AALTO UNIVERSITY
SCHOOL OF SCIENCE AND TECHNOLOGY
Faculty of Information and Natural Sciences
Department of Industrial Engineering and Management

Antti Manninen

APPLYING THE PRINCIPLES OF PROCESS MINING TO FINNISH HEALTHCARE

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering

16.3.2010

Supervisor: Professor Paul Lillrank
Instructor: M.Sc. Teemu Lehto

AALTO UNIVERSITY
SCHOOL OF SCIENCE AND TECHNOLOGY
Faculty of Information and Natural Sciences
Industrial Engineering and Management

ABSTRACT OF THE
MASTER´S THESIS

Author: Antti Ilari Manninen

Subject of the thesis: Applying the Principles of Process Mining to Finnish Healthcare

Number of pages: 12 + 98

Date: 16.3.2010

Library location: TU

Professorship: Industrial Engineering

Code of professorship: TU-22

Supervisor: Professor Paul Lillrank, Aalto University School of Science and Technology

Instructors: Teemu Lehto, Master of Science in Engineering, QPR Software Plc

Healthcare patient records contain a large amount of data, but this data is not used in a systematic, process development oriented way. The picture of what happens in the process across organizational boundaries is often unclear. Process mining is a technique for programmatically extracting process information from event logs. This study takes event logs from Finnish healthcare sector and applies process mining to these logs. The logs are taken from Meilahti and Töölö hospitals, both located in the Helsinki area.

In this study a product prototype applying a refined process mining algorithm is developed and validated. This prototype produces process model based on the activities and transitions found from the log and groups the separate process instances found from the log to classification groups. The resulting models are presented graphically. Statistics and figures are shown in the resulting analysis models to enable fast analysis of processes.

The process model created by process mining is a fact-based model describing patient episodes. This principle of drawing a process model is fundamentally different from traditional, expert driven process modeling. Process models created this way could be used, for example, for validation or to give insight into the current status of the processes.

The reception caused by the process mining based product prototype has been positive and the resulting process models drawn using it have aroused interesting discussions. Some challenges still exist, especially with visual properties of the drawn models as well as with the performance of the algorithm, but it looks clear that the tool would provide clear business benefits.

Keywords: Process mining, healthcare, activity, flow, episode, evidence-based medicine

Publishing language: English

AALTO-YLIOPISTON TEKNILLINEN KORKEAKOULU
Informaatio- ja luonnontieteiden tiedekunta
Tuotantotalouden tutkinto-ohjelma

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Antti Ilari Manninen

Työn nimi: Prosessilouhinnan periaatteiden hyödyntäminen suomalaisessa terveydenhuollossa

Sivumäärä: 12 + 98

Päiväys: 16.3.2010

Työn sijainti: TU

Professuuri: Teollisuustalous

Koodi: TU-22

Työn valvoja: Professori Paul Lillrank, Aalto-yliopiston teknillinen korkeakoulu

Työn ohjaaja: Diplomi-insinööri Teemu Lehto, QPR Software Oyj

Terveydenhuollon järjestelmiin tallennettavat potilastietokannat sisältävät suuren määrän dataa, mutta tätä dataa ei useinkaan hyödynnetä systemaattisesti prosessien kehityksessä. Organisaatiorajat ylittävä tieto prosessien tilasta on usein epäselvä. Prosessilouhinta (process mining) on tekniikka, jolla prosessitietoa kerätään ohjelmallisesti tietokannoista. Tässä tutkimuksessa käsitellään Suomen terveydenhuoltojärjestelmistä poimittuja potilastietokantoja ja sovelletaan prosessilouhintaa näihin kantoihin. Data on kerätty Helsingissä sijaitsevista Meilahden ja Töölön sairaaloista.

Tutkimuksessa kehitetään tuoteprototyyppi, jonka avulla prosessilouhinta-algoritmin toimintaa voidaan validoida. Tämä prototyyppi tuottaa prosessimallin tietokannasta löytyvistä aktiviteeteista ja niiden välisistä siirtymistä. Lisäksi se ryhmittelee datasta löytyvät prosessi-instanssit keskenään samankaltaisiin luokkiin. Tuloksena syntyvät prosessimallit esitetään graafisesti. Analyysimalleihin lasketaan myös erilaisia tunnuslukuja prosessianalyysin helpottamiseksi.

Prosessilouhinnan tuottama prosessimalli perustuu puhtaasti tosiasioihin ja kuvaa prosessia potilaan kokeman episodin kautta. Tämä periaate prosessimallien piirtämiseen on huomattavan erilainen perinteiseen, asiantuntijalähtöiseen prosessimallinnukseen verrattuna. Tällä tavalla tehtyjä prosessimalleja voidaan käyttää esimerkiksi mallien validointiin tai prosessin nykytilan kartoitukseen.

Tuoteprototyypin saama vastaanotto on ollut positiivista ja sen avulla piirretyt prosessimallit ovat herättäneet vilkasta keskustelua. Prototyypissä on toki edelleen joitakin haasteita, liittyen pääasiassa prosessimallien graafisiin ominaisuuksiin sekä algoritmin suorituskykyyn. Tuotteen kaupallinen potentiaali vaikuttaisi kuitenkin varsin suurelta ja hyödyt selkeiltä.

Asiasanat: Prosessilouhinta, terveydenhuolto, aktiviteetti, virta, episodi, näyttöihin perustuva lääketiede

Julkaisukieli: englanti

Acknowledgements

The topic for my thesis started to formulate when I one beautiful spring morning started asking at QPR Software whether they would have a suitable topic. Little did I know how interesting the topic would gradually evolve to! I must say it has been really fascinating to be able to work on a real business innovation of QPR. Seeing the results and customer responses has been very inspiring; for that I thank the great atmosphere at QPR Software.

I express my sincerest gratitude to my supervisor Professor Paul Lillrank who has been able to direct the course of my thesis to the right academic direction. My instructor at QPR, Teemu Lehto, as well as Antti Miettinen, the innovation manager at QPR, have both worked very hard alongside me in order to bring me the best ideas for further development regarding ProcessAnalyzer.

Researchers Paulus Torkki and Antti Peltokorpi also deserve my grateful thanks; they have helped me to acquire real life healthcare data they used in their own studies. Without their effort the fast development of ProcessAnalyzer would not have been possible. I also thank the people at Nordic Healthcare Group – Vesa Kämäräinen, Anssi Mikola, Pirjo Berg, Patrick Francke and Jussi Tan – who I have met during our discussions regarding the possibilities of ProcessAnalyzer.

From the healthcare sector I would like to thank Markku Mäkijärvi from HUS who provided interesting guidelines for the development of the ProcessAnalyzer prototype in our meeting in June 2009. Last, but not least, I thank Eero Hirvensalo from HUS for letting me utilize the data from Töölö hospital.

Antti Manninen

Espoo, March 2010

Table of Contents

List of figures

List of tables

Explanation of the terms and the abbreviations

AEU: Accident and Emergency Unit (tapaturma-asema in Finnish) at Töölö hospital, Helsinki

Algorithm: An effective and often computerized method of solving a problem using preset sequence of instructions

API: Application Programming Interface

Big O notation: A notation for indicating the time or space complexity of an algorithm

BPMN: Business Process Modeling Notation

CRU: Cardiovascular Research Unit (sydäntutkimusosasto in Finnish) at Meilahti hospital, Helsinki

CSV: Comma Separated Values. A simple format for storing data: each row contains one element and its values are separated by commas

D/F-graph: A graphical presentation of the information found when interpreting D/F-table

D/F-table: Dependency/frequency table describing the dependency relations and their volumes between different activities in a workflow log

Event: A concept related to mining algorithm based on event types. Event is a pair consisting of a task and the event type (started or completed).

Event trace: An ordered set of events in an event log where all events belong to a single case

Heuristic: A technique for rapidly finding a solution close to the best possible. Often used in situations where finding the optimal solution would be too time or space consuming.

HUS: Helsingin ja Uudenmaan Sairaanhoitopiiri (The hospital district of Helsinki and Uusimaa)

Noise: Erroneous data written to data warehouses. Can result from, among others, human error or incomplete logging in automated systems.

Pareto principle: A rule of thumb stating that about 80% of the effects come from 20% of the causes

Petri net: A tool for modeling interrelated activities and the types of their relations

Process mining: A technique for extracting process information from workflow logs

Variation: Deviation from a given target, such as dimensions of a produced machine part

Variety: Different but functionally equivalent targets, such as different colors of the product

VBA: Visual Basic for Applications. Scripting language developed by Microsoft to provide programming capabilities to programs such as Microsoft Office.

Workflow net: A Petri net that models the control-flow dimension of a workflow

1 Introduction

1.1 Background

Hospital patient record systems and ERP-like data warehouses contain a large amount of data on patient treatment history. Most treatment activities performed to a patient are recorded to patient records which contain millions of lines of medical treatment history. However, there is still no clear picture of what happens in a big hospital, let alone in a larger hospital district. Different hospitals have developed different ways of handling similar cases and doctors are often using their own best practices. This in essence means that there are very few people, if any at all, who actually know what happens between organizational boundaries in healthcare sector. (Mäkijärvi, 26.6.2009)

This lack of inter-organizational knowledge is somewhat conflicting with the recent literature describing patient episodes (Lillrank et al., 2009) as a fundamental unit of observation. This topic will be covered in more detail in later chapters; at this point it is sufficient to know that the terms process and episode are somewhat linked – they both describe longer course of related actions. However, where the process is a repeatable, planned execution order of activities that employs a set of productive resources, the episode, on the other hand, is a process perceived by a patient (Lillrank et al., 2009). Episode describes what happens to a patient in a certain medical condition from his or her own point of view. Recognizing episodes could be very useful for doctors or process planners, who often, because of reasons described earlier, do not simply know what happens to the patients in general.

This conflict of abundance of data and scarcity of knowledge is the main idea behind this study. The study is done for a Finnish software company, QPR Software Plc (later on referred to as simply QPR). The main products of QPR are QPR ProcessGuide, a tool for process modeling and analysis, and QPR ScoreCard, a tool for measuring, monitoring and reporting objectives in a scorecard format (www.qpr.com). Recently QPR has expressed interest in developing its rather mature business to new directions via innovation workshops and the concept of semi-automated process analyzer was recognized as an interesting field of study.

Process analyzer in the QPR context is a software tool that transforms log data to a clear and visual representation describing processes as automatically as possible. This differs from many traditional process mapping principles as they often require human interaction. As an example, the hermeneutic, understanding based perspective on process mapping means identifying, with the aid of brainstorming sessions and interviews, how the processes are experienced by the involved parties (Lillrank, 2010). Process analyzer, on the other hand, creates process illustrations programmatically and reflects the true status of the process. In this thesis the tool is called ProcessAnalyzer, reflecting the name the prototype has been given inside QPR.

A field of study called the process mining (for example, van der Aalst et al., 2003) is relevant for developing ProcessAnalyzer. Process mining is a technique for extracting process information from workflow logs; in this study process information is a process model and the workflow logs are taken from Finnish healthcare patient records.

1.2 Aims of the study and the research problem

The purpose of this study is to identify process mining techniques presented in the literature and to develop, based on these techniques and actual hospital data, a prototype of ProcessAnalyzer. The actual product is not implemented yet, but the necessary analyses are done on current hospital data to see if it is feasible to develop the idea further. The desired output of ProcessAnalyzer is an empirical and patient focused process model describing the common route that a patient belonging to a certain group goes through.

Achieving this goal requires finding relevant literature on process mining techniques and on the other hand finding and identifying real patient data from hospital patient records. These two main areas of the study provide answers to the research questions:

1. What kinds of process mining techniques are presented in the literature?

   a. What are the capabilities of the different techniques?
   b. What kind of data do the techniques use as a basis of the analysis?
2. What is the status of the data in the Finnish hospital patient records?
   a. How can data from different sources be combined?
   b. How can the data be transformed to the format required by process mining techniques?
3. Based on prototyping the process mining techniques, what are the possibilities and the challenges of ProcessAnalyzer?
   a. What is the validity of the mined process models?
   b. Does the prototype reveal hindrances with the process mining techniques?

1.3 Contents and structure

This thesis consists of six chapters: introduction, literature review, research setting and methodology, results, conclusions and suggestions and recommendations. The introduction explains the background of the study and the research questions answered. Chapter 2 contains literature review describing principles of process mapping, process mining and healthcare specific issues such as concepts of episodes and event-based medicine.

The focus of the study as well as the data is described in chapter 3, and chapter 4 describes the results. The results consist of describing the process mining algorithm used in the ProcessAnalyzer prototype and analyzing several cases where ProcessAnalyzer has been validated. Chapter 5 discusses the suitability of process mining regarding the cases analyzed and presents some challenges related to ProcessAnalyzer. Some ideas for further studies are also presented in this chapter. Finally, chapter 6 summarizes the conclusions of the study and gives recommendations for further actions.

2 Literature review

This chapter will first introduce the reader to the topic of processes and present general principles of mapping processes. After this, introduction to Petri nets is given as a basis for actual process mining techniques. The most relevant part regarding ProcessAnalyzer consists of the process mining techniques presented in chapter 2.3. After these techniques a brief summary of Pareto principle is given, and the chapter is concluded by describing concepts regarding recent literature findings in healthcare.

2.1 Processes and process mapping

This chapter consists of describing what processes are as well as the essential principles of process mapping. First the reader is introduced to the topic and the basic classes of process maps are discussed. After these chapters, the basics of visual modeling of processes are summarized. Finally a generic, traditional way of modeling processes is explained.

2.1.1 Introduction to processes

A fundamental concept in this study is the concept of process. Processes are everywhere and they are the source of many analysis and improvement activities performed in companies. Literature often defines process such that it tells how a company transforms inputs into outputs and what resources it uses to do so. Kai Laamanen (Laamanen, 2007, p. 18) defines business process as follows (translated from Finnish): “Business process is a set of interrelated actions and resources needed to perform these actions; using this set the inputs are transformed into products.”

The transformation view on business processes is however not the only presented in the literature. Lind suggests treating processes both as transformation and coordination tools (Lind, 2006). The coordination view on processes emphasizes the communication between process parties. Figure 1 presents an example of the communicative perspective of business processes. This action workflow loop is presented by Medina-Mora et al. in (Medina-Mora et al., 1992). The loop describes the iterative nature of the relationship between customer and supplier, or performer as called in this model. The customer makes a proposal for the performer and the customer and performer agree on the terms of performance. When the performer has performed whatever was ordered, the customer is to some extent satisfied and the iteration continues. This kind of a process emphasizes the communication between the customer and the performer and is thus handled in a different way compared to a transformation process.

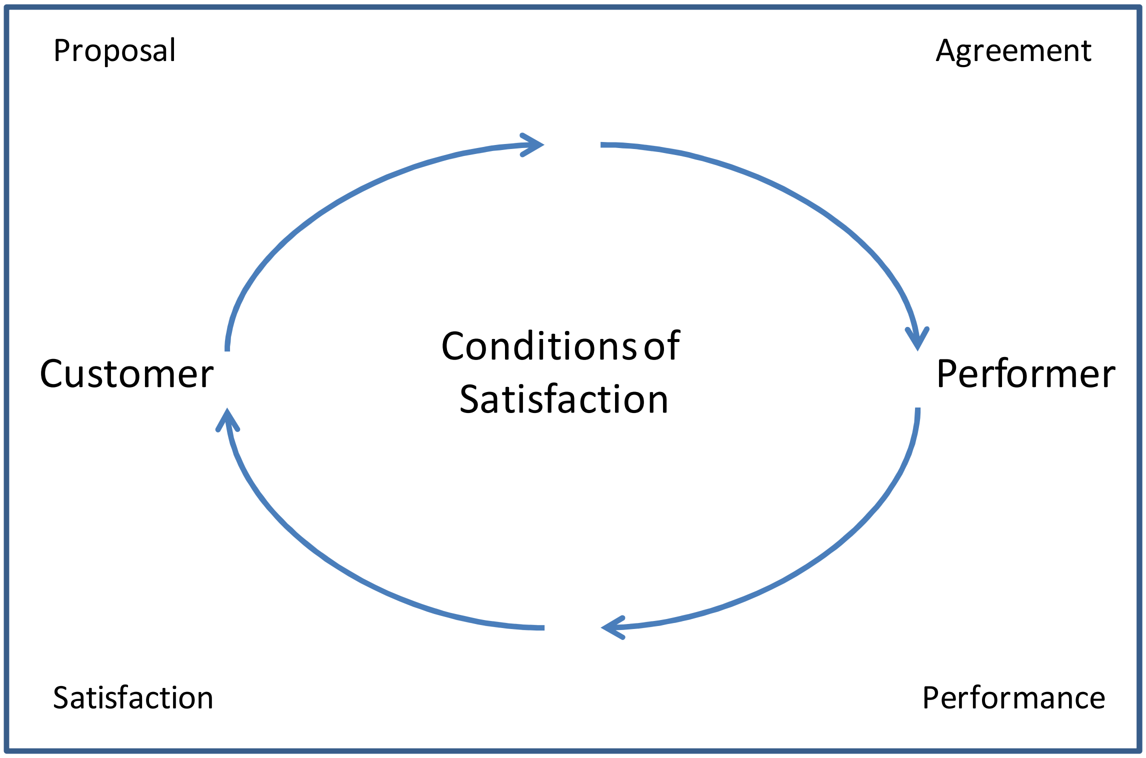


Figure 1: Action workflow loop (Medina-Mora et al., 1992)

Systematic processes are often used to bring order to the chaotic world (Laamanen, 2007, p. 23). Without a well-defined process the big picture is not clear; people are able to describe their own tasks but not their role in the larger perspective. Process thinking might also bring psychological benefits as when the organizations switch to process oriented thinking, the blame for mistakes is not directed towards employees any more, but instead to the process – analyzing the processes and changing the parts that do not work well is the way to improve the performance of the organization (Madison, 2005, p. 8).

2.1.2 Introduction to process mapping

The purpose of process mapping is to enable analysis. This is in practice done by illustrating a process visually by showing related activities and the types of their relations. Processes are modeled on many levels and from many perspectives – some process models describe high level activities while others drill deeper into the hierarchy and subprocesses. The general practice of modeling processes is called process mapping and is handled quite extensively in the literature (for example, Laamanen, 2007; Madison, 2005 and O'Connell et al., 2006).

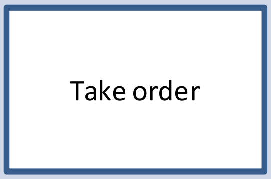
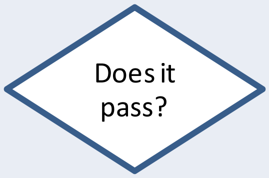
Laamanen describes several benefits of thoroughly mapping the processes of an organization in (Laamanen, 2007). Among others, these benefits include 1) increasing co-operation with customer, 2) helping people working at the organization to understand the big picture and their own role in it, and 3) reducing suboptimization when processes are investigated from a company-wide perspective. The same themes are also presented by Burlton who discusses that the objective of process modeling is to understand the underlying process (Burlton, 2001, p. 311). A process model should, according to Burlton, be treated as a tool that helps to understand the process and to communicate it to different stakeholders.

2.1.3 Visual process mapping conventions

The visual clarity and uniformity are important aspects of process mapping. The purpose of process mapping is to understand the processes and divide information on them easily. Using commonly agreed visual conventions helps to achieve this. These principles are discussed in a number of books; the ones presented here are taken from (Madison, 2005).

The basic blocks of process maps are the symbols used in the flowcharts. There are some differences in how different authors and different conventions use the rarer symbols, but the most commonly used ones are presented in the Table 1. A box represents activities, such as “Take order” in this example. They usually contain a short description of what is done in the activity. Reviews and decisions are marked by diamond shaped symbols, such as “Does it pass?” in this example. The diamond shape contains a question and the flows that start from the decision contain answers to that question. The direction of the flow in the process is indicated using arrows between the activities and decisions. Because log files generally do not contain decision steps, but instead just the performed activities, decisions cannot be distinguished when analyzing the files. Consequently the process models drawn by ProcessAnalyzer contain only activity boxes and process flow arrows.

Table 1: Commonly used flowchart symbols

Name	Symbol
Activity box	
Review and decision diamond	
Process flow arrow	→

The process maps are often drawn on different levels of detail for the same process – one level describes the big picture while more detailed levels describe parts of the big process in more detail. One of these classifications is presented in (Madison, 2005) in which the detail levels are called macro flowchart, functional-activity flowchart and task-procedure flowchart.

Macro flowcharts contain only the critical elements of a process – Madison recommends that there should be two to seven steps in the macro level flowchart altogether. The key benefit of a macro flowchart is that it defines the main elements and the scope of the process. Figure 2 shows an example of a macro flowchart, describing the five main activities of an order-to-delivery process.

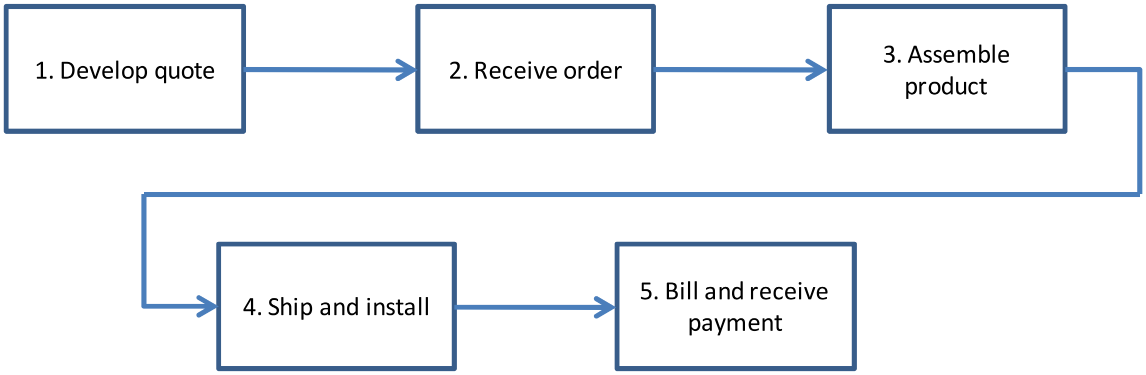


Figure 2: Example of a macro flowchart (Madison, 2005)

The next level in Madison's process map classification is the functional-activity flowchart. This flowchart type describes somewhat more detailed processes than those in the macro flowcharts and this time also the roles of people performing the activities are added to the model. Madison identifies seeing where the majority of the work is performed, where the value-adding steps are done, and where the problems occur as the most important benefits of showing who performs the activity (Madison, 2005).

Figure 3 shows an example of a functional-activity flowchart. This flowchart is a simplified version of an example presented in (Madison, 2005) and it describes an imaginary plan approval process. The flowchart contains four job titles that describe the functions. Customer is shown first; this recommendation is also given by Laamanen (Laamanen, 2007, p. 80). The activities are drawn to the functions that perform the actions, for example the administration receives the application and verifies whether the plan is complete. The mapping direction can be also horizontal in which case the functions are drawn to the left side of the process model.

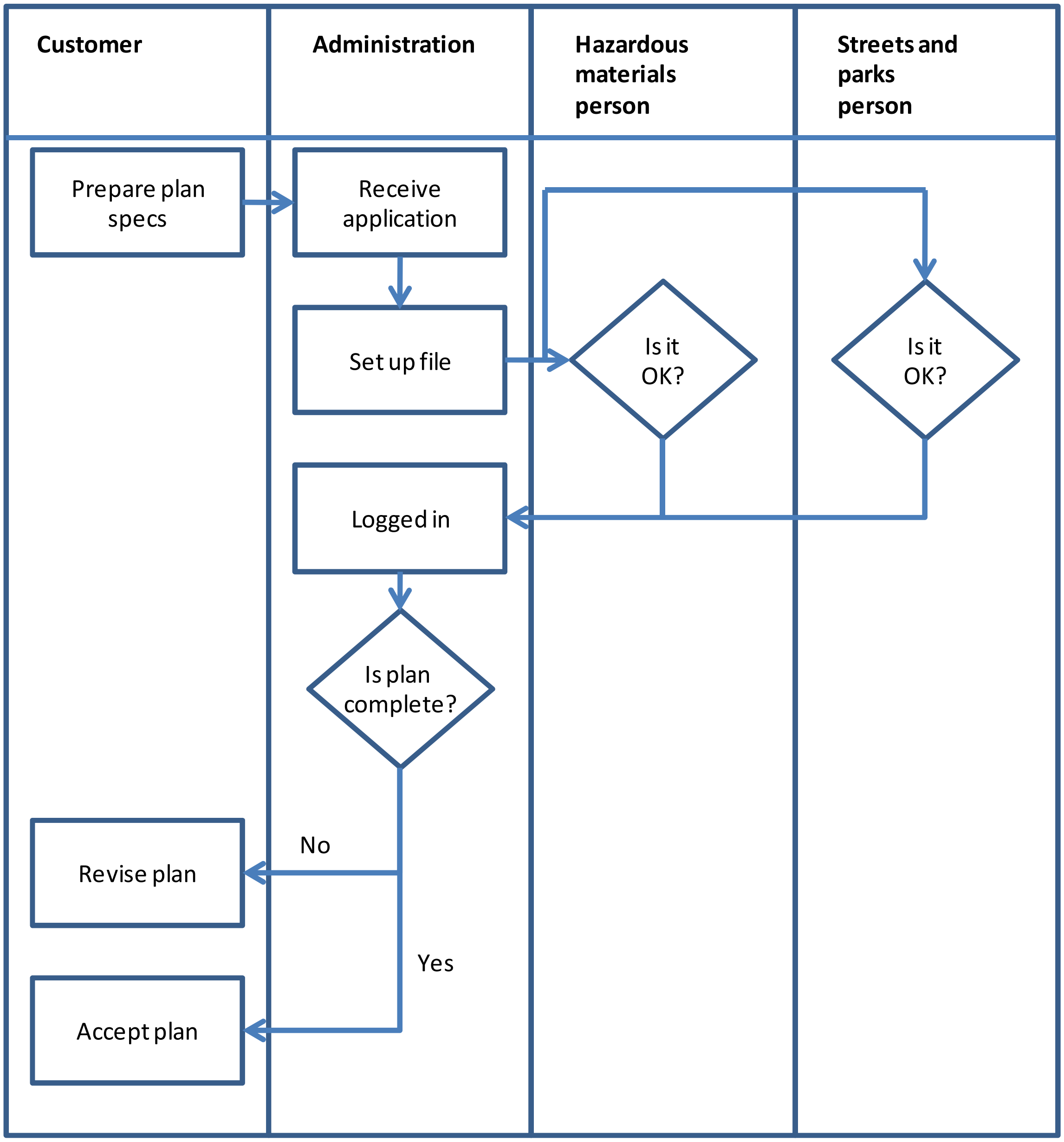


Figure 3: Example of a functional-activity flowchart, adapted from (Madison, 2005)

Madison defines task-procedure flowcharts as the most detailed flowchart type. These flowcharts go to the details of one activity or decision in the functional-activity flowchart; they describe the exact procedures done in that step in a format that can be understood by people who are not familiar with the process. In essence they are thus working instructions and do not much resemble the flowcharts discussed previously in this chapter. Continuing with the same example as in the previously mentioned process map detail levels, a functional-activity flowchart of application receiving could start by picking the application from the mailbox. This application is then read through, validated, sorted to right priority class and sent for further actions. Functional-activity flowchart lists the execution times and additional information for each of these steps.

