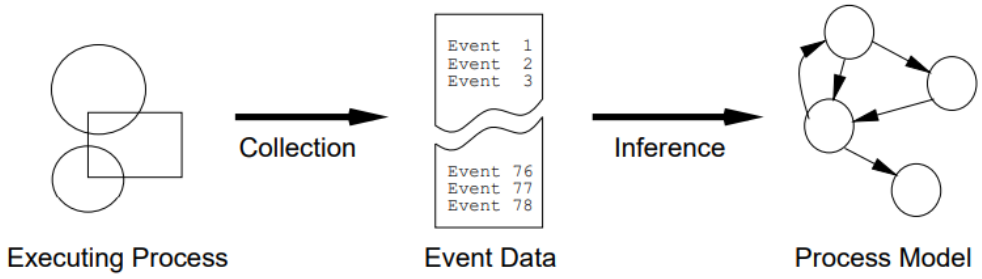


Figure 4.1: Process Mining Model Discovery (source:(Cook & Wolf 1998)

of Log data, algorithms that use them as an input as well as the most used tools, which execute the whole discovery process .

4.1.1 Log Data

The only and very important assumption of process mining technique is the possibility to gather a process log with data about the order the events take place. Regarding Cook & Wolf (1998), an event is used to characterize the dynamic behavior of a process in terms of identifiable, instantaneous actions, such as invoking a development tool or deciding upon the next activity to be performed.

In ECORO logs consist of transactions between ECORO and the Database Management System. Some part of it is shown in the figure 4.2. It will be analyzed in the following sections.

Offset	Zeit	Prozess	Typ	Transaktion in Gang	GlobalNode	Datenbank
150605448	2018-10-10 15:07:53	2056288	SET	Nein	^Protocol.REVWKundeDebBankvD	d:\db\ecoro_35_kunde\shdprotocol\
150605516	2018-10-10 15:07:53	2056288	SET	Ja	^Protocol.REVWKundeDebBankvD(373)	d:\db\ecoro_35_kunde\shdprotocol\
150605692	2018-10-10 15:07:53	2056288	BitSET	Ja	^Protocol.REVWKundeDebBankvI("DDLBEIndex",1)	d:\db\ecoro_35_kunde\shdprotocol\
150605776	2018-10-10 15:07:53	2056288	SET	Ja	^Protocol.REVWKundeDebBankvI("REVWKUNDEDEBBANKVPKey3",5211,3186124,373)	d:\db\ecoro_35_kunde\shdprotocol\
150605876	2018-10-10 15:07:53	2056288	ZKILL	Ja	^Protocol.REVWKundeDebBankvI("REVWKUNDEDEBBANKVPKey3",5211,3186124,373)	d:\db\ecoro_35_kunde\shdprotocol\
150605976	2018-10-10 15:07:53	2056288	SET	Ja	^Protocol.REVWKundeDebBankvI("DDLBEIndex",1)	d:\db\ecoro_35_kunde\shdprotocol\
150606156	2018-10-10 15:07:53	2056288	ZKILL	Ja	^Protocol.REVWKundeDebBankvD(373)	d:\db\ecoro_35_kunde\shdprotocol\
150606332	2018-10-10 15:07:53	2056288	ZKILL	Ja	^Protocol.RevisionInfoD(3186124)	d:\db\ecoro_35_kunde\shdprotocol\
150767040	2018-10-10 15:08:48	2056288	SET	Ja	^Protocol.RevisionInfoD(3186125)	d:\db\ecoro_35_kunde\shdprotocol\
150767116	2018-10-10 15:08:48	2056288	SET	Nein	^Protocol.REVWKundeDebBankvD	d:\db\ecoro_35_kunde\shdprotocol\
150767184	2018-10-10 15:08:48	2056288	SET	Ja	^Protocol.REVWKundeDebBankvD(374)	d:\db\ecoro_35_kunde\shdprotocol\
150767360	2018-10-10 15:08:48	2056288	BitSET	Ja	^Protocol.REVWKundeDebBankvI("DDLBEIndex",1)	d:\db\ecoro_35_kunde\shdprotocol\
150767444	2018-10-10 15:08:48	2056288	SET	Ja	^Protocol.REVWKundeDebBankvI("REVWKUNDEDEBBANKVPKey3",5211,3186125,374)	d:\db\ecoro_35_kunde\shdprotocol\
150767544	2018-10-10 15:08:48	2056288	ZKILL	Ja	^Protocol.REVWKundeDebBankvI("REVWKUNDEDEBBANKVPKey3",5211,3186125,374)	d:\db\ecoro_35_kunde\shdprotocol\
150767644	2018-10-10 15:08:48	2056288	SET	Ja	^Protocol.REVWKundeDebBankvI("DDLBEIndex",1)	d:\db\ecoro_35_kunde\shdprotocol\
150767824	2018-10-10 15:08:48	2056288	ZKILL	Ja	^Protocol.REVWKundeDebBankvD(374)	d:\db\ecoro_35_kunde\shdprotocol\
150768000	2018-10-10 15:08:48	2056288	ZKILL	Ja	^Protocol.RevisionInfoD(3186125)	d:\db\ecoro_35_kunde\shdprotocol\
150773480	2018-10-10 15:08:48	2055068	SET	Ja	^Protocol.RevisionInfoD(3186028)	d:\db\ecoro_35_kunde\shdprotocol\
150773556	2018-10-10 15:08:48	2055068	SET	Nein	^Protocol.REVWTelefonateD	d:\db\ecoro_35_kunde\shdprotocol\
150773620	2018-10-10 15:08:48	2055068	SET	Ja	^Protocol.REVWTelefonateD(403)	d:\db\ecoro_35_kunde\shdprotocol\
150773792	2018-10-10 15:08:48	2055068	BitSET	Ja	^Protocol.REVWTelefonateI("DDLBEIndex",1)	d:\db\ecoro_35_kunde\shdprotocol\
150774112	2018-10-10 15:08:48	2055068	SET	Ja	^Protocol.RevisionInfoD(3186029)	d:\db\ecoro_35_kunde\shdprotocol\

Figure 4.2: SHD ECORO transaction log example (source: SHD ECORO Cache System)

Log data should fulfill some pre-requirements. The events which are the smallest part of the log data:

- Should represent a task (activity), which can be for example a well-defined step in the process (Van der Aalst et al. 2004). It is not always clear what should be considered as an activity. For example, there is no column named either task or activity into ECORO transaction log. Some interpretations are needed here to come up with a reasonable representative of activities.

- Should represent a case, which can be for example a process instance (Van der Aalst et al. 2004). Every process have some instances, where every instance is considered to be a case with a corresponding case id. For instance, in ECORO transaction log, the processId can be presumed as caseId.

- Should be totally ordered, meaning that, in the log, events are recorded sequentially, even though tasks may be executed in parallel (Van der Aalst et al. 2004, van der Aalst et al. 2007).

- Can have a time-stamp (van der Aalst et al. 2007).

- Can have a performer also referred to as originator, that is the person executing or initiating the activity (van der Aalst et al. 2007).

According to van der Aalst et al. (2007), there can be distinguished three perspectives based on the fields of the log data; performer information, task-id information and case-id information. First is the organizational perspective, which focuses on *WHO* executes the process. Second comes the process perspective, which focuses on the control flow, finding existing paths of the processes. Lastly is the case perspective, which focuses on the properties of cases that can be characterized by their path in the process, by the originators working on a case as well as by the values of the corresponding data elements. For the aim of this thesis, it will be considered the process perspective to find out some work flow paths for SHD ECORO.

4.1.2 Algorithms and Tools

After defining the set of data valuable as an input in the process mining technique another crucial part of it are the *tools* used to analyze the data as well as the *algorithms* needed to mine the behaviour and provide process models. In this thesis are used two tools namely ***Disco*** and ***ProM***.

Disco is a complete process mining toolkit from Fluxicon that makes process mining fast and easy. According to Niels Lohmann (2012), the main functionality of Disco tool include:

- *Data import.* Disco opens a CSV or Excel file and configures which of the columns hold the case ID, timestamps, activity names and which of the other attributes should be also included in the analysis. Then the import can be started.

- *Automated Process Discovery.* The Disco miner is based on the Fuzzy miner, but combined with experience from practice and testing. The result is a mining algorithm that can be operated and understood efficiently by domain experts with no prior experience in process mining.

- *Process Statistics.* Disco provides detailed performance metrics about the process in terms of the number of cases and events in the data set, the time frame covered,

and performance charts like, for example, about the case duration. Other statistic views generate frequency and performance information for all activities and resources in the process. Moreover, all the additional columns information can have statistical results.

- *Variants and Individual Cases.* In this part Disco goes down to the individual case level and shows the raw data. In addition to a complete list of all cases in the data set, the user also gets direct access to the variants in the process, which are specific sequences of activities.

- *Filtering.* Disco has six powerful filter types available, respectively timeframe filter (filtering based on the time), performance filter (filtering based on a variety of different performance metrics like, for example, the case number), attribute filter (filtering by focusing on (or excluding) certain activities, resources or process categories based on data attributes), variation filter (filtering by focusing on the analysis based on either the mainstream behavior or precisely the exceptional cases by making use of the variants), endpoints filter (filtering by selecting cases based on their start and end activities) and follower filter (filtering by including a 4- Eyes filter option that can be used to check for segregation of duty violations). All of the above can be combined and stacked in any order.

- *Performance Highlighting.* Additionally to the frequency-based process map, the time that is spent in the process can be also analyzed.

- *Animation.* The process flow is visualized over time right in the discovered process map.

- *Project Management.* Disco supports project work through the management of multiple data sets in one project view.

Most of the above functionalists will be used for the data extracted from ECORO.

ProM is a powerful and the world-leading tool in the area of process mining (Van der Aalst et al. 2009). The input the log data but not in csv format. Disco helps to change

the data file format into either xes or mxml which are both valuable to ProM. According to Van Dongen et al. (2005), the main plug-in-s of this tool and their functionality are as follows:

- Mining plug-ins which implement some mining algorithm that construct a model based on some log data.
- Export plug-ins which implement some save as functionality for some objects (such as graphs).
- Import plug-ins which implement an open functionality for exported objects, e.g., load instance-EPCs from BMPN model.
- Analysis plug-ins which typically implement some property analysis on some mining result. For example, there are also analysis plug-ins to compare a log and a model.
- Conversion plug-ins which implement conversions between different data formats, e.g., from EPCs to Petri nets.

In this phase of the thesis, the tools are used to achieve the goal of discovering existing process models from the log data. Based on some research results ((Van Dongen et al. 2009), (Rozinat et al. 2007)), three of the most known mining plug-in-s in ProM are:

1. *Alpha miner* which contains alpha algorithm and the other extended ones such as alpha ++ algorithm
2. *Heuristic Miner*
3. *Inductive Miner*

All the above categories will be considered as possible mining algorithms for further usage in the mining technique applied on ECORO. According to Leemans et al. (2013), the challenge in process discovery is to find the best process model given recorded historical behaviour. Four important quality criteria that define a good model are*fitness*,

precision, *generalization* and *simplicity* But to decide which of the algorithms provide the best model, first pros and cons of them are analyzed based on previous research.

Alpha ++ algorithm is implemented in the ProM tool and its result is a model in the form of Petri nets. It is an extension of the alpha algorithm mining certain implicit dependencies, in particular non-free choice constructs (Bergenthum et al. 2008). In other words, it considers the completeness of direct succession by including long-term successions as well, meaning that if a transition can follow itself with some transition in between, then this has occurred at least once in the event log. This fact implies that the discovered model model does not allow for additional behaviour that is not described in the log, guaranteeing a high precision. Alpha ++ algorithm assumes a direct correspondence between the events-classes in the log and the transitions in the model, e.g each transition refers to a single event class. This is translated to an over-fitted model that does not generalize enough because it is fitted only to a specific log file (Van Dongen et al. 2009). Moreover, alpha++ algorithm has problems to return fitting models because it cannot correctly discover the skipping of tasks (Macintosh & Whyte 2008). Therefore, the value of fitness is very low. Furthermore, it cannot give satisfactory solutions when there exist difficult cycles (Bergenthum et al. 2008).

Heuristic algorithm is implemented in Prom as well. According to Weijters et al. (2006), the heuristics miner mines the control-flow perspective of a process model considering the order of the events within a case, where the order of events among cases is not important. Heuristic algorithms has completeness assumptions relating to the frequencies of succession. Thus, if there is a causal dependency between two transitions, then it should be reflected in the direct successions of these transitions. Furthermore, in some cases a transition does not refer to a respective event, indicating that heuristic miners provide an under-fitted model, meaning that it sacrifices precision to satisfy the criteria of generalization for a good model. Additionally, the algorithm can deal with noise and low frequent behavior. (Van Dongen et al. 2009). In practical situation, e.g. in event log with a large number of traces, low frequent behavior and some noise, the Heuristics Miner can focus on all behavior in the event log, or only the main behavior. Its parameters are:

- *relative to best threshold*, which defines the highest accepted value of the difference between the dependency measure and best dependency measure
- *positive observations*, which defines the lowest accepted value of the activity relation frequency measure
- *dependency threshold*, which defines the lowest accepted value of the activity dependency measure.
- *AND-threshold*, which defines the level of noise

The parameters of the Heuristics Miner can be used to catch only high frequent behavior or also low frequent behavior. But mining for also the low frequent behavior has always the risk to catch some noise (Weijters et al. 2006).

Inductive Miner is also implemented in ProM. It uses divide-and-conquer approach to discover the model. Thereby the results of inductive miner consist always in sound process models (Conforti et al. 2017), which are models that according to Caballero et al. (2017) have an initial marking and a finite set of final markings, have a path in the state space from the initial to each final marking, have always the possibility to reach some final marking and also have no transitions that can never fire.

Same as Heuristic miner also Inductive miner deals with noise. Additionally it also filters out the outlines, which are the infrequent events in a case. The model discovered by it always fits, meaning that it can generate the traces in the log and the value of the fitness is relatively high. This miner provides generalization, but suffers in precision (Leemans et al. 2013).

Criteria	Alpha++ miner	Heuristic Miner	Inductive Miner
Fitness	-	+	+
precision	+	-	-
generalization	-	+	+

Table 4.1: Algorithms performance based on quality criteria

The table 4.1 summarizes the behaviour of the previously described miners referring to the quality criteria. As it is shown, the heuristic as well as the inductive miner perform better than the alpha algorithm. Thus, both of them will be taken into consideration for process discovery. After generating some models in the following steps, simplicity will be used to choose between the results.

4.2 Process Mining for SHD ECORO

In this part the whole process mining is executed for SHD ECORO. The input data are mined proceeding in some steps which are explicitly described for SHD ECORO in the following subsections.

