Anti-pattern detection and analysis in data-driven product design

Pietraru, A.

Award date:
2018
Anti-pattern detection and analysis in data-driven Product Design

Master Thesis

A. (Andrada) Pietraru

Supervisors:
  dr. ir. N. (Natalia) Sidorova
  ir. A. (Angelique) Brosens-Kessels, PDEng (Philips Healthcare)
  dr. A.M. (Anna) Wilbik
  ir. M. (Maikel) L. van Eck (Advising member)

Final version

Eindhoven, July 2018
Abstract

The study of user behaviour is an important part of the product design process. This process is particularly more difficult when dealing with products that require very complex user interaction. Therefore, obtaining as much product usage information as possible is needed, since it can reveal patterns in the user behaviour that indicate a misalignment between the use intended by product designers and the actual use. Most often, there are a lot of assumptions made when designing that are either based on domain knowledge or on personal expertise. Traditional techniques such as interviews or reviewing customer complaints provide a restricted view of the behaviour of a small set of users. A more effective data-driven approach, which is presented in this thesis, is to improve the design process by facilitating the understanding of usability-related issues through analysis of product usage data. Product usage data is stored nowadays for many smart products, and it records numerous activities that are performed during the usage of a system, which can be used to search for or investigate different usage patterns of unforeseen or undesired user behaviour.

We have applied this approach in a real-life case-study at Philips, to analyse the usage of Interventional X-ray (iXR) systems used during surgery and discovered a new way of using the system. Several research goals were met: (1) choose a way to specify usage patterns and anti-patterns and to detect them in the product usage data, (2) define a method to investigate how users respond after an anti-pattern occurs, and (3) find correlations between anti-pattern occurrences and other system variables such that it enables root-cause analysis. A formalism based on Metric Temporal Logic was chosen to specify patterns and anti-patterns in a way that is easier to read and understand by everyone, and not just experts. The specifications were then mapped to a detection algorithm, which acted as the basis for building a Python framework that connects to a database, and mines, retrieves, and stores relevant data for the case-study under investigation. Then, the same formalism based on Metric Temporal Logic was used to define and mine responses after an anti-pattern occurs. Different techniques were used to enable insights and input for root-cause analysis. Because of the nature of the logging, data had to be reduced, and several strategies were implemented, to find the most optimal one. Then, different process mining techniques were applied to obtain clearer, more concise process models. Further analysis was performed in order to uncover unknown trends, or correlations between the occurrence of the anti-patterns, the responses and other system variables.

To conclude, the results of the case-study performed showed that we can learn from data and several questions can be answered by this type of analysis before going to talk to the users, such as what to ask, when to ask, at what time to observe, and what to look for. This makes the old methods used by experts more precise when talking to or observing users. One example is presented, where based on the investigation results and the following actions taken by a system expert, we have uncovered a new way of using the system, with two users working simultaneously on the system, which is something experts were not aware of. When one user was asked about it, he did not recall it since interaction is not always done consciously. This is a clear example that shows how traditional techniques of studying user behaviour cover a restricted view of a small subset of users, by taking only one perspective into account, and can benefit from analysing product usage data and making product design more data-driven.
Preface

First of all, I would like to start by saying I appreciate I was given the opportunity to work on this project. The past months have helped be grow both personally and professionally. It has been a very challenging project, but in the end we managed to get some useful results that can be used for product design, but also input for future improvements. This would have not been possible without the help of my supervisors, whom I want to thank for all their time, guidance and feedback. Natalia has helped me widen my boundaries and strengthen my way of thinking, by always challenging every step of the process, but also providing support when needed. Her feedback was always valuable, and thanks to it, I managed to structure my project and thesis as well as I did.

I would also like to express my gratitude towards Angelique and Maikel, who took the time to thoroughly introduce the (very-complicated) system and topic, and guided my weekly activities in numerous meetings. Without their knowledge and advices, I would not have been able to achieve all the goals we set six months ago. As well, I would like to thank Anna Wilbik, for agreeing to be a member of the assessment committee, and taking the time to read and assess this thesis.