The most relevant flowchart type regarding this study is the functional-activity flowchart as all the process models drawn by ProcessAnalyzer are horizontal functional-activity flowcharts containing activities and flows. Different detail levels can also be achieved by simply using these flowcharts for mapping first higher level processes with less detail and then subprocesses of those processes with higher detail level.

It should be noted that the conventions presented in this chapter are not the only ones used when mapping processes. The approach for this study was selected mostly for practical reasons – the products of QPR are based on these conventions and thus it was natural to select these also for the development of a possible future QPR product. Literature presents several other visual conventions for process mapping in addition to the one presented in this chapter. An example of a more technical one that is also quite often used is BPMN (Business Process Modeling Notation), maintained by OMG (Object Management Group). The details of BPMN do not belong to the scope of this study so it should be sufficient to say that the standard activities and flows are present also in the BPMN notation. The decision steps are modeled in BPMN with more detail – they are called gateways and are divided to different types of splits and merges of flows: exclusive, inclusive and parallel (www.bpmn.org).

2.1.4 Process map categorization

Literature often divides process maps into several categories depending on how fact-based processes they describe. Laamanen distinguishes four different levels of process mapping (Laamanen, 2007, p. 87): 1) current process, 2) slightly improved process, 3) radically improved process, 4) ideal process. He recommends that the process mapping should focus on alternative number 2, the slightly improved process, which in practice means a process that can be achieved in about six months. According to Laamanen, the process maps that focus on more significant improvements are only relevant when some external factor forces the company to make radical changes in its structure. On the other end of the spectrum, he also comments that mapping current process might not be always a good idea, especially if the current process is not working very well. The reason for this is that mapping a process can make people believe in it more than before, and thus mapping a not so well planned process enhances its bad effect on the company.

Another opinion on the subject can be found from (Burlton, 2001, p. 311). He specifically comments that for understanding the process, the mapping must focus on what actually happens in the process. This means disregarding what written guides say about the organization or what the process owners think about the processes. This is naturally contradicting with the opinion of Laamanen – however, the points of view of these two authors are different. Laamanen discusses improving processes while Burlton describes understanding the current processes.

Lillrank divides processes mapping into three categories – normative, hermeneutic and positivistic (Lillrank, 2010). Normative process maps describe the processes as they should be, often from the management point of view. Hermeneutic process maps describe how processes are expressed by the involved parties. They are usually derived by interviewing or arranging brainstorming sessions – the outcome tries to grasp the reality of business processes as well as the attendants can describe that. The positivistic process maps focus on a general process characteristic such as flow, volume or throughput time. As this study focuses on real life data collected from the hospital patient records, the resulting process models are, according to Lillrank's categorization, positivistic representations of the processes. Process models like this offer a valuable tool for understanding the actual status of the processes in the organization as they are based on facts.

2.1.5 Traditional process mapping

When processes are mapped using traditional methods, the mapping is done based on the expertise of the process owners. Laamanen discusses that this in essence requires presence of the top management as they know the high level processes. These people naturally are busy and tend to delegate work to those below them in the organizational hierarchy. However, this is work that cannot be delegated; it has to be done by the process owners themselves (Laamanen, 2007, p. 82). Also Madison emphasizes the role of process owners in process mapping (Madison, 2005, p. 18). He discusses that mapping needs to be done by a group of individuals that know the process well; the same approach is also described by Laamanen.

A common process mapping method is to arrange a workshop consisting of process owners who think about how the processes are organized. This method produces, according to the categorization by Lillrank, a hermeneutic process map (Lillrank, 2010). It is also described extensively in literature; one of these explanations is presented by Laamanen (Laamanen, 2007, p. 86). He discusses that a process mapping workshop should start by introducing the subject of process mapping to the group of people. Once this is clear, simple questions are asked to identify basic facts about the process. These questions include such elementary issues as who the customer is or who provides the basic input for a particular step of the process.

When the basic facts regarding the process have been identified, the next step presented by Laamanen is to let individual people in the process mapping group to think about the processes from their own perspective. This lets the people to form their own opinion instead of just adapting the opinions of others. Once enough time is spent with individual thinking, the collective idea sharing can start. Practically this means that everyone gets a turn to explain one new activity in the mapped process. When all activity candidates have been recognized, the results are prioritized by the process owners – only those activities that these people value as the most relevant ones are selected to the actual process map. Finally the session is wrapped up by giving instructions to the process owners. (Laamanen, 2007, p. 86)

The actual process maps are drawn using specialized software designed for this purpose. Laamanen recommends using tools that enable publishing the maps in intranet. On the other hand, if the point of process mapping is just to understand the processes, simply using basic drawing tool, such as Microsoft PowerPoint, for drawing the activities and their relations is sufficient. (Laamanen, 2007, p. 88)

2.2 Petri nets

As a necessary prerequisite to understanding process mining techniques I will present brief introduction to Petri nets before moving to actual process mining. Even though the ProcessAnalyzer does not rely on Petri nets, they are an important concept frequently presented in the process mining literature. For more in-depth discussion on Petri nets the reader is advised to refer to (Reisig & Rozenberg, 1998).

2.2.1 Introduction to Petri nets

Petri nets are a formal way of modeling tasks and flows between the tasks. They consist of three basic element types: places, transitions and arrows connecting places and transitions. There is always a transition between two places, and one place can lead to and start from one or more transitions. When initialized, a so called token is put into the starting place of the Petri net. A transition is enabled if each of its input places contains at least one token. An enabled transition may fire, in which case it consumes one token from each of its input places and produces one token for each of its output places. The execution ends when there is a token in the ending place.

Even though Petri nets are designed to give a generic and technical representation of any set of interrelated tasks, they can also be used to map processes. Interrelation between Petri nets and processes can be seen when comparing terms presented in the literature. For example, Lillrank uses the term mobilization when accepting the input and working according to that (Lillrank, 2010) – this is analogous with the token conceptualization discussed above.

2.2.2 Example of a Petri net

The following example is taken from (Weijters & van der Aalst, 2003).

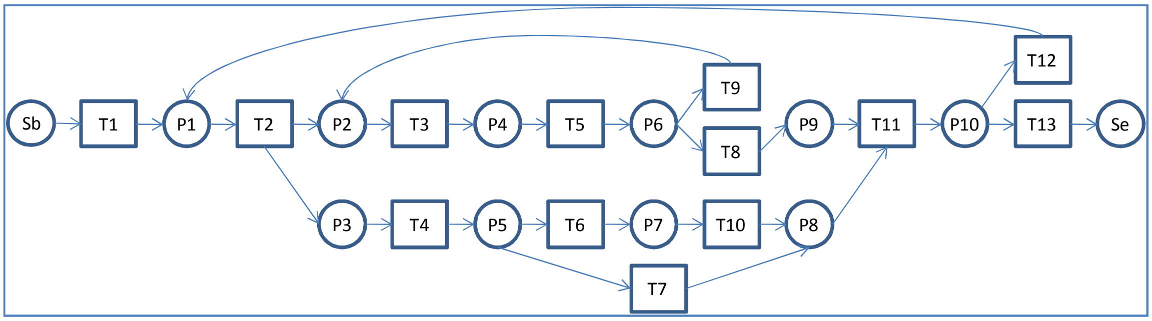


Figure 4: Example of a Petri net (Weijters & van der Aalst, 2003)

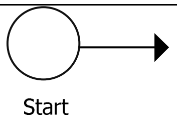
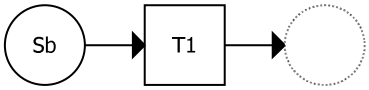
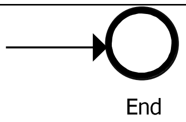
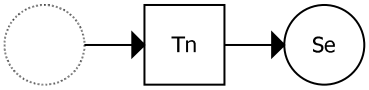
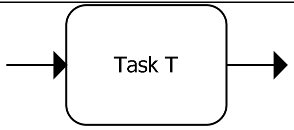
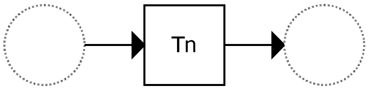
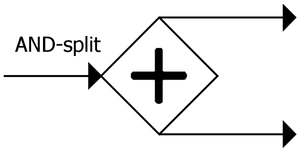
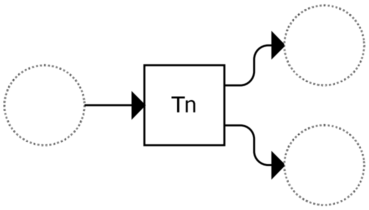
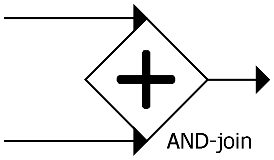
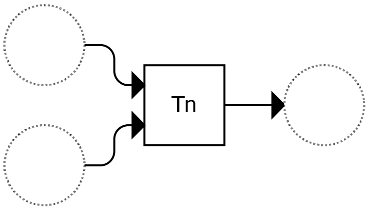
In the figure above (Figure 4) the boxes represent transitions which again represent tasks in a process model and circles represent places which in turn represent causal dependencies between tasks. When a case described by the Petri net above starts, a token is put in the starting place Sb. Transition T1 may then fire as it has token in its input place. This consumes token from Sb and produces a token to P1. T2 is fired next and in the following split tokens are put to both place P2 and place P3. This is repeated until token it put to the ending place Se.

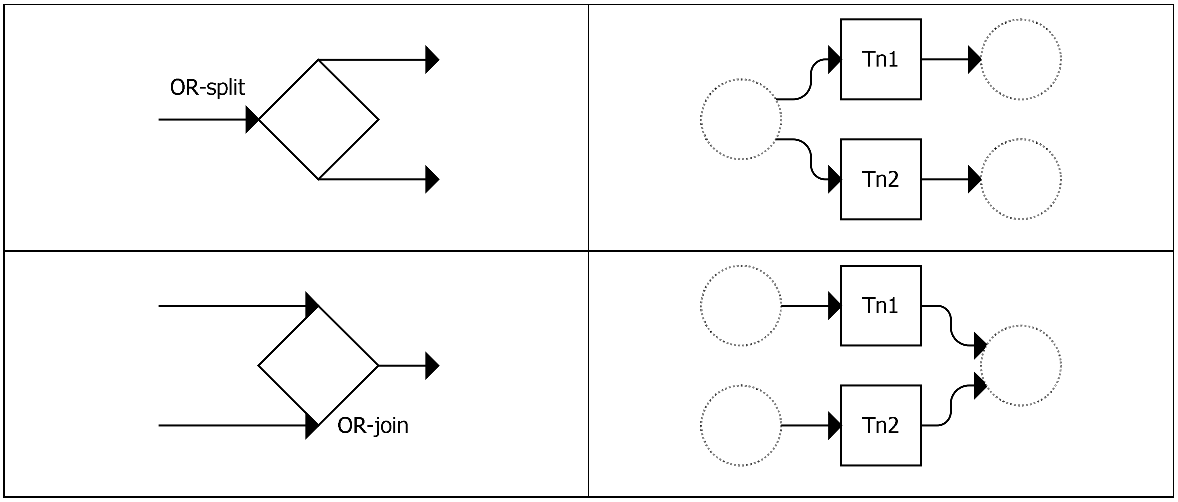
The different parallel routes in the example model can be distinguished to splits and joins of type AND- and OR. AND-splits and –joins mean routes where both ways are executed in parallel – for example transition T2 leads to places P2 and P3 in the example. When T2 is fired, both P2 and P3 get a token and can be executed in parallel. On the other hand both P8 and P9 function as an input place to T11, which indicates an AND-join – for T11 to fire both P8 and P9 must have a token. OR-splits and –joins, on the other hand, indicate alternate routes. For example, place P5 leads to T6 and T7 which means that either of the transitions T6 and T7 can fire when there is a token in P5.

2.2.3 Petri nets and process models

Petri nets are formal and generic descriptions of activities and flows connecting them. Thus they can be used for a large variety of purposes. One of the interesting research areas regarding this study is their applicability in describing processes. Dijkman et al. (Dijkman et al., 2008) describe a tool for converting standard BPMN process models (www.bpmn.org) to a Petri net. Table 2 presents a summary of these conversion rules. The grayed out places in Petri net illustrations indicate the implicit places that occur in the Petri net after the actual element described by the picture.

Table 2: Converting a BPMN process model to a Petri net (Dijkman et al., 2008)

BPMN	Petri net
	
	
	
	
	



Start and end task conversions are quite trivial – they just generate respective places in Petri nets. Same applies to simple tasks: place and transition to that place is added to the Petri net. The different splits and joins require more effort, but after all they are quite simple as well. In the case of AND-gateways, two or more places are connected to one place with a transition. In the case of OR-gateways, on the other hand, two or more places are connected to one place using separate transitions for each input place.

Because BPMN models can be converted to Petri nets and BPMN is simply one notation for process modeling, Petri nets can be used to model any process. This notion is important for this study as most mining algorithms generate Petri nets as result. These Petri nets can be converted back to traditional process mapping notations by applying the conversion rules in reverse.

2.3 Process mining

Process mining is a technique for extracting process information from event logs that contain process execution information. It has been discussed in literature in the recent years; in particular by W.M.P. van der Aalst (for example, van der Aalst et al., 2004). His underlying thought is that modeling processes is a very time-consuming process and at the same time a large number of data on real transactions is recorded in different event logs. If this data could be used to generate process models, a lot of time would be spared for other activities. Modeling processes based on real life data also reveals the true state of processes in the organization and can reveal interesting conflicts with how the managers or other process owners thought processes to be.

In the literature this technique of extracting process models from logs is called process mining. It means taking data from any system, such as an ERP system of a company, and using various algorithms to derive the underlying process model that generates those entries to the log. According to van der Aalst (van der Aalst et al., 2004) a log file is suitable for process mining if it consists of events which 1) each refer to a task, 2) each refer to a case, and 3) are in the order of execution. Process mining algorithms are capable of identifying sequential and parallel activities from these kinds of logs and generating the process model based on these findings.

An example might illustrate better the principles of process mining. This example can be found from (van der Aalst et al., 2007). Table 3 contains an imaginary and simplified log generated by, for example, an ERP system performing different activities. The important parts of the log regarding process mining are the case identifiers, which distinguish different cases from each other, the activity identifiers, which tell what activities are done, and the order the activities occur in. The actual timestamps are not necessary unless needed, for example, for lead time calculation.

Table 3: An event log example (van der Aalst et al., 2007)

Case id	Activity id	Timestamp
Case 1	Activity A	9.3.2004 15:01
Case 2	Activity A	9.3.2004 15:12
Case 3	Activity A	9.3.2004 16:03
Case 3	Activity B	9.3.2004 16:07
Case 1	Activity B	9.3.2004 18:25
Case 1	Activity C	10.3.2004 9:23
Case 2	Activity C	10.3.2004 10:34
Case 4	Activity A	10.3.2004 10:35
Case 2	Activity B	10.3.2004 12:34
Case 2	Activity D	10.3.2004 12:50

Case 5	Activity A	10.3.2004 13:05
Case 4	Activity C	11.3.2004 10:12
Case 1	Activity D	11.3.2004 10:14
Case 3	Activity C	11.3.2004 10:44
Case 3	Activity D	11.3.2004 11:03
Case 4	Activity B	14.3.2004 11:18
Case 5	Activity E	17.3.2004 12:22
Case 5	Activity D	18.3.2004 14:34
Case 4	Activity D	19.3.2004 15:56

The log reveals that each of the five identified cases starts with activity A and ends with activity D. For cases 1–4 the activities B and C occur between activities A and D. The execution order of these two activities can change: for cases 1 and 3 the execution order is first B, then C – for cases 2 and 4 the execution order is reversed. For case 5 only activity E occurs between start and end activities. Figure 5 contains one possible Petri net representation of the process that generates such log as presented in Table 9. Process mining techniques attempt to find these representations algorithmically by iterating through the log and finding connections between different activities.

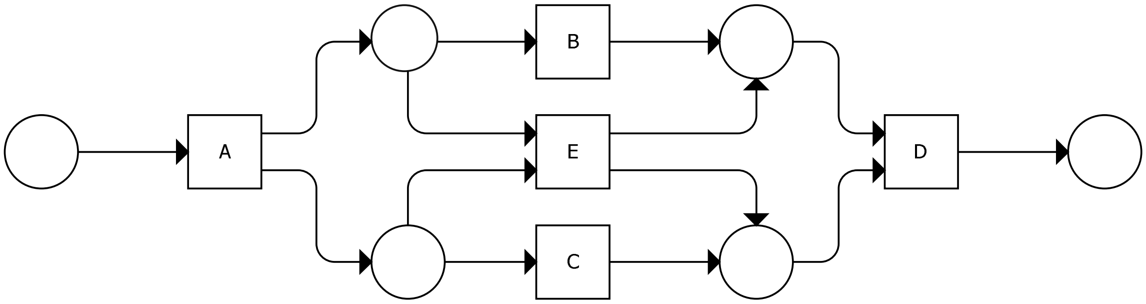


Figure 5: Petri net representation of the example data (van der Aalst et al., 2007)

In this study I have found several interesting algorithms from literature and investigated more deeply two of them: the heuristic miner and the mining algorithm based on event types. I will next present these and in the chapter 4.1 the modified algorithm which the ProcessAnalyzer uses.

2.3.1 Heuristic mining algorithm

If data was always perfect, mining the process model from it would not be very difficult. This would simply require recording each implicit flow between the activities in the log and drawing the process model based on that information. However, real life data usually contains elements that are called noise in literature (e.g. Cook & Wolf, 1998; Weijters & van der Aalst, 2003). Noise can be a result of, among others, wrong data accidentally written to database or missing data. One of the most promising algorithms to deal with noise is the heuristic mining algorithm (Weijters & van der Aalst, 2003). The authors have given the name “Little Thumb” for the algorithm, partly because the algorithm is based on heuristics (e.g. Russell & Norvig, 2003, chapter 4) which are often called “rules of thumb”. The algorithm has also been tested in healthcare environment (Mans et al., 2009).

Heuristic mining algorithm is based on constructing a so called D/F-table (dependency/frequency table) from the workflow log. This table is used as the basis of deriving what activities depend on each other and what interaction class they belong to. Finally the algorithm produces a workflow net (WF-net) (van der Aalst, 1998) which in essence is a more strictly defined Petri net and which thus can be converted to traditional process modeling notation. I will present the basic idea of constructing D/F-table as it is presented in (Weijters & van der Aalst, 2003). However, through careful investigation, I have identified some strange issues with the notations in this article and cannot find any other explanation to them than simply typing mistakes. I have fixed these and will also present the original version here for reference.

Several metrics are extracted from the workflow log in order to calculate the values necessary for D/F-table. It should be noted that each of these metrics is calculated within one case only, meaning that even if the log contains adjacent activities, they are not considered related to each other unless they belong to the same case. The metrics are calculated for each activity and are presented in Table 4.

Table 4: The metrics required for calculating a D/F-table.

Metric	Description
#A	The overall frequency of activity A
#B<A	The frequency of activity A directly preceded by another activity B
$\$A\rightarrow{}^{L}B$	The local metric indicating the close range dependency relation between activities A and B
$\$A\rightarrow B$	The global metric indicating the longer range dependency relation between activities A and B
DS(A,B)	The combined dependency score describing the overall degree of dependency between activities A and B

The first two metrics are calculated simply by browsing through the workflow log. On each row the count of the activity of that row is increased by 1, and the follower and predecessor of the activity is checked and counted to the #A<B and #B<A metrics. After these operations the local metric is calculated using formula

$$\$A\rightarrow{}^{L}B = \frac{\#A<B-\#B<A}{\#A<B+\#B<A+1}.$$

This metric describes the general direction of relation between two activities and its purpose is to reduce the effect of noise. In the article by Weijters and van der Aalst (Weijters & van der Aalst, 2003) this metric was calculated slightly differently – each of the relations was of type follower (>), not predecessor (<). This, in my opinion, is not according to what was explained in the article so I decided to turn the directions of the relation. As an example of this metric, consider a case where activity A is directly followed by B 10 times and never the other way around. The value of this metric is

$$\$A\rightarrow{}^{L}B = \frac{10-0}{10+0+1} = \frac{10}{11} = 0{,}909.$$

We cannot thus be sure that the dependency relation is such that B follows A as the metric is not very close to 1. On the other hand, if the situation was such that activity A is directly followed by B 1000 times, the value of this metric would be quite high even though there was, say, 2 cases caused by noise that indicate the opposite direction. In this case the value would be

$$\$A \rightarrow {}^{L}B = \frac{1000-2}{1000+2+1} = \frac{998}{1003} = 0{,}995$$

and it would be quite safe to assume that the relation is A<B – the noise would be eliminated.

The global metric indicates, as the name suggests, a longer range dependency between tasks. It is calculated such that for each activity in the log the log is searched backwards for activities that occur before it. For each activity like this the $A→B dependency counter is incremented by δ^n where δ is so called dependency fall factor and n is the number of activities between activities A and B. Weijters and van der Aalst (Weijters & van der Aalst, 2003) have experimentally set the value of δ to 0.8 – in theory it can be set anywhere between [0.0 … 1.0]. After searching the log backwards and reaching either the beginning of the case or another occurrence of activity A, the search continues forward. Each activity that occurs after activity A is similarly handled and this time the dependency counter is decreased by δ^n. After the whole log is processed the counters are each divided by minimum of the frequencies of the tasks that belong to the activities of the counter, that is $A→B is divided by min(#A, #B).

Dependency score DS(A,B) is the ultimate score for deriving the best candidates for relations. In the article by Weijters and van der Aalst (Weijters & van der Aalst, 2003) this is calculated using formula

$$DS(X,Y) = \frac{(\$\mathrm{A}\rightarrow{}^{\mathrm{L}}\mathrm{B})^2 + (\$\mathrm{A}\rightarrow\mathrm{B})^2}{2}.$$

However, this appears to cause strange consequences with large negative values of the local and global metrics. Negative values should indicate that the relation is a reverse relation, but raising this to the second power brings a positive result. As the algorithm assumes that the activities with high dependency scores should be considered as the related activities, this gives wrong results. For this reason I refined the dependency score by removing the squares from it – the refined dependency score is calculated using formula

$$DS(X,Y) = \frac{\$A \rightarrow^{L} B + \$A \rightarrow B}{2}.$$

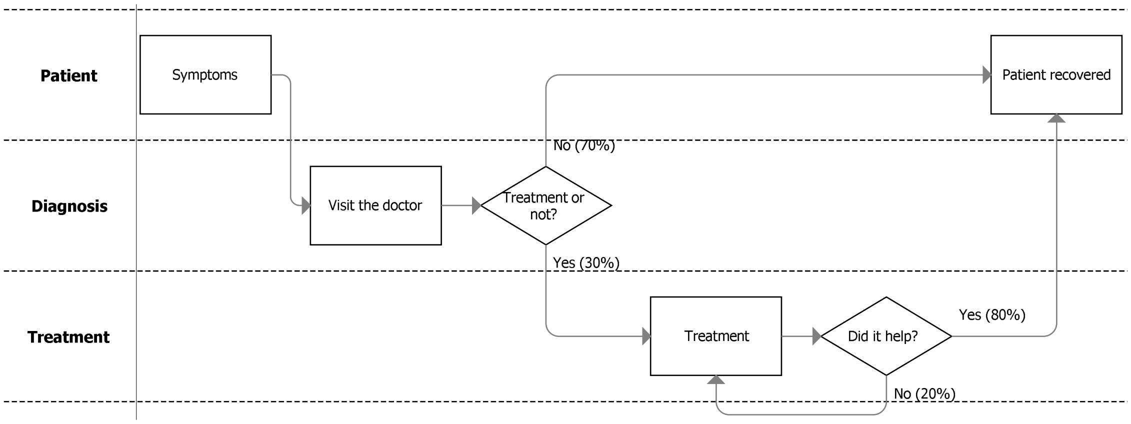


Figure 6: Example model for heuristic mining algorithm

Figure 6 contains an example model I created for acquiring data for testing the heuristic mining algorithm. The model is quite a simple process describing a patient seeking medical help for his or her symptoms. The main point behind the model is to illustrate the algorithm, not to try to model any real treatment process. Once the process described in Figure 6 is transferred to a simulation model in QPR ProcessGuide, the simulation creates a log that can be used as source data for heuristic mining algorithm. The mining algorithm calculates D/F-tables for each of the six activities in the model. The full D/F-tables can be found from Appendix 1, but I will present the D/F-table of decision "Treatment or not?" here for illustration purposes.

Table 5: A sample D/F-table

A	B	#B	#B<A	#A<B	local metric	global metric	DS
Treatment or not?	Symptoms	1000	0	0	0	-0,64	-0,32

Diagnosis	Visit the doctor	1000	994	0	-0,99899	-0,8	-0,8995
	Treatment or not?	1000	0	0	0	0	0
	Patient recovered	1000	0	687	0,998547	0,701146	0,849846
	Treatment	369	0	307	0,996753	0,665583	0,831168
	Did it help?	369	0	0	0	0,532466	0,266233

Once the D/F-tables are generated, finding the successor-relations is relatively trivial – they are the ones with the highest DS-scores. For detecting parallel or alternative execution paths multiple routes should be selected – Weijters and van der Aalst actually suggest that all activities which have dependency score of over 95% of the maximum DS should be selected as potential related activities (Weijters & van der Aalst, 2003). The identified successors are then drawn to a D/F-graph (dependency/frequency graph) which is a good starting point for creating a full process model. D/F-graph contains the frequencies and successors of all activities, as well as the dependency scores of the succession-relations. Figure 7 contains the D/F-graph derived from the test model. The result looks quite a lot like the original process model. The types of splits and joins are not identified yet. These can usually be determined from the counts of flows – in this example the D/F-table shows that in 695 cases the activity “Treatment or not?” has been followed by “Patient recovered” and in 302 cases by “Treatment”. Because the sum of these numbers is close to the total of 1000 cases going through this activity, it seems that this split is an alternative one – either patient recovers or he or she gets treatment. Other splits and joins can in simple cases be recognized similarly.