4.2.1 Generate the Log data

As it is emphasized in the previous section, log data is the baseline for the process mining. Thus, the first thing to check is whether this input can be provided or not. The nature of the log data can differ from one system to another, due to diverse information that is saved. In the case of SHD ECORO, the transaction logs are stored into the cache in the form of journals. According to the database specialists the reason behind it is to recover the objects that could be lost due to integrity problems. The data is saved every day in separate journals for each day. Every journal has an average size of around 500 MB and they are kept for no more than five days. Because of the size it is assumed that the data is large enough for process mining technique. Since the transactions are generated from all the actions done on the system by employees (developers and system testers), it is expected an incompleteness of the data. It can be the case when just some part of the possible features of the system are tested or further developed during those days. But, in order to have a good representative model from process mining the input data should represent the system. A possible solution for this issue is conducting some automated tests for all the feasible actions in SHD ECORO and afterwords save the data in a text file. For further procedures it is needed to have another format of the file containing the

data. For example, Disco opens only CSV (Comma-separated values) files. Due to this fact, the text file should be transformed into CSV (Comma-separated values) file by using Microsoft Excel.

4.2.2 Analyze the log data

Analyzing is a very crucial step in process mining due to the fact that the data should fulfill three most required features also mentioned in the log data subsection such as representing a task (activity), representing a case and being totally ordered. The third property is performed by the provided log data from SHD ECORO because they are totally ordered based on the time stamp. Regarding the task representation, there is no attribute named either task or activity in the events. But, one of the columns, namely *GlobalNode*, represents what is done in a specific event in terms of database tables. It means that in every instance one table is changed reflecting an action done from the user.

Offset	Zeit	Prozess	Typ	Transaktion in Gang	GlobalNode	Datenbank
150605448	2018-10-10 15:07:53	2056288	SET	Nein	^Protocol.REVWKundeDebBankvD	d:\db\ecoro_35_kunde\shdprotocol\
150605516	2018-10-10 15:07:53	2056288	SET	Ja	^Protocol.REVWKundeDebBankvD(373)	d:\db\ecoro_35_kunde\shdprotocol\

Figure 4.3: SHD ECORO event and its attributes (source: SHD ECORO Cache System)

For example in the figure 4.3 a short sequence of events is shown where the *GlobalNode* column holds the name of the table that is changed. This attribute can be a satisfying representative of the activity. But, here the names of the tables are associated with some global names like *User* and *Protocol*. User contains all the existing tables in SHD ECORO, whereas Protocol involves tables that store the changes in SHD ECORO. For example, the REVWKundeDebBankvD is the global name corresponding to REV_WKundeDebBankvD which has all new and updated values of every customer that is a debtor. Since the log data should reflect the behavior of a system and every time a protocol table is changed indicates a behavior, then the protocol global-s can be enough for the aim of process mining. However, this decision is further argued in the following paragraphs.

Another must-be attribute is the Process Id which indicates individual cases. In the log data presented there is a column containing the process id. In this case it is not the identifier of the business process, but it is the unique number that identifies each of the running processes in the operating system.

Offset	Zeit	Prozess	Typ	Transaktion in Gang	GlobalNode	Datenbank
137356	**2018-10-18 13:57:30**	**3177948**	**BeginTrans**	**Ja**		
137372	2018-10-18 13:57:30	3177948	BeginTrans with level	Nein		
137683	2018-10-18 13:57:30	3177948	SET	Ja	^User.AJobDefinitionD(301)	d:\db\ecoro_35_kunde\shd\
137760	2018-10-18 13:57:30	3177948	CommitTrans Pending with level	Nein		
137796	**2018-10-18 13:57:30**	**3177948**	**CommitTrans**	**Ja**		
137812	**2018-10-18 13:57:30**	**3177948**	**BeginTrans**	**Ja**		
137828	2018-10-18 13:57:30	3177948	BeginTrans with level	Nein		
138136	2018-10-18 13:57:30	3177948	SET	Ja	^User.AJobDefinitionD(102)	d:\db\ecoro_35_kunde\shd\
138208	2018-10-18 13:57:30	3177948	CommitTrans Pending with level	Nein		
138228	**2018-10-18 13:57:30**	**3177948**	**CommitTrans**	**Ja**		
138244	**2018-10-18 13:57:30**	**3177948**	**BeginTrans**	**Ja**		
138260	2018-10-18 13:57:30	3177948	BeginTrans with level	Nein		
138616	2018-10-18 13:57:30	3177948	SET	Ja	^User.AJobDefinitionD(2301)	d:\db\ecoro_35_kunde\shd\
138688	2018-10-18 13:57:30	3177948	CommitTrans Pending with level	Nein		
138708	**2018-10-18 13:57:30**	**3177948**	**CommitTrans**	**Ja**		
196624	**2018-10-18 13:57:55**	**3177948**	**BeginTrans**	**Ja**		
196640	2018-10-18 13:57:55	3177948	BeginTrans with level	Nein		
196948	2018-10-18 13:57:55	3177948	SET	Ja	^User.AJobDefinitionD(801)	d:\db\ecoro_35_kunde\shd\
197020	2018-10-18 13:57:55	3177948	CommitTrans Pending with level	Nein		
197040	**2018-10-18 13:57:55**	**3177948**	**CommitTrans**	**Ja**		
198432	2018-10-18 13:57:55	3177948	SET	Nein	^%SYS("sql","cursor counter")	d:\cachesys\mgr\
198496	2018-10-18 13:57:55	3177948	SET	Nein	^rtNDEXSQL("sqlidx",1,"G1mzLPgC5oHtgQhqJT3BvCA2DM8=",0)	d:\db\ecoro_35_kunde\shd\
198632	2018-10-18 13:57:55	3177948	SET	Nein	^rtNDEXSQL("sqlidx",1,"G1mzLPgC5oHtgQhqJT3BvCA2DM8=",1)	d:\db\ecoro_35_kunde\shd\
198884	2018-10-18 13:57:55	3177948	SET	Nein	^rtNDEXSQL("sqlidx",1,"G1mzLPgC5oHtgQhqJT3BvCA2DM8=",1,1)	d:\db\ecoro_35_kunde\shd\
198976	2018-10-18 13:57:55	3177948	SET	Nein	^rtNDEXSQL("sqlidx",1,"G1mzLPgC5oHtgQhqJT3BvCA2DM8=","rtn","%sqlcq.SHDu35.cls1654.1")	d:\db\ecoro_35_kunde\shd\
199092	2018-10-18 13:57:55	3177948	SET	Nein	^rtNDEXSQL("sqlidx",2,"%sqlcq.SHDu35.cls1654.1","G1mzLPgC5oHtgQhqJT3BvCA2DM8=")	d:\db\ecoro_35_kunde\shd\
199200	2018-10-18 13:57:55	3177948	SET	Nein	^rtNDEXSQL("sqlidx",1,"G1mzLPgC5oHtgQhqJT3BvCA2DM8=","plan")	d:\db\ecoro_35_kunde\shd\
199688	2018-10-18 13:57:55	3177948	SET	Nein	^rtNDEXSQL("sqlidx",1,"G1mzLPgC5oHtgQhqJT3BvCA2DM8=","plan",1)	d:\db\ecoro_35_kunde\shd\
201072	2018-10-18 13:57:55	3177948	SET	Nein	^rtNDEXSQL("sqlidx",1,"G1mzLPgC5oHtgQhqJT3BvCA2DM8=","plan",2)	d:\db\ecoro_35_kunde\shd\
201716	2018-10-18 13:57:55	3177948	SET	Nein	^rtNDEXSQL("sqlidx",1,"G1mzLPgC5oHtgQhqJT3BvCA2DM8=","plan",3)	d:\db\ecoro_35_kunde\shd\
202092	2018-10-18 13:57:55	3177948	SET	Nein	^rtNDEXSQL("sqlidx",1,"G1mzLPgC5oHtgQhqJT3BvCA2DM8=","plan",4)	d:\db\ecoro_35_kunde\shd\
202188	2018-10-18 13:57:55	3177948	SET	Nein	^rtNDEXSQL("sqlidx",1,"G1mzLPgC5oHtgQhqJT3BvCA2DM8=","planinfo")	d:\db\ecoro_35_kunde\shd\
202388	2018-10-18 13:57:55	3177948	SET	Nein	^rtNDEXSQL("sqlidx",1,"G1mzLPgC5oHtgQhqJT3BvCA2DM8=","priv",1,SQLUser.WAlParaInhFiliale","s")	d:\db\ecoro_35_kunde\shd\
202536	2018-10-18 13:57:55	3177948	SET	Nein	^rtNDEXSQL("sqlidx",1,"G1mzLPgC5oHtgQhqJT3BvCA2DM8=","rel","SQLUSER.WALPARAINHFILIALE")	d:\db\ecoro_35_kunde\shd\
202648	2018-10-18 13:57:55	3177948	SET	Nein	^rtNDEXSQL("sqlidx",3,"SQLUSER.WALPARAINHFILIALE","G1mzLPgC5oHtgQhqJT3BvCA2DM8=")	d:\db\ecoro_35_kunde\shd\
202756	2018-10-18 13:57:55	3177948	SET	Nein	^%SYS("sql","cursor counter")	d:\cachesys\mgr\
202820	2018-10-18 13:57:55	3177948	SET	Nein	^rtNDEXSQL("sqlidx",1,"rUICSn5zZjq0hsijz/BU9EhFcN4=",0)	d:\db\ecoro_35_kunde\shd\
202956	2018-10-18 13:57:55	3177948	SET	Nein	^rtNDEXSQL("sqlidx",1,"rUICSn5zZjq0hsijz/BU9EhFcN4=",1)	d:\db\ecoro_35_kunde\shd\
203256	2018-10-18 13:57:55	3177948	SET	Nein	^rtNDEXSQL("sqlidx",1,"rUICSn5zZjq0hsijz/BU9EhFcN4=",1,1)	d:\db\ecoro_35_kunde\shd\
203348	2018-10-18 13:57:55	3177948	SET	Nein	^rtNDEXSQL("sqlidx",1,"rUICSn5zZjq0hsijz/BU9EhFcN4=","rtn","%sqlcq.SHDu35.cls1716.1")	d:\db\ecoro_35_kunde\shd\
203464	2018-10-18 13:57:55	3177948	SET	Nein	^rtNDEXSQL("sqlidx",2,"%sqlcq.SHDu35.cls1716.1","rUICSn5zZjq0hsijz/BU9EhFcN4=")	d:\db\ecoro_35_kunde\shd\
203572	2018-10-18 13:57:55	3177948	SET	Nein	^rtNDEXSQL("sqlidx",1,"rUICSn5zZjq0hsijz/BU9EhFcN4=","plan")	d:\db\ecoro_35_kunde\shd\
204028	2018-10-18 13:57:55	3177948	SET	Nein	^rtNDEXSQL("sqlidx",1,"rUICSn5zZjq0hsijz/BU9EhFcN4=","plan",1)	d:\db\ecoro_35_kunde\shd\
205408	2018-10-18 13:57:55	3177948	SET	Nein	^rtNDEXSQL("sqlidx",1,"rUICSn5zZjq0hsijz/BU9EhFcN4=","plan",2)	d:\db\ecoro_35_kunde\shd\
205848	2018-10-18 13:57:55	3177948	SET	Nein	^rtNDEXSQL("sqlidx",1,"rUICSn5zZjq0hsijz/BU9EhFcN4=","plan",3)	d:\db\ecoro_35_kunde\shd\
206224	2018-10-18 13:57:55	3177948	SET	Nein	^rtNDEXSQL("sqlidx",1,"rUICSn5zZjq0hsijz/BU9EhFcN4=","plan",4)	d:\db\ecoro_35_kunde\shd\
206320	2018-10-18 13:57:55	3177948	SET	Nein	^rtNDEXSQL("sqlidx",1,"rUICSn5zZjq0hsijz/BU9EhFcN4=","planinfo")	d:\db\ecoro_35_kunde\shd\
206520	2018-10-18 13:57:55	3177948	SET	Nein	^rtNDEXSQL("sqlidx",1,"rUICSn5zZjq0hsijz/BU9EhFcN4=","priv","1,SQLUser.WAlParaInhFiliale","s")	d:\db\ecoro_35_kunde\shd\
206668	2018-10-18 13:57:55	3177948	SET	Nein	^rtNDEXSQL("sqlidx",1,"rUICSn5zZjq0hsijz/BU9EhFcN4=","rel","SQLUSER.WALPARAINHFILIALE")	d:\db\ecoro_35_kunde\shd\
206788	2018-10-18 13:57:55	3177948	SET	Nein	^rtNDEXSQL("sqlidx",3,"SQLUSER.WALPARAINHFILIALE","rUICSn5zZjq0hsijz/BU9EhFcN4=")	d:\db\ecoro_35_kunde\shd\
214760	**2018-10-18 13:57:55**	**3177948**	**BeginTrans**	**Ja**		
214920	2018-10-18 13:57:55	3177948	SET	Ja	^User.AAnschriftD(76635)	d:\db\ecoro_35_kunde\shd\
214984	2018-10-18 13:57:55	3177948	SET	Ja	^User.AAnschriftI("IXAAnschriftOrt0","DGGDFGDGD",76635)	d:\db\ecoro_35_kunde\shd\

Figure 4.4: Example of how many different tasks does one process of the operating system. (source: SHD ECORO Cache System)

The figure 4.4 depicts a simple example of one operating system process and how it can execute many diverse tasks. The OS process identifier cannot be used as a case indicator because one operating system process runs activities coming from different business processes. Therefore, the tasks executed in one business process can be processed by different operating system processes. Some of the situations that can happen based on the figure 4.5 are as follows:

1. All the tasks can be executed by the same OS process. For example it can be the case where P1 runs Task1, Task2, Task3, Task4 as well as Task5, Task6, Task7.

2. Tasks from different work-flows are executed as a group by the same OS process. For example, it can be the case where P1 runs Task1, Task2, Task3 and Task5, whereas P2 runs Task4, Task6 and Task7.

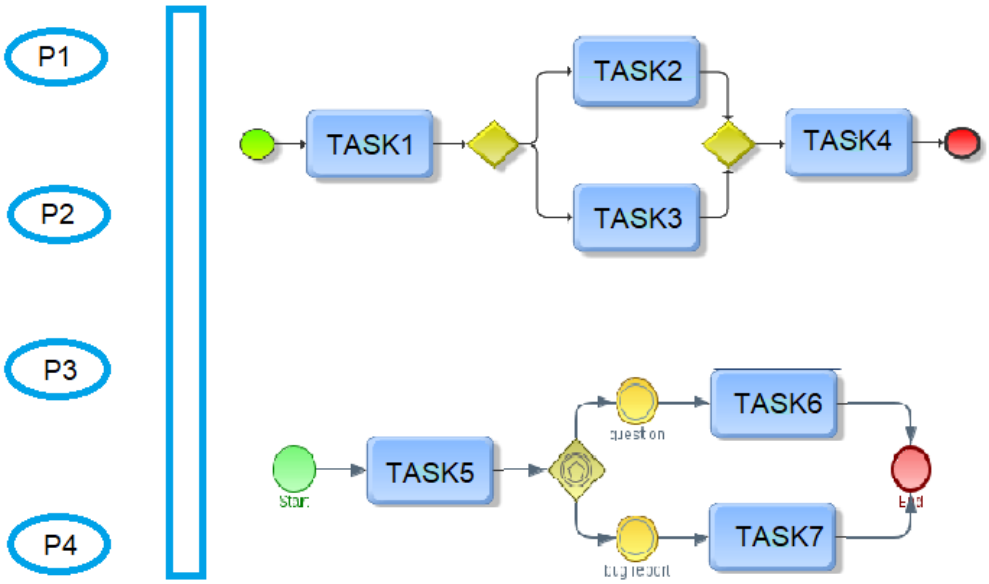


Figure 4.5: OS processes vs User processes (Tasks) (source: self-made)

The process id which is provided in the log data cannot help for the purpose of process discovery by using the process mining algorithms and tools. Therefore another possible solution in this case could be considering the transactions as individual tasks. This notion is supported by the fact that:

1. Every operation in SHD ECORO starts by either a save or update of software objects; for instance a sell/buy contract (kaufvertrag), an offer (angebot), an article (artikel) etc.