Lastly, I would like to thank my parents for always supporting, loving and encouraging me, even when I was under a lot of stress and pressure. My parents have given me the freedom to grow in the direction I wanted, even if it meant being so far away from them, and I will always be grateful for that. Their love and help kept me going, as I achieved all my goals, one by one.
# List of Figures

1.1 An example of a Philips Azurion System. ........................................... 3  
1.2 Top-level overview of the chosen approach........................................... 5  
2.1 Example event log.................................................. 7  
2.2 Overview MTL templates (1)........................................... 12  
2.3 Overview MTL templates (2)........................................... 12  
3.1 Overview of all the artifacts modelled for the generic example. ............ 17  
3.2 Example events in the input test log, pressing pedal while X-ray disabled. .... 22  
3.3 Example anti-pattern instance found, pressing pedal while X-ray disabled. .... 22  
3.5 Overview results for anti-pattern where fluoroscopy or exposure pedal is pressed while X-ray is disabled; many occurrences. ........................................... 25  
3.6 Overview results for an anti-pattern that lead to finding only a few occurrences. .... 25  
3.4 Overview of all the artifacts modelled for the case-study 1. .................. 26  
4.1 Example of process model for one anti-pattern without any data reduction. .... 29  
4.2 Example - 3 anti-pattern occurrences within same exam leads to creation of 3 different event logs. .................................................. 31  
4.3 Mapping between anti-patterns, MTL templates, activities and system variables. .... 32  
4.4 Example of relative frequency plot; divided per response category: green-positive, red-negative, yellow-miscellaneous; all sum up to 1 such that it is clear which activities are correlated more with one of the response categories. ............... 33  
4.5 Inductive Miner - Process model after reduction based on support, lift, confidence; running example case-study, miscellaneous responses ........................................... 39  
4.6 Inductive Miner - Process model after reduction based on support, lift, confidence; running example case-study, negative responses ........................................... 39  
4.7 Inductive Miner - Process model after reduction based on support, lift, confidence; running example case-study, positive responses before further reduction .......... 40  
4.8 Inductive Miner - Process model after reduction based on support, lift, confidence; running example case-study, positive responses after further reduction ............ 40  
4.9 Inductive Miner - Process model after entropy-based filtering; running example case-study, miscellaneous responses ........................................... 41  
4.10 CSM Miner - Process model after reduction based on support, lift, confidence; running example case-study, negative responses ........................................... 41  
4.11 CSM Miner - Process model after reduction based on support, lift, confidence; running example case-study, miscellaneous responses (1) ........................................... 42  
4.12 CSM Miner - Process model after reduction based on support, lift, confidence; running example case-study, miscellaneous responses (2) ........................................... 42  
4.13 CSM Miner - Process model after reduction based on support, lift, confidence; running example case-study, positive responses (1) ........................................... 43  
4.14 CSM Miner - Process model after reduction based on support, lift, confidence; running example case-study, positive responses (2) ........................................... 43
4.15 Two Local Process Models after entropy-based filtering; running example case-study, miscellaneous responses ........................................... 44

5.1 Computing difference between the time the anti-pattern occurs and lab start, end, and scaling it ................................................................. 46

5.2 Histogram after scaling AP occurrences for running example; red is negative response, green is positive and blue is miscellaneous ........................................... 47

5.3 KDE plot after scaling AP occurrences for running example; red is negative response, green is positive and blue is miscellaneous ........................................... 47

5.4 Distribution of frequency per exam - X-axis show the number of instances during procedure, Y-axis shows the number of procedures with that many instances .. 47

5.5 Distribution over the time since last occurrence, where X-axis has 5 intervals (“0-5 min”, “5-30 min”, “30-60 min”, “1-6 hours”, “6-12 hours”) and Y-axis shows the number of anti-pattern instances ........................................... 47

5.6 Association rules found for running example case-study ........................................... 49

5.7 Occurrences grouped per response category and per month; running example case-study ........................................... 49

5.8 Occurrences grouped per response category and per machine; running example case-study ........................................... 49

5.9 Occurrences grouped per response category and per protocol; running example case-study ........................................... 49

5.10 Occurrences grouped per response category and per country; running example case-study ........................................... 49

5.11 Heatmap anti-pattern occurrences over number of exams per machine (Y-axis) per month (X-axis); running example case-study ........................................... 50

5.12 Overview input and output for hazardous situations ........................................... 51

5.13 Percentage of pattern instances (green) versus anti-pattern instances (red); running example case-study ........................................... 52

5.14 Two doctors working on the same set of pedals ........................................... 53

5.15 Two different pedals sets used in the same exam room ........................................... 53

E.1 Overview RapidMiner workflow for mining association rules ........................................... 74
List of Tables

2.1 Overview of the templates defined in [4] and adapted to be used for the anti-pattern formalization. ................................................................. 13

3.1 Overview of the artifacts and the set of possible states for each artifact; generic example. ................................................................. 16
3.2 Overview of the artifacts and the set of possible states for each artifact, case-study 1. 23
3.3 Overview of the artifacts and the set of possible states for each artifact; case-study 2. 27