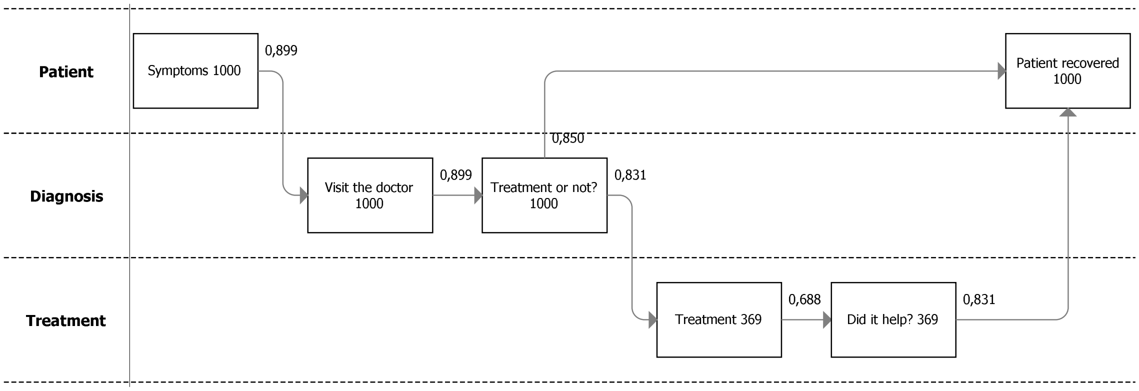


Figure 7: D/F-graph of heuristic mining algorithm

However, the D/F-graph in Figure 7 shows a more severe drawback – the flow from "Did it help?" back to "Treatment" is not visible at all. This is caused by the rather low dependency score this flow gets. In the original model it was specified that 20% of the cases go back to "Treatment", so the majority of the routes are recognized correctly. However, in many cases it would actually be interesting to know what the special cases look like and this problem of heuristic mining algorithm becomes more relevant. This is actually one of the problematic dependency relations mentioned by Weijters and van der Aalst that are not recognized by the mining algorithm (Weijters & van der Aalst, 2003). Weijters and van der Aalst call these interconnected activities "short loops", meaning same activity run multiple times, or one activity and another one alternating their execution. This major drawback causes problems with many real life models and renders the promising heuristic mining algorithm quite useless in the ProcessAnalyzer context. However, some of its principles can be utilized when refining the mining algorithm – and in the case of a simple process but noisy log it can actually provide quite good results.

2.3.2 Mining algorithm based on event types

Because the heuristic mining algorithm presented in the previous chapter is unable to recognize short loops I needed to find another approach to process mining. Heuristic mining algorithm also cannot reliably identify parallel execution paths as the activities belonging to these paths occur in arbitrary order and the algorithm tries to interpret these execution orders as sequential relations. These problems are largely solved by the mining algorithm based on event types, presented by Wen et al. (Wen et al., 2009).

The fundamental idea behind this mining algorithm is that it takes into use both the start and the end times of the task execution. These times are usually saved to workflow logs and they can be used to detect parallelism directly (Wen et al., 2009) – for example, if some task starts while another one is still being executed, it is safe to assume that the tasks are parallel. The workflow log is modified so that the log rows each indicate one event type occurrence – a task is either started or completed at a given time in a given organizational unit. This effectively doubles the size of the needed data table.

The algorithm starts by browsing through the log and searching for successions and intersections between tasks. These two basic relations between tasks form the basis of higher level analyses. For understanding some of the vocabulary in the next chapters also the terms event and event trace are needed. Event means a pair consisting of a task and the event type – effectively it says that task A is either started or completed. Event trace is an ordered set of events in an event log where all events belong to a single case. (Wen et al., 2009)

Succession relation is defined such that task A is succeeded by task B if in at least one event trace there is not another complete task occurrence between the two task occurrences A and B. This practically means that the log needs to contain at least one event trace where the execution of new task starts immediately after previous task has ended. This can be detected by searching for start occurrences after a task has ended – each start occurrence means succession and this can be repeated until complete occurrence is found. Succession relation is indicated using notation $A >_w B$ which means that A succeeds B.

Intersection relation, on the other hand, is defined such that task A intersects with task B if there is start or complete occurrences belonging to task B between the start and complete occurrences of task A. This can be detected by searching for occurrences belonging to other activities after a starting activity has been found – each occurrence means parallelism and this can be repeated until ending occurrence of the original activity is found. Intersection relation is indicated using notation $A \times_w B$ which means that A intersects B. Intersection relation is symmetric; if A intersects B, then also B intersects A.

Using the same example model as with illustrating heuristic miner the succession and intersection tables can be calculated. The tables are read so that the column label indicates the preceding task and row label indicates the following task. For example, in the Table 6 the value "1" in "Did it help?" – "Patient recovered" means that there is at least one direct flow from "Did it help?" to "Patient recovered".

Table 6: Succession relations of the example model

Succession >	**Symptoms**	**Visit the doctor**	**Treatment or not?**	**Patient recovered**	**Treatment**	**Did it help?**
Symptoms	0	1	0	0	0	0
Visit the doctor	0	0	1	0	0	0
Treatment or not?	0	0	0	1	1	0
Patient recovered	0	0	0	0	0	0
Treatment	0	0	0	0	0	1
Did it help?	0	0	0	1	1	0

Table 7: Intersection relations of the example model

Intersection x	Symptoms	Visit the doctor	Treatment or not?	Patient recovered	Treatment	Did it help?
Symptoms	0	0	0	0	0	0
Visit the doctor	0	0	0	0	0	0
Treatment or not?	0	0	0	0	0	0
Patient recovered	0	0	0	0	0	0
Treatment	0	0	0	0	0	0
Did it help?	0	0	0	0	0	0

The example model does not contain any parallel execution of tasks so the intersection counts (Table 7) are all 0. The succession table is thus a lot more interesting in this case – in particular it shows correctly all the splits and joins of the model. It should be noted that the actual counts of relations are not calculated in the algorithm presented in (Wen et al., 2003) – this drawback is countered with refined mining algorithm presented later. These basic relations form the basis for derived relations that are used in the actual Petri net calculation.

The derived relations and their definitions are (¬ indicates negation) (Wen et al., 2003)

- $A \rightarrow_w B$ if and only if $A >_w B$ and $\neg(A \; x_w \; B)$
- $A \parallel_w B$ if and only if $A \; x_w \; B$
- $A \#_w B$ if and only if $\neg(A >_w B)$ and $\neg(A \; x_w \; B)$
- $A \nparallel_w B$ if and only if $\neg(A \; x_w \; B)$

It is quite trivial to notice that the last derived relation contains also relations A $\rightarrow_w$ B and A $\#_w$ B. The last relation is also actually not used in the mining algorithm so the simple rules for extracting correct derived relations are 1) if tasks A and B intersect, the relation is A $\|_w$ B, 2) if 1) does not apply and B succeeds A, the relation is A $\rightarrow_w$ B, and 3) if neither 1) nor 2) apply, the relation is A $\#_w$ B. Table 8 contains these derived relations for the example model. As there are no intersections, also the relation A $\|_w$ B is not present in the derived relations.

Table 8: Derived relations of the example model

	Symptoms	**Visit the doctor**	**Treatment or not?**	**Patient recovered**	**Treatment**	**Did it help?**
Symptoms	$\#_w$	$\rightarrow_w$	$\#_w$	$\#_w$	$\#_w$	$\#_w$
Visit the doctor	$\#_w$	$\#_w$	$\rightarrow_w$	$\#_w$	$\#_w$	$\#_w$
Treatment or not?	$\#_w$	$\#_w$	$\#_w$	$\rightarrow_w$	$\rightarrow_w$	$\#_w$
Patient recovered	$\#_w$	$\#_w$	$\#_w$	$\#_w$	$\#_w$	$\#_w$
Treatment	$\#_w$	$\#_w$	$\#_w$	$\#_w$	$\#_w$	$\rightarrow_w$
Did it help?	$\#_w$	$\#_w$	$\#_w$	$\rightarrow_w$	$\rightarrow_w$	$\#_w$

The actual mining algorithm browses through the log file and attempts to find three distinct sets needed for constructing a Petri net: 1) a set of transitions, 2) a set of places, and 3) a set of arcs. The tasks identified from the log are directly used as transitions. Places require more work. To determine them, all tasks are checked and their $\rightarrow_w$ relations recorded. The pairs of <PS, SS> are collected where both PS and SS are a subset of all tasks and for each A belonging to PS and B belonging to SS applies A $\rightarrow_w$

B and that no tasks belonging to PS are mutually parallel and that no tasks belonging to SS are mutually parallel. The largest of these identified pairs are taken into account as places. Finally, arcs are drawn between all the identified transitions and places.

In the Table 8 above the potential pairs for place selection are

1. <{Symptoms}, {Visit the doctor}>
2. <{Visit the doctor}, {Treatment or not?}>
3. <{Treatment or not?}, {Patient recovered, Treatment}>
4. <{Treatment}, {Did it help?}>
5. <{Did it help}, {Patient recovered, Treatment}>
6. <{Treatment or not?, Did it help?}, {Patient recovered, Treatment}>

The candidates 1 – 5 are quite obvious; however, candidate 6 requires more thinking. It can be recognized using the fact that for both elements in the pair's first set, "Treatment or not?" and "Did it help?", the follower relations $\rightarrow_W$ occur with the same tasks. It can also be noted after determining these pairs that the pair number 6 already contains numbers 3 and 5 and thus, as we are interested in the largest subsets, only candidates 1, 2, 4 and 6 are taken as places for the Petri net. Figure 8 shows the Petri net drawn based on this algorithm. It correctly shows all the alternate routes that exist in the example model and thus provides better results than the heuristic mining algorithm.

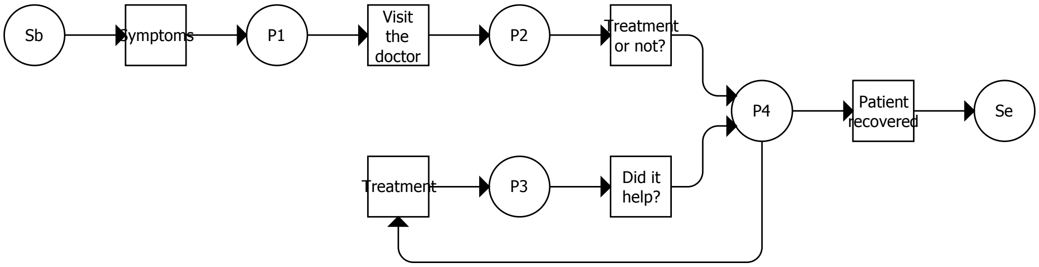


Figure 8: Petri net drawn using algorithm based on event types

A process model that is similar to that presented in Figure 6 can be drawn based on this Petri net. Even though the correct model could be found in this case there are still some issues present in the event type based algorithm. The most evident one became visible with real healthcare data where many activities and flows are encountered and the resulting process model can easily become too confusing. The algorithm was thus slightly modified to fit the needs of ProcessAnalyzer better – the actual mining algorithm used in the ProcessAnalyzer prototype is presented later in the chapter 4.1.

2.4 Pareto principle

Pareto principle is often used in business context as a rule of thumb stating that a large amount of the effects comes from a small amount of the causes. The term was suggested by Joseph Juran and the name was based on an Italian economist Vilfredo Pareto. Pareto discovered at the turn of the 20th century that a large amount of land was owned by a small amount of people in Italy (see for example Pareto, 1892 and Pareto, 1897). The principle is sometimes also called 80/20-rule – 80% of the effects are caused by 20% of the causes. Also other divisions are possible, depending on the case.

Nowadays the principle is commonly used to direct the efforts to the most relevant causes. It is often not useful to try to optimize small outliers as optimizing the vital few that account for high portion of the effects will provide much better results. This is likely the case in healthcare sector as well – it probably is not possible to find a process that fits all the cases found from treatment histories. However, what can be done is to identify the most common processes and filter out the less relevant cases. For more information on Pareto principle, the reader is advised to refer to, for example, (Juran & Gryna, 1988, Section 22)

2.5 Special characteristics of the healthcare sector

This chapter describes three recent topics presented in the literature regarding healthcare processes. First, the division between standard, routine and non-routine processes is discussed. After this discussion, the concept of episode and its relation to the processes are explained. The discussion on event-based medicine concludes the chapter.

2.5.1 Standard, routine and non-routine processes

In recent literature the classification of processes is often based on the terms variation and variety. As an example of these classifications is the division of processes into three basic categories: standard, routine and non-routine processes (Lillrank, 2002; Lillrank, 2003; Lillrank & Liukko, 2004). Before explaining these classes I will first explain the difference between variation and variety as they form a major theoretical background for process classification.

According to Lillrank, variation means how much the output of the process deviates from a preset target, for example the difference between the dimensions of a produced machine part and the dimensions of the planned part (Lillrank, 2003). It can be measured and be a subject of analysis and improvement actions. Variety, on the other hand, means different but functionally equivalent targets, such as different colors of the product (Lillrank, 2003). Variation is thus indicating errors, whereas variety means desired distinction and customization of products. In practice these terms are, however, often used as synonyms which can cause some confusion when reading literature.

When classifying processes, the most basic process type is often referred to as standard process. Standard processes mean processes where process input accepts and process output produces only one type of variety and where the variation of process output is measured (Lillrank, 2003). If the variation is within pre-set tolerance limits, the output of the process is accepted. For example, some traditional food producing processes fall into this category. A milk production line only accepts milk as an input and produces ready-to-drink milk with desired amount of fat.

Routine processes differ from standard process in that they accept more than one input varieties and produce more than one output varieties (Lillrank, 2003). The acceptance criteria are no longer simple as the amount of process output varieties can be large, though not infinite. If the milk production line mentioned above encounters huge variety of fat levels in its inputs, it is likely that the outputs vary in their fat level as well. The production can thus be separated into different varieties of milk – some with more fat, some with less fat.

In non-routine processes the input variety consists of so many elements that they all cannot be described or known beforehand (Lillrank, 2003). In these cases responding individually to each case can often be the only solution. However, the objective of a non-routine process is always clear which distinguishes this process type from truly chaotic processes. Quite often the processes in healthcare sector fall into the category of non-routine processes: the variety of inputs, who in healthcare sector are patients, is large as each person is individual (Lillrank & Liukko, 2004).

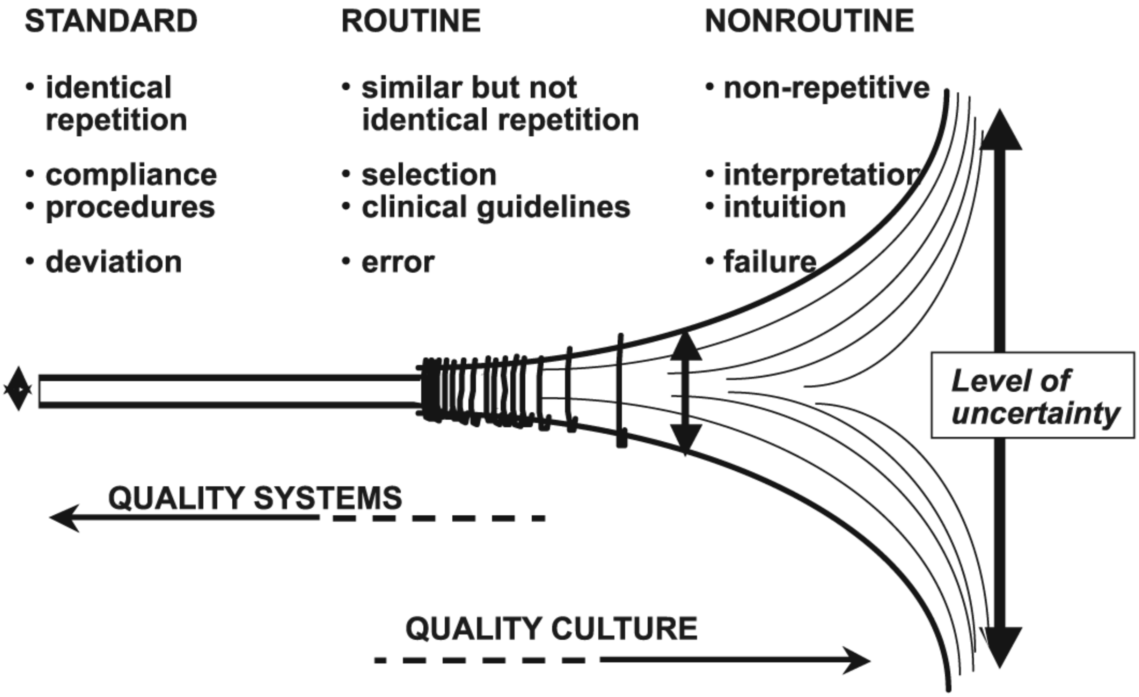


Figure 9: Quality broom (Lillrank & Liukko, 2004)

Lillrank has concretized these process types with his metaphor of a quality broom (Lillrank, 2002; Lillrank & Liukko, 2004). Figure 9 shows a classic presentation of a quality broom as expressed in (Lillrank & Liukko, 2004). The idea is that the stick of the broom depicts the standard processes – the stick is not flexible, it just does whatever it is designed to do. However, the stick is rigid and concrete which describes well the nature of the standard processes. The other extreme of the broom are the bundle of straws which describe non-routine processes. They are flexible and behave in a non-repetitive manner. Routine processes are somewhere in the middle where the rigid stick is connected to the flexible straws. The increasing level of uncertainty is indicated in the Figure 9 by the vertical arrows – the stick is quite thin indicating well-formulated processes whereas the wide straws cause increasing amount of uncertainty in the process.

2.5.2 Episodes in healthcare

As has been stated earlier, the focus of this study is to investigate techniques and develop a tool for automatically modeling processes based on log files. In the healthcare focus this practically means finding and mapping the medical treatment activities that the patient encounters when he or she seeks treatment for a certain illness or ailment. In literature the processes observed by patient are called episodes. The concept of episode was first formulated by Solon et al. already in the 1960s (Solon et al., 1967); their definition is as follows: "An episode of medical care is a block of one or more medical services received by an individual during a period of relatively continuous contact with one or more providers of service, in relation to a particular medical problem or situation."

There are three distinctive basic elements in this definition: patient focus, limited period of time and a particular problem. An episode describes from the patient's point of view what he or she encounters in a recovery process. This matches very well with the patient centric focus of this study. Episode does not thus attempt to describe how certain patients are generally treated, and more importantly, it does not attempt to visualize the perception of the current treatment processes the process owners have – instead it shows what the situation actually is, based on facts.

Another important element in the definition of episode is the limited period of time. The importance of time is very fundamental in an episode – if the time between activities is long, the activities are not likely to be associated with each other and belong to different episodes. There is, however, no universal law stating a time limit after which an activity can be considered to belong to a different episode. In some cases the nature of the patient's condition is such that treatment activities can be only performed once a month or, for example, every two months. Still these activities clearly belong to the same patient episode aiming at recovery from the medical condition. In another case the patient might suffer from two illnesses in a period of one month, and if he or she seeks medical help for these illnesses, the two cases are considered as different episodes, even though the illness would be the same.

The third fundamental element in the episode definition by Solon et al is that an episode always describes a single medical problem or situation – if the same patient undergoes treatments for several medical conditions at the same time, each of these treatments constitutes a separate episode. This naturally means also that a patient can be part of several different episodes concurrently if he or she suffers from several medical conditions.

Even though the concept of episodes is over 40 years old, its importance has grown larger in practical situations only in the last 10 years. In 1997 Brailer & Hackett suggested that the time would be right to start applying episodes in practice as the data collection had evolved substantially (Brailer & Hackett, 1997). In the recent years the analogue between episodes of healthcare and customer order-to-delivery chains in traditional industry has been emphasized (Kujala et al., 2006). The idea is that as manufacturing industry tries to reduce the amount of work in progress (WIP) to streamline the processes, the healthcare industry should try to reduce the amount of patients in progress (PIP) (Kujala et al., 2006) by eliminating non-value-adding times from the patient episodes.

There is a fundamental difference between patient episodes and processes. While patient episode describes the activities that are done to or done by an individual patient, the process describes the organizational healthcare system performing the activities. When a patient interacts with the healthcare service providers, a so called service event occurs: the service event links the patient episode and the process together. (Lillrank et al., 2009).

Figure 10 illustrates the basic concepts presented in (Lillrank et al., 2009). The circles in the picture describe an atomic health event in the patient episode while the rectangles illustrate steps in the process. These two intersect in a service event which is initiated by certain prerequisites for mobilization. After a service event the process is passed through a handover to the next state. The thicker line represents the actual patient episode which, as shown in the picture, consists also of time spent in non-value-adding activities outside the treatment process. In these steps the patient can be classified as a patient-in-progress.

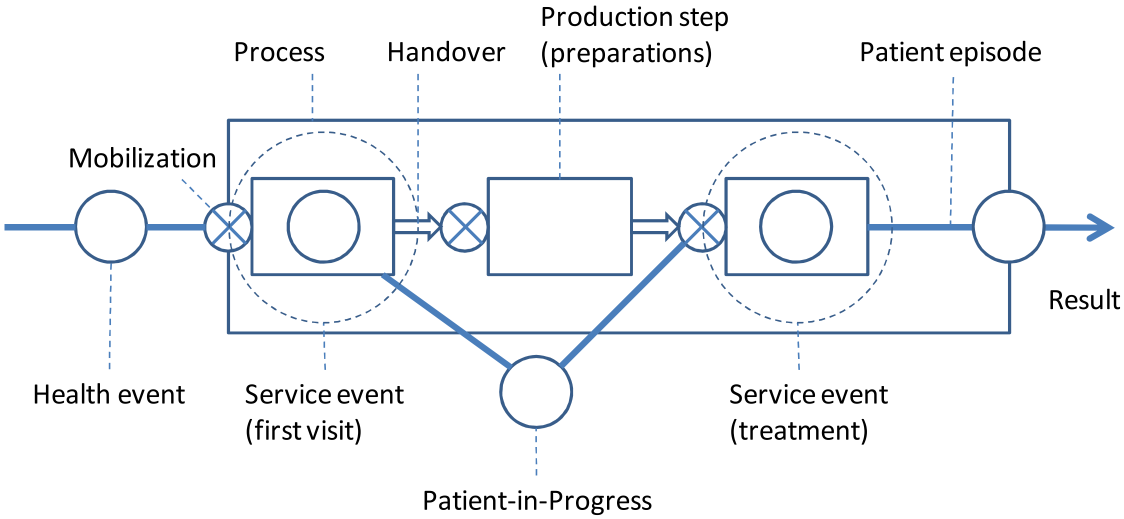


Figure 10: Illustration of episodes in healthcare (Lillrank et al., 2009)

The focus of this study and the ProcessAnalyzer is on analyzing the episodes. Mapping them using process mining techniques allows process owners to see the actual episodes, after which they can start to optimize the service events also from the patient’s point of view. The actual patient perceived lead times come also concrete when mapping the patient episodes; the implication of this is that focus can be put on reducing the amount of patients in progress.

2.5.3 Evidence-based medicine

This study focuses on identifying real, fact-based processes from the log data. As was explained in chapter 2.1.4, the process maps can be generally classified to three basic categories: normative, hermeneutic and positivistic. The processes modeled by experts usually are hermeneutic. Normative processes describe things as they should be or as the person doing the modeling wants them to be. They can be used to dictate the correct ways of working or steer the process to a new direction, but they should not be used as a guideline of how things currently work. Positivistic process models address this problem. They explain quantitative facts of the process concentrating on measurable properties such as patient flows. As the processes modeled by process mining techniques use real life data as the basis, the outcome is a positivistic process model.

When combined with information on what actually happens to the patient, the positivistic process model supports well the ideology of evidence-based medicine. The origins of evidence-based medicine come from mid-19th century Paris (Sackett et al., 1996), but in the last decade it has gained more attention. Especially Dr. David Sackett has addressed the topic in the literature, and for example in 1997 he defines evidence-based medicine as “the conscientious, explicit and judicious use of current best evidence in making decisions about the care of individual patients” (Sackett, 1997). In reality, however, the doctors quite often use the methods they have most experience in when treating the patients and do not rely on the factual, evidence-based findings presented in the vast amount of medical literature available (Pfeffer & Sutton, 2006).

The decision making should of course not be solely based on external evidence – the patient cases are different and one solution does not fit them all. This is why the evidence-based approach should, according to literature, be used together with doctor’s clinical expertise (Sackett et al., 1996). The evidences found from literature do not provide clear instructions on how to perform in different situation; the doctors need to interpret them in order to identify the relevant ones for each case.

ProcessAnalyzer approaches the problem of mapping the processes from the evidence-based point of view. This is somewhat different from the general view of evidence found from literature, but it still tells the user how a particular process is being run from the patient perspective. By using the ProcessAnalyzer the doctors could easily see what treatment processes are really like and they could also quite easily extract the best cases as the baseline of the optimal treatment. Then they could apply their own expertise and adapt the best practices to each case individually.

3 Research setting and methodology

This chapter describes the focus of the study as well as the data collected to validate the ProcessAnalyzer. Also the principle of how data is combined from different sources is explained.

3.1 Focus of the study

The focus of this study is on healthcare sector – the data is taken from two hospitals located in Helsinki area in Finland. This enables focusing on a limited group of processes in an interesting market area. This thesis has so far presented two process mining tools and the event-based mining algorithm is selected for prototyping with small modifications as it appears to give the most valid results.

The study does not analyze the treatment processes of the units where data is taken from. The data is used simply for verifying the data analysis tools that are present in the current literature, not for finding new medical or process related results from the data.

Despite the study being focused on healthcare, the ProcessAnalyzer prototype is a generic tool for analyzing process data and there should be no need to restrict its use to only healthcare. This is something QPR needs to think of as a business opportunity – this study is not going to evaluate financial potential.

3.2 Data

The modified process mining algorithm presented later in chapter 4.1 is validated in this study by using data from different sources. As the focus of the study is on healthcare sector, data from healthcare systems is naturally used. However, during the prototyping phase of the study also other data was used when healthcare data was not available. This data was generated by simulating processes and logging the results. In later phases of the study real healthcare data was acquired from two sources: Meilahti hospital's cardiology unit and Töölö hospital. This chapter describes the different kinds of data that are used in the algorithm validation.

3.2.1 Data generated by simulation

In the early phases of ProcessAnalyzer development it was already clear that the focus is on healthcare sector. However, acquiring sensitive data from real patient treatment records turned out to be quite time-consuming so it was decided that the early phases of ProcessAnalyzer are validated with data generated by simulating an existing process model. This could be easily accomplished using QPR ProcessGuide's simulation capabilities.

QPR ProcessGuide's existing simulation test model was used to generate simulation data. The model is shown in Figure 11 and Figure 12 below – basically it is a simple test model describing the demand-supply chain of an artificial product. QPR ProcessGuide allows adding simulation properties, such as distribution of lead time and input and output conditions to activities and flows connecting the activities and this enables running simulation. Slight adjustments were needed to the logging of simulation results to produce the log in a format required by the process mining algorithms.

Even though QPR ProcessGuide supports extracting log from simulation, this log is not usable for the mining algorithms as such because it consists of aggregated results and case-by-case results in a wrong format. This is why I modified QPR ProcessGuide slightly in order for it to produce the simulation log in a format that is better suited for this purpose. As QPR ProcessGuide is written in Delphi, I also made my modifications regarding improved logging in Delphi. After these modifications the simulation tools of QPR ProcessGuide produce log in a simple CSV-like (Comma-Separated Values) (Shafranovich, 2005) format. The definition of CSV files states that each value is separated by comma but I decided to use semicolon instead as it is more rarely used when naming the actual activities or organization units. Consequently, the refined log file contains one row for each row in the desired log table and each table cell is separated by a semicolon.