2. Each of the operations is a single save/update transaction created after a call from the client to the server.

The situation is illustrated bellow, where other tasks related to the operating system are excluded.

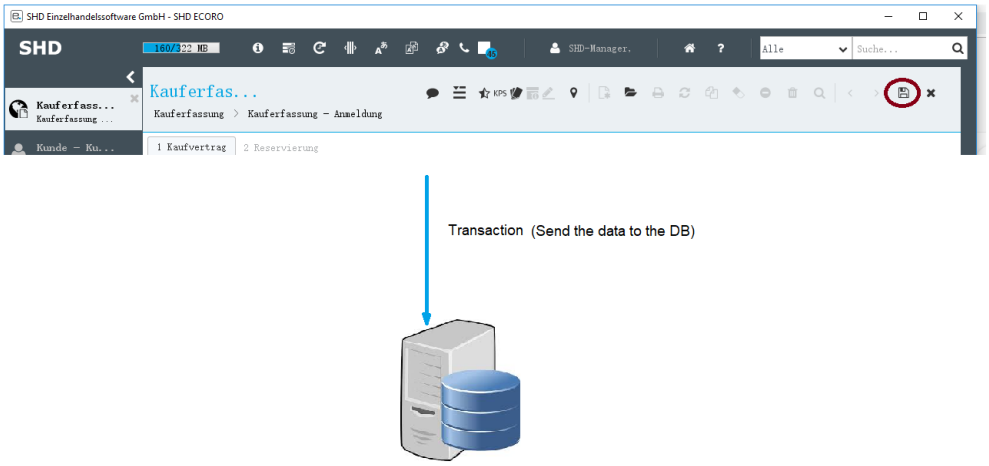


Figure 4.6: Example of a transaction that sends the data from client to be saved into the server. (source: self-made)

The example in figure 4.6 shows the whole information saved when a sale contract is created. Therefore, as also mentioned before one transaction can represent a single case. But, the transaction id is not provided in the log data. Nevertheless, there is a column called ***Type***, which indicates apart from the type of each sub-transaction also the begin and commit information of each connection represented as a big transaction. Thus, a BT will infer a begin and CT a commit of that transaction. In this situation should be considered two tricky possibilities:

1. Parallel transactions from different clients, which happen at the same time but are executed from different users. In this case is interesting to look whether they are overwritten or not.

2. Cases where the transactions are not ideally nested but could exist an undesired example as shown in the figure 4.7.

To confirm if any of those situations may occur, it is simulated the situation of two different clients where one creates a sale contract and the other creates an article at the same

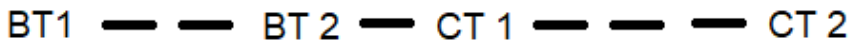


Figure 4.7: Example of transactions are not ideally nested. (source: self-made)

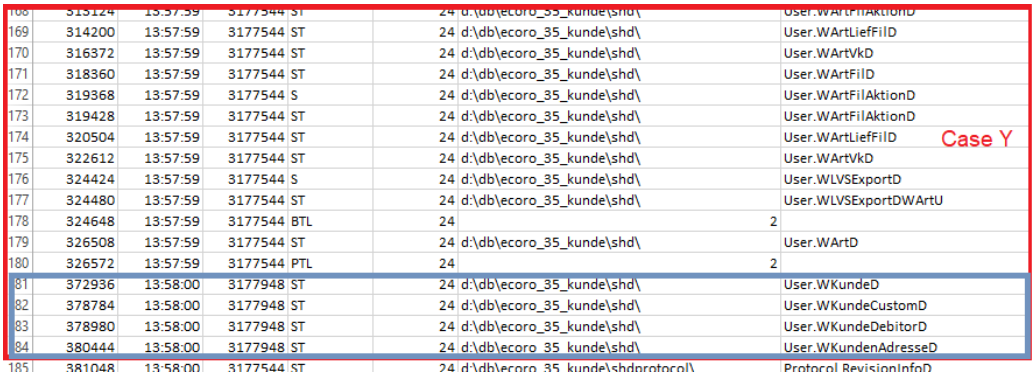
time. The created sample of log data is used for further analyses. It is afterwards manually checked for the above situations since the data is small enough to do so.

Since the business processes that are represented into the log data are known as being a creation of a sale contract and in parallel also an article is separately created, it is expected that the data reflect the processes as well. But, based on what is done, the data associated to the global called *User* is not ordered and somehow mixed into diverse transactions. It is noticed, as also shown in the figure 4.9, that the data about creation of a customer related to the sale contract (Case X) is stored as part of the transaction that include the data about the creation of the article (Case Y). In this case the timestamp is the same, therefore cannot help in ordering.

90	214760	13:57:55	3177948	BT	23			
91	214920	13:57:55	3177948	ST	23	d:\db\ecoro_35_kunde\shd\	User.AAnschriftD	
92	216024	13:57:55	3177948	ST	23	d:\db\ecoro_35_kunde\shd\	User.APersonD	Case X
93	217540	13:57:55	3177948	BTL	23	2		
94	217872	13:57:55	3177948	ST	23	d:\db\ecoro_35_kunde\shd\	User.AAnschriftD	
95	218112	13:57:55	3177948	PTL	23	2		
96	224664	13:57:55	3177544	BT	24			
97	224680	13:57:55	3177544	BTL	24	2		
98	224756	13:57:55	3177544	PTL	24	2		
99	229976	13:57:55	3177544	S	24	d:\db\ecoro_35_kunde\shd\	User.WLVSExportD	Case Y
100	230032	13:57:55	3177544	ST	24	d:\db\ecoro_35_kunde\shd\	User.WLVSExportDWArtU	
101	231108	13:57:55	3177544	ST	24	d:\db\ecoro_35_kunde\shd\	User.WArtDTest	
102	248024	13:57:55	3177544	ST	24	d:\db\ecoro_35_kunde\shd\	User.WArtLiefD	
103	249988	13:57:55	3178160	BT	25			

Figure 4.8: Example of two different cases. (source: ECORO log data)

One solution to this issue could be to use as a *case id* in the process mining both the *transaction id* and the *process id* of the OS in order to differentiate the cases. But, by doing so the saving of a customer as well as situations that belong to the same transaction

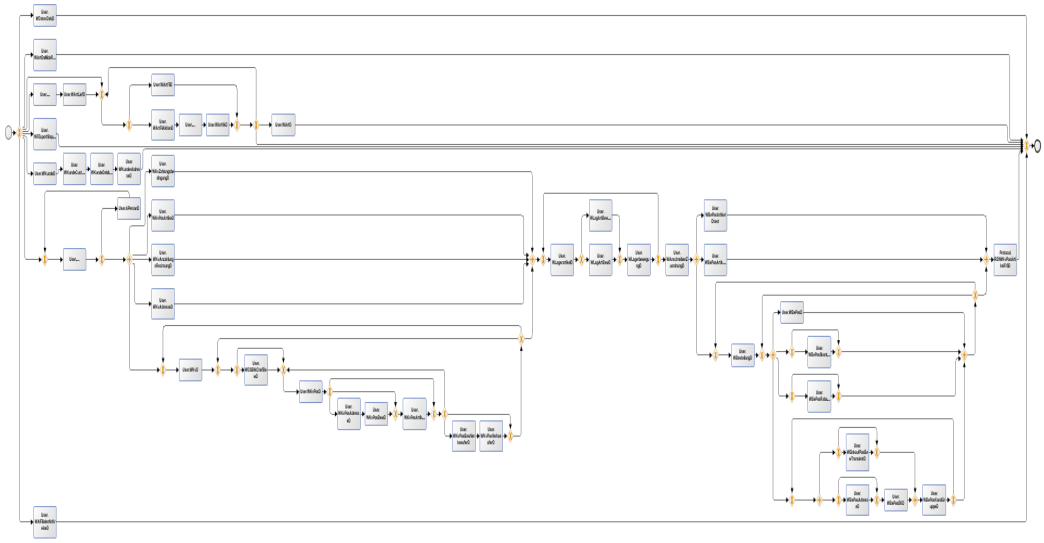


169	314200	13:57:59	3177544	ST	24	d:\db\ecoro_35_kunde\shd\	User.WArtLiefFilD	
170	316372	13:57:59	3177544	ST	24	d:\db\ecoro_35_kunde\shd\	User.WArtVkD	
171	318360	13:57:59	3177544	ST	24	d:\db\ecoro_35_kunde\shd\	User.WArtFilD	
172	319368	13:57:59	3177544	S	24	d:\db\ecoro_35_kunde\shd\	User.WArtFilAktionD	
173	319428	13:57:59	3177544	ST	24	d:\db\ecoro_35_kunde\shd\	User.WArtFilAktionD	
174	320504	13:57:59	3177544	ST	24	d:\db\ecoro_35_kunde\shd\	User.WArtLiefFilD	Case Y
175	322612	13:57:59	3177544	ST	24	d:\db\ecoro_35_kunde\shd\	User.WArtVkD	
176	324424	13:57:59	3177544	S	24	d:\db\ecoro_35_kunde\shd\	User.WLVSExportD	
177	324480	13:57:59	3177544	ST	24	d:\db\ecoro_35_kunde\shd\	User.WLVSExportDWArtU	
178	324648	13:57:59	3177544	BTL	24	2		
179	326508	13:57:59	3177544	ST	24	d:\db\ecoro_35_kunde\shd\	User.WArtD	
180	326572	13:57:59	3177544	PTL	24	2		
81	372936	13:58:00	3177948	ST	24	d:\db\ecoro_35_kunde\shd\	User.WKundeD	
82	378784	13:58:00	3177948	ST	24	d:\db\ecoro_35_kunde\shd\	User.WKundeCustomD	
83	378980	13:58:00	3177948	ST	24	d:\db\ecoro_35_kunde\shd\	User.WKundeDebitorD	
84	380444	13:58:00	3177948	ST	24	d:\db\ecoro_35_kunde\shd\	User.WKundenAdresseD	

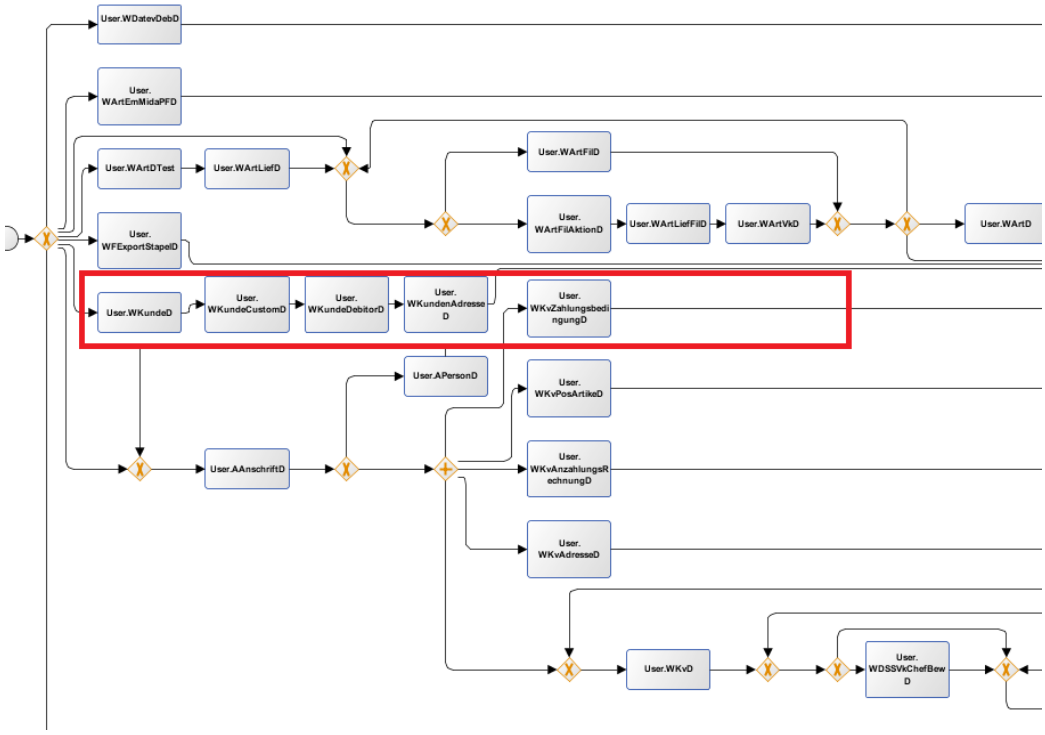
Figure 4.9: Example of not ordered cases. (source: ECORO log data)

but are executed by different processes would be completely diverse cases and would not belong to the right business process. Therefore, this solution will extract the activities merged into transactions where they do not belong to, but will not provide the right place for them. For example, the figure 4.10 depicts the model created by ***ProM*** while using the *User* data and the *process id* as well as the *transaction id* as case id for the mining algorithm. It is clear that the creation of a customer is not part of either the sale contract creation or the article creation in the example. However, the data related to the *Protocol* global are not mixed and are ordered.

Moreover, the not ideally nested transactions are not present in the data. It might be supposed that possibly this situation can be found for a large amount of log data, but the employees responsible for the database insure that it is not likely to happen for SHD ECORO.



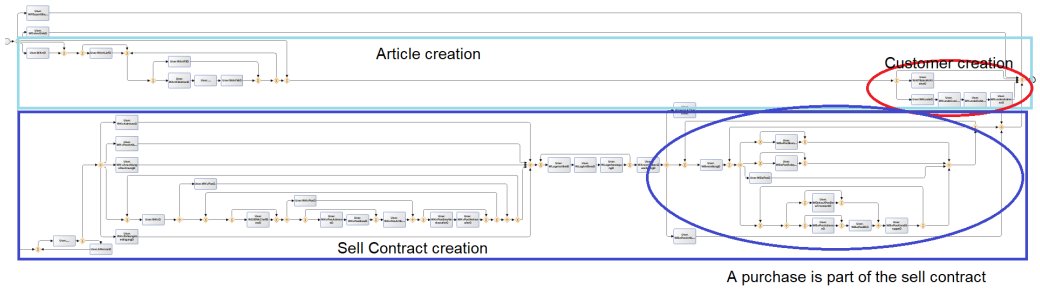
(a) Complete Model



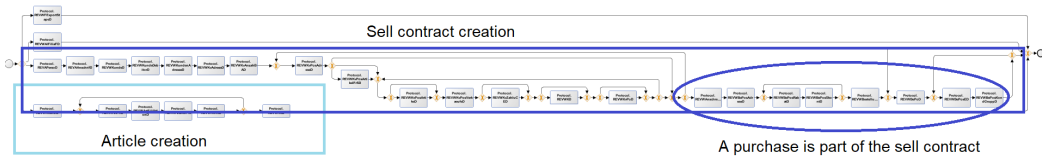
(b) Part of the Model

Figure 4.10: Model created in ***ProM*** by using the *transaction id* as well as the *process id* as a case id for the mining algorithm (source: ***ProM*** results)

Afterwards, the data is used from ***ProM*** to discover some business processes. The main goal in this step is to reveal if the model reflects the two business processes, respectively creation of a Sale Contract as well as creation of an article.