4.1 Mapping relational data to event data by columns ............................................. 30
4.2 Overview of the number of cases, events and activities left before and after filtering based on entropy; pedal press while X-ray disabled, miscellaneous responses. .... 33
4.3 Overview of the number of cases, events and activities left after filtering based on support, lift and confidence; pedal press while X-ray disabled. .................. 35
4.4 Overview of the number of cases, events and activities left before filtering; pedal press while X-ray disabled. ..................................................... 35

D.1 Summary of the results of investigating the anti-patterns and their responses. .... 73
Chapter 1

Introduction

The study of user behaviour is an important part of designing a product, and consequently of the product development process. This process is particularly more difficult when dealing with products that require very complex user interaction. Therefore, obtaining as much product usage information as possible is needed, since it can reveal patterns in the user behaviour that indicate a misalignment between the use intended by product designers and the actual use. Furthermore, it can indicate that there are missing features, or the contrary, features that are hardly ever used and may be removed or changed.

Most often, it can be difficult for designers to imagine how someone without expert knowledge would do certain things. Hence, there are a lot of assumptions made that are either based on domain knowledge or on personal expertise. The reason behind this is the fact that obtaining detailed information on user behaviour can be a very costly and time consuming task, since it requires both the time of system experts but also of system users, and one cannot interview all the users of a system. Another source of input for getting information on product usage comes from user complaints. When a user complains about a feature, the product designer does not know how other users experience it. Hence, it is important to investigate the scale of impact before deciding to remove or change a system feature. If most users do not experience that problem, it is best to investigate that one particular user that complained. Or, the exact opposite, it might happen that a lot of users encounter the same problem, and then the issue is much bigger than anticipated and has to be addressed globally. Moreover, techniques such as interviews or reviewing customer complaints provide a restricted view of the behavior of a small set of users, by taking only one perspective into account. Therefore, the following question arises: How to analyse user behaviour systematically? One approach could be to define a survey and send it to as many users possible. But then again, you might miss relevant questions, and users might not recall what they do. Most of the time you find only parts of the answers, since users sometimes come up with unexpected ways of using a product and they might not recall how and if they do certain things them when asked. Additionally, it also can happen that the product designer does not know what to specifically ask. A more effective data-driven approach, which will be discussed in this thesis, is to look into the data stored from the automatic collection of product usage data. Product usage data is stored nowadays for many smart products, and it records many activities that are performed during the usage of a system, which can be used to search for or investigate different patterns of expected and unexpected user behaviour.

The approach presented throughout this thesis will enable product designers to integrate usage data in their design processes, and, eventually, help improve the product. By systematically analysing user behaviour from the product usage data, we will have a better overview on how many users experience or not a particular problem issued in a customer complaint. Furthermore, it will help improve the traditional practices used currently to study user behaviour, such as usability testing, semi-structured interviews or observations, by knowing what, when, to whom,
CHAPTER 1. INTRODUCTION

and how often something happened. Overall, data-driven design is based on more representative results that give a different view of reality, also taking into account aspects such as interaction between users, and not just between product and user.

1.1 Project Context

Philips IGT (Image Guided Therapy) Systems department is part of Philips Medical Systems. This business unit creates iXR (Interventional X-ray) machines like Azurion, which are designed to have highly intuitive usability that supports performing interventional, surgery-like procedures on patients. Azurion machines rely on the use of radiological image guidance, such as X-ray fluoroscopy and exposure. Fluoroscopy is a type of medical imaging technique that shows a continuous X-ray image of the interior of an object on a monitor [1]. X-ray exposure uses a higher dose of radiation than fluoroscopy, and can give live or recorded images, with a higher precision and better quality. The Azurion machines have several built-in protocols a user can select, depending on the type of the procedure and on the area of the body that is being investigated. These are complex machines, with many different components that need to be in tight orchestration with the medical staff using it, not just doctors, but also technicians. Figure 1.1 depicts an Azurion machine, which has seven main components [16]:

1. C-arm: the part of the system that is used to position the detector and X-ray tube at the right angle.
2. Detector: the place where X-ray image is captured.
3. Table: the place where the patient lays down.
4. FlexVision: the screen on which medical staff can (re)view images and position the system.
5. Touch Screen Module(TSM): used by medical staff to choose certain settings, or give commands to the system. There can be several TSMs connected to the same system, and in different locations (e.g. operation room and control room).
6. Control Module(TSO): module that can be used to position the C-arm and the table.
7. Pedals: used to enable fluoroscopy and exposure.