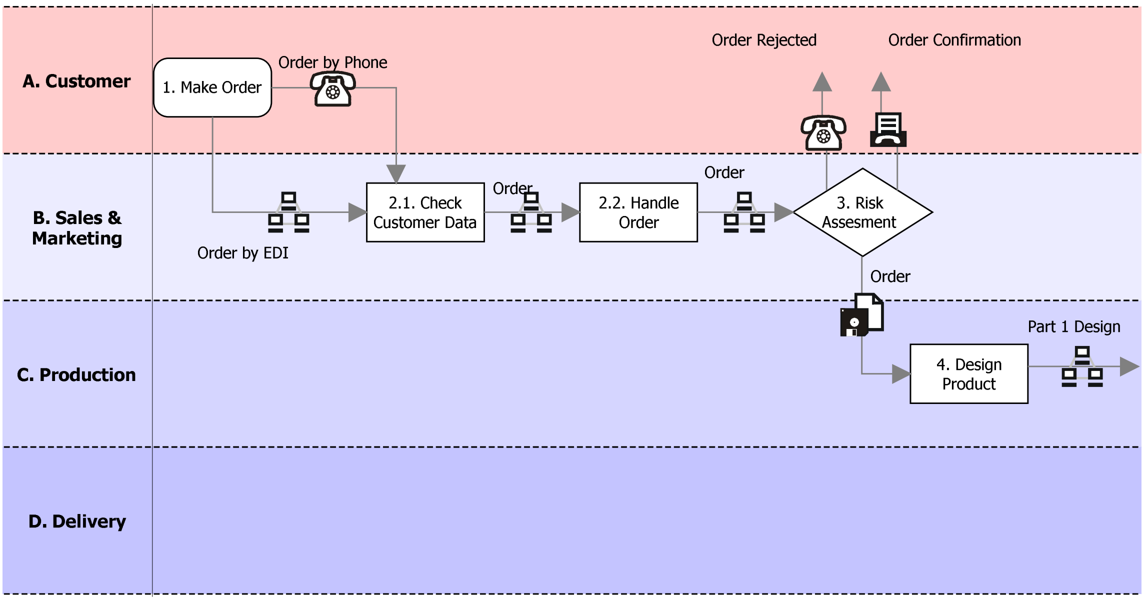


Figure 11: Simulation example model 1/2

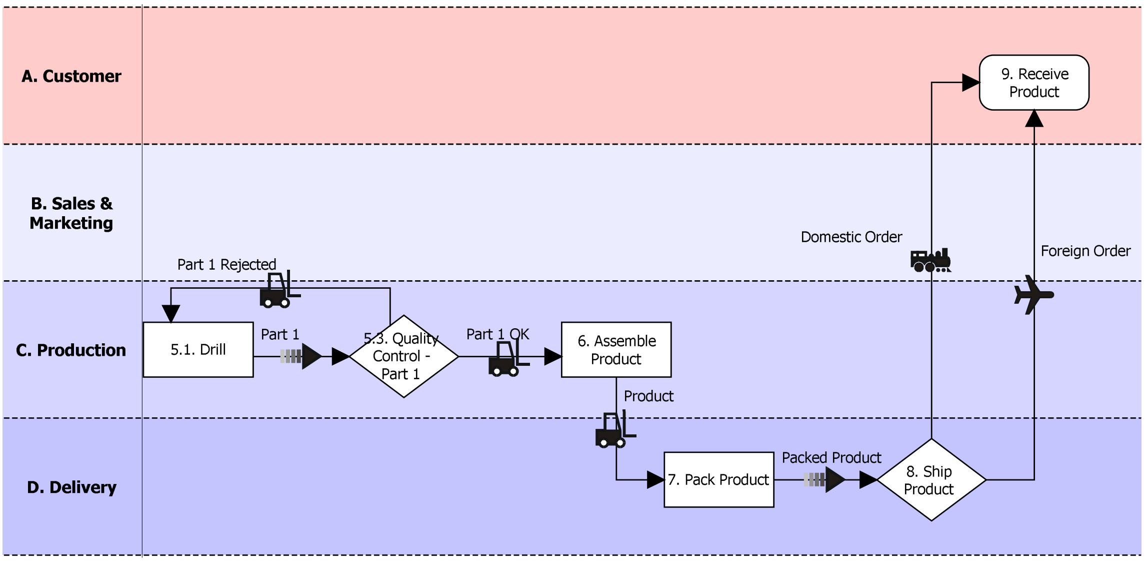


Figure 12: Simulation example model 2/2

Once the log is in a CSV format it is easily copied to, for example, Excel and presented in a table format. This artificial data generated by simulation tries to replicate a real workflow log by showing information that usually is always present in logs – the performing organization unit, the performed activity, case identifier and start and end times. As an example the table below (Table 9) presents a snapshot of the data generated.

Table 9: Example data generated by simulation

Case id	Activity	Organization unit	Event type	Time
102471374	1. Make Order	A. Customer	START	6.7.2008 23:33
102471374	1. Make Order	A. Customer	COMPLETE	7.7.2008 23:33
102471374	2.1. Check Customer Data	B. Sales & Marketing	START	7.7.2008 23:43
102471374	2.1. Check Customer Data	B. Sales & Marketing	COMPLETE	8.7.2008 23:39
102471374	2.2. Handle Order	B. Sales & Marketing	START	8.7.2008 23:44
102471374	2.2. Handle Order	B. Sales & Marketing	COMPLETE	9.7.2008 21:36
102471374	3. Risk Assesment	B. Sales & Marketing	START	9.7.2008 21:41
102471374	3. Risk Assesment	B. Sales & Marketing	COMPLETE	10.7.2008 17:35

Similar data was created using other test models as well. The resulting data is always in a similar format as that presented in Table 9; the values are naturally different but the basic idea remains the same. Data generated by different simulation models was used in early phases of the study while waiting for real healthcare data.

3.2.2 Data acquired from hospitals

The real patient records used in this study were acquired from two hospitals located in Helsinki, Finland. First data consists of data from cardiovascular research unit in Meilahti hospital as well as related first aid and cardiology polyclinics at Meilahti and Maria hospitals and the wards at Meilahti hospital. The second data is collected from three units at Töölö hospital – the accident and emergency unit, the surgery unit and the wards.

The data from Meilahti hospital consists of the same data that was used on a study conducted by Nordic Healthcare Group (NHG) at the cardiology unit of Meilahti hospital (Eklund et al., 2007). NHG consultant Paulus Torkki agreed to remove sensitive patient information from the data, including social security numbers which he replaced by sequential patient ids. This enabled access to tens of thousands of rows of real healthcare data and the prototyping of ProcessAnalyzer could start in a more realistic situation.

Similar case applies to the other test data set – the data has been collected by Antti Peltokorpi who uses it in his dissertation study at Töölö hospital (Peltokorpi, forthcoming in 2010). As with the Meilahti data, social security numbers were replaced by sequential patient ids. The major contrast with this data compared to Meilahti data is that it contains cases where patient, according to the log, appears to be in both a ward and the surgery at the same time. This is caused by the way the patient records are updated at Töölö – if the patient is operated he or she is logged to the patient records of the surgery unit but the longer period of ward treatment is not interrupted.

3.2.3 Combining healthcare data acquired from different sources

Even though the healthcare data used in validation was collected from a relatively narrow area, it still consists of data from different units. For example the Meilahti hospital data includes data from cardiology and first aid polyclinics at Meilahti and Maria hospitals, the cardiovascular research unit and different wards. Unfortunately, the database formats at different units seem to differ quite a lot – especially the wards seem to log a larger amount of data to the patient records than the polyclinics. The same problem is present in the Töölö hospital data as well as it consists of unit specialized in accidents and emergencies, the surgery unit and the ward, each with their own database formats.

It is clear that there is no single and universal way of logging service events at the different units of the Finnish healthcare sector. This means that a cross-functional analysis carried out by ProcessAnalyzer requires data manipulation in order to reach a generic database format. In essence, the data has to be handled and combined to a simple table showing only the relevant information needed for process mining.

QPR FactView is used for this purpose. It allows combining data from different sources into a single model that can be analyzed and filtered. The scripting language used to collect the data is quite extensive and allows for example filtering and correcting simple errors in the data and concatenating and manipulating different textual data presentations. The major benefit of using QPR FactView for data manipulation is that as it collects the data using a written script, the script can be reused if the data changes. As an example, if this same data handling was done in Microsoft Excel, it would be necessary to do it again when a new analysis on similar data is done as the data will again be in the original, unhandled format. On the other hand, once the script for QPR FactView model is written, it can be rerun on a new data, assuming that the data format remains the same.

Both Meilahti and Töölö data are written as separate QPR FactView models and the resulting combined tables are exported in Microsoft Excel format. This produces similar table to that presented in Table 9, but in this case it describes real patient data. The data can also be filtered to show only interesting cases by using QPR FactView’s built-in filtering tools. For example, only some diagnosis codes can be selected for further analysis.

4 Results

This chapter presents the results of the study. The chapter begins with describing the refined mining algorithm used in ProcessAnalyzer. After this the ProcessAnalyzer prototype produced while conducting this study is introduced. The three cases presented afterwards form a major part of this chapter. They describe and evaluate the suitability of ProcessAnalyzer in three different situations – simulated data, cardiology data from Meilahti hospital and accident and emergency polyclinic data from Töölö hospital. After the case descriptions the concept of process classification is discussed. The chapter is concluded by addressing performance of the process mining algorithm used in ProcessAnalyzer.

4.1 Refined mining algorithm

The refined mining algorithm is largely based on the event type based algorithm presented in chapter 2.3.2 and in (Wen et al., 2009). It also applies some of the principles from heuristic mining algorithm presented in chapter 2.3.1 and in (Weijters & van der Aalst, 2003). It is the algorithm used in QPR ProcessAnalyzer and is also evolving with the product development of the analyzer. The algorithm presented in this study describes the algorithm at the time this thesis was written. It should also be noted that the full algorithm is part of confidential code that is property of QPR Software, which means that not all parts of it can be published in a thesis.

4.1.1 Counts of flows

One of the problems of event type based mining algorithm is that it only recognizes whether some flow is present in the log or not. This did not cause troubles with the example model as it contained very few activities and all the flows in the log were relevant. However, in real life data this rarely is the case. There can often be a lot more activities and some flows are only present in very few cases. Adding all these to the process map adds the complexity of the mined process model and makes it hard to analyze the results. To counter this, counting the number of different flows was added to the mining algorithm.

The principle is quite simple; similarly as with heuristic mining algorithm (Weijters & van der Aalst, 2003) the matrix showing flows is updated each time a new flow is found by adding the count of that flow. The resulting matrix is similar to that presented in Table 6, except that the values now also show the actual number of cases where that particular flow is present. Table 10 shows the succession relations and their counts from the example model. The basic idea behind this is that the derived matrix, based on which the process model is drawn, can be calculated taking only some of the flows into account. For example, if a flow is present only in 2 cases out of 1000, it can probably be discarded from the general process model.

Table 10: Succession relations derived with refined mining algorithm

Succession >	**Symptoms**	**Visit the doctor**	**Treatment or not?**	**Patient recovered**	**Treatment**	**Did it help?**
Symptoms	0	1000	2	0	0	0
Visit the doctor	2	0	998	0	0	0
Treatment or not?	0	0	0	720	280	0
Patient recovered	0	0	0	0	0	0
Treatment	0	0	0	0	0	358
Did it help?	0	0	0	280	78	0

4.1.2 Ignoring intersection relations

Another, and more radical, change to the original event based mining algorithm is discarding the intersection relation altogether. The reason is that when analyzing the healthcare processes we are interested in what happens to a patient, and the patient cannot be in multiple places at the same time. Even though the data sometimes indicates this by, for example, showing that patient first arrives to the ward and then goes to surgery while still being at ward, the patient actually proceeds from the ward to surgery and back to the ward again.

To counter this problem the mining algorithm was adjusted further. The intersection relation is no longer recorded; the algorithm simply checks succeeding activities and derives process map based on those. This naturally means that the cases of true parallelism, such as manufacturing products in a factory, are not mined correctly by this revised algorithm. When applied to healthcare, however, this is a natural way of thinking – the patient whose patient episodes we are interested in cannot be in multiple places at the same time.

The refined event type based algorithm keeps track of previously executed activity. When a new activity starts it assumes that there is a flow between the previous and the current activities. In addition to these, when an activity execution ends, the algorithm checks if there are open activity executions – if this is the case, a flow between the ended activity and the open activity is added. This enables adding the aforementioned flow from surgery to ward even though this flow is not actually visible in the log.

4.1.3 Adding special flows

The ProcessAnalyzer uses flow counts between activities to indicate for example the major flows and to show the amount of cases that go to each route starting from an activity. The indication of major flows is achieved by adjusting the thickness of the arrows. The algorithm is also done so that the flows that less than 20% of all the cases are part of are shown with transparency. The transparency level is higher when the amount of cases that pass through the flow goes lower.

The activities that start and end the process need separate handling for the counts of cases to work, as otherwise the number of incoming and outgoing flows from these activities would not match. Special flows are thus added in the refined mining algorithm. When a new case starts a new flow is counted from a special “start step” to the first actual activity of the case. Particularly with the healthcare data these special flows are visible in almost every activity as the data is collected from only a fraction of the hospitals and there are plenty of cases that contain only one activity. Similar handling is present also when the case ends – a flow from the last actual activity of the case to a special “end step” is added.

This special handling makes the counts of incoming and outgoing flows related to an activity match. For analyzing purposes as well as usability this is very important – the user is not confused because he or she can check that the amount of incoming flows is the same as the amount of outgoing flows.

4.1.4 Drawing the process model

Once the algorithm presented in the previous chapters has calculated the necessary dependency relations, the next step is to transform these numeric results to a process model. This is done by using QPR ProcessGuide. More specifically, the ProcessAnalyzer uses the QPR ProcessGuide API (Application Programming Interface) that in essence is an alternative way to control QPR ProcessGuide. Instead of visually drawing the process elements, the API provides a programmatic way to perform basic process modeling and this is ideal solution for an algorithmic approach.

In practice this means that the mining algorithm, while calculating the dependency relations presented in the previous chapters, also calls the QPR ProcessGuide API to draw the calculated activities and flows. Using the API requires determining quite specifically where the elements are drawn to the resulting process model and these coordinates are calculated in the ProcessAnalyzer. In practice, the ProcessAnalyzer first determines which activity is the best candidate as the first element in the process. This is done by identifying the special starting flows of the model – the activity with most of these flows is selected as the first activity and drawn to the leftmost column of the process model.

Once the first element is determined, the other elements are drawn in the order so that the elements that are connected to the first element most often are drawn closest to the first element. Once connections are all iteratively handled, the activities that are not yet drawn to the model are handled separately. This logic tries to arrange the elements to a logical order in the process model. Naturally this arrangement does not fit to all cases and some manual rework might be needed afterwards; however, this is easily done as the model is created using a tool specifically designed for process modeling.

When the activities are all added to the process model, the next step in drawing the model is to add the arrows indicating flows. These are naturally added between the activities that have been identified as the relevant relations in the mining algorithm, that is, activities with succession relation value higher than 0. The QPR ProcessGuide API has some limitations regarding the visual properties of flows – in practice, the start and end positions of flows between activities cannot be determined, but instead QPR ProcessGuide calculates them automatically. This causes problems as the automatic calculation does not always result in flows that are optimal for this situation, especially as all the flow labels indicating the volume appear on top of each other.

The special flows indicating cases arriving to and exiting from the system are also drawn by the ProcessAnalyzer. The incoming flows start from above the activity and the outgoing ones lead to this same area; Figure 13 explains these special cases. The QPR ProcessGuide API supports giving the exact coordinate of the starting point of incoming flow or ending point of outgoing flow as they are not tied to any activity. However, the point that connects these flows to activity is calculated automatically by QPR ProcessGuide and this results in somewhat strange process models where the incoming flows are connected to the left side of the activity and outgoing flows start from the right side of the activity. Forcing the QPR ProcessGuide to draw these as presented in Figure 13 required some manual editing of QPR ProcessGuide source code.

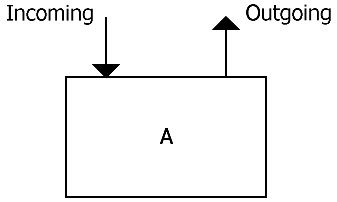


Figure 13: Special flows

4.1.5 Pareto principle

A brief introduction to Pareto principle was given in chapter 2.4. This principle is also used to some extent in ProcessAnalyzer. The tool allows selecting a threshold for drawn process models; any flow with volume of less than the threshold will not be drawn. If hiding flows this way also causes some activities to have no flows associated to them, they are not drawn either. In essence this means that the process model will only show major flows between activities, enabling easier interpretation. Another way of applying Pareto principle is to classify the process instances and to ignore classes with low volumes of cases. This differs from the previous perspective so that it still preserves full process instances in the aggregated model.

4.2 ProcessAnalyzer prototype

An important part of this study has been creating a prototype of ProcessAnalyzer. This prototype is used to validate the aforementioned refined mining algorithm (chapter 4.1) with real life data, as well as to demonstrate the product concept of ProcessAnalyzer both inside and outside QPR. The prototype consists of three quite distinct parts (Figure 14); the full product naturally should try to minimize the need for these interfaces, especially if it is intended for direct end consumer usage.

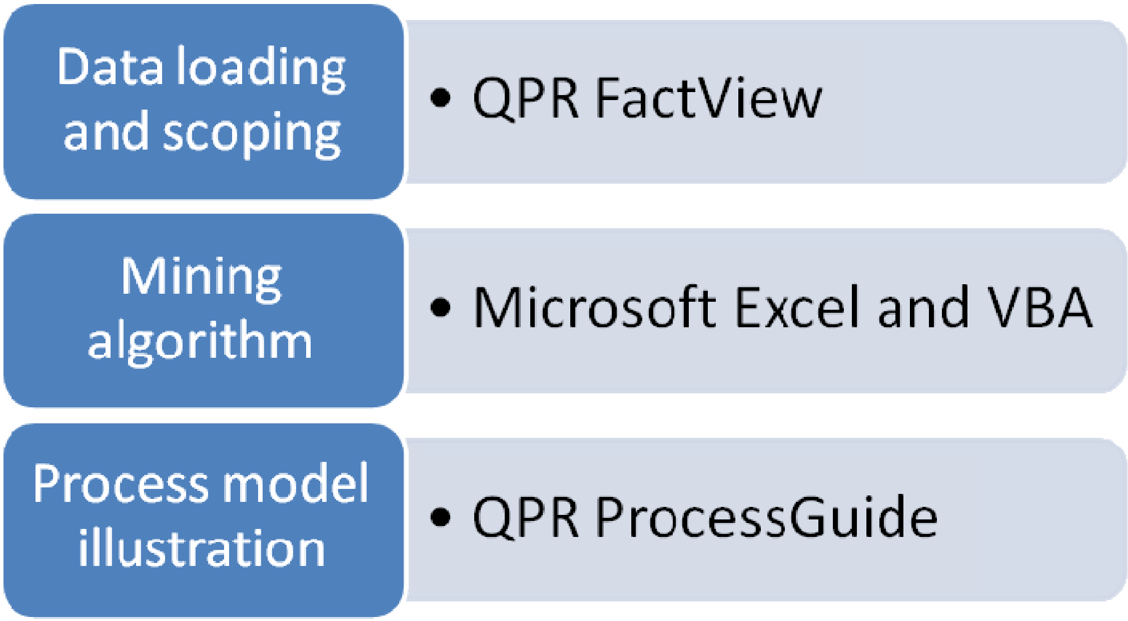
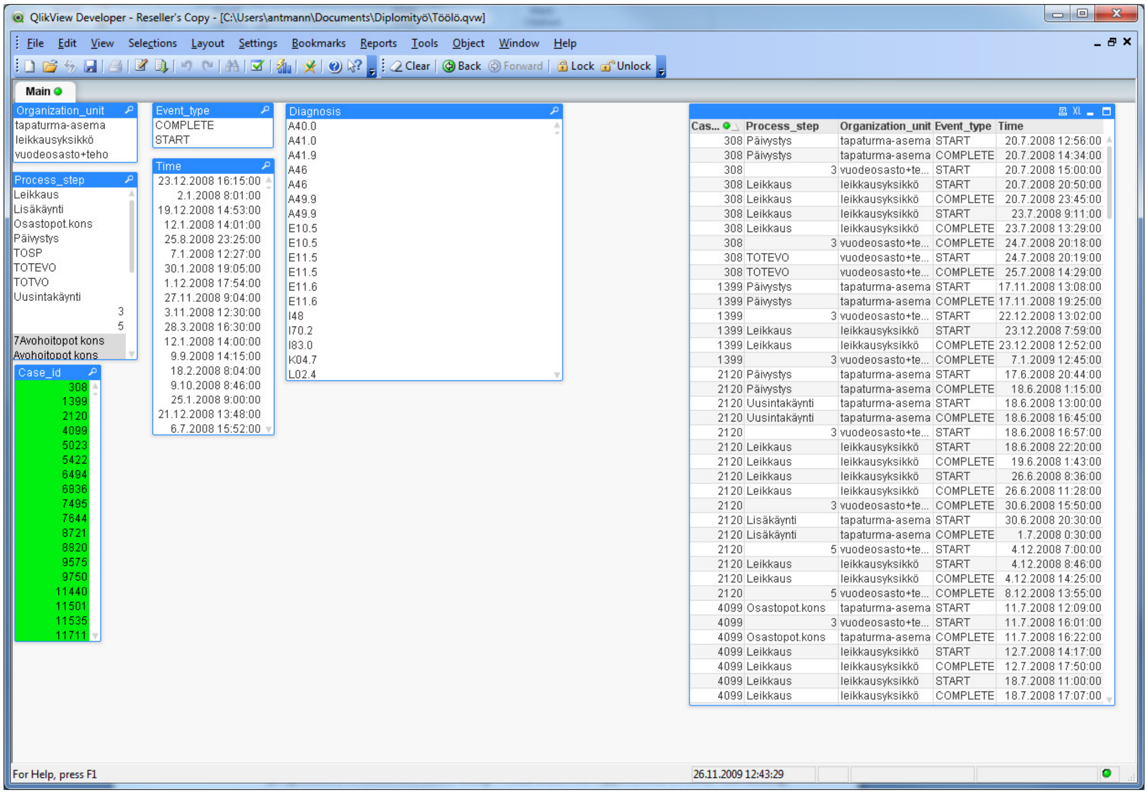


Figure 14: ProcessAnalyzer prototype

The first part of the prototype is using QPR FactView to load data from different sources. This has already been discussed in chapter 3.2.3. To summarize the basic idea, the QPR FactView enables writing data load operations to a script that can be rerun as many times as wanted. The data can also be quite extensively manipulated in the script which reduces need for major data adjustments in the source data. Once data is loaded to a FactView model, it can be scoped by selecting only some of the data rows for further analysis. In the ProcessAnalyzer prototype the data is grouped to the required format in QPR FactView. In practice this means adding a table box as a sheet object inside the FactView model and configuring the box so that it shows columns for case identifiers, activity names, organization units, event types and event occurring times. This table box can then be directly exported to Microsoft Excel format.

Figure 15 shows a screenshot taken from QPR FactView illustrating the first phase of making a process analysis using ProcessAnalyzer. The items on left part of the screen indicate items that can be used to scope only part of the data. The table box on the right part of the screen contains the table that is exported to Excel for analysis. Items marked by green color are used to scope the data – in this case only some case identifiers have been selected.