(a) Model from the ***User*** data



(b) Model from the ***Protocol*** data

Figure 4.11: Models created in ***ProM*** by using the *transaction id* as a case if for the mining algorithm separately for the ***User*** and ***Protocol*** data(source: ***ProM*** results)

In the figure 4.11 is easy to see that both of the models reflect the two separated processes, but the model corresponding to the ***Protocol*** data does not include the case of the customer creation into the article creation as the other model does. Hence, the model that best represents the undertaken processes is the one extracted from the ***Protocol*** data.

4.2.3 Filter the input data

Before performing any mining algorithm it is crucial to have clear and representative input data. Therefore, filtering is a very important step in the process mining. For filtering diverse aspects should be considered based on the nature of the data. In the case of SHD ECORO log data is filtered focusing on database name, global name, attribute name,

ActivityStream attribute name and disco filtering.

Filtering by the database name helps to get rid of the log data saved in the cache system that are not related to SHD ECORO. In this situation are picked up from the whole data set just the transactions that interact with ***shdprotocol*** database, where are saved the data related to SHD ECORO. As also represented in the figure above, the information of the database name is part of the log data and filtering consists of deleting the rows that deal with other databases rather than ***shdprotocol***.

From two existing *globals*, the ***protocol*** is a better representative of the log data, because it fulfills all the prerequisites of it. The argument about this case has been fully explained in the previous subsection.

The column named "attribute name" aaves very detailed information, meaning that every transaction that notifies a save into one table is followed by transactions which set every index value in that table. For instance in the figure 4.12 is shown an example when an article is added into a warehouse which is represented by the very first transaction; *At 10:14:46 the windows system process with the id 86568 has set into the User.WLagerartikelID a value corresponding to a warehouse article*. All the information that follows is a very specific one that is actually not needed for the aim of this step.

The ActivityStream is the table that contains the name of the activity that SHD ECORO performs in relation to the customers. Some of them are: Creation of a Sales Contract, Change of a Sale Contract, Creation of an Offer, Change of an Offer etc. For each of the *Protocol.REVActivityStream* global is attached the value saved in this table (depict in the table) instead of the name of the table. Values of the ActivityStream table are depicted in the table 4.2

Disco provides many filters also mentioned in the subsection related to *Disco*, but in the case of SHD ECORO is used the filter to first add out all the remaining activities that do not provide any crucial information for the process; such as for example those related to the *Print*, and then delete all the cases that have one event as not being significant.

As a conclusion, the filtered data have three important columns as shown in the table 4.3 that contain the needed information for the process mining algorithms in ***ProM***.

```
22379292 10:14:46   869568  ST User.WLagerartikelD(52313)=$lb("",0,2,79972,"",1,"",2,"",43581,0,"2018-10-02 00:00:00","1902-12-31 10:14:49.083",'
22379356 10:14:46   869568  ST OBJ.DSTIME("User.WLagerartikel",5609376885,52313)=1
22379436 10:14:46   869568  bT User.WLagerartikelI("$WLagerartikel",1)="52314,1"
22379516 10:14:46   869568  bT User.WLagerartikelI("IXBMWLagerartikelBVPos0",1,1)="52314,1"
22379604 10:14:46   869568  bT User.WLagerartikelI("IXBMWLagerartikelBestandsfi0",2,0,1)="52314,1"
22379700 10:14:46   869568  bT User.WLagerartikelI("IXBMWLagerartikelBeste0",0,1)="52314,1"
22379788 10:14:46   869568  bT User.WLagerartikelI("IXBMWLagerartikelIsGesperrt0",0,1)="52314,1"
22379884 10:14:46   869568  bT User.WLagerartikelI("IXBMWLagerartikelIsInUmbuch0",0,1)="52314,1"
22379980 10:14:46   869568  bT User.WLagerartikelI("IXBMWLagerartikelIsReklaLie0",0,1)="52314,1"
22380076 10:14:46   869568  bT User.WLagerartikelI("IXBMWLagerartikelIsReklaOff0",0,1)="52314,1"
22380172 10:14:46   869568  bT User.WLagerartikelI("IXBMWLagerartikelIsTeilebes0",0,1)="52314,1"
22380268 10:14:46   869568  bT User.WLagerartikelI("IXBMWLagerartikelMenge0",1,1)="52314,1"
22380356 10:14:46   869568  bT User.WLagerartikelI("IXBMWLagerartikelQMMerker0"," ",1)="52314,1"
22380448 10:14:46   869568  bT User.WLagerartikelI("IXBMWLagerartikelRekla0",1,1)="52314,1"
22380536 10:14:46   869568  bT User.WLagerartikelI("IXBMWLagerartikelResPo0",1,1)="52314,1"
22380624 10:14:46   869568  bT User.WLagerartikelI("IXBMWLagerartikelSperrgrund0",-100000000000000,1)="52314,1"
22380720 10:14:46   869568  bT User.WLagerartikelI("IXBMWLagerartikelVerka0",1,1)="52314,1"
22380808 10:14:46   869568  bT User.WLagerartikelI("IXBMWLagerartikelVorga0",1,1)="52314,1"
22380896 10:14:46   869568  bT User.WLagerartikelI("IXBMWLagerartikelabgerechne0",0,1)="52314,1"
22380992 10:14:46   869568  ST User.WLagerartikelI("IXWLAGERARTIKELRETORR0",-100000000000000,52313)=""
22381076 10:14:46   869568  ST User.WLagerartikelI("IXWLagArtLetzteRekla",-100000000000000,52313)=""
22381160 10:14:46   869568  ST User.WLagerartikelI("IXWLagerartikelAbgerechnetB0",0,2,0,-100000000000000,-100000000000000,-100000000000000,0," ",
22381280 10:14:46   869568  ST User.WLagerartikelI("IXWLagerartikelArtikelIDVer0",79972,-100000000000000,-100000000000000,-100000000000000,-1000(
22381396 10:14:46   869568  ST User.WLagerartikelI("IXWLagerartikelArtikelI0",79972,52313)=""
22381484 10:14:46   869568  ST User.WLagerartikelI("IXWLagerartikelBVPosi0",-100000000000000,52313)=""
22381568 10:14:46   869568  ST User.WLagerartikelI("IXWLagerartikelBePosAbr0",-100000000000000,52313)=""
22381656 10:14:46   869568  bT User.WLagerartikelI("IXWLagerartikelBestan0",2,1)="52314,1"
```

Figure 4.12: Example of transactions that set an Index in a table. (source: SHD ECORO log data)

4.2.4 Results of data mining

In this step all the filtered data is available for being used in mining algorithms. They contain events related to:

1. *Protocol* global, which contains the changes in database tables.

2. *ActivityStream* attribute name, which is the name of the activity generated into the respective table and it represents just the activities undertaken with the customer.

In the process mining of the data both the *Inductive* & *Heuristic* algorithms are used. The results of *Protocol* data is depicted in the following the figure 4.13 and 4.14.

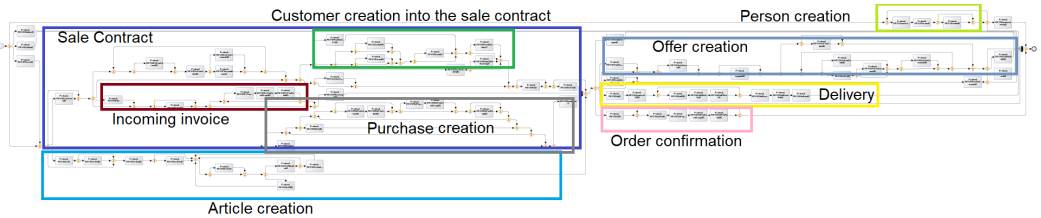


Figure 4.13: SHD ECORO model resulted from the Inductive algorithm

***ActivityStream* Table**
Angebot erfasst (Offer recorded)
Angebot geaendert (Offer changed)
Auslieferabrechnung fuer KV vorgenommen (Delivery settlement for KV made)
Chance erfasst (Chance recorded)
Chance geaendert (Chance changed)
Kaufvertrag erfasst (Sale contract recorded)
Kaufvertrag geaendert (Sale contract changed)
Kaufvertrag komplett lieferfaehig (Purchase contract completely deliverable)
Kaufvertrag nicht komplett lieferfaehig (Purchase contract not completely deliverable)
KD position erfasst (Customer service position recorded)
Kundendienstmeldung fuer KV erfasst (Customer service message for KV recorded)
Kundendienstmeldung fuer KV bearbeitet (Customer service message for KV changed)
Kundenliefertermin geaendert (Customer delivery changed)
Notiz erfasst (Notification recorded)
Telefonat erfasst (Telephonat recorded)
Tourenplanung fuer KV abgeschlossen (Tour planning for KV completed)
Uebernahme in KV (Takeover in KV)

Table 4.2: Some of the activity values attached to *ActivityStream* table

Column name	***Information***
Time	Time when a transaction is executed
Transaction ID	Unique ID of a transaction
Activity	Global name which represents the table that is changed

Table 4.3: Important columns in the Log Data

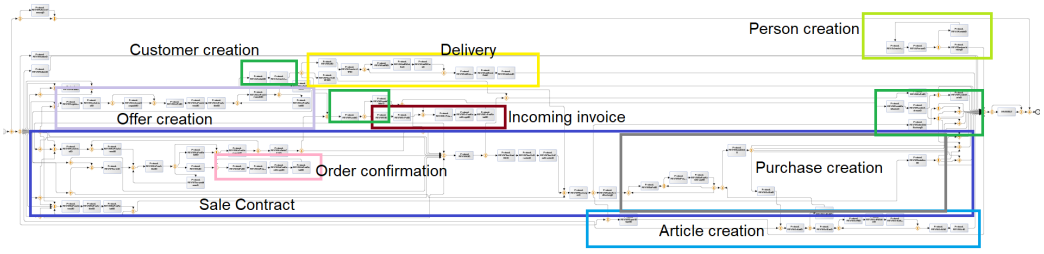
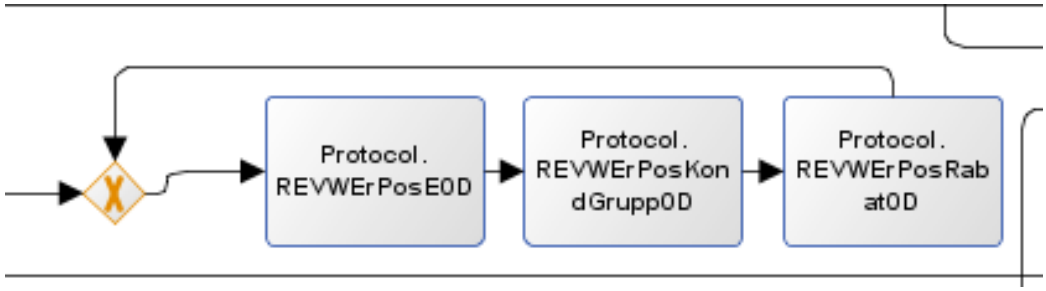


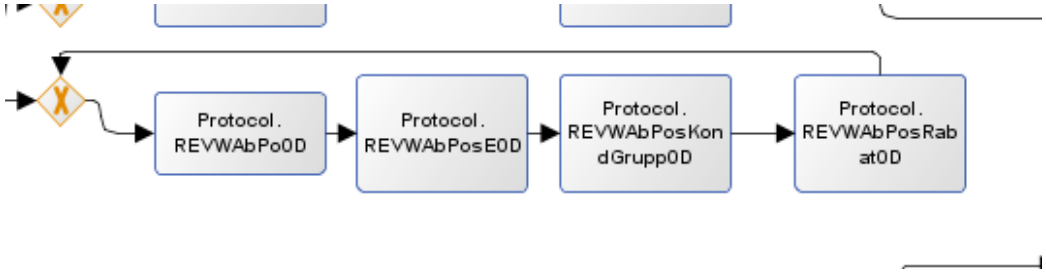
Figure 4.14: SHD ECORO process model resulted from the Heuristic algorithm

Results from both mining algorithms provide visible parts of the process that correspond to sub-processes as presented in the table 4.4 along with a short information about what activities they contain. The heuristic miner involves some infinite loops displayed in the figure 4.15, which correspond to interactions with other layers in the business process management process model such as:

- Incoming invoice, which is a receiving activity that waits for the invoice to come from the supplier.
- Order confirmation, which is a sending activity that transmits the confirmation to the supplier.



(a) Incoming invoice



(b) Order confirmation

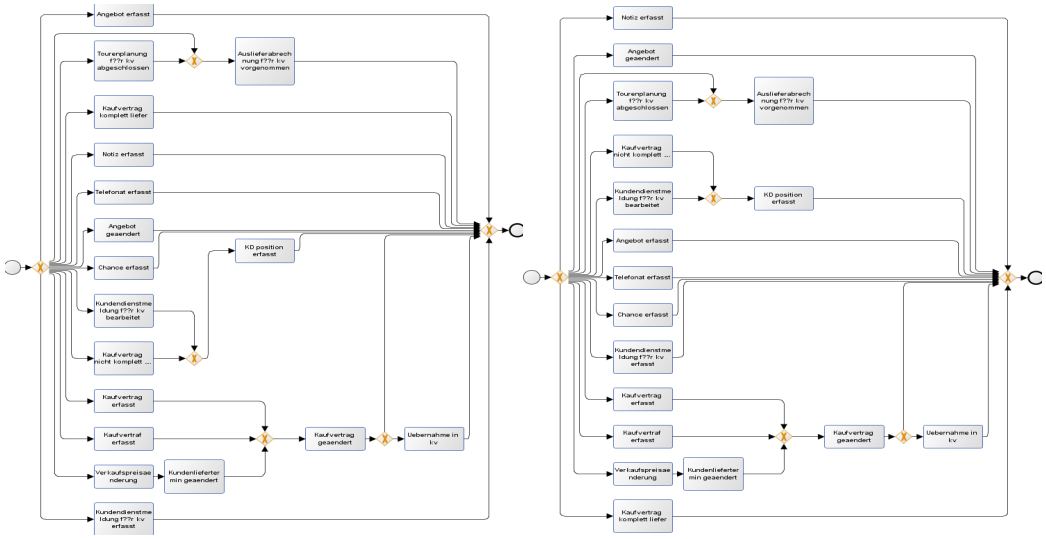
Figure 4.15: Infinite loops in the model resulted from the Heuristic miner

However, the overall workflow of the system is not very clear and other external information is required in this case in order to create an authentic process model. Moreover, the scope will be narrowed in the selling process. To do so, just the sub-processes which are part of the big process of selling will be taken into consideration to create a final model.