Good, reliable usability tests can help Philips continuously improve their system’s design. Usability tests are used to see how intuitively the design of such a machine is, by looking at the user interaction after giving him a set of tasks to accomplish. The user is part of the target group of the product, usually from the medical field. Usability tests do not refer to validation, or testing the functionality of the system. It is essential to incorporate critical tasks of a user in a scenario when usability is tested, since the users do not have a lot of spare time, and usability testing tends to be costly. These tasks are usually affected by a change in the system that needs to be tested, or they are believed to potentially lead to some issues. However, since usability testing is costly, it can be done only on a small subset of users that will likely show a biased and incomplete view of reality. A less costly and more feasible way to improve this is to look at the data that is generated by the machines deployed in the field, in order to analyze certain usage patterns. Philips retrieves and stores logs from their medical systems across the globe, including the Azurion machines. The main driver of this project is the data, but the main goal is to improve the design process by facilitating the understanding of usability issues through analysis of system usage data.

So far, previous work has been done for developing a workflow model that incorporates different procedures by using real-life data from one hospital in the Netherlands [16]. The main goal there was to find more about the way doctors are working with the deployed systems by applying process mining techniques to obtain workflow models. Next to the machine log data, there were two more data sources available that were used: location data from users that wear a tag that can be
 CHAPTER 1. INTRODUCTION

Figure 1.1: An example of a Philips Azurion System.

tracked, and manually entered data by users through a tablet during procedures to record the main steps. However, not all hospitals log the additional data, and the fine-grained nature of logging makes the datasets available extremely large. Therefore, instead of converting all machine data into higher-level event data and creating very complex process models, this project focuses only on subsets of activities, looking for undesired expected use of the system where under a certain condition, the expected user goal is not met. These situations are well defined and constrained in scope. Focusing only on a subset of activities is more feasible, since creating a mapping between low-level machine data and high-level activity data for all data is complex, time-consuming and requires very specific domain knowledge, and it is out of the scope of this project.

Currently, the main feedback from the field comes from complaint reports filed by hospitals, but mainly for critical life-threatening events, when the damage is visible. Out of those complaints, only a subset is related to usability issues, since they could be caused not only by user error, but also system failures. On the other hand, when a complaint is reported, it is important to check how other hospitals are experiencing the feature that is in the complaint, before deriving a biased, unrealistic conclusion. In other words, it is important to check the scale of impact in other hospitals, before deciding that a certain feature is faulty, and should be removed or changed. In order to make a fair comparison, we also investigate the desired expected use of the system.

The second source of input for situations that involve undesired use of the systems is based on the risk management Philips does, by having expectations on what could go wrong, and by defining hypothetical hazardous situations and estimating the probability of occurrence. Risk control measures are set in place to prevent these situations, however their efficiency is again estimated. In case of new added features, system experts brainstorm on defining the situations and come up with an estimation based on their knowledge and experience. Once the features are deployed in the field, those estimations are updated based on the customer complaints. The project described by this thesis will improve this process by making it more data-driven, as the results are based on real data from the field, instead of intuition or personal experience. The user responses to risk control measures after an unexpected use of the system occurs are split into categories, and further analyzed in order to uncover correlations, tendencies or hidden patterns, that can enable
CHAPTER 1. INTRODUCTION

root-cause analysis. The findings result into more precise questions about the undesired use of the system that can be used as input for semi-structured interviews or observations, in order to understand better the interaction between users and between user and system. The defined approach can be used in the future for other products and datasets, whether they are similar or not.

1.2 Approach and Research goals

The feedback from deployed systems is limited, infrequent, and not representable for all users. Currently, Philips defines hypothetical hazardous situations based on customer complaints, and domain knowledge and estimates the probability of occurrence. Risk control measures are set in place to prevent these situations, however their efficiency is again estimated. The goal of Philips is to improve the design process by facilitating the understanding of usability-related issues through analysis of system usage data.

The so-called anti-patterns investigated for this thesis define the expected undesired use of the system that can lead to an usability issue. Patterns are the opposite of anti-patterns, and define the expected desired use of the system. By analysing them, we can see how the systems are used in the field in some situations. The results obtained can reveal ways of working that a usability engineer or system designer did not think of. Moreover, using data from multiple systems allows for a more realistic view on how the system is indeed used, instead of just taking user complaints into account. It is important to see how other users deal with certain usage patterns and anti-patterns, and provide input for comparison to the current estimations made in risk analysis and management. To achieve this, three research goals were defined:

Research goal 1 (RG1): Choose a way to specify usage patterns and anti-patterns and to detect them.