QlikView Developer - Reseller's Copy - [C:\Users\antmann\Documents\Diplomityö\Töölö.qvw]
File Edit View Selections Layout Settings Bookmarks Reports Tools Object Window Help
Clear Back Forward Lock Unlock
Main
For Help, press F1
26.11.2009 12:43:29

Organization_unit
tapaturma-asema
leikkausyksikkö
vuodeosasto+teho

Event_type
COMPLETE
START

Diagnosis
A40.0
A41.0
A41.9
A46
A46
A49.9
A49.9
E10.5
E10.5
E11.5
E11.5
E11.6
E11.6
I48
I70.2
I83.0
K04.7
L02.4

Process_step
Leikkaus
Lisäkäynti
Osastopot.kons
Päivystys
TOSP
TOTEVO
TOTVO
Uusintakäynti
3
5
7Avohoitopot kons
Avohoitopot kons

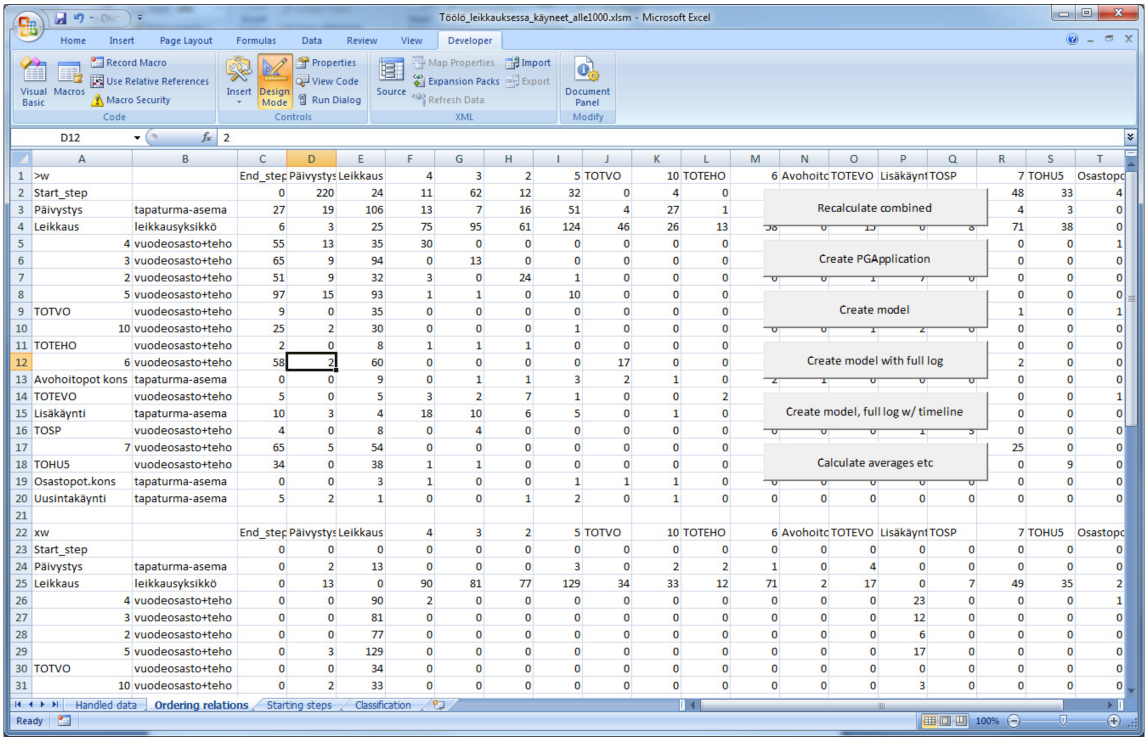
Time
23.12.2008 16:15:00
2.1.2008 8:01:00
19.12.2008 14:53:00
12.1.2008 14:01:00
25.8.2008 23:25:00
7.1.2008 12:27:00
30.1.2008 19:05:00
1.12.2008 17:54:00
27.11.2008 9:04:00
3.11.2008 12:30:00
28.3.2008 16:30:00
12.1.2008 14:00:00
9.9.2008 14:15:00
18.2.2008 8:04:00
9.10.2008 8:46:00
25.1.2008 9:00:00
21.12.2008 13:48:00
6.7.2008 15:52:00

Case_id
308
1399
2120
4099
5023
5422
6494
6836
7495
7644
8721
8820
9575
9750
11440
11501
11535
11711

Cas...	Process_step	Organization_unit	Event_type	Time
308	Päivystys	tapaturma-asema	START	20.7.2008 12:56:00
308	Päivystys	tapaturma-asema	COMPLETE	20.7.2008 14:34:00
308	3	vuodeosasto+te...	START	20.7.2008 15:00:00
308	Leikkaus	leikkausyksikkö	START	20.7.2008 20:50:00
308	Leikkaus	leikkausyksikkö	COMPLETE	20.7.2008 23:45:00
308	Leikkaus	leikkausyksikkö	START	23.7.2008 9:11:00
308	Leikkaus	leikkausyksikkö	COMPLETE	23.7.2008 13:29:00
308	3	vuodeosasto+te...	COMPLETE	24.7.2008 20:18:00
308	TOTEVO	vuodeosasto+te...	START	24.7.2008 20:19:00
308	TOTEVO	vuodeosasto+te...	COMPLETE	25.7.2008 14:29:00
1399	Päivystys	tapaturma-asema	START	17.11.2008 13:08:00
1399	Päivystys	tapaturma-asema	COMPLETE	17.11.2008 19:25:00
1399	3	vuodeosasto+te...	START	22.12.2008 13:02:00
1399	Leikkaus	leikkausyksikkö	START	23.12.2008 7:59:00
1399	Leikkaus	leikkausyksikkö	COMPLETE	23.12.2008 12:52:00
1399	3	vuodeosasto+te...	COMPLETE	7.1.2009 12:45:00
2120	Päivystys	tapaturma-asema	START	17.6.2008 20:44:00
2120	Päivystys	tapaturma-asema	COMPLETE	18.6.2008 1:15:00
2120	Uusintakäynti	tapaturma-asema	START	18.6.2008 13:00:00
2120	Uusintakäynti	tapaturma-asema	COMPLETE	18.6.2008 16:45:00
2120	3	vuodeosasto+te...	START	18.6.2008 16:57:00
2120	Leikkaus	leikkausyksikkö	START	18.6.2008 22:20:00
2120	Leikkaus	leikkausyksikkö	COMPLETE	19.6.2008 1:43:00
2120	Leikkaus	leikkausyksikkö	START	26.6.2008 8:36:00
2120	Leikkaus	leikkausyksikkö	COMPLETE	26.6.2008 11:28:00
2120	3	vuodeosasto+te...	COMPLETE	30.6.2008 15:50:00
2120	Lisäkäynti	tapaturma-asema	START	30.6.2008 20:30:00
2120	Lisäkäynti	tapaturma-asema	COMPLETE	1.7.2008 0:30:00
2120	5	vuodeosasto+te...	START	4.12.2008 7:00:00
2120	Leikkaus	leikkausyksikkö	START	4.12.2008 8:46:00
2120	Leikkaus	leikkausyksikkö	COMPLETE	4.12.2008 14:25:00
2120	5	vuodeosasto+te...	COMPLETE	8.12.2008 13:55:00
4099	Osastopot.kons	tapaturma-asema	START	11.7.2008 12:09:00
4099	3	vuodeosasto+te...	START	11.7.2008 16:01:00
4099	Osastopot.kons	tapaturma-asema	COMPLETE	11.7.2008 16:22:00
4099	Leikkaus	leikkausyksikkö	START	12.7.2008 14:17:00
4099	Leikkaus	leikkausyksikkö	COMPLETE	12.7.2008 17:50:00
4099	Leikkaus	leikkausyksikkö	START	18.7.2008 11:00:00
4099	Leikkaus	leikkausyksikkö	COMPLETE	18.7.2008 17:07:00

Figure 15: ProcessAnalyzer prototype, loading and scoping data

The second step in the ProcessAnalyzer prototype is the actual mining algorithm. This has been programmed to Microsoft Excel using Visual Basic for Applications (VBA). ProcessAnalyzer takes data from one Excel worksheet: this is where the data exported from QPR FactView is copied to. Next it calculates the dependency relations described in chapter 4.1. A process model can then be drawn using a simple button. Figure 16 illustrates this by showing an Excel spreadsheet that contains the dependency matrices calculated by the mining algorithm.

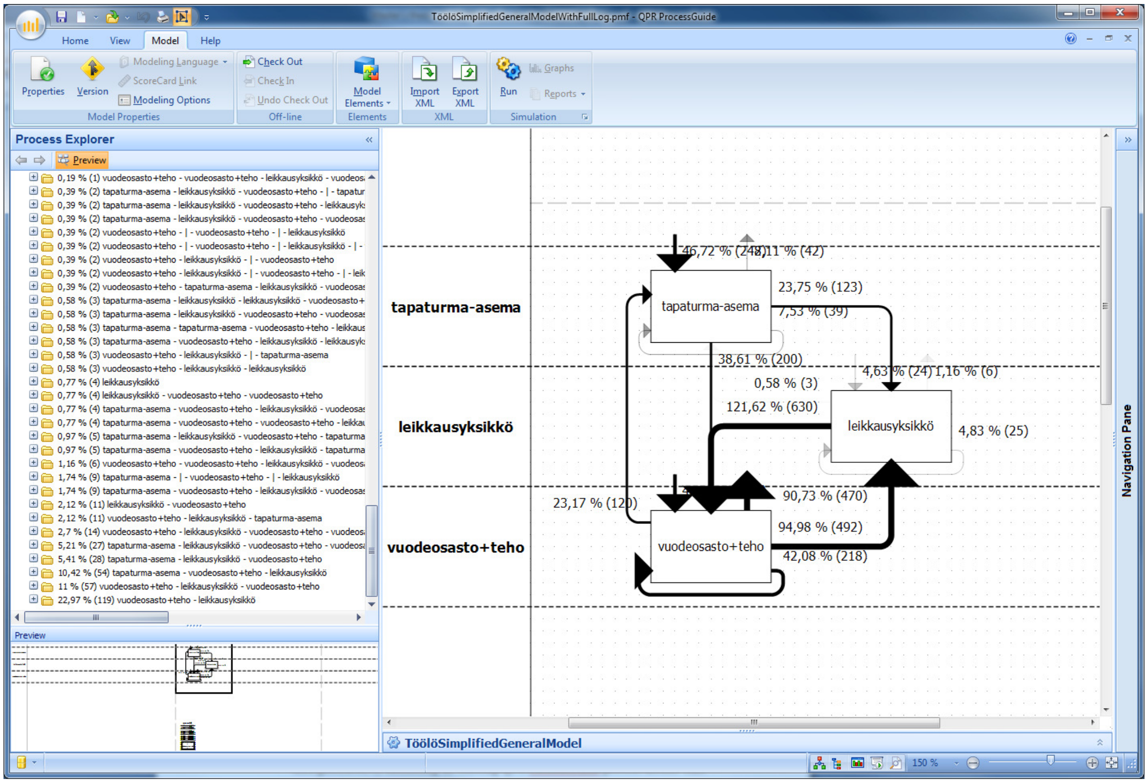


D12 2

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	>w		End_step	Päivystys	Leikkaus	4	3	2	5	TOTVO	10	TOTEHO	6	Avohoitc	TOTEVO	Lisäkäynt	TOSP	7	TOHU5	Osastopc
2	Start_step		0	220	24	11	62	12	32	0	4	0						48	33	4
3	Päivystys	tapaturma-asema	27	19	106	13	7	16	51	4	27	1						4	3	0
4	Leikkaus	leikkausyksikkö	6	3	25	75	95	61	124	46	26	13	58	0	15	0	8	71	38	0
5	4	vuodeosasto+teho	55	13	35	30	0	0	0	0	0	0						0	0	1
6	3	vuodeosasto+teho	65	9	94	0	13	0	0	0	0	0						0	0	0
7	2	vuodeosasto+teho	51	9	32	3	0	24	1	0	0	0	0	0	1	7	0	0	0	0
8	5	vuodeosasto+teho	97	15	93	1	1	0	10	0	0	0						0	0	0
9	TOTVO	vuodeosasto+teho	9	0	35	0	0	0	0	0	0	0						1	0	1
10	10	vuodeosasto+teho	25	2	30	0	0	0	1	0	0	0	0	0	1	2	0	0	0	0
11	TOTEHO	vuodeosasto+teho	2	0	8	1	1	1	0	0	0	0						0	0	0
12	6	vuodeosasto+teho	58	2	60	0	0	0	0	17	0	0						2	0	0
13	Avohoitopot kons	tapaturma-asema	0	0	9	0	1	1	3	2	1	0	2	1	0	0	0	0	0	0
14	TOTEVO	vuodeosasto+teho	5	0	5	3	2	7	1	0	0	2						0	0	1
15	Lisäkäynti	tapaturma-asema	10	3	4	18	10	6	5	0	1	0						0	0	0
16	TOSP	vuodeosasto+teho	4	0	8	0	4	0	0	0	0	0	0	0	0	1	3	0	0	0
17	7	vuodeosasto+teho	65	5	54	0	0	0	0	0	0	0						25	0	0
18	TOHU5	vuodeosasto+teho	34	0	38	1	1	0	0	0	0	0						0	9	0
19	Osastopot.kons	tapaturma-asema	0	0	3	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0
20	Uusintakäynti	tapaturma-asema	5	2	1	0	0	1	2	0	1	0	0	0	0	0	0	0	0	0
21																				
22	xw		End_step	Päivystys	Leikkaus	4	3	2	5	TOTVO	10	TOTEHO	6	Avohoitc	TOTEVO	Lisäkäynt	TOSP	7	TOHU5	Osastopc
23	Start_step		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	Päivystys	tapaturma-asema	0	2	13	0	0	0	3	0	2	2	1	0	4	0	0	0	0	0
25	Leikkaus	leikkausyksikkö	0	13	0	90	81	77	129	34	33	12	71	2	17	0	7	49	35	2
26	4	vuodeosasto+teho	0	0	90	2	0	0	0	0	0	0	0	0	0	23	0	0	0	1
27	3	vuodeosasto+teho	0	0	81	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0
28	2	vuodeosasto+teho	0	0	77	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0
29	5	vuodeosasto+teho	0	3	129	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0
30	TOTVO	vuodeosasto+teho	0	0	34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	10	vuodeosasto+teho	0	2	33	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0

Figure 16: ProcessAnalyzer prototype, calculating dependency relations

When the dependency relations have been calculated in ProcessAnalyzer prototype, the user can draw a process model based on these relations. When drawing the model, the prototype executes VBA code which extracts the information stored to the dependency matrices. Further on, it communicates with QPR ProcessGuide using the QPR ProcessGuide API (Application Programming Interface). It tells QPR ProcessGuide to create a new model and to add the objects to the model to the calculated positions. Full details are omitted here but the result is a process model similar to that presented in the screenshot below (Figure 17).



```
0,19 % (1) vuodeosasto+teho - vuodeosasto+teho - leikkausyksikkö - vuodeos
0,39 % (2) tapaturma-asema - leikkausyksikkö - vuodeosasto+teho - | - tapatur
0,39 % (2) tapaturma-asema - leikkausyksikkö - vuodeosasto+teho - leikkausyk
0,39 % (2) tapaturma-asema - leikkausyksikkö - vuodeosasto+teho - vuodeosas
0,39 % (2) vuodeosasto+teho - | - vuodeosasto+teho - | - leikkausyksikkö
0,39 % (2) vuodeosasto+teho - | - vuodeosasto+teho - | - leikkausyksikkö - | -
0,39 % (2) vuodeosasto+teho - leikkausyksikkö - | - vuodeosasto+teho
0,39 % (2) vuodeosasto+teho - leikkausyksikkö - | - vuodeosasto+teho - | - leik
0,39 % (2) vuodeosasto+teho - tapaturma-asema - leikkausyksikkö - vuodeosas
0,58 % (3) tapaturma-asema - leikkausyksikkö - leikkausyksikkö - vuodeosasto+
0,58 % (3) tapaturma-asema - leikkausyksikkö - vuodeosasto+teho - vuodeosas
0,58 % (3) tapaturma-asema - tapaturma-asema - vuodeosasto+teho - leikkaus
0,58 % (3) tapaturma-asema - vuodeosasto+teho - leikkausyksikkö - leikkausyk
0,58 % (3) vuodeosasto+teho - leikkausyksikkö - | - tapaturma-asema
0,58 % (3) vuodeosasto+teho - leikkausyksikkö - leikkausyksikkö
0,77 % (4) leikkausyksikkö
0,77 % (4) leikkausyksikkö - vuodeosasto+teho - vuodeosasto+teho
0,77 % (4) tapaturma-asema - vuodeosasto+teho - leikkausyksikkö - vuodeosas
0,77 % (4) tapaturma-asema - vuodeosasto+teho - vuodeosasto+teho - leikkau
0,97 % (5) tapaturma-asema - leikkausyksikkö - vuodeosasto+teho - tapaturma
0,97 % (5) tapaturma-asema - vuodeosasto+teho - leikkausyksikkö - tapaturma
1,16 % (6) vuodeosasto+teho - vuodeosasto+teho - leikkausyksikkö - vuodeos
1,74 % (9) tapaturma-asema - | - vuodeosasto+teho - | - leikkausyksikkö
1,74 % (9) tapaturma-asema - vuodeosasto+teho - leikkausyksikkö - vuodeosas
2,12 % (11) leikkausyksikkö - vuodeosasto+teho
2,12 % (11) vuodeosasto+teho - leikkausyksikkö - tapaturma-asema
2,7 % (14) vuodeosasto+teho - leikkausyksikkö - vuodeosasto+teho - vuodeos
5,21 % (27) tapaturma-asema - leikkausyksikkö - vuodeosasto+teho - vuodeos
5,41 % (28) tapaturma-asema - leikkausyksikkö - vuodeosasto+teho
10,42 % (54) tapaturma-asema - vuodeosasto+teho - leikkausyksikkö
11 % (57) vuodeosasto+teho - leikkausyksikkö - vuodeosasto+teho
22,97 % (119) vuodeosasto+teho - leikkausyksikkö
```

Figure 17: ProcessAnalyzer prototype, examining calculated process model

4.3 Case 1: Simulated data

4.3.1 Introduction

As was explained in chapter 3.2.1, the mining algorithm was developed by testing it with data generated by simulating an existing QPR ProcessGuide process model. This allowed focusing on developing a functional algorithm instead of spending time waiting for real healthcare data. The simulation model is a slightly modified version of the process model included in QPR ProcessGuide for illustrating its simulation capabilities. The process model has been already shown in Figure 11 and Figure 12, so I won’t repeat that here. It should be sufficient to say that the model describes an artificial demand-supply chain of a simple product consisting only of one part.

The example model was selected for the test for several reasons. First of all, it has been already thought of thoroughly and thus it looks quite realistic, even though it is very simple. A more important criterion, however, is that the model contains decision steps (from “3. Risk Assessment” and “5.2. Quality Control – Part 1”) and a loop; these are necessary for evaluating how the algorithm handles these special cases.

4.3.2 Research setting

The objective of this case study is to take the simulated data presented in chapter 3.2.1 and use it as input for the ProcessAnalyzer prototype. As the simulation model is constructed so that the result contains a feasible number of data rows, no further data scoping is needed. Thus the research setting is quite simple – the whole data set generated by simulation is used. A process model is drawn using ProcessAnalyzer prototype and this model can be compared to the original process model.

4.3.3 Mining results

The data generated by QPR ProcessGuide’s simulation functionalities contains a total of 1000 cases. The log containing details on each of these cases is used as input for ProcessAnalyzer in order to validate the model it generates. Figure 18 contains this mined process model. Comparing this to the original model (Figure 11 and Figure 12) reveals similarities and differences between the models.

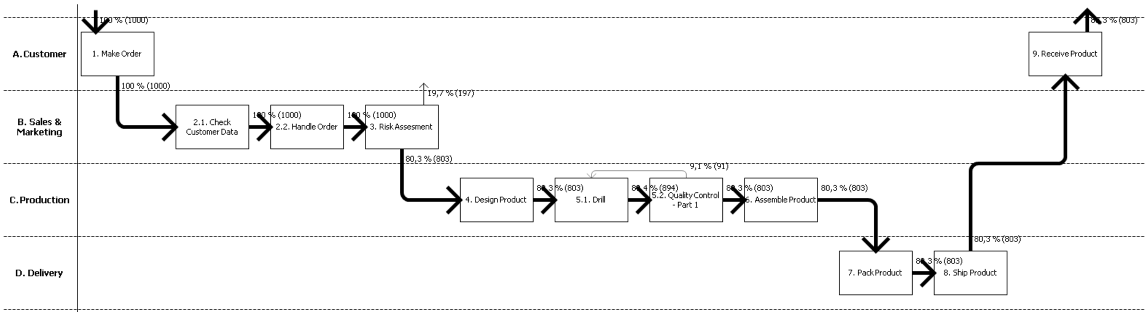


Figure 18: Results from simulation example model

The most obvious difference between the models is the absence of visual properties distinguishing activities and flows from each other. The original model contained information on, for example, the flow type attached to the actual modeling elements, but the ProcessAnalyzer, at least in its current form, does not support this. However, this is merely a matter of illustration that can be solved by adding more logic to the drawing capabilities. What is more important is the order of the activities and the flows between them.

The mined process model contains all the activities present in the original model, and in the correct order. The flows indicate correct transitions between activities and the two decision steps (“3. Risk Assessment” and “5.2. Quality Control – Part 1”) contain two outgoing flows. The thickness of the flows and the numbers associated with them indicate the amount of cases that go through that route. In a simple case like this it is quite natural that all the cases proceed through the first four activities. In “3. Risk Assessment” approximately 20% of the cases exit the process and 80% proceed further. This is in line with the original model where 20% of the orders are rejected in this activity.

In the second decision step (“5.2. Quality Control – Part 1”), about 9% of all the original cases, meaning all cases that started from the first activity, return to “5.1. Drill”. In the original process model this decision was defined such that 10% of the cases that pass through it will go back to “5.1. Drill”. A portion of 10% of the original 80% of cases that passed activity “3. Risk Assessment” would be 8%, and the volume of 9% present in the simulation data is quite close to this figure. This loop that in real life situation probably would indicate rework also explains why flow between “5.1. Drill” and “5.2. Quality Control – Part 1” is higher than the amount of cases arriving to “5.1. Drill”. The loop also explains why slightly higher amount than the anticipated 8% proceed back to drill from quality control – some cases have to be reworked several times.

Special attention should also be given to the inputs and outputs of the whole system. These are indicated by the flows starting from and leading to nowhere. The only input of the process leads to “1. Make Order” which indicates that this was the first executed activity in all of the cases. Outputs are divided to two – about 20% of the cases exit the process after risk assessment and the rest after the customer receives the product.

It is quite promising that the only real issues with the process model generated by the ProcessAnalyzer from the simulated data are related to the visual properties of the model. These are quite easily solved when further developing the tool. Based on these results testing with real healthcare data can be started.

4.4 Case 2: Cardiology data

4.4.1 Introduction

The second test case for ProcessAnalyzer is taken from the cardiology treatment unit at Meilahti hospital in Finland. Cardiology is a field of medical science that studies and treats cardiovascular disorders, meaning disorders of heart and blood vessels. The nature of these disorders is naturally quite complex and most cases have to be individually treated, but there might still be some patient episodes that occur more frequently than the others and identifying these would help to streamline the treatment processes.

The cardiology data handled in this study consists of all visits to the first aid units at Meilahti and Maria hospitals, all visits to the Cardiology polyclinics at Meilahti and Maria hospitals, all visits to certain wards at Meilahti and all visits to Cardiovascular Research Unit (term CRU used from now on) in the years 2004 and 2005. The data is exactly the same as that collected by NHG in 2006 in the so called Ihannesairaala (ideal hospital) project (Eklund et al., 2007).

The purpose of this data selection is to identify longer treatment paths between these different units and to handle real life healthcare data instead of the optimal simulated data presented in chapter 4.3. This enables validating the process mining algorithm in a realistic situation. Because the same data has already been used earlier by NHG in their study, the case is familiar with the consultants interviewed while conducting the study and the validation of the results becomes easier.

4.4.2 Research setting

Even though the data was selected to only consist of treatment activities in a few units in a limited time span of two years, the amount of data is still quite large. What causes even more problems to the clarity of the mined process models is the heterogeneous nature of the data – there are multitude of totally different cases and trying to fit them all to one process model causes some confusion to say the least. This is why the data was filtered even further when validating the algorithm – the scope was selected based on some mutual factors that exist between many patient cases and the scoping could have also been done in a number of other ways.

For this thesis the cases were filtered so that only cases where the patients had at some point diagnosis codes I20–I25, I50 or R07 were selected. In order to reduce the effect of dispersion of the cases into many different process classes the filtering continued by selecting only cases with patient identifier of less than 2000. The diagnosis code based selection of cases was similar to that done in the NHG study; however that study did not limit the cases further. Once the patients having these special diagnosis codes had been found from the data, the specific patients were selected as the main data and the filter from diagnosis code was removed. This removes the problem that in many cases the diagnosis code changes during the process and simply taking the situation when the patient actually has a specific diagnosis code discards some of the earlier steps in the patient episode. These filters applied to the source data dropped the amount of cases from over 72000 to less than 350. The number of cases is still quite high after these limitations and the filtered data can be safely used as source data for process mining algorithm.

It is still likely that the resulting process model containing all the activities performed to the 350 identified cases becomes quite complex. The reason is that it contains every individual treatment event performed to every patient and some of the events or event combinations are quite rare. Special analyses are thus made by limiting the shown treatment events and flows between the events to the largest masses. In practice a Pareto rule is applied – after drawing the process model containing all the activities and flows, another process model is drawn by selecting only top 80% of the activities and flows.

The result should follow the Pareto principle and consist of only a fraction of the original activities and flows.

4.4.3 Mining results

The results presented in this chapter include the generic process model generated by the process mining algorithm when showing all the activities and flows and the simplified process model generated when showing only the main activities and flows. Both of these models are analyzed and the applicability of ProcessAnalyzer is evaluated in the case of the cardiology data.

The pictures below (Figure 19 and Figure 20) contain the beginning of the process model generated from this data. The full model can be found from Appendix 2. The process model is quite complex and contains many flows between activities that occur only in a handful of cases. It is already clear at first glance that model like this is not very useful for thorough analysis, but it can be used to recognize the main flows between activities. Because of the high amount of flows between the activities, the labels indicating the amount of cases passing through each flow is not clear in this overall picture. However, the thickness of the arrow indicates the amount of cases. In this model the cases are divided quite equally between small individual activities and large flows only exist around the first aid polyclinics of Maria and Meilahti hospital, more specifically the units MRSPPKL and PÄ.

The picture here does not tell the whole truth of the generated process model. The ProcessAnalyzer can even at its current state show flows related to an activity when user clicks it. This further information enables reviewing the flows and their volumes in higher detail. For a complex model like the one presented here this is the only way of showing all the information. However, when simplifying the model also the flow labels indicating volumes become important.

An overall picture such as that presented by these figures is, however, good for making rough analyses of the whole situation. For example, when looking at Figure 19 it seems that the first aid polyclinics at both Maria and Meilahti hospitals are in a heavy use – 28% of the patients arrive to the first aid polyclinic at the Maria hospital and 27% to the respective polyclinic at the Meilahti hospital. An interesting remark is, however, that a relatively small amount of patients move from them to any other unit. The reason for this might be that the majority of patients are sent straight back home or that the patients are distributed evenly to the smaller units, thus making their flows low. The latter case can be evaluated by looking more closely the amounts of patients moving to either of the cardiology polyclinics. As the pictures reveal, these amounts are not very high which indicates that majority of the patients actually just visit the first aid polyclinic and go home straight after this.

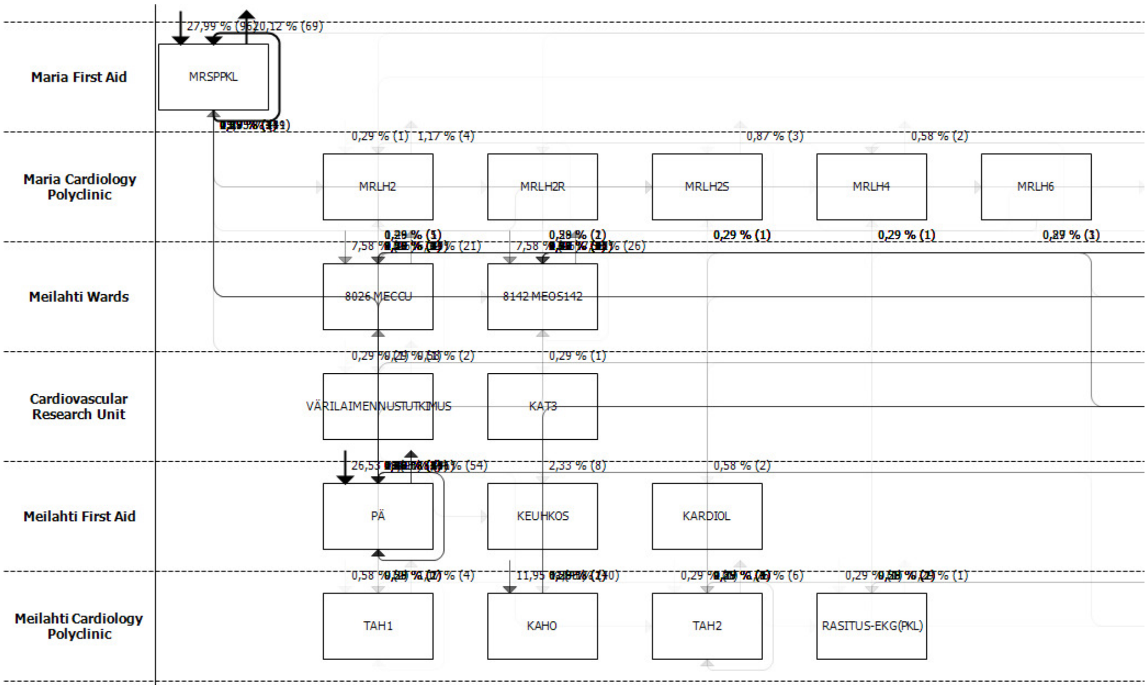


Figure 19: Cardiology data, mined process model 1/2

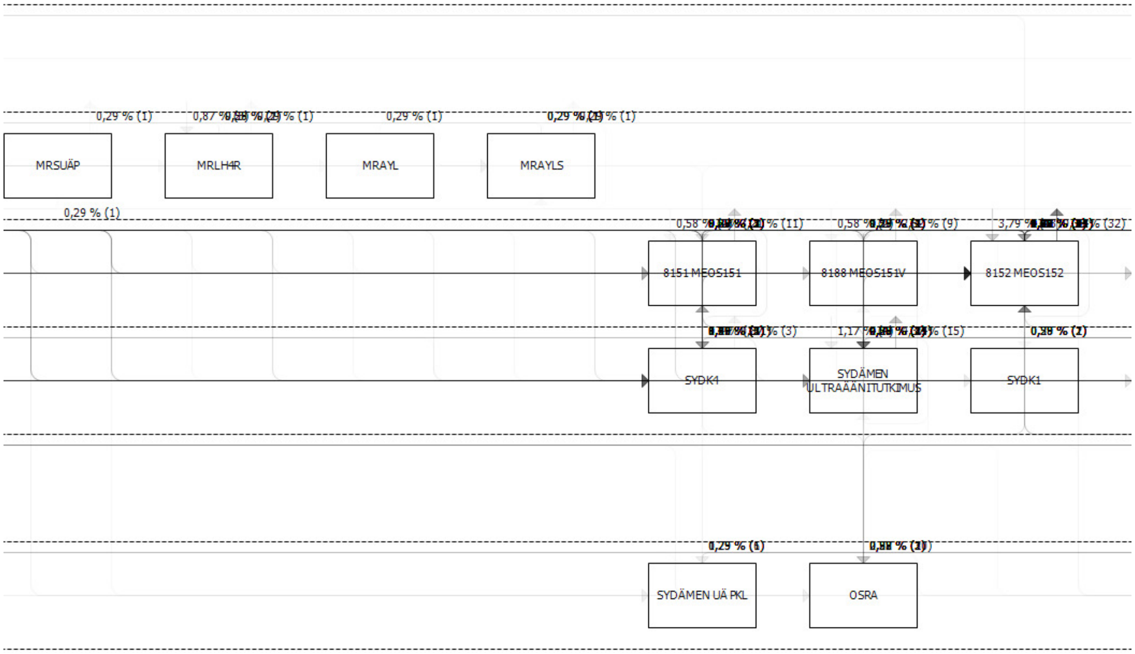


Figure 20: Cardiology data, mined process model 2/2

The process model reveals also that none of the flows between activities are very large. The highest figures in these cases are around 25%, meaning that there is no single activity that all or even a majority of the cases go through. When actually analyzing process like this, the next step would probably be to separate the Meilahti and Maria first aid polyclinics and draw separate process models for both of them. However, as medical study is not the focus of this study, I will not go into details here.