From the models is visible that *Protocol* data preform differently from the the *ActivityStream* data. As shown in the figure 4.16, the depicted model presents single transactions separately.

Sub process	***Information***
Article creation	Includes the workflow where the tables containing information about **article** are changed. (*REVWArtSetD, REVWArtD, REVWArtLiefD, REVWArtEanD, REVWArtFilD, REVWArtFilAktionD* etc.)
Sale Contract	Includes the workflow where the tables containing information about **sale contract** are changed. (*REVWKvAdress0D, REVWKvPosArtike0D, REVWKvPosAdress0D, REVWKvZahluCE10D* etc.)
Offer creation	Includes the workflow where the tables containing information about **offer** are changed. (*REVWAngebo0D, REVWAnAdress0D, REVWAnAnzahlungspla0D, REVWAnPosArtike0D, REVWAnPosRabat0D* etc.)
Customer creation	Includes the workflow where the tables containing information about **customer** are changed. (*REVWKundeD, REVWKundeDebitorD, REVWKundenAdresseD* etc.)
Incoming invoice	Includes the workflow where the tables containing information about **incoming invoice** are changed. (*REVWErPo0D, REVWErPosRabat0D, REVWErPosKondGruppe0D* etc.)
Purchase creation	Includes the workflow where the tables containing information about **purchase** are changed. (*REVWBePosAdress0D, REVWBePosArtike0D, REVWBePosRabat0D* etc.)
Order confirmation	Includes the workflow where the tables containing information about **order confirmation** are changed. (*REVWAbPo0D, REVWAbPosKondGruppe0D, REVWAbPosRabat0D* etc.)
Delivery	Includes the workflow where the tables containing information about **delivery** are changed. (*REVWLiefD, REVWLiefFilD, REVWLiefLandD* etc.)
Person creation	Includes the workflow where the tables containing information about **person** are changed. (*REVAPersonD, REVAKontaktD, REVAAnschriftD* etc.)

Table 4.4: Main sub-processes in the models



(a) *Inductive* algorithm (b) *Heuristic* algorithm

Figure 4.16: Activity stream SHD ECORO model

Each of the activities actually represent a separate process in ECORO where the sequence behind it is not shown. Therefore, this indicates that the model is not a good representative of the overall system workflow.

4.2.5 Further analyses

Some statistics about SHD ECORO log data can be delivered from *Disco* . It gives an overall description of the cases, activities and events mostly in terms of duration and frequency. The total number of cases is 163 distributed in 60 variants. Whereas the total number of activities is 83, where 30 of them are start activities and 34 are end ones. Some of the activities take more time than the others. Commonly the duration is extremely low, but there are some exception same as shown in the table 4.6.

All the activities are storing actions in the database. Therefore the time of them does not indicate anything rather than the time spent to save the data. Large data require more time. The amount of activities is not very large but the model includes distinguished processes. However, the main process in SHD ECORO is ***Selling*** which proceeds with

	Statistics
Activities	83
Minimal frequency	1
Maximal frequency	155
First activities in case	30
Last activities in case	34
Variants	60
Cases	163

Table 4.5: Satistics from ***Disco***

Activity name	***Duration***
Protocol.REVWARTFilAktionD	18 sec
Protocol.REVFBankverbindungD	06 sec
Protocol.REVWAnPosVerkaeufe0D	18 sec
Protocol.REVWZusatztextD	14 sec

Table 4.6: Activities that have a higher duration

the registration of a sale contract. Therefore, it is necessary to gather just the activities related to sale contract creation in order to discover the selling process. In this situation the values which are stored in the ActivityStream table come in hand. The approach that helps is to pick in a chronological way all the transactions that have *Kaufvertrag erfasst (Sale contract recorded)* among other activities. Due to the fact that the ActivityStream values correspond to just the actions that SHD ECORO undertakes with the customer, there is no activity named for example *Purchase recorded*. Nevertheless it is expected that the purchase creation would be enclosed in the whole process of sale contract creation.

Both of the miners provide a similar workflow of the selling process as shown in the figures 4.17 and 4.18. However other diverse resources will be considered to come up with an authentic model.

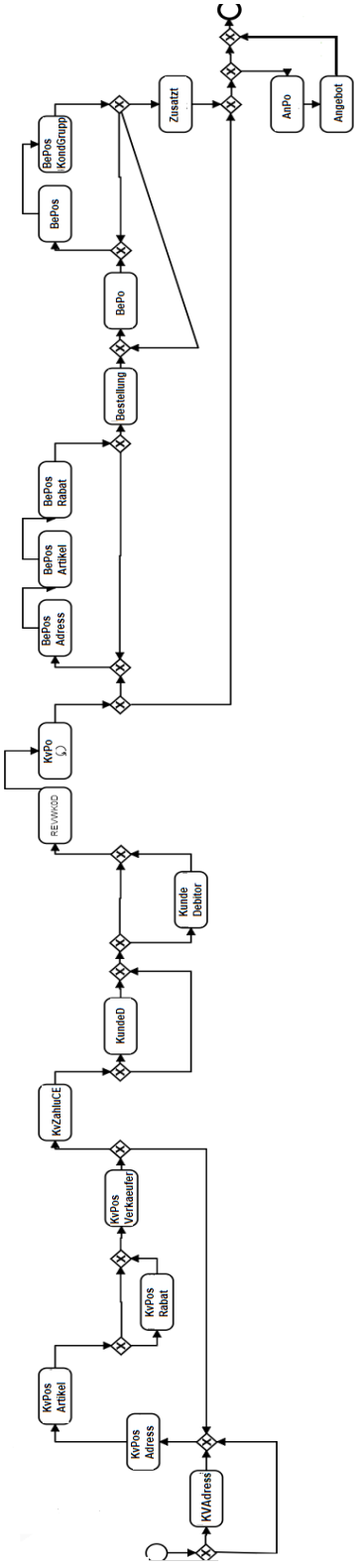


Figure 4.17: Model provided by the Inductive miner.

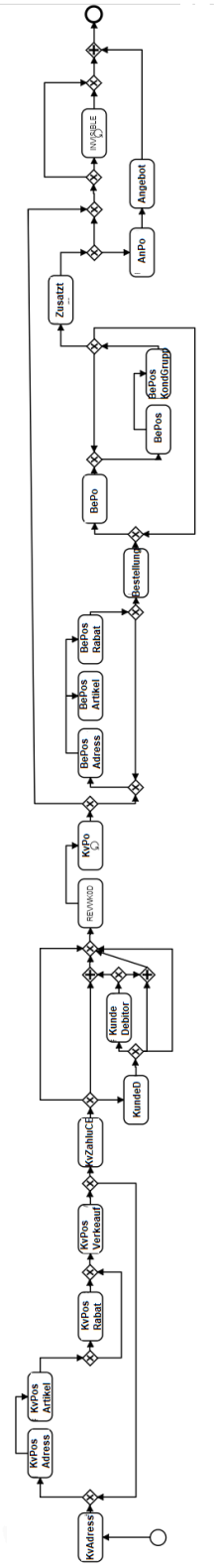


Figure 4.18: Model provided by the Heuristic miner.

4.3 Other Process discovery techniques

Process mining is a good approach for process discovery, but it is fully dependent on the input data, which are known as log data. Some of the mining algorithms such as heuristic deal with incompleteness and noise (add section), but in the case of SHD ECORO the issue is more related to the nature of the data rather than their quality. Therefore, manual technique of process discovery is needed in this situation.

4.3.1 Manual discovery approach

This sub chapter provides a manual technique for process discovery in ERP systems. Porters value chain entails viewing organizational units as systems composed of interrelated subsystems or processes that involve consumption and transformation of resources such as money, work force, materials, infrastructure and management (Dekker 2003). Therefore, Porters Value Chain concept can help in developing a big but generalized picture of all the activities that are involved in the creation of finished products and how these processes are interrelated.

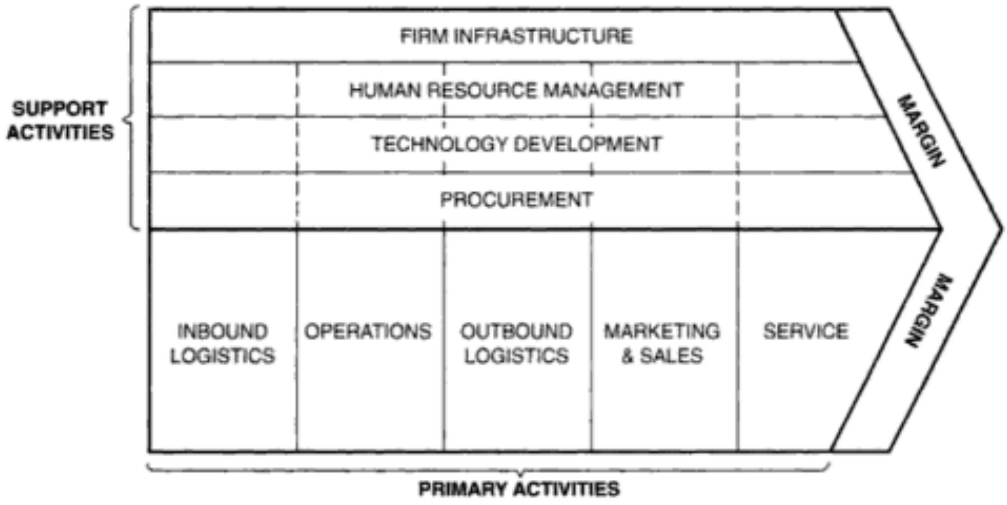


Figure 4.19: The generic value chain. (source: Porter (2001))

The figure 4.19 shows the value chain activities which are divided in two main sets namely

support and primary activities. It is important to focus on the primary activities because they describe the main processes.

To have a better overview, some examples of each primary activities in the Porter's value chain are described in the table 4.7.

Primary activity name	***Examples***
Inbound Logistics	receiving materials, quality control, raw materials control etc
Operations	manufacturing, packaging, production control, maintenance etc
Outbound Logistics	finishing goods, order handling, dispatch, delivery, invoicing etc
Marketing and **Sales**	customer management, order taking, promotion, sales analysis etc
Service	warranty, maintenance, education and training etc

Table 4.7: Primary activities in Porter's Value Chain

In the case of SHD ECORO are undertaken well defined steps with the aim to find some specific processes of the system. Previously discovered properties and models as well as Porter's value chain are the input of the further work.

One important step is the identification of some main processes specific for SHD ECORO as well as their relation. Selling is one main process which is supported by others, such as inbound and outbound logistics, to fulfill the whole business process. The first model provided by the process mining technique will perform as a baseline for other steps. Moreover, the interaction with the system gives also some initial hints on the work flow of the process. For instance, creating a sale contract in SHD ECORO crosses the main steps which indicate the activities processed while selling. In this step are extracted some activities and their relation is created. The project managers come in help to give the final touch after listing the main tasks generated from both process mining models and the manual interaction with SHD ECORO. They are the people who represent the system in the workshops and know better the functionalists of the system in an upper level.

4.3.2 Results

Before showing the final model of the selling process in SHD ECORO a description of every step happening behind the selling process will help in defining the main activities. Starting with the selling process, which can either initiate with a customer creation/update or directly with an article input for the contract creation. Both activities are sub processes which happen one after another and their order is configurable in the system by using a parameter. If the person is not an existing customer then he/she should be saved as a new customer. In the case of being a customer, the profile can be updated.

Afterwords is executed the articles input, which is part of a separate sub process and checks whether they are in stock or not. If it is not the case, then an order with the missing articles is created and it is sent to the supplier.

There are times when the seller can add a discount to the customer for the required articles. After this step the order is send to the supplier and then they are expected to be received in the warehouse. The waiting time is different for diverse suppliers. If the time is exceeded then an invoice is created where is stated the amount of money depended on the delayed days that the supplier should pay as a fine. Again the process returns to the item-receiving activity.

When the items come in the warehouse a tour to deliver them is created only in the case that there does not already exist one. The sale contract articles are added to a tour to be then delivered to the final customer.
Activities part of this process as well as their relations will be created based on the above description. Afterwards, the model will be designed in ***Camunda Modeler*** using ***bpmn*** language. In the figure 4.20 is the model which is delivered after the examinations. The table 4.8 contains a description for each activity part of the process model.

4.4 Usability and conclusions

Business Process Discovery is knows as a discipline of business process management,that scientifically extracts the implemented business processes and aims to create a AS-IS

Activity name	***Description***
Create contract	It is a user activity that creates a contract with all its components; articles, discount, customer etc...
Create Order	It is a service activity that creates an order to the supplier if the articles are not in stock
Send Order	It is a sending activity that sends a notification of the order tot he supplier
Receive Item	It is a receiving activity that notifies the arrival of the items in the warehouse.
Create Invoice for the supplier	It is a service activity where an invoice is created to the supplier when it has delayed the order
Tour planning	It is a manual activity where a contract is added to a tour.

Table 4.8: Activities of the final process

baseline of them (Jadhav 2011, Vercruysse 2018, **?**). Manual and automatic techniques are part of the whole discovery process. The manual technique includes interviews, organizational structure and generic value chain analysis. Whereas, the automatic technique uses the ABPD (Automatic Business Process Discovery) tools for discovery of business processes through databases. Sometimes, same as for SHD ECORO, it is required to use both techniques due to existing complexity of the system. Process mining is an automatic approach of learning from the previous behavior which aims to extract aggregate, high-level information about several aspects of the process from some so-known event-logs. It can be applicable to all kind of Information Systems during Process Model Discovery, since the only requirement is that the system should record the actual behavior. The only and very important assumption of process mining technique is the possibility to gather a process log with data ordered based on the time that the events take place. Three pre requirements of the log data are; to present a task which is a well defined step in the process, to represent a case which is a process instance and to be totally ordered based on the time stamp. From the input can be discovered three perspectives of a business organizational, process and case prospective. (Cook & Wolf 1998, Van der Aalst et al. 2004, van der Aalst et al. 2007, Mans et al. 2008)However, the focus in this thesis is to discover

the business processes. The tools that help in process mining are ***Disco***, which comes in hand generally for data import, data filtering, process statistics, format converting etc and ***ProM***, which helps in mining the data by using diverse algorithms, analyzing the data etc. Some important algorithms used for mining are alpha ++, heuristic and inductive algorithm. (Van Dongen et al. 2009, Rozinat et al. 2007) Among them, heuristic and inductive miner are selected as most effective ones based on three quality criteria; fitness, precision and generalization. The forth quality criteria is simplicity which indicates the simplicity of the extracted model. Extended description of pros and cons for each algorithm are provided in subsection 4.2.2. In process mining approach extracting, analyzing and filtering the data are crucial steps to deliver an authentic and representative model. The performance of this technique is not very effective in the case of SHD ECORO due to following reasons:

1. *Complexity of the nature of the data.* The log data do not have specified activity name. An in depth analyzing process was necessary to restructure the input data for the mining algorithms.