Pareto principle can still be applied here, even though there is no clear 80% majority in the model. High amounts in this case seem to be close to 10% of all the cases, so I will next present process model drawn from the same data but taking only account those flows where over 10% of the cases have been active. Figure 21 presents this model.

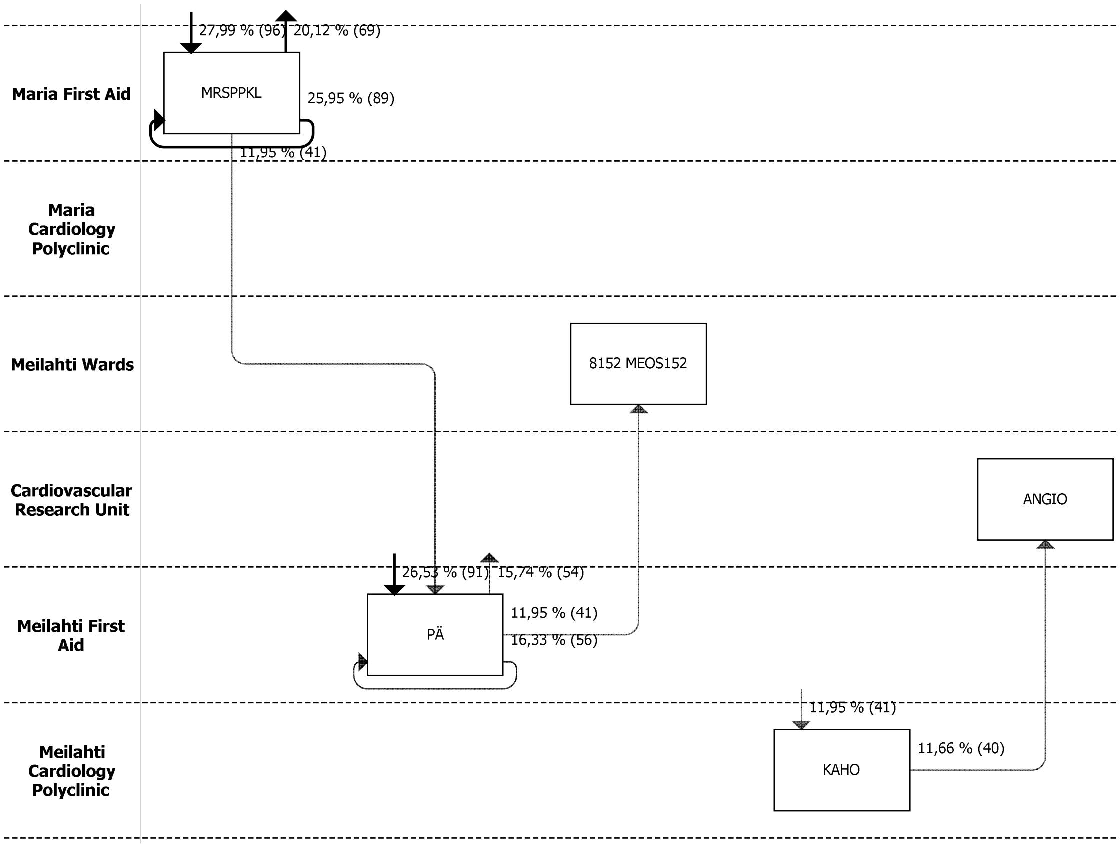


Figure 21: Cardiology data, mined process model with major flows

It is obvious that this process model is a lot clearer than that presented in earlier figures (Figure 19 and Figure 20). The major flows are naturally the same as those presented in the process model containing all cases, but this time the absence of the irrelevant small activities enables the focus of analysis to stay on the right track. Of course, if the point of the analysis is to examine a certain special activity, this will not be revealed in a Pareto analysis, but instead requires a special case constructed for this purpose.

To illustrate the capabilities of ProcessAnalyzer further I will also present process model created from the same data, but this time the activities are simplified so that each organization unit contains only one similarly named activity. This reduces largely the complexity of process model presented in pictures earlier (Figure 19 and Figure 20) and should serve as demonstration of how the visual properties of process model can be largely affected by how activities are selected. The process model is presented in Figure 22.

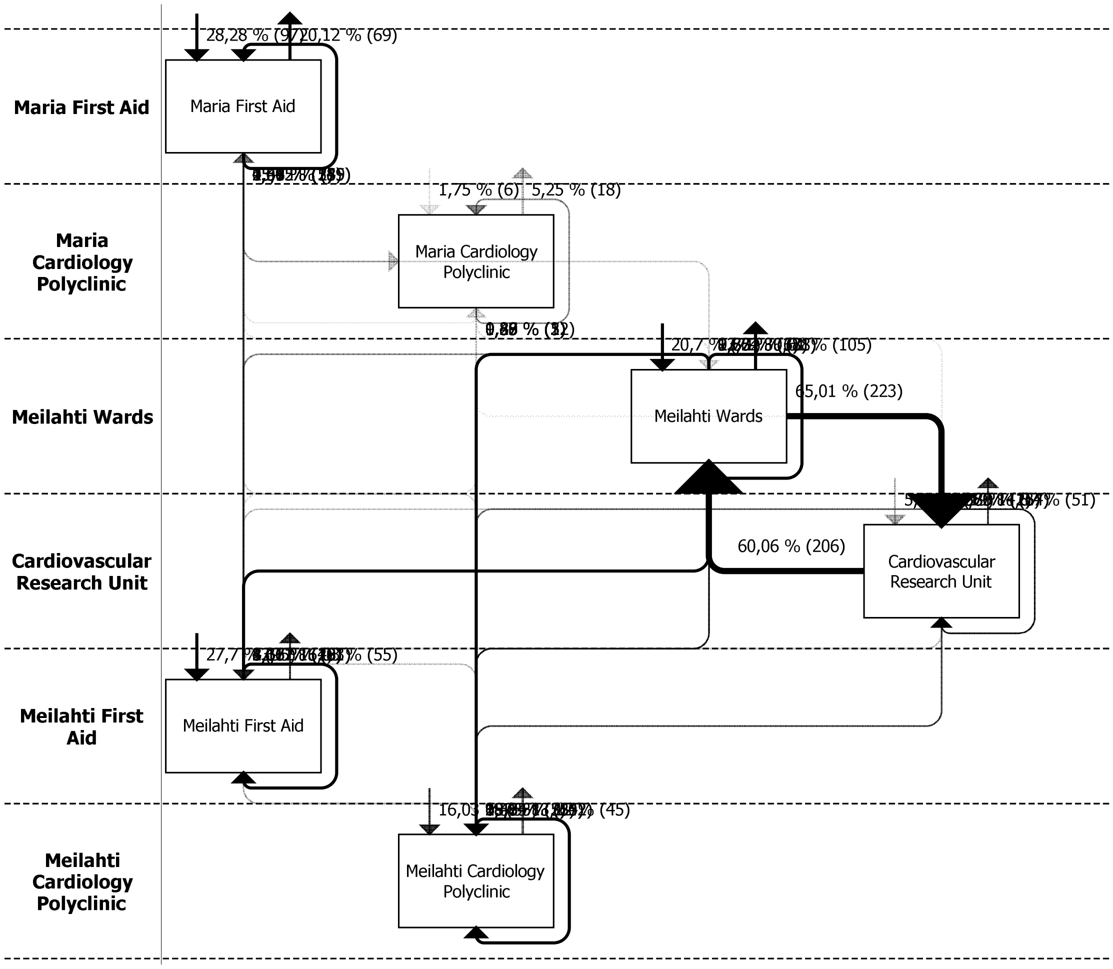


Figure 22: Cardiology data, simplified process model

When comparing this simplified process model with the process model presented earlier in this chapter, an interesting conclusion can be made. The major flows have moved from around the first aid polyclinics to the flows between the wards and CRU. More specifically, the flows around the first aid polyclinics are of course still present as before, but combining the different wards and CRU's units together significantly increased their importance. Actually 65% of the cases found from the data move from ward to CRU and about 60% of the cases move back to wards. This highlights the importance of carefully selecting the data for ProcessAnalyzer such that it can answer to the questions that are asked from it.

The nature of the data collected from the cardiology polyclinics is such that it does not contain very long treatment activity chains. The consequence of this is that large number of cases only consist of one activity – the patient arrives from somewhere outside the logs from which the data was taken and after inspection is sent to another unit. This can happen, for example, if the patient arrives from municipal healthcare. The problem with the data collected for this study is that it is not possible to identify whether the first activity for a case is actually the first activity the patient really encounters, or whether it is just the first activity that can be found from the analyzed log files. This causes some problems for interpreting the results generated by the ProcessAnalyzer.

Another problem was also already discussed briefly when showing the more complicated cardiology processes. The process models can easily be confusingly scattered and consist of high amount of flows between almost all the possible activity pairs. This problem can, however, be somewhat alleviated by showing only the major flows and related activities. Also the source data selection affects the results of the ProcessAnalyzer quite a lot – this can easily be seen when comparing Figure 22 with Figure 19 and Figure 20.

A clear visualization problem of the current prototype of ProcessAnalyzer is its inability to distribute evenly the starting points of different flows starting from an activity; instead it draws them all starting from the same point. In some cases this cannot be solved; in practice these are the cases where so many flows start from a single activity that distributing them is not possible in the limited amount of space. In many simpler cases, however, this distribution could be done and this is something that should be taken into account when developing the product further.

4.5 Case Töölö hospital

4.5.1 Introduction

Another set of real life data from the healthcare sector was acquired from Töölö hospital. This hospital is situated in Helsinki, Finland and is specialized in treating severe accidents. The Accident and Emergency Unit (hereafter AEU) specializes in orthopedics and traumatology, hand surgery, plastic surgery and neurosurgery (the internet pages of HUS).

This data was originally collected by Antti Peltokorpi and it serves as basis for his forthcoming dissertation (Peltokorpi, forthcoming in 2010). The data is somewhat different than that presented in the chapter 4.4 – it consists of data collected at the AEU, several wards and the surgery unit. The data also illustrates ProcessAnalyzer's capability of handling parallel activities as in most cases the patient is, according to the logs, at the ward concurrently with his or her surgery.

The aim for selecting this data was to validate ProcessAnalyzer with another, somewhat different data set. As the resulting process model from the cardiology polyclinic turned out to be quite complicated, it was hoped that this data would provide more clear results. The whole dataset consists of all the visits in the aforementioned units of Töölö hospital in the year 2008. In total this means about 19000 patients.

4.5.2 Research setting

Because the data consists of all the patients that come to the Töölö hospital, regardless of their diagnosis, the data is quite heterogeneous. However, this time no single set of diagnosis codes was selected for further analysis as most of the diagnosis codes would have returned quite a small subset of the whole data and not very interesting results. This time the filtering was made based on the activities the patients go through in their episodes. Only those patients that have been performed a surgery were selected for further analysis.

Once this restriction was made, further filtering of the data was done based on the patient identifier numbers. Only those patients that had patient identifier of less than 1000 were selected to limit the data. As the result, 518 cases out of the original 19258 were actually selected and used as the source data for ProcessAnalyzer. Once these cases had been selected, the filter on activity was naturally removed to include also other activities in the source data. The main reason to restrict the amount of handled data rows comes from the performance limitations of the ProcessAnalyzer prototype.

The data was also selected so that the wards and the AEU were separated to smaller process activities. The obvious way to divide the ward data is to handle each ward as separate activity. The AEU data does not contain very elaborate information based on which cases could be distinguished, so the visit type was selected to distinguish the different reasons why patients comes to the polyclinic. All surgery activities are simplified as generic surgery in this mined process model. Of course, this is not a limitation of the ProcessAnalyzer, but a limitation of the selected data set – if the person doing the analysis wants, she can naturally select the drawn activities as she wishes.

4.5.3 Mining results

The results presented here follow the same logic as those presented in the case of cardiology data. First I will present the generic process model generated from the data. This process model illustrates all the possible flows between different activities. To simplify the process model, I will present another process model from the same case showing only the major patient flows. Finally I will illustrate how the process model changes when selecting the handled activities differently. Each of these cases is also analyzed and finally the ProcessAnalyzer's capability of analyzing this kind of processes is evaluated.

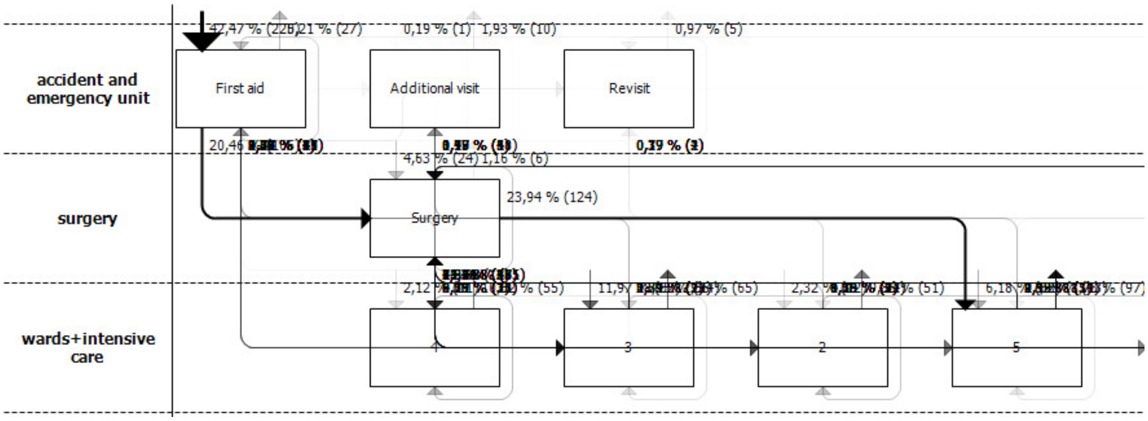


Figure 23: Töölö data, mined process model 1/3

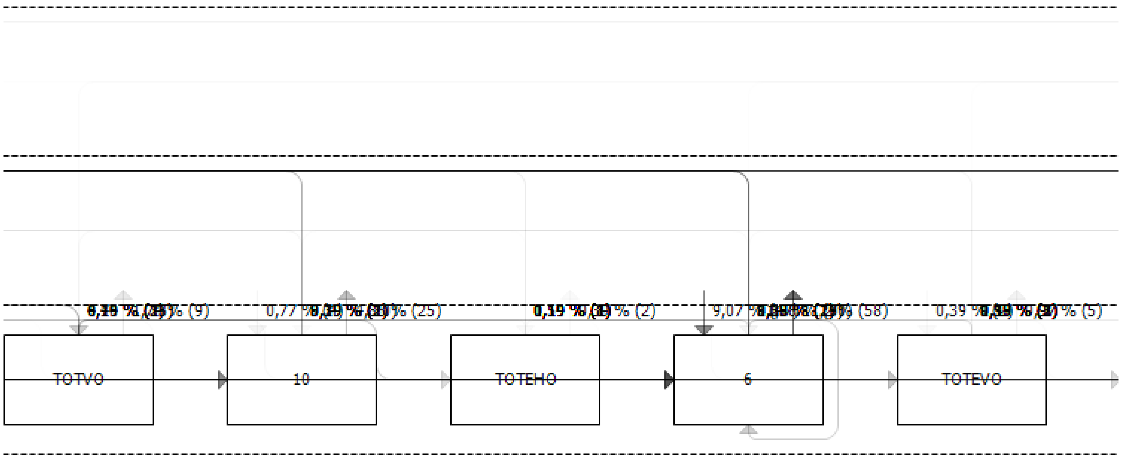


Figure 24: Töölö data, mined process model 2/3

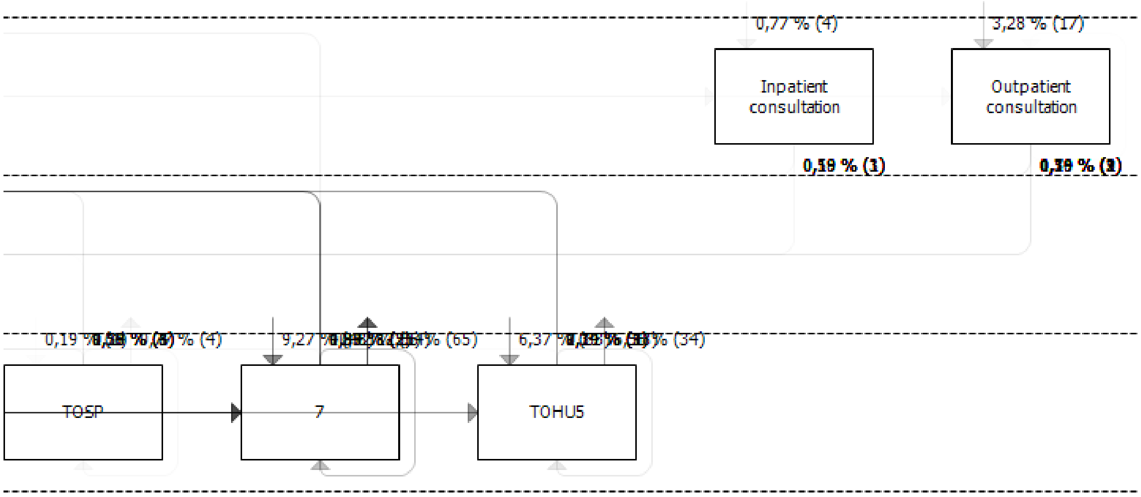


Figure 25: Töölö data, mined process model 3/3

The attached pictures (Figure 23, Figure 24 and Figure 25) show the generic process model mined from the Töölö hospital data. The process model contains activities performed in the three organization units: AEU, surgery unit and the wards.

In addition to these the picture contains the flows between different activities. As can be seen from the thickness of the flows, the largest volume of patients flows between the AEU and surgery (about 20%), as well as surgery and ward number 5 (about 24%). The largest amount of patients enters the system through the AEU, which accounts for about 42% of all the cases. It is interesting to notice, however, that many cases begin by patient arriving to various wards. The explaining behavior for this might be that the patient is arriving from some other hospital or from the municipal healthcare centers.

The next step in testing the ProcessAnalyzer is to limit the amount of shown activities and flows. As was done also in the case of cardiology data, only those flows that contain 10% or more of the total number of cases investigated are shown in this refined process model. The pictures below (Figure 26 and Figure 27) illustrate this slightly simpler case. The major flows of course stay the same as only the smaller ones were removed from the model. The situation does not change as dramatically as it did in the cardiology case as this time the source data was quite simple already in the first place.

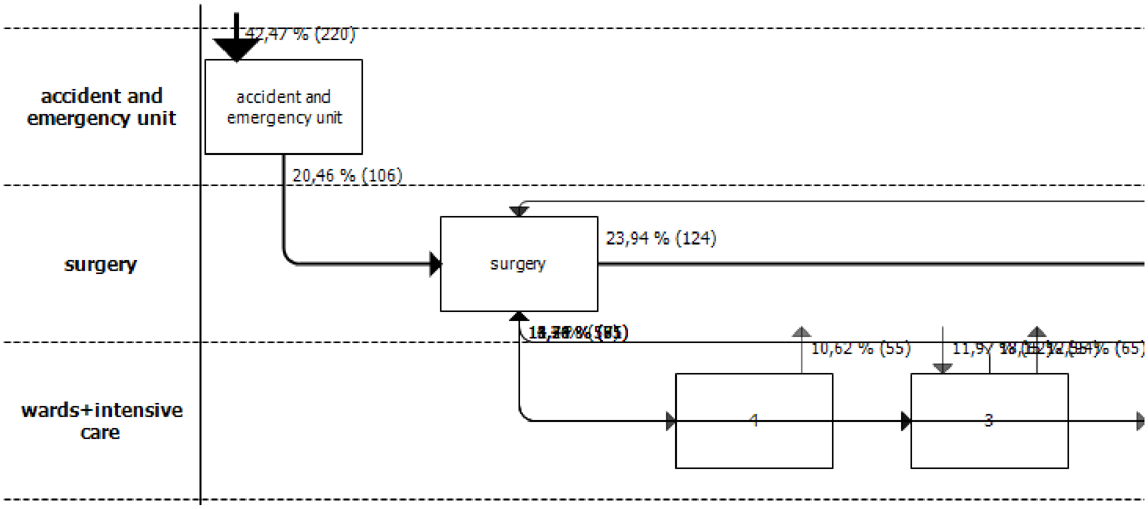


Figure 26: Töölö data, mined process model with major flows 1/2

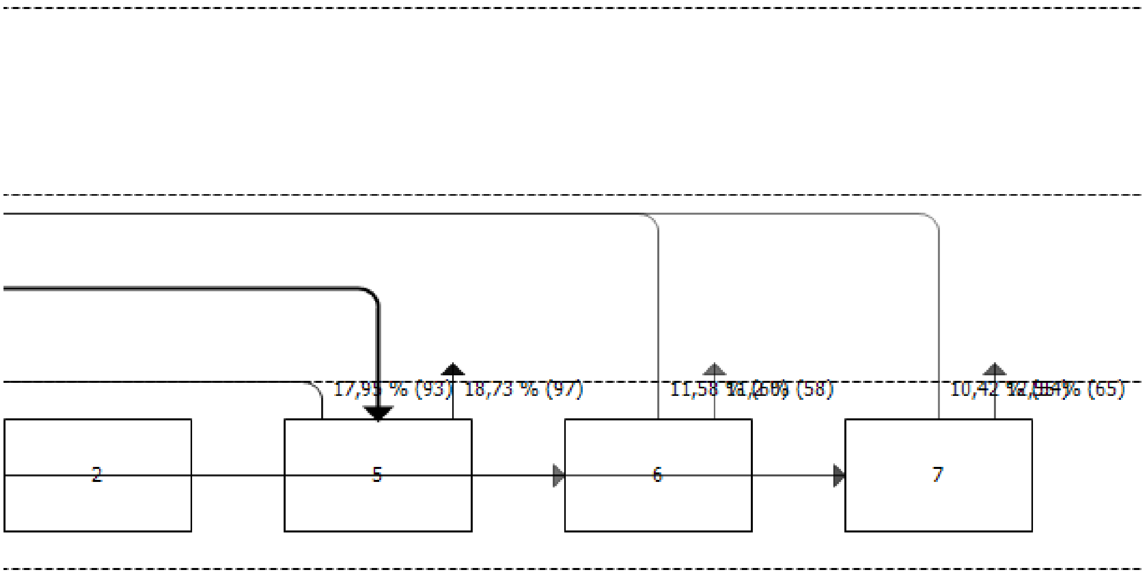


Figure 27: Töölö data, mined process model with major flows 2/2

It should be noted that the mining algorithm was constructed for the healthcare sector purposes such that when some activity happens concurrently with another activity in the workflow log, the activities are still handled separately and additional flows are added. This case was not present in the cardiology data, but in the Töölö data this applies to many of the cases. It seems that the logs show long periods of time of patient staying at a ward, and while the patient is still logged in the ward, he or she is performed surgery.

Consequently, the surgery appears to overlap the ward in the process models. This is solved in the ProcessAnalyzer so that the patient moves away from the ward when the surgery begins and back to the ward when the surgery ends. This is visible in the flows between different wards and the surgery unit.

To illustrate better the large flows of patients, a more simplified process model was created. This time the division to different wards was removed, and the process model was configured such that each of the organization units contains only one generic activity associated to them. This allows forming the big picture of how patients flow between these units. Figure 28 shows this simplified process model.

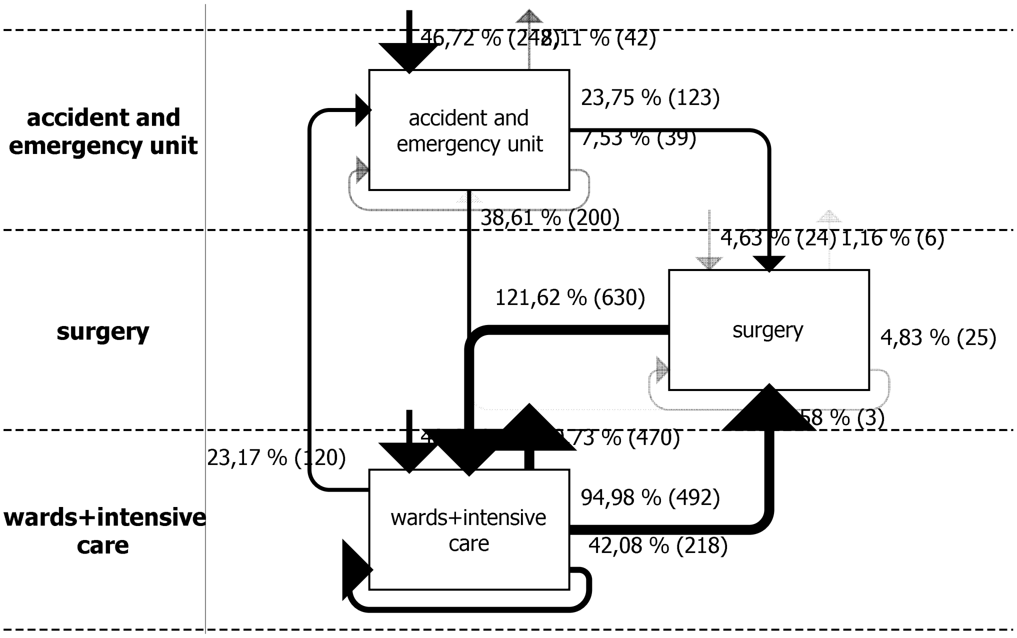


Figure 28: Töölö data, simplified process model

The simplified model reveals some interesting insights on the process. First of all, the patient flows between wards and surgery unit are very high. Actually, the flow from surgery to ward occurs on average 1.2 times per each case. The explanation for this is the way parallel executions of activities are handled, as explained above. Nearly all of the patients operated at a surgery at Töölö hospital reside in some of the wards during the operation; however, ProcessAnalyzer assumes they are moved away from the ward when surgery starts and then moved back to the ward after the surgery.

Another interesting insight on the process is revealed by unifying the wards as well – the ward is in 91% of the cases the place through which the patients exit the system. Also the flows inside the ward activity are common, occurring in about 42% of all the cases. Because of the simplified nature of the process, there is no need to anymore limit the amount of flows shown as they all fit quite nicely to the picture.

In the case of the cardiology case, the visual way how mined process models are drawn was identified as one weakness of the current prototype. This problem is present in the Töölö data as well, especially in the cases where larger amount of activities are visible. Even the last picture presented, Figure 28, required some manual reallocation of flows so that the labels do not overlap. In the future development of ProcessAnalyzer this probably can be alleviated by redefining the logic of drawing the flows. It is possible, however, that some manual work needs to be done after the mining algorithm has finished drawing the process model.

4.6 Classification

4.6.1 Introduction to classification

When analyzing the resulting process models calculated from the real life healthcare data, it becomes clear that the process models can become very complex. This complexity comes from the fact that there probably is not a small amount of distinct processes in the analyzed situation, but instead many different process classes. When all of these are drawn to a single process model, the model naturally contains many flows between activities. Part of the problem can be solved by combining activities into larger activity groups, such as combining different wards to one generic ward for analysis. However, this is not the optimal solution for cases where smaller activities are subject of analysis.

To counter these problems, support for identifying different process classes was added to the ProcessAnalyzer prototype. In essence this means recognizing different trails of flows from the log that is analyzed and classifying each of these different process types as a separate class. This makes it possible to see the amount of different process types in the source data as well as the distribution of process types – some of them are common while others only exist in very few cases. It is also possible to use these classifications as a tool for selecting interesting process types for further analysis. A doctor could, for example, select from these visual process classes just the cases that visit her department in the hospital and draw the generic process model from those cases.

4.6.2 Classification principles

The classification principle of ProcessAnalyzer is based on browsing the log and finding different transition trails. In this context transition trail means a set of transitions that is present in the log. As an example, consider the simple process described in Figure 29. In this case activity A leads with a 50% probability to activity B and with a 50% probability to activity C. In essence this means that the possible transition trails on the log are either A, B, D or A, C, D. When the log generated by this process would be analyzed, these two process classes would be found.

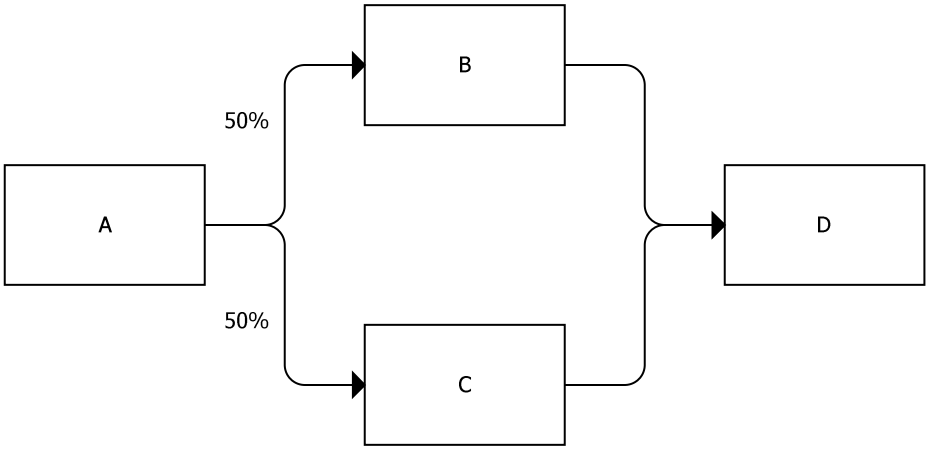


Figure 29: Classification example

Because the classification also records the distribution of different process classes, the number of instances belonging to each class is recorded while browsing through the log. In the previous example, the frequencies of classes A, B, D and A, C, D would be close to each other as there is a 50% chance of both of them to happen in the original model. In a real case the model naturally is not this simple, and the distribution becomes much more complex.

The classification provides basis for a lot of new features for ProcessAnalyzer. One of these ideas was presented by Torkki in our meeting at Helsinki University of Technology (Torkki, 11.9.2009) and it handles the distinction of long and short delays between activities. Because ProcessAnalyzer generally uses case identifiers to distinguish cases from each other, the problem might be that even though the case identifier between the activities stays the same, a long time between the activities actually means the cases are not related to each other and should thus be handled separately. This happens particularly in healthcare sector, as the social security number is used as a natural identifier. If the scoping of the data is not very narrow, it is likely that the cases contain medical treatment history records from several treatments the person has received. Recognized episodes are not actually episodes but instead collections of them.

This problem is tackled by treating the activities with long delays between them as separate cases compared to those with short delays between them. The definition of long and short of course depends on case and is something the user of ProcessAnalyzer can select. Currently the distinction of delay categories is done simply when constructing the different process classes, but when developing the product further this could also mean that the parts of the process divided by the long delay are actually considered as separate processes altogether.

4.6.3 Analyzing process classes

The classification of process types also enables analyzing these different classes separately. When statistics for classes are calculated, the classes can be quite easily compared with each other to recognize differences. Benefits for analysis purposes include, among others, the possibility to identify differences between the execution lead times of different healthcare units as well as the ability to compare proportion of value-adding times between different units.

The basic statistics that are calculated by ProcessAnalyzer in the current prototype are the average lead times of activities and the average relative start times from the start of the process. Both statistics include also the standard deviation for identifying whether the distribution is narrow or wide. This naturally leaves a lot of room for improvement for the actual product development – for example, showing the distributions graphically would make them a lot clearer.

In order to calculate these statistics, the information on the actual cases belonging to each class is needed. The individual cases are shown graphically so that the person using the ProcessAnalyzer can easily see which cases belong to that class. As the log contains the exact start and end timestamps for each activity, the information can be aggregated to the process class level. The details of these operations are omitted here, but the basic principle is storing the timestamps of cases in a data structure which can then be read when calculating the statistics for process class.

4.6.4 Visualizing process classes

Calculating statistics related to the process classes provides good basis for showing the classes also visually. The visual approach selected for the ProcessAnalyzer is actually closer to a Gantt chart (for example Artto et al., 2006) familiar from the project business than to a traditional process model. As a distinction from the general level process model that contains simply equal boxes representing related activities, the classification visualization provides boxes with sizes relative to the actual duration of the activity. Gantt chart provides also timeline which can easily be used to show the scale of the process class visualization. However, not all properties of the general level process model are discarded – the organization units are still present in the class diagram to show the same performers of the activities as are shown on the main level.

To demonstrate the visualization, I will return back to the Töölö case described in chapter 4.5.3. One process model drawn from this data is found from Figure 28. Even though the general process model in this case is quite simple, there are still many different types of processes actually present in the log. In total there are 144 classes of which 31 contain more than one actual case in the log. Table 11 contains ten most common process classes found in this data – the most common by a clear margin is a simple flow from wards to surgery which accounts for about 23% of total cases. The vertical line between activities in the last example indicates a long delay between the activities. The table also presents the same process class without the vertical lines – this shows that these two cases are handled as separate classes.

Table 11: Most common process classes in Töölö data

Frequency (% of total)	**Class**
119 (22.97%)	wards+intensive-care – surgery
57 (11%)	wards+intensive-care – surgery – wards+intensive-care
54 (10.42%)	accident and emergency unit – wards+intensive-care – surgery
28 (5.41%)	accident and emergency unit – surgery – wards+intensive-care
27 (5.21%)	accident and emergency unit – surgery – wards+intensive-care – wards+intensive-care
14 (2.7%)	wards+intensive-care – surgery – wards+intensive-care – wards+intensive-care
11 (2.12%)	wards+intensive-care – surgery – accident and emergency unit
11 (2.12%)	surgery – wards+intensive-care
9 (1.74%)	accident and emergency unit – wards+intensive-care – surgery – wards+intensive-care
9 (1.74%)	accident and emergency unit - | - wards+intensive-care - | - surgery

Each of the process classes found from the data is drawn using the aforementioned Gantt presentation method. As an example of this visualization, Figure 30 below contains visualization of the class “accident and emergency unit - wards+intensive care - surgery”. Some remarks can already be made by simply looking at this picture. First of all, the whole process of patient arriving to accident and emergency unit and finally leaving the ward takes a bit longer than five days. However, the patient only spends few hours of the first day at the accident and emergency unit and then arrives to the ward after three days. During his stay at the ward he or she goes through a surgery which takes a few hours.

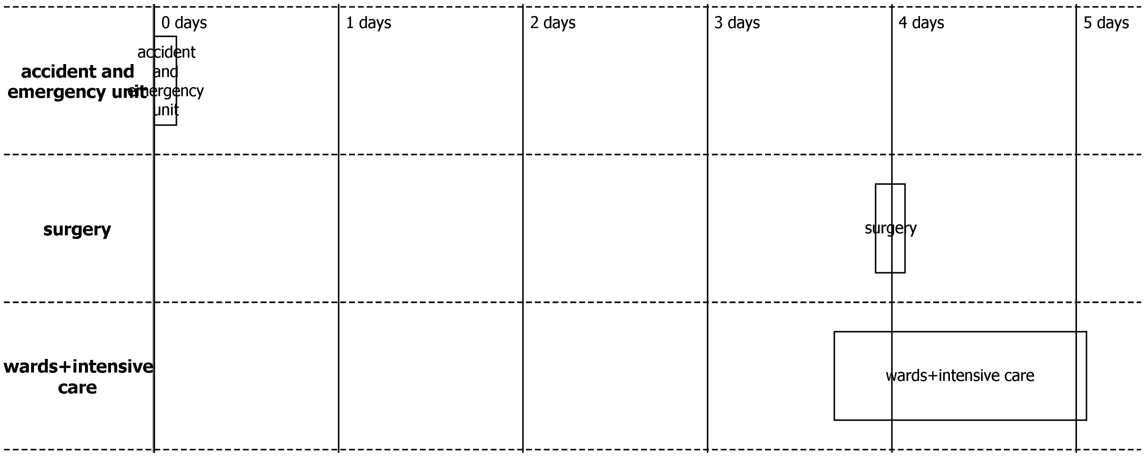


Figure 30: Process class "accident and emergency unit - wards+intensive care - surgery"

The process class representation is drawn using the average values of cases belonging to that class. Averages naturally do not tell the whole truth – for example in the case of the previous picture, the average lead time of “wards+intensive care” is 1.37 days with standard deviation of 1.32 days. Large standard deviation such as the one in this case means that the durations in the actual cases spread quite widely: there are some cases where the actual time spent at ward is short but there are also cases where it is significantly larger than 1.37 days.

Each of the cases belonging to a specific class are drawn to their own Gantt-like charts, and the person analyzing this process can then browse through these cases to see how actual cases behave in the process.

4.7 Performance of the mining algorithm

Even though this thesis is not focused on the technical details of the mining algorithm, a few words about the performance needs to be said. The reader should be aware of the basic concepts of computational complexity – average case of running time and memory usage. This chapter handles only running time as in this context it is most likely the critical factor. Memory usage of the mining algorithm is simply so small compared to the time usage.

A commonly used notation for running time complexity is so called “Big O notation” which means the upper bound of the growth rate of the function. For example, an algorithm that simply browses through all the data rows is of complexity O(n) where n is the number of rows in the data. On the other hand, if for each element in the data the data is browsed through a second time – for example to find relations – the complexity is O(n^2). Complexity notation only tells the most significant exponent as the smaller exponents are not relevant. In addition to exponents, logarithms are also often found in complexity notations. For more information on Big O notation see for example (Kingston, 1998, chapter 2).

The complexity of mining algorithm used in the ProcessAnalyzer can be analyzed from two perspectives – the time usage of dependency relation calculation and the time usage of drawing the process model. When calculating the dependency relations, the mining algorithm presented in the chapter 4.1 is applied to the log and the relations presented in chapter 2.3.2 are calculated. This practically calculates all the information that is necessary for creating the generic process model. The second part of model creation – drawing the process model – is carried out after the dependency relations are calculated. In this step ProcessAnalyzer uses the calculated dependency relations to draw the necessary items to the process model.

The dependency relations are usually calculated less often than the model is drawn as configurations of what exactly to draw to the model can be made after calculating the full dependency relation matrix. If n means the number of rows in the source data and m means the number of distinct activities in the source data, the complexity of deriving dependency relations is O(n*m). This is a relatively good result – in practice it means that the data rows are looped once, and inside that loop the distinct activities are browsed through. Result like this means that the execution speed of calculating the dependency relations does not dramatically increase when the size of the source data increases. However, the complexity is not linear which means that the time required for the operations goes higher at an increasing speed when the log file becomes larger.

A more time-critical element of ProcessAnalyzer is drawing the actual process models. Unfortunately this element also takes significantly more time. The drawing can be separated to two cases – either only the top level general model or the full, detailed model containing process classifications is drawn. When only the top level model is drawn, the performance is quite good – $O(m^2)$ using the same symbols as before. As m means the number of distinct activities, even exponential growth is not very severe as the number of activities in most cases is relatively low even with high amount of data. It should be noted that this only evaluates the time complexity of the VBA code – the complexity of code used to draw elements inside QPR ProcessGuide is not evaluated here. The situation is, with this remark, a bit worse than the complexity notation would indicate as most of the time seems to be spent when drawing the elements using QPR ProcessGuide API.

When drawing the detailed model containing process classifications, the complexity starts to be more of a problem. Even though the complexity of the algorithm itself is still moderate, O(n*m) as with calculating the dependency relations, the execution seems to be taking quite long. This time the O(n*m) notation also describes the amount of elements created using QPR ProcessGuide API. Element creation seems to be in practice quite slow using this interface; however, I will not dig deeper into the QPR ProcessGuide source code here to investigate the exact reason. With practical tests it seems that drawing elements takes several minutes when data size is a few thousand rows and increases quite linearly – indicating that handling a few hundred thousand rows would take a few hundred minutes. Tests handling such large data sets have not been carried in this study. However, if the general time it takes to draw elements with QPR ProcessGuide API could be reduced, handling large data volumes should not be a very big problem for ProcessAnalyzer.

5 Discussion and evaluation of results

This chapter sums up the findings from the previous chapters and evaluates the suitability of process mining regarding Finnish healthcare sector. In addition to these, four challenges of ProcessAnalyzer are described and their effects on the further product development are discussed. Finally some ideas for further research are presented.

5.1 The suitability of process mining

The most important requirement for the source data analyzed by ProcessAnalyzer is that the data must be in a log format, meaning it must contain the history of performed events. There are actually quite many systems that record only the current situation and delete the history log altogether. Data from systems like these simply cannot be used as the source data for mining algorithms, as the algorithms cannot derive the relations of activities from a snapshot of current situation. However, the actual format of the log is not relevant as the tools used for data manipulation contain quite a large amount of different techniques for extracting the relevant data from the log.

When the data on a general level is suitable for mining algorithms, the actual process mining algorithm can be evaluated based on the three different cases analyzed in the previous chapters. Chapter 4.3 presents results from the simulated data. Major characteristic of this data is that it contains whole chains of process instances, each of which have the same start and end activities. The underlying assumption on this data is also that it is generated by a perfect process where each instance fits the process and where there are not a vast amount of different execution paths.

When analyzing simulated data, the ProcessAnalyzer provides results identical to the source model as a result. The model contains all the activities in the right places and the flows between the activities show correct volumes. The visual placement of flows is not optimal, but can be solved by developing the product further. This shows that the mining algorithm works, assuming the data is of good quality.

When switching to real life data, the validity and uniformity of the data becomes relevant. As was identified in chapter 4.4, the ProcessAnalyzer calculates quite a complex process model from the cardiology data. The problem is partly caused by how the data was gathered – it is taken from many different patient record systems but it still only contains those parts of the episodes where the patient is in the particular units. This means the episodes might not be complete in some of the cases. This is a generic problem with healthcare data. The data usually contains patients handled in one unit and deriving whole episodes would require extracting data from each of the units the patient has visited. However, this still does not render the ProcessAnalyzer useless. It can be used to identify the processes between specific units.

When the data is limited so that only major flows are present, the process models become much clearer. This is quite natural as there are fewer activities and thus fewer intersecting flows in the process model. It would seem that, based on these findings, drawing the process model should be done on a sufficiently low detail level if the data is very heterogeneous. Once more uniform data is identified process modeling can be done in more detail.

The other real life hospital data set was taken from Töölö hospital. Analyzing this data provided more promising results, and the reason was simple: in the Töölö case the data is actually largely created by a nearly uniform process. Most patients visit the accident and emergency unit before arriving to the ward. During the ward visit the patients go through a surgery and after some time spent at the ward they can go home. Drawing a process model from a simple process like this is naturally easier than drawing it from a more heterogeneous environment at the cardiology polyclinic. A slightly less important factor describing why the analysis worked better with Töölö data was that the data contained fewer units and thus the drawn process models became simpler.

The cases illustrated quite effectively the importance of selecting the dimensions presented in the process model. The most logical way to select the dimensions is to show persons or organizational units as horizontal bars, sometimes called swim lanes (for example Burlton, 2001, p. 317), and the performed activities inside one or more of the bars. The detail level of selecting the performers and activities is important and depends on the type of analysis – if the big picture of the process is drawn, there should not be very many performers and activities as the process model easily becomes confusing. On the other hand, if a more in-depth analysis is done, the activities and performers should be selected so that they answer to the question presented.

This shows an important notion of how ProcessAnalyzer works: it answers the questions presented to it but it cannot invent the questions by itself. In practice this means that using the tool is an iterative process as such – especially if the person using the analyzer is not very familiar with the data. First step of using the ProcessAnalyzer is calculating the process model at some quite arbitrary level that somehow relates to the analysis. Once generic models are drawn this way, the analysis requires input from experts familiar with the analyzed process – these experts can point the analysis to a right way and enable scoping the data. When performing process analyses with this scoped data, the results usually start to look better; the questions are being formulated. When this iteration is done a few rounds, the mined process models start answering these questions, assuming a process model exists.

An important characteristic enabling the iterative nature of process analyses is the fast and easy way of using ProcessAnalyzer. The traditional methods of process analysis have involved calculating dependencies in Microsoft Excel or some other tool, spending days preparing the analysis. Iteration cannot be done in such an environment, especially in time critical situations. On the other hand, as ProcessAnalyzer calculates the process significantly faster, the iteration becomes a much more natural way of analyzing processes.

Classifying the generic model into separate process classes was implemented to the ProcessAnalyzer prototype. The main reasons behind classification are to help the iterative analysis by pointing out the relevant issues and to enable richer analysis by providing some calculated statistics of the process classes. As the volumes of individual cases belonging to classes can be identified, the most relevant process classes are easily found. The analysis can then, for example, be limited to include only the cases belonging to the major process classes. A simple Pareto principle seems to be present also in most of the data identified in this study – most cases fall into a few large process classes, while the rest of the process classes only contain one or two cases.

The statistics calculated for the process classes provide useful information on the lead times and waiting times of individual activities. This can already as such be used as basis for analysis. It also provides many potential ways of how ProcessAnalyzer can be developed further from the prototype stage. Showing the statistics, such as distribution of lead time of an activity, graphically enables the person doing the analysis to see easily the relevant information on the process class.

The ProcessAnalyzer prototype has been presented to various parties during the course of this study. The reception has been encouraging – the healthcare sector seems to be eager to find out the ways ProcessAnalyzer can be used in process development. The prototype has been so far presented to the consultants at NHG and to the process development group of HUS. The points of view of these two groups are naturally somewhat different – NHG seeks a tool aiding directly their consultancy projects while HUS identifies the possibilities of ProcessAnalyzer in a more indirect way. HUS would benefit from buying analyses from external consultants that use the tool to carry out some of their actual analyses.

Evaluating the commercial potential of ProcessAnalyzer is not the subject of my thesis. However, it is clear that the tool is mainly designed for companies performing process analyses, not directly for the companies these analyses are performed at. The tool is intended for experts, which probably, should the tool be developed further, will in the first phases be experts inside QPR. If the tool proves to be useful, it can be developed further into a separate product that can be sold to external experts for use in their analysis projects.

5.2 Challenges

The current prototype presents also some challenges. These are related to visualization, drawing parallel routes, classification and performance. Some of them are simply caused by the restrictions of the ways how prototype is programmed while others require more thinking to be solved.

Visualization presents challenges especially with complex process models that contain many flows. The problems come from the fact how flows are drawn by QPR ProcessGuide API. The flows are all drawn so that they start from the same spot, which does not generally cause troubles with simple models, but when the activity can in the case of complex data lead to a high number of activities, the result is a fuzzy collection of intersecting flows. The labels of flows are formulated so that they contain the volumes going through the flow, but as QPR ProcessGuide draws these labels on top of each other when the flows start from the same spot, the result does not look very professional.

These problems can most likely be alleviated by moving the ProcessAnalyzer functionality from the current Microsoft Excel and VBA based implementation to the actual source code of QPR ProcessGuide. This way drawing the flows can be much more freely controlled; for example, using the whole edge of an activity for drawing multiple flows starting from it should make the flows less overlapping. The position of the label boxes could also be calculated more freely so that they do not overlap each other.

Another challenge comes from the parallel routes. Even though they are not present in the healthcare sector where the study was conducted at, they do exist in manufacturing industry. For example, when building a machine, the parts are assembled in parallel and then built together. If data from this kind of process was used as input for ProcessAnalyzer, the result would not look correct. The assumption regarding healthcare sector was that the patient cannot be in multiple places at a single time, which means that if the person according to the data, as in the Töölö case, is at a ward and surgery at the same time, it is assumed that there are flows between these units. This naturally leads to wrong interpretation in the manufacturing industry case.

This challenge is the result of a simplification made at the prototyping stage. The product should be developed further so that it enables also extracting processes containing parallel execution of activities. In practice this most likely means that there are various configurable ways how the product works; one of these configurations enables selecting whether parallel routes are handled with true parallelism or whether they are simplified like in the healthcare case. Implementing this requires some more work to be done and is not relevant in the scope of this thesis.

The classification related challenges are related to the quite simple way the classification is made. Classification simply takes into account the starting times of activities and generates execution order based on these times. This means that quite different process instances can be categorized to same class. As an example, the figures below (Figure 31 and Figure 32) contain two process instances that both belong to same class: “surgery - wards+intensive care”. In the first case (Figure 31) the patient goes to surgery and after about a day proceeds to the ward. In the second case (Figure 32) the surgery starts and almost instantly the patient is also recorded to the ward.

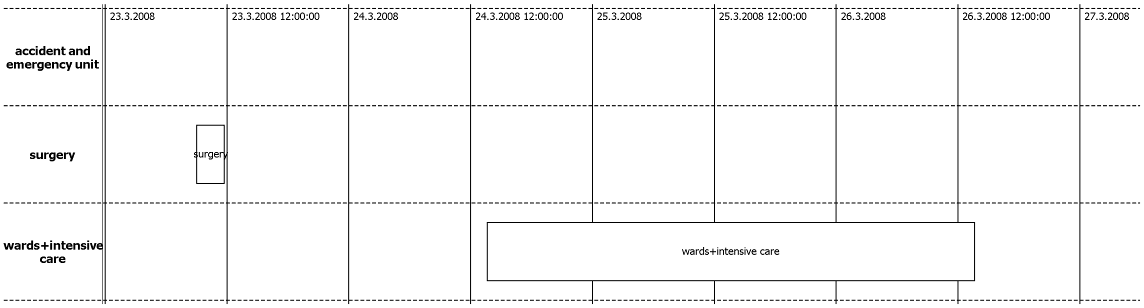


Figure 31: Example of classification challenges 1

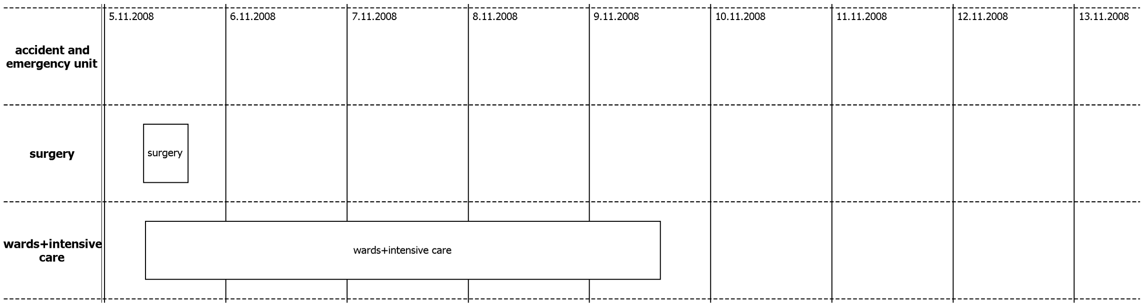


Figure 32: Example of classification challenges 2

The challenges with classification are somewhat related to the challenges with parallel activities. In this case the classification that puts these two activities to same class is right because the ProcessAnalyzer assumes that there is an implicit flow after the activity execution ends in the case another activity is still going on. However, if the process would be such that it really contains true parallelism, these two cases should belong to different classes: one where the activities overlap and one where they do not.

Lastly, performance causes some troubles with the current prototype of ProcessAnalyzer. The execution takes quite a long time and the time increases exponentially as the amount of data increases. In practice this means that if the amount of data is below few thousand rows, the results can be calculated in some minutes. If the amount of data becomes close to 50000, the execution takes several hours. Larger data sets were not handled in this study at all as the smaller volumes of data provided results as well.

This is a real challenge that needs to be solved in order for ProcessAnalyzer to be able to analyze complex real life cases consisting of hundreds of thousands or millions of rows. Most likely moving the execution from VBA to a better performing environment such as Delphi or .NET will radically improve the execution time. Further improvement can be made by optimizing the code by removing unnecessary loops and utilizing already calculated information in a more optimal way. Before that is done, the problem needs to be avoided by selecting data so that it does not contain too high amount of cases.