2. *Lack of information.* The case id, which in the log data should represent the business process, in SHD ECORO is the process id of the operating system. Since a transaction in SHD ECORO is a specific process happening in the system the solution for this issue was using the transaction id generated by transaction type which indicates the beginning and ending of one transaction.

3. *Complex output.* A big model is provided from the filtered data, which requires further interpretations, such as highlighting parts in the model that represent specific sub-processes and then relate them to each other.

4. *Additional techniques needed.* The extracted model has as activities the names of the tables which are changed during the process flow. Further manual approaches are needed to create a representative final model.

According to Szirbik (2005), the logic embedded in any ERP system can be divided into two main parts: the business logic and the session logic. The automatic technique which

is totally based on process mining provides an insight of the session logic of an ERP. It consist of steps that a user undertakes to complete a process in the system. For example in the case of SHD ECORO, this logic is hidden behind the saving process of a contract. Both models extracted from heuristic and inductive miner present the same scenario. They contain sequences of contract saving process steps in SHD ECORO. However, the manual technique come in hand to discover the business logic, which is the logic that is executed automatically in the background in an ERP system. For example, one user is aware of the task “receive items from the supplier”, but are not explicitly called by the user session.

As a conclusion, the business discovery process for SHD ECORO is implemented as a join of both automatic and manual techniques. In the manual approach, Porters Value Chain concept is helpful in developing a big but generalized picture of all the activities that are involved in the creation of finished products and how these processes are interrelated. Models delivered from the automatic technique as well as outer information from developers and project managers were useful to establish a final representative model.

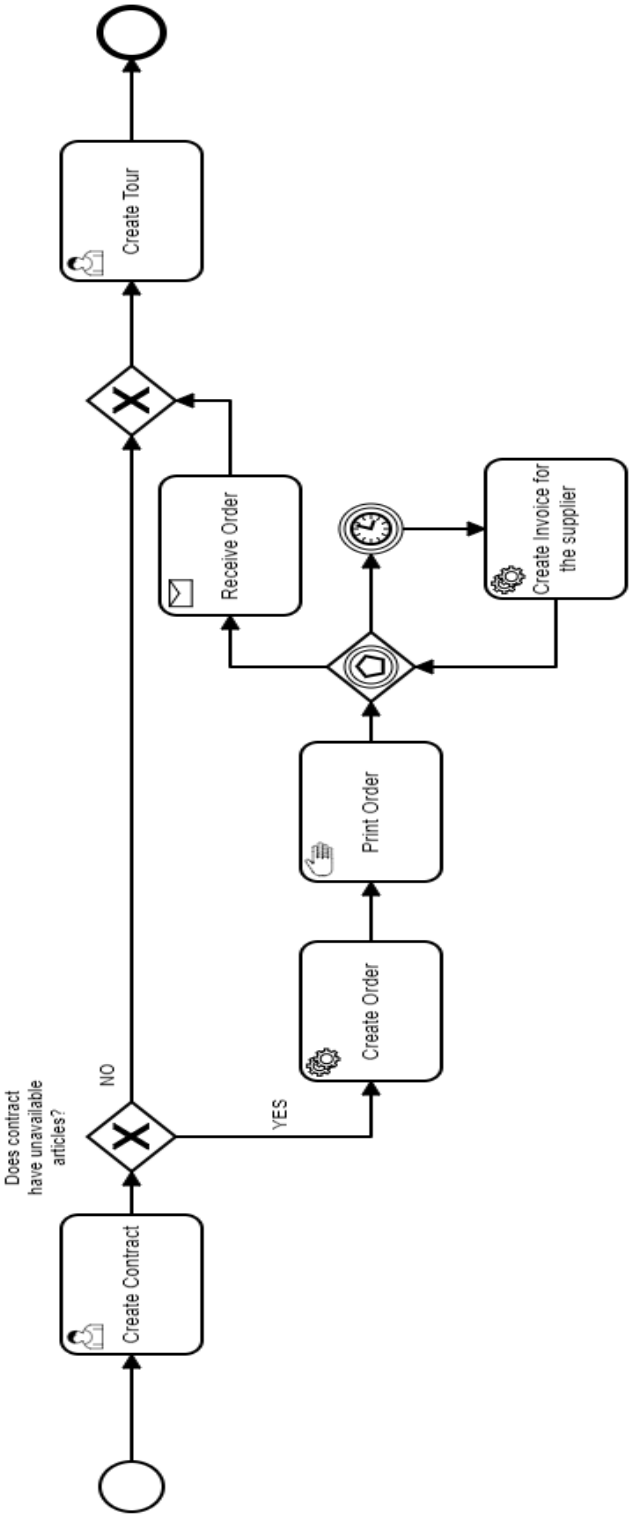


Figure 4.20: Process Model

Chapter 5

Modeling and executing a Business Process

Referring to Cardoso et al. (2004), in ERP systems like SHD ECORO apart from configured parameters there is no inherent flexibility of process execution due to being hard or semi-hard-coded applications. Differently from WfMS the workflow model in ERP systems is not explicitly specified because it is embedded in the applications and the parameter tables. Therefore, the recent trend from ERP vendors is to integrate a WfMS into ERP systems (Cardoso et al. 2004). This chapter provides an introduction to Camunda as a WfMS as well as its integration to SHD ECORO, where the main decision making parts of the model (mostly known as gateways) should be implemented in accordance to the identified business rules.

5.1 Camunda BPM platform and its implementation

Camunda workflow engine executes most of the symbols defined in the BPMN and it contains two very useful components such as; Camunda tasklist where the end users can

organize and work on BPMN user tasks they are supposed to complete and Camunda cockpit which provides a dashboard of running BPMN process models that allows an overview of the process instances running in the background (*Camunda Documentation* 2019). The Camunda Framework is very flexible and can be integrated into various ways (figure 5.1) such as: embedded, shared and standalone engine. All of them are integrated by following diverse configuration steps for different scenarios.

The first case is to embed the process engine as an application library in a server application (Embedded). If there are several applications on a server that use a process engine, this must be integrated in every single application and started in the application, if necessary even configured. Therefore it brings extra effort since there are parallel instances that perform the same tasks. Moreover, it is not applicable in programs written in other programming languages rather than Java. This approach has its advantages including, easy to use Java API, possibilities to write proper unit tests and also integrating transaction tasks. The deployment path is clear since the engine starts whenever the application is started. Nevertheless, a high number of deployed versions are generated, which disturb the visibility of the business tasks background in a given moment in time and the central monitoring like Camunda cockpit is missing.

The second case is to use the process engine as a container service, which can be shared by all applications deployed inside the container. This prevents the problem of additional server load due to instances of the process engine running in parallel created in the first case, because there is a one to one relation between deployments and applications. However, there are scenarios that cannot be satisfied using this method and the other integration method explained in the following paragraph should be used.

The third case is to apply standalone process engine, in which Camunda runs in its own separate server and communicates via network services with other applications. This approach is applicable in situations when multiple applications run on different servers but in the same network and should use a (same) process engine. Furthermore, it allows Camunda to interact with non java applications. Nevertheless, the provided web services like REST API (**R**epresentational **S**tate **T**ransfer **A**pplication **P**rogramming **I**nterface) do

not complete the full functionality of the JAVA API in Camunda.

The third approach is used in the case of SHD ECORO because of the following reasons:

1. According to Szirbik (2005), embedded engine have usually only peripheral role (mainly in document flows) and they have no operational link with the main business processes supported by the core of the ERP system. Therefore, this approach is excluded because it is not the desired scenario for SHD ECORO, where there should be also a control of the workflow.

2. The process logic in the code of SHD ECORO is hardwired to other legacy applications. Thus, in this situation a WfWS should be as a "middleware" platform serving to integrate diverse applications maybe running in various servers, meaning that the lifecycle of the engine should depend neither on the applications nor on the server lifecycle.

3. Camunda standalone provides a fairly good separation of complexity so that everything is not an over-complex information mess.

In a WfMS, the idea is to first be able to model processes by using visual tools such as Camunda Modeler. The next section deals with it.

5.2 Process modeling in compliance to business rules

As also mentioned in the beginning of this thesis, designing is fundamental and crucial step of BPM lifecycle. Until this chapter, every undertaken approach has contributed to finalize the design of a process model for SHD ECORO. The whole phase includes business rule and business process discovery and their integration together. Afterwards the flow logic found in SHD ECORO system is being separated out and captured in the WfMS process model. Hence, the aim in this section is to model the process based on the information generated from the automated techniques which have shown the inner process running in SHD ECORO and the manual technique in which are specified the steps of a

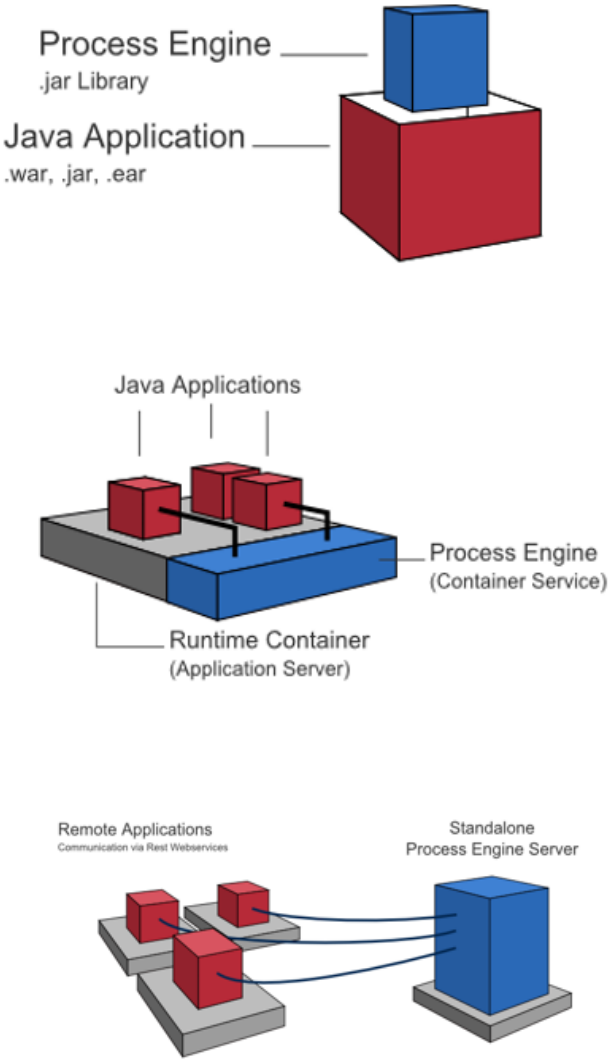


Figure 5.1: Camunda BPM Platform architecture. (source: *Camunda Documentation* (2019))

sales process from an upper layer. Some conclusions can be drawn by comparing the generated models, one shown in the figure 4.20 and the other depicted in figures 4.18 and 4.17:

1. The smallest instance indicator, which is the unique business key that references a single instance, is in this case the position of either a contract or an order. It

represents the articles part of the contract/order that is being proceeded. Hence, one process instance will have a business key equal to the position id either of a contract or an order depending on the initial starting task.

2. Some activities like adding a discount or an address to the contract are executed whenever the button of saving a contract is pressed since SHD ECORO is page oriented not field oriented meaning that it deals with the data in a session only after the user submits the data (Szirbik 2005). Therefore, the task of saving a contract is a black box task that is invoked when the button save is clicked.

Accordingly, the extracted process workflow is displayed in the figure 5.4. Additionally to business process tasks also the business rules can be implemented in business rule tasks which help in the configuration of the exclusive gateways. A simple example will be presented in the process of SHD ECORO as displayed in the figure 5.3, where the status of a contract retrieved from the system decides whether it has articles that are not in stock and should be ordered or not. In SHD ECORO the status of a contract position can have the value “B” (Bestellung) which indicates an unavailable article that should be ordered, or it can have the value “L” (Lager), which tells that an article is available in the warehouse. One of the implementation methods of a business rule task is using a DMN (Decision Model and Notation) tables, which make decisions based on their data entries specified as input and output same as depicted in the figure 5.2.

Decision 1

check-status

U	Input +	Output +
	status status	is Order
	string	boolean
1	"B"	true
2	"L"	false
+	-	-

Figure 5.2: DMN Table in Camunda

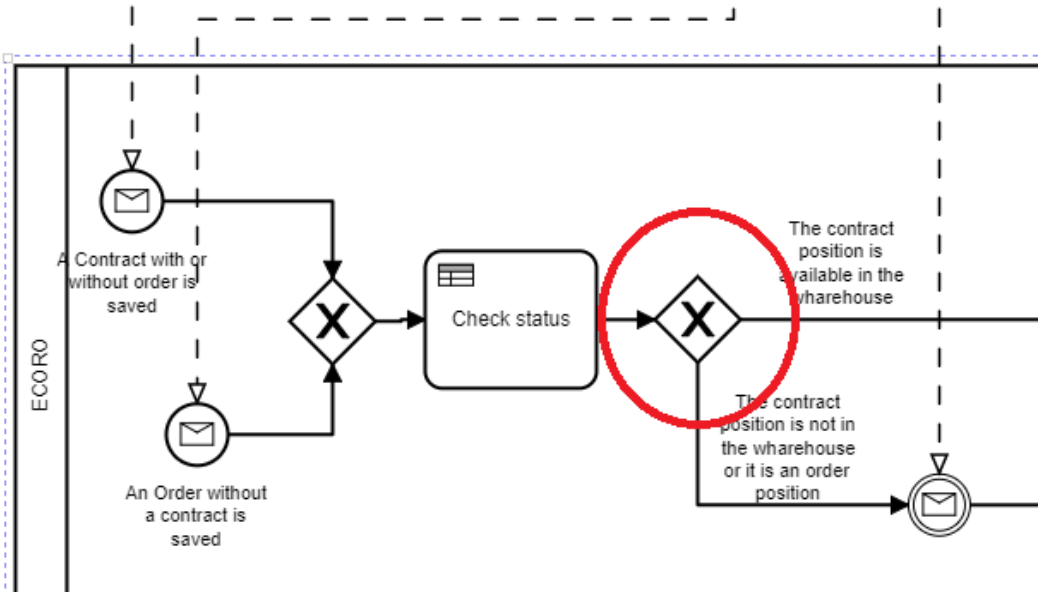


Figure 5.3: Business Rule Task integration in the Process Model

5.3 Execution of the process and test of the integration

The WfMS interacts with the ERP through web services. The process engine is used as a standalone one, which communicates with SHD ECORO via REST API. SHD ECORO generates messages/signals at various times, and Camunda listens for them and executes processes based on those messages/signals or are activated manually by users in SHD ECORO. The process workflow will be shown afterwards in Camunda cockpit. The way of how the existing tasks are generated in Camunda is explained in the table 5.1.

The next step is to find the right places in SHD ECORO source code from where to sent requests via REST services. In this case the help of the developers of SHD ECORO is required. After implementing the required code in SHD ECORO, the process can be started whenever a contract or an order is created in the system. To test the integration, two process instances are generated by; first creating a contract which has one article not available in the warehouse, secondly creating a separated order that has one article. After this execution both instance processes will be waiting for the orders to be sent to

Activity name	***Description***
A contract with or without order is saved	It is a message which states that a contract position (being or not being available in the warehouse) is saved by the seller. This message starts the process workflow.
An order without a contract is saved	It is a message which states that an order position (not having a contract position attached) is saved by the seller. This message starts a process instance.
The order position is sent	It is a message which notifies that an order position is printed, that means sent to the supplier.It is correlated by the business key which is either the contract position id or the order position id, depending on the starting instance conditions.
The order position is received	It is a message which informs that an order position is received and available in the warehouse. Same as the previous task it is correlated by the business key of the process instance.
The tour is created	It is a message generated every time the seller creates a tour for a contract position.

Table 5.1: Activities of the final process

the supplier as also shown in Camunda cockpit in figure 5.5. It might happen that in another moment in time the order arrives in the warehouse. This activity is generated by the message “The order position is received”. The new view of the process workflow is presented in the figure 5.6.

Diverse process variables can be attached to every instance, providing more information about it as well as a better overview of the activities standing by. For example, when a contract position is created variables such as the contract number, the customer name can be added. Whereas when an order position is created the information about the order number and the supplier can be set to the message.

Moreover, in the WfMS can be implemented a java application which executes diverse service tasks in the process with the aim of providing a higher-level view of what is going on. A simple example in the case of SHD ECORO could be to add a service task after

receiving the items, which performs statistical calculations about the suppliers. More specifically, the days required to complete the delivery of the items will be saved for each supplier. The data can be used to answer some performance questions about the suppliers that directly influence the process execution such as:

1. What is the average number of days needed to receive the items from a given supplier.

2. Which are the most/less reliable suppliers, the ones that do not have delays in their deliveries?

Some benefits expected from the implementation of Camunda for SHD ECORO are; flexibility in monitoring business process instances timing and state and tight control around compliance by having a better view into gateways configurations.

5.4 Usability and Conclusions

The Camunda Framework is very flexible and can be integrated into various ways such as: embedded in which the engine is added as a jar (Java ARchive) file, shared, where the engine and the application stand in the same container and standalone engine, in which the engine runs in its own remote server and communicates with the application via REST services. Even though, the best supported method of Camunda engine is the shared one, there are cases when it cannot be used such as situations when the programming language of the software is not java. But, for SHD ECORO it is used the standalone Camunda due to possibility to have more than one application integration, more than just a workflow documentation and a fairly good separation of complexity.
The implemented WfMS in this case is a continues communication between Camunda and SHD ECORO with the aim to display the running process instances and their conditions, which in ECORO are configured using system parameters. When Camunda as a WfMS interacts with SHD ECORO as an ERP system, the system is the source of record/data, and the WfMs is the source of the steps that are followed.

In the WfMS can be implemented also a Java application that can execute diverse service tasks which help in generating reports used to analyze the process by providing a higher view level.

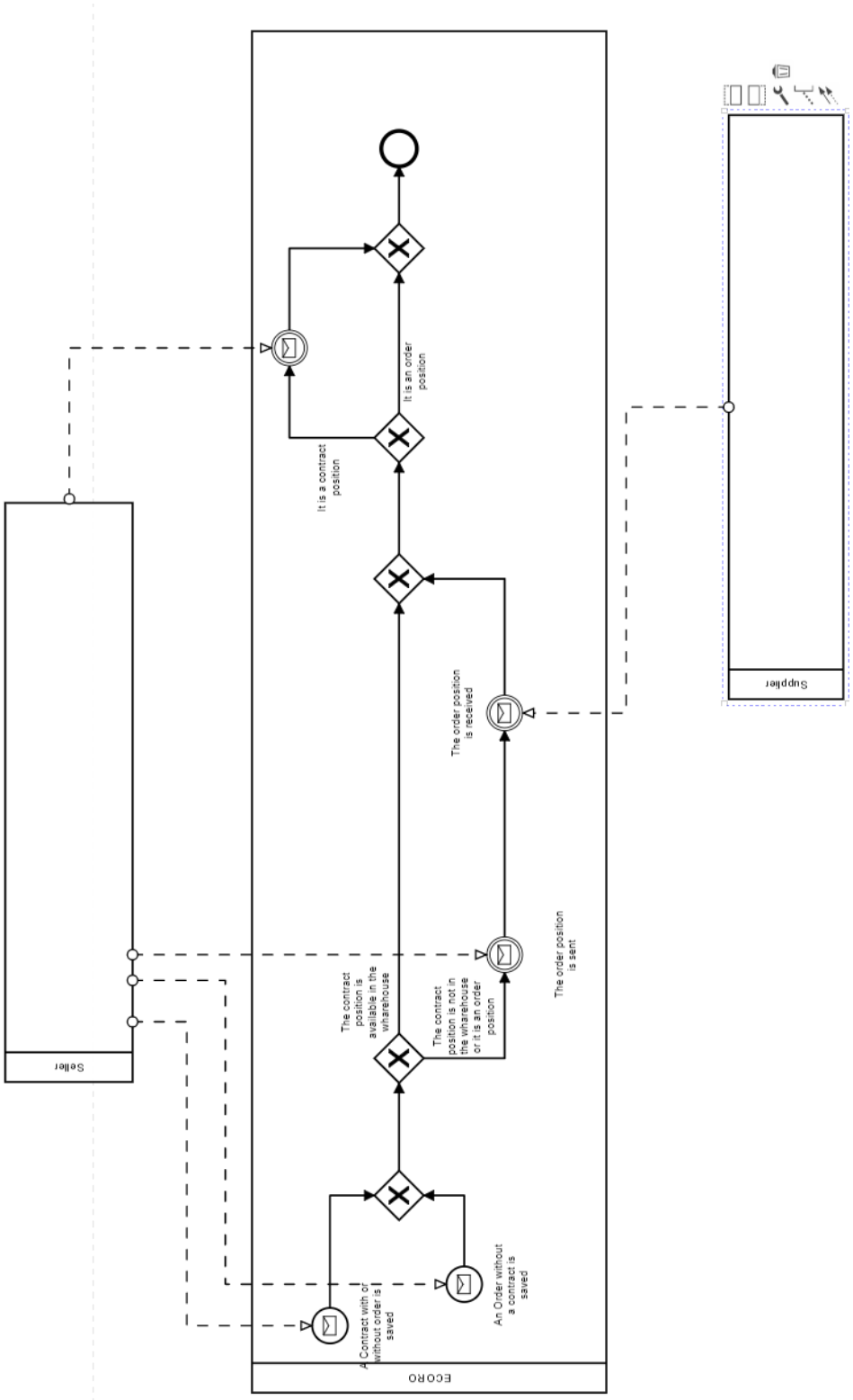


Figure 5.4: Process Model

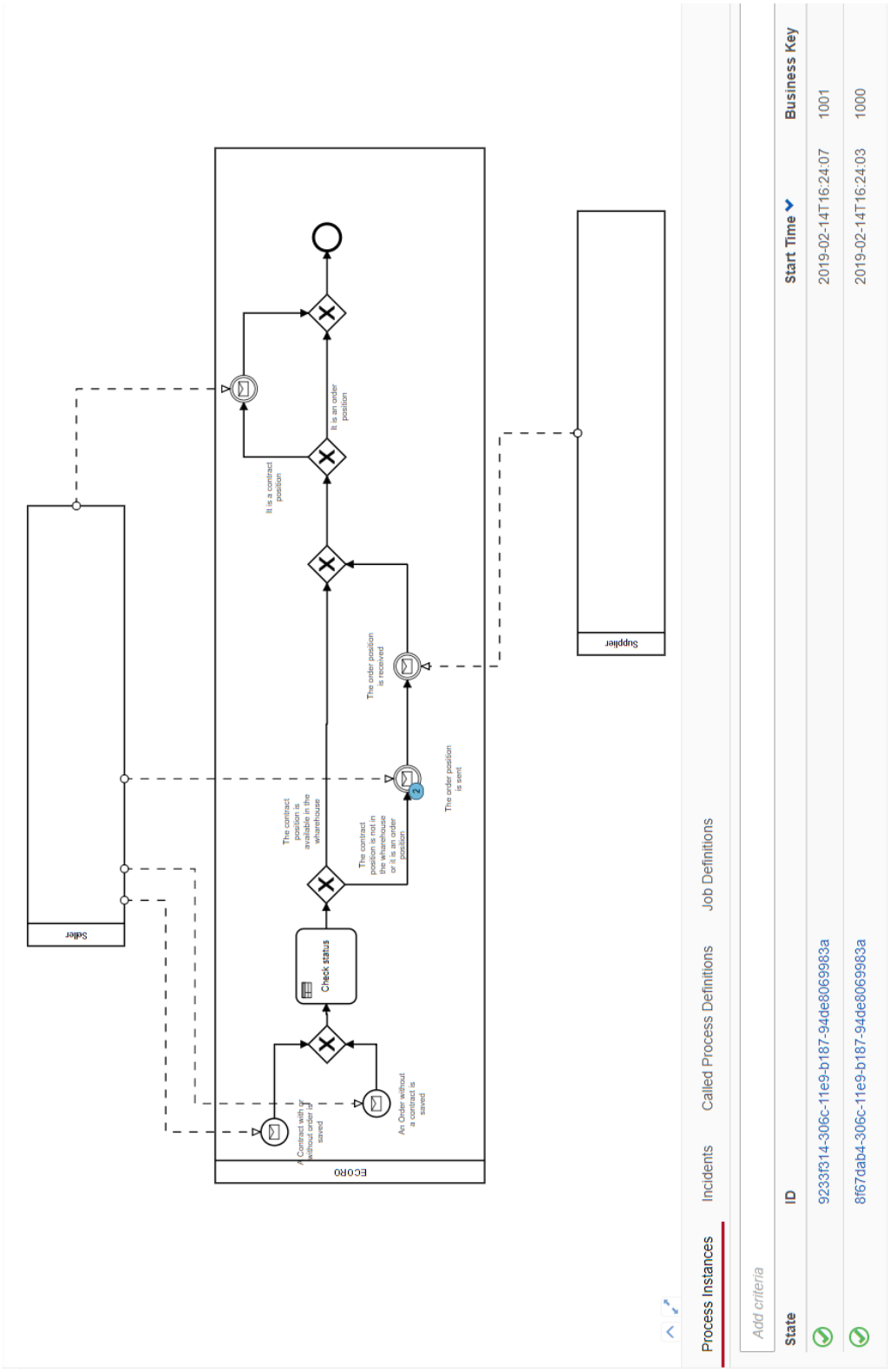


Figure 5.5: Two Process Instances Running

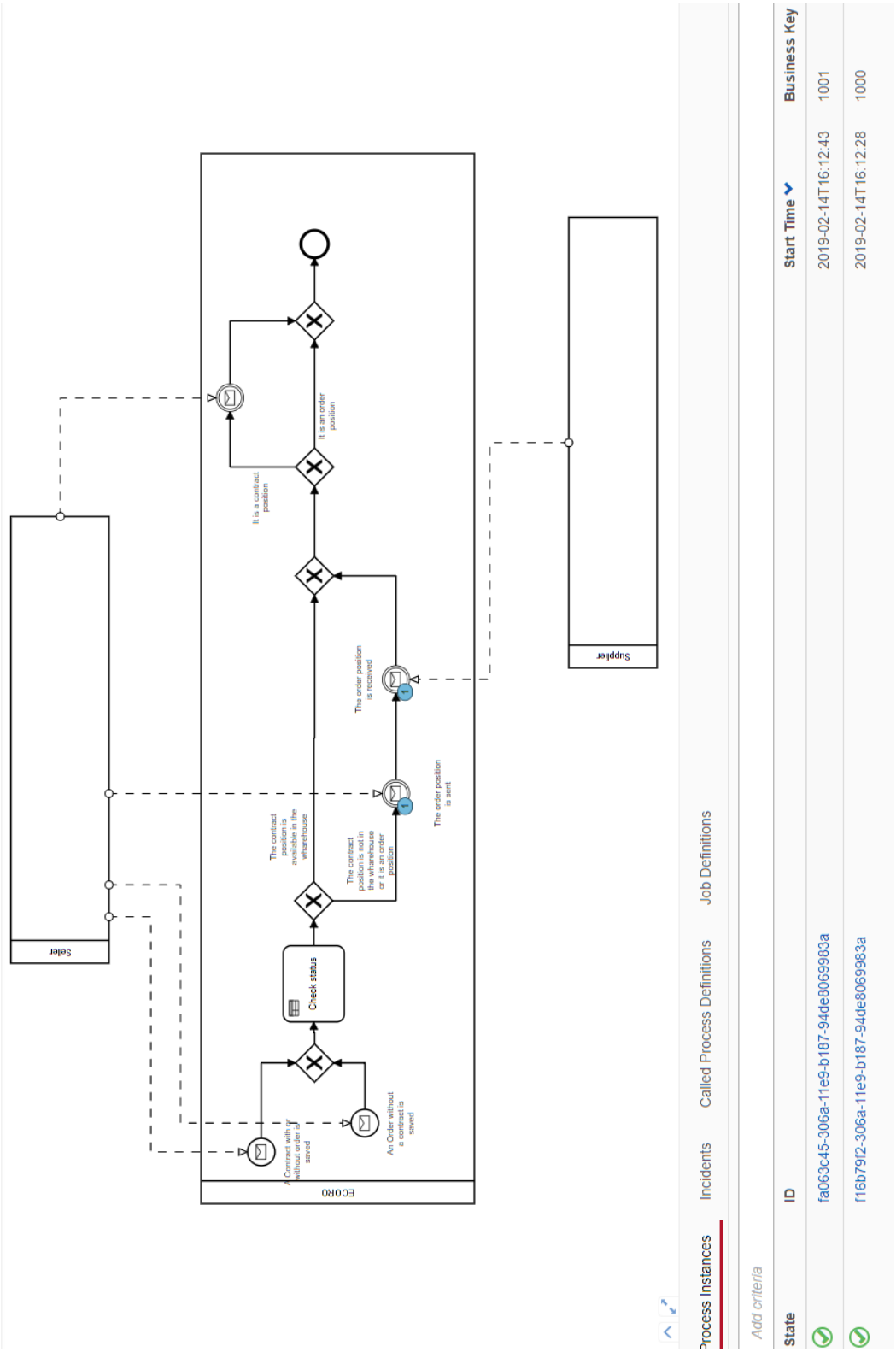


Figure 5.6: Two Process Instances Running (next execution)

Chapter 6

Conclusion

This chapter provides an outline of the way the research goals are recognized and accomplished. Since every chapter of this thesis contains a section of conclusions, here they are summarized and explained in short terms. Lastly, this chapter presents possible future directions of the research.

6.1 Summary

This thesis starts with a presentation with SHD ECORO and how it is related to workflow control and compliance issues. Afterwards it lays the basic concepts of Business Process Management, its integration advantages, what requirements should a WfMS fulfill and which are the most used alternatives of them in the market. After a small analyse based on users comments, Camunda is selected for the following work in this research.