5.3 Ideas for further research

When developing the prototype of ProcessAnalyzer it became apparent that classification is quite a relevant concept in this context. However, as this thesis focuses on process mining, this subject is not handled very thoroughly. I think that investigating more sophisticated classification methods and developing the classification further would form an interesting topic for further studies. The current classification is based on very simple identification of exactly similar process instances, where similar means same execution order between instances. Classification would become much more elegant if some more sophisticated technology was taken into use.

There is multitude of techniques that could be applied to the classification. Most notably different intelligent systems could probably be used to recognize patterns and similar processes more effectively. This brings the discussion to the field of artificial intelligence. For example, learning agents (Russell & Norvig, 2003, p. 37) could be an interesting field of study as they provide a way for the algorithm to adapt to the data they handle. In practice this could mean that in the case of noisy data where simple classification techniques generate very large amount of different process classes, these techniques could adapt to the situation and start forming the classes less strictly.

Another topic mentioned by my instructor while creating the prototype is neural networks (Russell & Norvig, 2003, chapter 20.5). In computer science the term artificial neural network attempts to simulate biological neural networks which form the basis on how electrical information passes inside living organisms. They are, as well as the learning algorithms mentioned above, adaptive so that the information they handle can change the structure of the network. Studying their applicability in the process classification context could provide interesting new methods.

The actual focus of this study was on process mining, and there are still issues with the current process mining techniques used in the prototype. Separate study could be made to develop the mining algorithm further. Especially the noise removing capabilities of the heuristic mining algorithm presented in chapter 2.3.1 could be useful if applied to the algorithm used by ProcessAnalyzer. In the current prototype the only way to remove noise from the data is to ignore flows and related activities with low enough volume. This, however, can cause problems as sometimes the removed information might be exactly the interesting outlier that needs to be investigated by the analysis.

Noise reduction capabilities of heuristic mining algorithm, on the other hand, are based on the frequencies of relations and their inverse relations – if there are significantly more of either of the relation types, the other can quite safely be discarded as noise in the data. Investigating this topic could provide a new, alternative mining technology suitable for some cases. Of course, applying such an algorithm should be done only after careful thinking as in this case the algorithm already makes assumptions from the data – if the data actually contains valid inverse relations between some activities, these would be erroneously discarded.

Finally, the field of statistics provides lots of possibilities in developing the ProcessAnalyzer further. The prototype only handles very basic statistics such as average and standard deviation. Applying techniques of statistical hypothesis testing (see, for example, Milton & Arnold, 2003) could provide answers to questions such as whether the lead time has decreased compared to the situation six months ago or whether the uniformity of the process has gone higher. The experts interviewed during the course of this study have often expressed that ProcessAnalyzer could provide a way to measure, for example, the effects of change or the difference between lead times of different units. A simple figure describing the magnitude of the difference can naturally be calculated quite easily, but this figure gets a true meaning when it statistically proves the difference between samples.

6 Conclusions and suggestions

The process mining seems to be an interesting concept and the results from creating the prototype are very promising. Even though the generated models are not always very clear, they have in every case managed to start discussion, even among people that do not know much about the source data. This is a fundamentally different situation compared to, for example, looking at the source data in a spreadsheet program. No one can draw similar conclusions from simply looking at the data – the visual information is necessary for passing understanding quickly.

The ProcessAnalyzer prototype has aroused interest also among potential customers. At this point the discussion inside QPR is based on whether to productize the idea so far that it could be used by end customers or whether to offer it as a service provided by QPR or trained partners. We have negotiated with both types of potential customers – consulting companies that could use ProcessAnalyzer as tool to aid their standard analyses as well as the actual end customers whose processes would be analyzed. The reception has been positive.

Many of the potential customers to whom the prototype has been demonstrated have expressed the same delight: ProcessAnalyzer provides something new for them. In some cases it is a tool for seeing and understanding how the process really works in the company; in other cases it is a tool for quickly collecting data from various sources and drawing process models to enable the analysis to go to right direction. There are multiple possibilities where the tool could be used, and the most important part from business benefit point of view is that there are no known direct competitors in this market.

Based on these clear benefits of ProcessAnalyzer, I strongly recommend QPR to focus some of its resources in developing the tool further. The first step would probably be offering analyses as services provided either by QPR itself or through its partner network. In the later phases the product could also be developed further to offer an easy-to-use solution for process analyzing also for the end customers.

7 References

Van der Aalst W. (1998): *The Application of Petri Nets to Workflow Management*. The Journal of Circuits, Systems and Computers. Vol. 8, No. 1, pp. 21–66. Available online at http://wwwis.win.tue.nl/~wvdaalst/publications/p53.pdf.

Van der Aalst W., van Dongen B., Herbst J., Maruster L., Schimm G., Weijters A. (2003): *Workflow Mining: A Survey of Issues and Approaches*. Data and Knowledge Engineering. Vol. 47, No. 2, pp. 237–267. Available online at http://is.tm.tue.nl/staff/wvdaalst/publications/p206.pdf

Van der Aalst W., Weijters A., Maruster L. (2004): *Workflow Mining: Discovering process models from event logs*. IEEE Transactions on Knowledge and Data Engineering. Vol. 16, No. 9, pp. 1128–1142. Available online at http://wwwis.win.tue.nl/~wvdaalst/publications/p245.pdf

Van der Aalst W., Reijers H, Weijters A., van Dongen B., Alves de Medeiros A., Song M., Verbeek H. (2007): *Business process mining: An industrial application.* Information Systems. Vol 32, pp. 713–732.

Artto K., Martinsuo M., Kujala J. (2006): *Projektiliiketoiminta*. 1st Edition. WSOY Oppimateriaalit. ISBN 951-0-31482-x. In Finnish.

Brailer D., Hackett T. (1997): *That pioneer spirit: Implementing clinical episodes of care in an imperfect world.* Health Systems Review. Vol. 30, Issue 5, pp. 26–29.

Burlton R. (2001). *Business Process Management – Profiting from Process*. 1st edition. Sams Publishing. ISBN 0-672-32063-0.

Cook J., Wolf A. (1998). *Discovering models of software processes from event-based data*. ACT Transactions on Software Engineering and Methodology. Vol. 7, No. 3, pp. 215–249. Available online at http://www.cs.colorado.edu/department/publications/reports/docs/CU-CS-819-96.pdf.

Dijkman R., Dumas M., Ouyang C. (2008): *Semantics and analysis of business process models in BPMN*. Information and Software Technology. Vol 50, pp. 1281–1294.

Eklund F., Kämäräinen V., Tolkki O., Torkki P. (2007): *Geneerinen prosessimalli ja integroitu kustannuslaskenta*. HUS, Ihannesairaala project. In Finnish. Available online at http://www.hus.fi/default.asp?path=1,28,820,13120,17956,25572,25573,25574,25640

The internet pages of BPMN. http://www.bpmn.org

The internet pages of HUS. http://www.hus.fi

The internet pages of QPR Software Plc. http://www.qpr.com

Juran J. (editor-in-chief), Gryna F. (associate editor) (1988): *Juran's quality control handbook*. 4th Edition. McGraw-Hill. ISBN 0-07-033176-6.

Kingston J. (1998): *Algorithms and Data Structures – Design, Correctness, Analysis.* 2nd Edition. Addison Wesley Longman Ltd. ISBN 0-201-40374-9.

Kujala J., Lillrank P., Kronström V., Peltokorpi A. (2006): *Time-based management of patient processes.* Journal of Health Organization and Management. Vol. 20, No. 6, pp. 512–524.

Laamanen, K. (2007): *Johda liiketoimintaa prosessien verkkona – ideasta käytäntöön*. 7th edition. Laatukeskus Excellence Finland. ISBN 978-952-5136-16-6. In Finnish.

Lillrank P. (2002): *The Broom and Nonroutine Processes: A Metaphor for Understanding Variability in Organizations*. Knowledge and Process Management. Vol. 9, No. 3, pp. 143–148.

Lillrank P. (2003): *The Quality of Standard, Routine and Nonroutine Processes*. Organization Studies. Vol 24, No. 2, pp. 215–233.

Lillrank P., Liukko M. (2004): *Standard, routine and non-routine processes in health care*. International Journal of Health Care Quality Assurance. Vol. 17, No. 1, pp. 39–46.

Lillrank P., Groop J., Venesmaa J. (2009). *Processes, Episodes and Events in Health Service Supply Chains*. Supply Chain Management: an International Journal. Forthcoming.

Lillrank P. (2010): *Service Processes*. Chapter 16 in Salvendy G. (editor), Karwowski W. (2010): *Introduction to Service Engineering*. Wiley. ISBN 978-0-470-38241-7.

Lind M. (2006): *Determination of Business Process Types Founded in Transformation and Coordination.* Systems, Signs & Actions: An International Journal on Communication, Information Technology and Work, Vol. 2, No. 1, pp. 60-81. Available online at http://www.sysiac.org/uploads/2-4-Lind.pdf.

Madison, D. (2005). *Process Mapping, Process Improvement, and Process Management*. Paton Press LLC. ISBN 978-I-932828-04-7.

Mans R., Schonenberg M., Song M., van der Aalst W., Bakker P. (2009): *Application of Process Mining in Healthcare: A Case Study in a Dutch Hospital*. Biomedical Engineering Systems and Technologies, Vol. 25 of Communications in Computer and Information Science, pp. 425–438. Available online at http://wwwis.win.tue.nl/~wvdaalst/publications/p499.pdf

Medina-Mora R., Winograd T., Flores R., Flores F. (1992): *The Action Workflow Approach to Workflow Management Technology*. In Mantel M., Baecker R. (editors): *Proceedings of the 1992 ACM conference on Computer-supported cooperative work.* Pp. 281–288. ACM Press.

Milton J., Arnold J. (2003): *Introduction to probability and statistics: principles and applications for engineering and the computing sciences.* 4th Edition. McGraw-Hill. ISBN 0-07-246836-X.

O’Connell J., Pyke J., Whitehead R. (2006): *Mastering Your Organization’s Processes – A Plain Guide to Business Process Management.* 1st edition. Cambridge University Press. ISBN 978-0-521-83975-4.

Pareto V. (1892): *State Expenditures in Italy as Compared with the National Wealth.* The Economic Journal. Vol. 2, No. 7, pp. 561–564.

Pareto V. (1897): *The New Theories of Economics*. The Journal of Political Economy. Vol. 5, No. 4, pp. 485–502.

Peltokorpi, A: Improving Efficiency in Surgical Services: Production Planning and Control Approach. Forthcoming dissertation, estimated to be ready in 2010.

Pfeffer P., Sutton R. (2006): *Evidence-Based Management*. Harvard Business Review. Vol. 84, Issue 1, pp. 62–74.

Reisig W., Rozenberg G. (1998): *Lectures on Petri Nets I: Basic Models*. Lecture Notes in Computer Science.

Russell S., Norvig P. (2003): *Artificial Intelligence: A Modern Approach*. 2nd Edition. Pearson Education. ISBN 0-13-080302-2.

Sackett D., Rosenberg W., Gray J., Haynes R., Richardson W. (1996). *Evidence based medicine: what it is and what it isn't – It's about integrating individual clinical expertise and the best external evidence.* British Medical Journal, Vol. 312, pp. 71–72.

Sackett D. (1997). *Evidence-Based Medicine.* Seminars in Perinatology. Vol. 21, No. 1, pp. 3–5.

Shafranovich Y. (2005): *RFC 4180 – Common Format and MIME Type for Comma-Separated Values (CSV) Files*. Available online at http://tools.ietf.org/html/rfc4180.

Solon J., Feeney J., Jones S., Rigg R., Sheps C. (1967): *Delineating episodes of medical care.* American Journal of Public Health and the Nation's Health. Vol. 57, No. 3, pp. 401–408.

Weijters A., van der Aalst W. (2003): *Rediscovering Workflow Models from Event-Based Data using Little Thumb*. Integrated Computer-Aided Engineering. Vol. 10, No.

2, pp. 151–162. Available online at http://wwwis.win.tue.nl/~wvdaalst/publications/p188.pdf

Wen L., Wang J., van der Aalst W., Huang B., Sun J. (2009): *A Novel Approach for Process Mining Based on Event Types.* J Intell Inf Syst. Vol. 32, pp. 163–190. Available online at http://wwwis.win.tue.nl/~wvdaalst/publications/p232.pdf

7.1 Interviews

12.6.2009: Paul Lillrank, Helsinki University of Technology (HUT)

17.6.2009: Timo Itälä, HUT

26.6.2009: Markku Mäkijärvi, Meilahti hospital, HUS

5.8.2009: Paulus Torkki, HUT/NHG

6.8.2009: Paul Lillrank, HUT

11.9.2009: Paulus Torkki, HUT/NHG

15.9.2009: Antti Peltokorpi, HUT

29.9.2009: Paulus Torkki, HUT/NHG

2.11.2009: Paul Lillrank, HUT

8 Appendices

8.1 Appendix 1: D/F-tables of heuristic mining algorithm test

The first column contains the activity for which the D/F-table is calculated. The value on the third row, first column indicates the organization unit the activity belongs to.

A	B	#B	#B<A	#A<B	local metric	global metric	DS
Symptoms	Symptoms	1000	0	0	0	0	0
Patient	Visit the doctor	1000	0	996	0,998997	0,8	0,899498
	Treatment or not?	1000	0	0	0	0,64	0,32
	Patient recovered	1000	0	0	0	0,448734	0,224367
	Treatment	369	0	0	0	0,425973	0,212986
	Did it help?	369	0	0	0	0,340778	0,170389

A	B	#B	#B<A	#A<B	local metric	global metric	DS
Visit the doctor	Symptoms	1000	996	0	-0,999	-0,8	-0,8995
Diagnosis	Visit the doctor	1000	0	0	0	0	0
	Treatment or not?	1000	0	994	0,998995	0,8	0,899497
	Patient recovered	1000	0	0	0	0,560917	0,280458
	Treatment	369	0	0	0	0,532466	0,266233
	Did it help?	369	0	0	0	0,425973	0,212986

A	B	#B	#B<A	#A<B	local metric	global metric	DS
Treatment or not?	Symptoms	1000	0	0	0	-0,64	-0,32
Diagnosis	Visit the doctor	1000	994	0	-0,99899	-0,8	-0,8995
	Treatment or not?	1000	0	0	0	0	0
	Patient recovered	1000	0	687	0,998547	0,701146	0,849846
	Treatment	369	0	307	0,996753	0,665583	0,831168
	Did it help?	369	0	0	0	0,532466	0,266233

A	B	#B	#B<A	#A<B	local metric	global metric	DS
Patient recovered	Symptoms	1000	0	0	0	-0,44873	-0,22437
Patient	Visit the doctor	1000	0	0	0	-0,56092	-0,28046
	Treatment or not?	1000	687	0	-0,99855	-0,70115	-0,84985
	Patient recovered	1000	0	0	0	0	0
	Treatment	369	0	0	0	-0,53247	-0,26623
	Did it help?	369	307	0	-0,99675	-0,66558	-0,83117

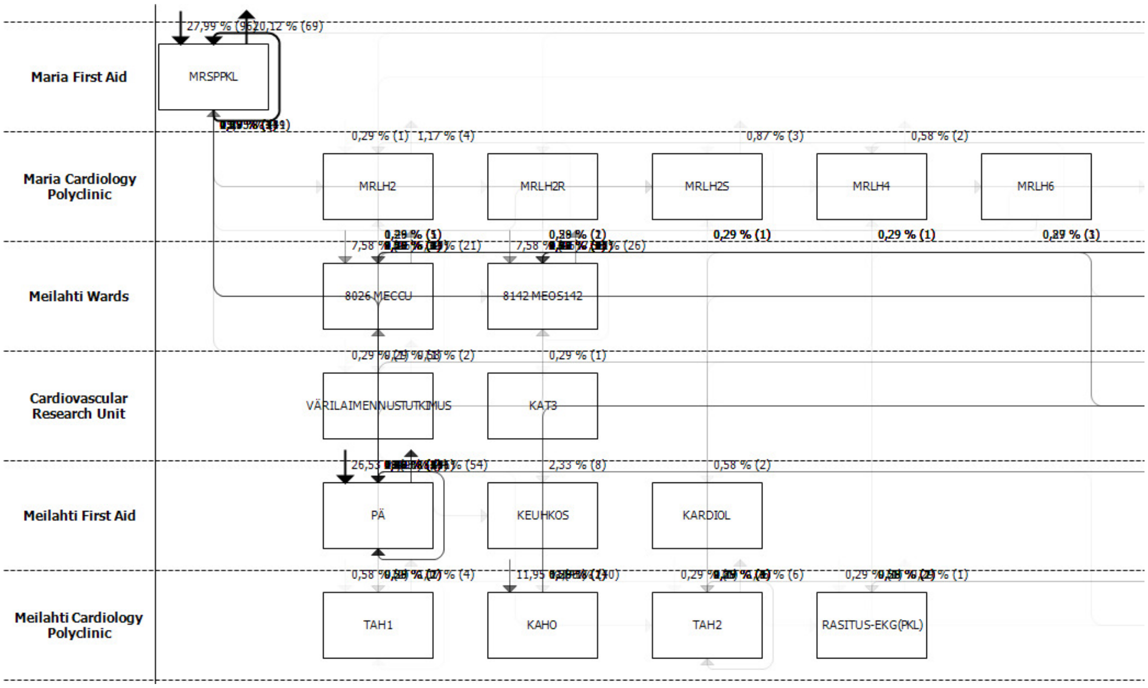
A	B	#B	#B<A	#A<B	local metric	global metric	DS
Treatment	Symptoms	1000	0	0	0	-0,42597	-0,21299
Treatment	Visit the doctor	1000	0	0	0	-0,53247	-0,26623

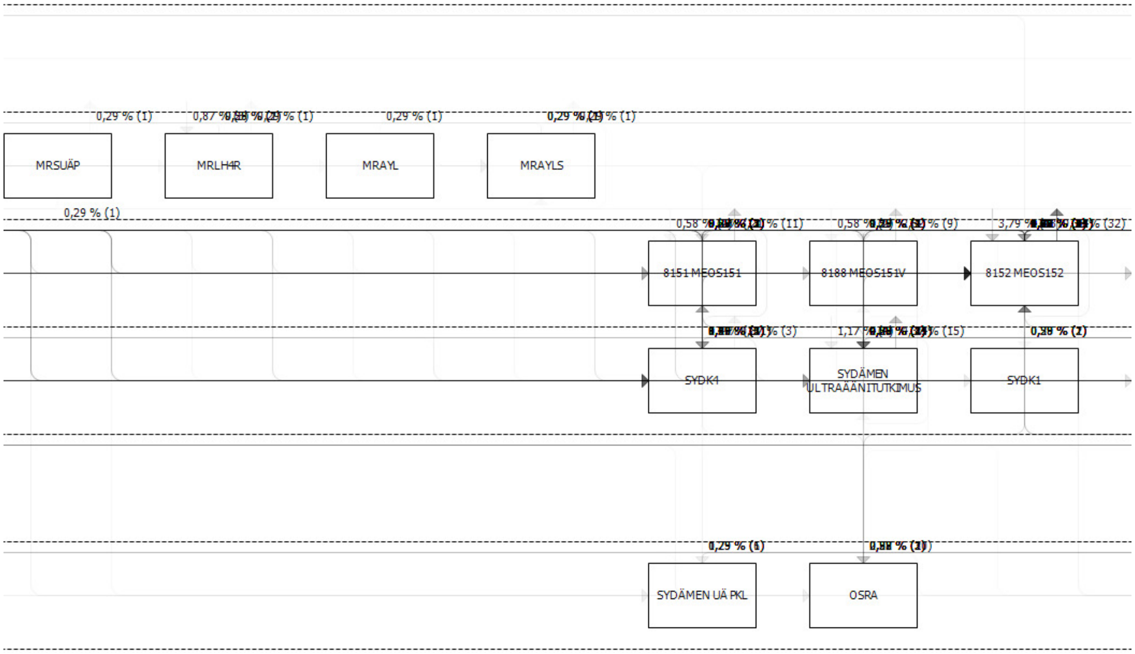
	Treatment or not?	1000	307	0	-0,99675	-0,66558	-0,83117
	Patient recovered	1000	0	0	0	0,532466	0,266233
	Treatment	369	0	0	0	0	0
	Did it help?	369	62	368	0,709977	0,665583	0,68778

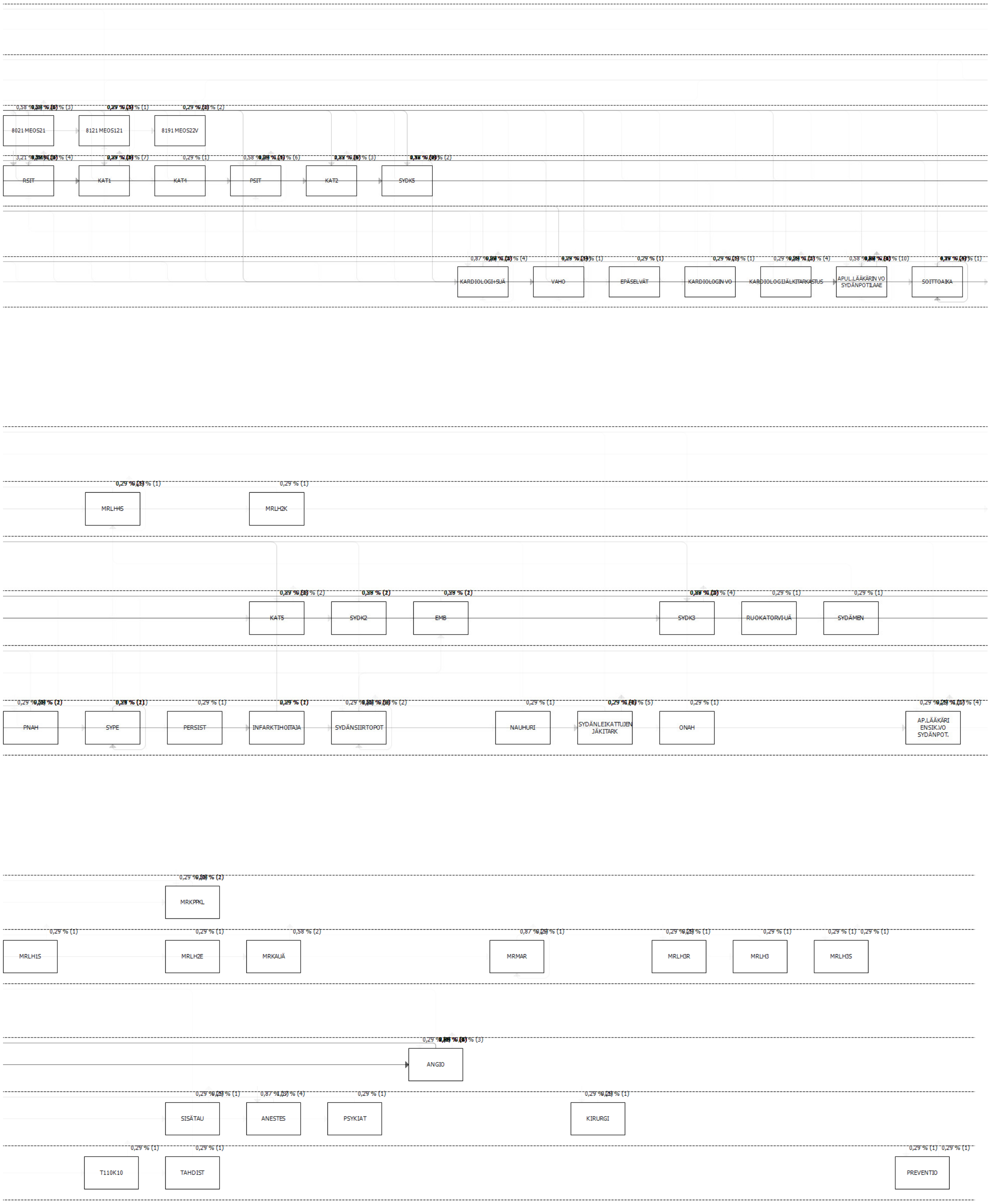
A	B	#B	#B<A	#A<B	local metric	global metric	DS
Did it help?	Symptoms	1000	0	0	0	-0,34078	-0,17039
Treatment	Visit the doctor	1000	0	0	0	-0,42597	-0,21299
	Treatment or not?	1000	0	0	0	-0,53247	-0,26623
	Patient recovered	1000	0	307	0,996753	0,665583	0,831168
	Treatment	369	368	62	-0,70998	-0,66558	-0,68778
	Did it help?	369	0	0	0	0	0

8.2 Appendix 2: Full process model generated from cardiology data

The first and most relevant activities are shown in greater detail in the first two pictures. The last three pictures show the tail of the process model as smaller and less detailed pictures.


Maria First Aid
Maria Cardiology Polyclinic
Meilahti Wards
Cardiovascular Research Unit
Meilahti First Aid
Meilahti Cardiology Polyclinic
MRSPPKL
27,99 %
20,12 % (69)
MRLH2
MRLH2R
MRLH2S
MRLH4
MRLH6
0,29 % (1)
1,17 % (4)
0,87 % (3)
0,58 % (2)
8026 MECCU
8142 MEOS142
7,58 %
(21)
(26)
VÄRILAIMENNUSTUTKIMUS
KAT3
0,29 % (1)
(2)
PÄ
KEUHKOS
KARDIOL
26,53
(54)
2,33 % (8)
0,58 % (2)
TAH1
KAHO
TAH2
RASITUS-EKG(PKL)
0,58 %
(4)
11,95
(6)
0,29 %
(1)


MRSUÄP
MRLH4R
MRAYL
MRAYLS
0,29 % (1)
0,87 %
8151 MEOS151
8188 MEOS151V
8152 MEOS152
0,58 %
(11)
(9)
3,79 %
(32)
SYDK4
SYDÄMEN ULTRAÄÄNITUTKIMUS
SYDK1
1,17 %
(3)
(15)
(2)
SYDÄMEN UÄ PKL
OSRA
0,29 % (1)


8021 MEOS21
8121 MEOS121
8191 MEOS22V
RSIT
KAT1
KAT4
PSIT
KAT2
SYDK5
KARDIOLOGI+SUÄ
VAHO
EPÄSELVÄT
KARDIOLOGIN VO
KARDIOLOGIJÄLKITARKASTUS
APUL.LÄÄKÄRIN VO
SYDÄNPOTILAAE
SOITTOAIKA
MRLH4S
MRLH2K
KAT5
SYDK2
EMB
SYDK3
RUOKATORVI-UÄ
SYDÄMEN
PNAH
SYPE
PERSIST
INFARKTIHOITAJA
SYDÄNSIIRTOPOT
NAUHURI
SYDÄNLEIKATTUJEN
JÄKITARK
ONAH
AP.LÄÄKÄRI
ENSIK.VO
SYDÄNPOT.
MRKPPKL
MRLH1S
MRLH2E
MRKAUÄ
MRMAR
MRLH3R
MRLH3
MRLH3S
ANGIO
SISÄTAU
ANESTES
PSYKIAT
KIRURGI
T110K10
TAHDIST
PREVENTIO