Since business rules are an integral part of a software modernization project, the performance of extraction approaches of finding encoded rules in SHD ECORO source code are presented in a separated chapter of this thesis. Three methods were used to achieve this goal and several limitations were encountered.

The first method is called JBreX. It is a tool that deals with java applications by classifying some pre selected domain variables, identifying chunks of code related to them

and then presenting the results as text and graph format. Problems raised in this case are linked to the large number of variables that should be selected, the enormous number of lines of the code which directly affects the performance of the tool. This approach does not provide results for SHD ECORO and all similarly big softwares.

The second approach is based on parsing the source code to find if statements in the business logic layer. It focuses on the conditional shape of rules in general and the fact that business rules are not encoded either in UI (User Interface) logic or database access. The results in this approach are not satisfying because of the huge amount of retrieved data, poor representation of the variables present in the code and the fact that a large part of the SHD ECORO business logic is written as SQL statements. The manual work to identify if statements that represent a real business rule makes the overall results of this method questionable.

The third method focuses on parsing the source code to find if statements in the data access layer. Same as the previous approach it is based on the conditional form of the rules. But it considers also database access parts because most SHD ECORO business logic is written as SQL statements. Different from previous methods, the results here are generally better. However there are found more structural than behavioral rules. The problem is that not all structural rules can be considered as business rules. Therefore it is needed a manual check of the results which is difficult due to poor representation of the variables as well as large amount of if statements.

Apart from business rules, business processes are a fundamental part of a WfMS (Georgakopoulos et al. 1995, Szirbik 2005). Therefore a whole chapter of this thesis is dedicated in discovering processes embedded in an ERP system. Manual as well as automated techniques are used to generate a representative process model from SHD ECORO. Firstly the log data are used as an input of process mining algorithms. It is concluded that heuristic and inductive miner perform better in terms of fitness, precision and generalization. The problem faced during analyzing log data is that the process id represents the operation system process and not the business process. Hence the beginning and ending of a database transaction is used to attach unique identifiers for a transaction. The results

display a transnational process in SHD ECORO, where the name of tables changing during the process represent the tasks part of the workflow. The manual technique based on Porterś value chain creates a better representation of the selling process which is taken as an example in this thesis. Finally, models delivered from the automatic technique as well as outer information from developers and project managers were useful to establish a final representative model.

Last step of the research consists on integrating a WFMS based on the elements found in the previous work. It helps in controlling the workflow and insuring the compliance of the process with the business rules. The implemented WfMS in this case is a continues communication between Camunda and SHD ECORO with the aim to display the running process instances and their conditions, which in SHD ECORO are configured using system parameters. When Camunda as a WfMS interacts with SHD ECORO as an ERP system, the system is the source of record/data, and the WfMs is the source of the steps that are followed.

As a conclusion, the compliance analysis can be done upon an ERP system by integrating a WfMS system with it, but the undertaken steps to achieve it require loads of work and analysis which increase with the growth of system size.

6.2 Future Work

One of the possible directions of future work is closely related to the rule extraction approaches. It would be interesting to investigate the results of the methods proposed in this research by:

1. Including other similarity measures rather than string based ones to compare the domain variables with the condition in the if statement. According to Vijaymeena & Kavitha (2016), words are similar in two possible ways; lexically and semantically. In this thesis the approaches check only whether the if-condition and the domain variable are lecixally similar. But, considering the fact that the variables are not very representative this method provides questionable results. Thereby,

corpus-based and knowledge-based algorithms should be used to discover the semantic similarity.

2. Clustering the results by using clustering algorithms based on the similarity measures (Lin et al. 2014), for example string-based. It would strongly help in narrowing the manual work done during processing of the final data.

Another direction is specifically related to SHD ECORO and its WfMS integration. In this thesis Camunda is integrated as a standalone engine that just listens to the ERP system and shows the workflow. But it is possible to generate some tasks by calling SHD ECORO from Camunda. For instance, the task “The order position is sent” can be implemented as an external activity in the model that is executed automatically by some job generator in SHD ECORO. This approach provides automated workflow controlled by Camunda.

Appendix A

Pseudo code of the implementation in SHD ECORO

```
if a new contract is created
{
   generate all contract positions paired with their availability status (L/B);

   for each contract position

   {
       create a request body that contains the businessKey = contract position
       id and variables status (that indicates the availability) and contract
       number.

       then send the message request to Camunda
   }
}
```

Figure A.1: Activation of a starting event generated from contract creation.

```
if a new order is created
{
   generate the order position id-s that do not have any contract id attached to them.

   for each order position

   {
        create same request body but here the businessKey = order position id, status
        is always B and it is attached the order number.

        then send the message request to Camunda
   }
}
```

Figure A.2: Activation of a starting event generated from order creation.

```
if some order positions are sent
{
   for order positions that have a contract position attached to them
   {
        create message request that contains businessKey=contract position id and
        contract number

       send the request to Camunda
   }

   for order positions that do not have a contract position attached to them
   {
        create message request that contains businessKey=order position id and
        order number

       send the request to Camunda
   }
}
```

Figure A.3: Execution of a task that indicates that the order is sent to the supplier.

Bibliography

Bajwa, I. S., Lee, M. G. & Bordbar, B. (2011), Sbvr business rules generation from natural language specification., *in* 'AAAI spring symposium: AI for business agility', pp. 2–8.

Bergenthum, R., Desel, J., Kölbl, C. & Mauser, S. (2008), Experimental results on process mining based on regions of languages, *in* 'Proceedings of the Workshop CHINA, Petri Nets', pp. 73–87.

Bisbal, J., Lawless, D., Wu, B. & Grimson, J. (1999), 'Legacy information systems: Issues and directions', *IEEE software* **16**(5), 103–111.

BPMN Tool Matrix (2018), `https://bpmnmatrix.github.io/`.

Caballero, H. S. G., Westenberg, M. A., Verbeek, H. M. & van der Aalst, W. M. (2017), Visual analytics for soundness verification of process models, *in* 'International Conference on Business Process Management', Springer, pp. 744–756.

Camunda Documentation (2019), `https://docs.camunda.org/manual/7.10/`.

Cardoso, J., Bostrom, R. P. & Sheth, A. (2004), 'Workflow management systems and erp systems: Differences, commonalities, and applications', *Information Technology and Management* **5**(3-4), 319–338.

Chaparro, O., Aponte, J., Ortega, F. & Marcus, A. (2012), Towards the automatic extraction of structural business rules from legacy databases, *in* 'Reverse Engineering (WCRE), 2012 19th Working Conference on', IEEE, pp. 479–488.

Conforti, R., La Rosa, M. & ter Hofstede, A. H. (2017), 'Filtering out infrequent behavior from business process event logs', *IEEE Transactions on Knowledge and Data Engineering* **29**(2), 300–314.

Cook, J. E. & Wolf, A. L. (1998), 'Discovering models of software processes from event-based data', *ACM Transactions on Software Engineering and Methodology (TOSEM)* **7**(3), 215–249.

Cosentino, V., Cabot, J., Albert, P., Bauquel, P. & Perronnet, J. (2012), A model driven reverse engineering framework for extracting business rules out of a java application, *in* 'International Workshop on Rules and Rule Markup Languages for the Semantic Web', Springer, pp. 17–31.

De Weerdt, J. (2012), 'Business process discovery: new techniques and applications.'.

Dekker, H. C. (2003), 'Value chain analysis in interfirm relationships: a field study', *Management accounting research* **14**(1), 1–23.

Financesonline (2018), *What Is Business Process Management Software, Analysis of Features, Types, Benefits and Pricing*. `https://financesonline.com/business-process-management-software-analysis-features-types-benefits-pri`

Gayialis, S., Papadopoulos, G., Ponis, S., Vassilakopoulou, P. & Tatsiopoulos, I. (2016), 'Integrating process modeling and simulation with benchmarking using a business process management system for local government', *International Journal of Computer Theory and Engineering* **8**(6), 482.

Geambaşu, C. V. (2012), 'Bpmn vs uml activity diagram for business process modeling', *Accounting and Management Information Systems* **11**(4), 637–651.

Georgakopoulos, D., Hornick, M. & Sheth, A. (1995), 'An overview of workflow management: From process modeling to workflow automation infrastructure', *Distributed and parallel Databases* **3**(2), 119–153.

Gillot, J.-N. (2008), *The Complete Guide to Business Process Management*.

Günther, C. W. (2009), 'Process mining in flexible environments'.

Harmon, P. & Wolf, C. (2008), 'The state of business process management', *Business process trends* .

Herbert Gomez Tobon, A. F. J. F. (2010), 'Business rules extraction from business process specifications written in natural language', *Business Rules Journal* **11**(7).
URL: *https://www.brcommunity.com/articles.php?id=b543*

Huang, H., Tsai, W.-T., Bhattacharya, S., Chen, X., Wang, Y. & Sun, J. (1996), Business rule extraction from legacy code, *in* 'Proceedings of 20th International Computer Software and Applications Conference: COMPSAC'96', IEEE, pp. 162–167.

Huijsman, K. & Noordveld, P. (2017), 'Bpm based erp implementation'.

IT Central Station (2018), https://www.itcentralstation.com/categories/business-process-management.

Jadhav, S. (2011), 'Business process discovery', *BP Trends (February)* .

Ko, R. K. (2009), 'A computer scientist's introductory guide to business process management (bpm)', *Crossroads* **15**(4), 4.

Ko, R. K., Lee, S. S. & Wah Lee, E. (2009), 'Business process management (bpm) standards: a survey', *Business Process Management Journal* **15**(5), 744–791.

Kothari, C. R. (2004), *Research methodology: Methods and techniques*, New Age International.

Leemans, S. J., Fahland, D. & van der Aalst, W. M. (2013), Discovering block-structured process models from event logs containing infrequent behaviour, *in* 'International conference on business process management', Springer, pp. 66–78.

Lin, Y.-S., Jiang, J.-Y. & Lee, S.-J. (2014), 'A similarity measure for text classification and clustering', *IEEE transactions on knowledge and data engineering* **26**(7), 1575–1590.

LLC, U. (2017), 'Business rules extracted from code'.

Lourens, A. & Jonker, J. (2013), 'An integrated approach for developing a technology strategy framework for small- to medium-sized furniture manufacturers to improve competitiveness', *The South African Journal of Industrial Engineering* **24**(1), 50–67.
URL: *http://sajie.journals.ac.za/pub/article/view/500*

Macintosh, A. & Whyte, A. (2008), 'Towards an evaluation framework for eparticipation', *Transforming government: People, process and policy* **2**(1), 16–30.

Mans, R. S., Schonenberg, M., Song, M., van der Aalst, W. M. & Bakker, P. J. (2008), Application of process mining in healthcare–a case study in a dutch hospital, *in* 'International joint conference on biomedical engineering systems and technologies', Springer, pp. 425–438.

Niels Lohmann, S. M. (2012), Bpm 2012 demonstration track.

Paknikar Shantanu, Anand Abhishek, P. K. K. K. K. B. D. (2014), 'Rules harvesting from source code'.
URL: *https://www.happiestminds.com/whitepapers/Rules-Harvesting-from-Source-Code.pdf*

Perez-Castillo, R., Weber, B., Pinggera, J., Zugal, S., de Guzman, I. G.-R. & Piattini, M. (2011), 'Generating event logs from non-process-aware systems enabling business process mining', *Enterprise Information Systems* **5**(3), 301–335.

Pizzoleto, A. V. & do Prado, A. F. (2016), An approach for analysis and flexibility of business rules in legacy systems, *in* 'Proceedings of the International Conference on Software Engineering Research and Practice (SERP)', The Steering Committee of The World Congress in Computer Science, Computer , p. 139.

Porter, M. E. (2001), 'The value chain and competitive advantage', *Understanding Business Processes* pp. 50–66.

Quesada, H. & Gazo, R. (2007), 'Methodology for determining key internal business processes based on critical success factors: A case study in furniture industry', *Business Process Management Journal* **13**(1), 5–20.

Rico, A. (2017), 'Manufacturing kpis that matter most', `https://icharts.net/blog/netsuite-reporting-and-analytics/manufacturing-kpis-matter`.

Ross, R. G. (2003), *Principles of the business rule approach*, Addison-Wesley Professional.

Rozinat, A., De Medeiros, A. A., Günther, C. W., Weijters, A. & Van der Aalst, W. M. (2007), 'Towards an evaluation framework for process mining algorithms', *BPM Center Report BPM-07-06, BPMcenter. org* **123**, 142.

Rudden, J. (2007), 'Making the case for bpm-a benefits checklist', *BPTrends 2007* .

Sneed, H. M. (2001), Extracting business logic from existing cobol programs as a basis for redevelopment, *in* 'Program Comprehension, 2001. IWPC 2001. Proceedings. 9th International Workshop on', IEEE, pp. 167–175.

Sneed, H. M. & Erdos, K. (1996), Extracting business rules from source code, *in* 'Program Comprehension, 1996, Proceedings., Fourth Workshop on', IEEE, pp. 240–247.

Szirbik, H. W. . N. (2005), 'Erp and workflow systems. do they work together?', *Information Systems Cluster* .

Tammy, A. (2017), 'Determining your bpm software requirements', `http://www.bpminstitute.org/resources/articles/determining-your-bpm-software-requirements`.

van der Aalst, W. M., Reijers, H. A., Weijters, A. J., van Dongen, B. F., De Medeiros, A. A., Song, M. & Verbeek, H. (2007), 'Business process mining: An industrial application', *Information Systems* **32**(5), 713–732.

Van der Aalst, W. M., van Dongen, B. F., Günther, C. W., Rozinat, A., Verbeek, E. & Weijters, T. (2009), 'Prom: The process mining toolkit.', *BPM (Demos)* **489**(31), 2.

Van der Aalst, W., Weijters, T. & Maruster, L. (2004), 'Workflow mining: Discovering process models from event logs', *IEEE Transactions on Knowledge & Data Engineering* (9), 1128–1142.

Van Dongen, B. F., De Medeiros, A. A. & Wen, L. (2009), Process mining: Overview and outlook of petri net discovery algorithms, *in* 'Transactions on Petri Nets and Other Models of Concurrency II', Springer, pp. 225–242.

Van Dongen, B. F., de Medeiros, A. K. A., Verbeek, H., Weijters, A. & Van Der Aalst, W. M. (2005), The prom framework: A new era in process mining tool support, *in* 'International conference on application and theory of petri nets', Springer, pp. 444–454.

Vercruysse, J. (2018), '7 methods for business process discovery'.

Vijaymeena, M. & Kavitha, K. (2016), 'A survey on similarity measures in text mining', *Machine Learning and Applications: An International Journal* **3**(2), 19–28.

Weijters, A., van Der Aalst, W. M. & De Medeiros, A. A. (2006), 'Process mining with the heuristics miner-algorithm', *Technische Universiteit Eindhoven, Tech. Rep. WP* **166**, 1–34.

Weske, M. (2012), Business process management architectures, *in* 'Business Process Management', Springer, pp. 333–371.

Wohed, P., van der Aalst, W. M., Dumas, M., ter Hofstede, A. H. & Russell, N. (2006), On the suitability of bpmn for business process modelling, *in* 'International conference on business process management', Springer, pp. 161–176.