First, anti-patterns are defined. Figure 1.2 depicts the two main sources of input for defining anti-patterns that may lead to the situation where an issue may occur: customer complaints and predefined hazardous situations (or expectations based on experience). Then, the patterns follow, by reasoning what the correct usage should have been. Machine data is recreated for selected patterns and anti-patterns by using a test machine. The recreated data is cleaned and interpreted, such that it can be correlated to the activities performed. Next, after choosing a formalism, the anti-patterns and patterns have to be defined for the case study of this project. The formalization can be used to define other patterns and anti-patterns in the future in general. The definitions are then converted into code, based on a detection algorithm. A framework is built in order to connect to the database, and to mine, retrieve, and store relevant data for anti-patterns and patterns. Every time an anti-pattern occurrence is found, the data is retrieved based on a specific time frame. Time is included because looking only at the order in which some activities are executed is not enough, since there are so many events logged. In addition, the user expects a timely response from the system when interacting, so checking the time between certain activities is needed.

Research goal 2 (RG2): Define a method to investigate how users respond after an anti-pattern occurs.

As mentioned before, in some cases, there are functionalities designed to inform and prevent undesired use of the system, and the next step after finding instances of an anti-pattern, is to investigate what was the response of the user immediately after. The expected responses are defined similarly to anti-patterns, and the mapping to code is done as well, by using the previously mentioned framework. The responses are categorized into three main categories, such that we can distinguish between a positive, a negative and an "unexpected" response. The responses are depicted as process models. In order to reduce the large set of machine events logged and
obtain a set of understandable activities that are relevant to a specific anti-pattern and response category, different measures are computed to help filter some activities. Using the filtered data, we apply different process mining techniques such that the models obtained are easier to understand and show a more clear flow and sequence of activities. In order to achieve the last step, converting the retrieved data into event data is needed.

**Research goal 3 (RG3):** Find correlations between anti-pattern occurrences and other system variables such that it enables root-cause analysis.

Further analysis is done to summarize and plot frequencies for the anti-patterns and their response, and to uncover unknown correlations between them and other system variables, such as time, location of the machine, and others. Also, the estimations previously done for risk management will be compared with data-driven results, based on patterns and anti-patterns frequencies. The findings are used as input for causal analysis, which leads to talking to the users, people from clinical marketing that are in contact with users, or checking other customer complaints.

Lastly, input for improving future data logging is provided. The available data was collected in a period before this thesis. It was not possible to include any requirements or suggestions for the purpose of this research. All suggestions on the data will be included for future studies.

Some advantages of the chosen approach are:

- By using domain knowledge and recreating the machine data for specific usage patterns and anti-patterns, we can define exactly what events to search for, and increase the chances of getting traces that are more reliable, and that can be correctly interpreted.

- Searching for patterns blindly results in a large collection on uninteresting patterns related to normal behavior, this approach is more straightforward and faster, as it allows us to focus on usability issues and risks.

- Process discovery is normally limited to the discovery of a model capturing the behavior of process instances as a whole. Because the machine logging is so detailed, the users have so much freedom and the large variety in clinical procedures in which they are used all lead to complex workflows that are hard to read and interpret, so it is desirable to focus on local patterns within case instances. Moreover, the reason for mining for specific anti-patterns is that we are focused on investigating to see how often they happen, and what are the user

![Figure 1.2: Top-level overview of the chosen approach.](image-url)
responses to undesired behaviour. To do this, the whole workflow model is unnecessary, and even more, not suited.

- Converting low-level machine data into high-level event data is very time consuming and messy. This approach partly avoids this step.

1.3 Outline

The remainder of this thesis is structured as follows. Chapter 2 introduces several preliminary concepts that will be used throughout this thesis, such as basic process mining terms, composite state machines, and temporal logic. Chapter 3 refines the concepts introduced in the previous chapter, introduces the definition for patterns and anti-patterns, followed by responses. A running example is given to show how they are formalized. Then, the same chapter describes how the formulas can be converted into code based on a detection algorithm. Next, since we are dealing with a very complex and fine-grained dataset, different strategies in order to reduce it are implemented and explained in chapter 4, followed by the process mining techniques applied and the models obtained. Chapter 5 describes what analysis was further performed in order to uncover unknown trends, or correlations between the occurrence of the anti-patterns, the response categories and other system variables. Then, the same chapter depicts some interesting findings and how they can be used to make the design process more data-driven, and make user interviews and observations more precise. Lastly, the conclusions are presented in chapter 6, where limitations and future work are also addressed.
Chapter 2

Preliminaries

This chapter introduces preliminary concepts that will be used throughout this thesis. First, we define what process mining is, and all the related terms, such as activity, event, or trace. Next, we focus on laying a basis for the formalization in chapter 3, by looking into specifying patterns, composite state machines, and temporal logic.

2.1 Process mining

Process mining is an emerging discipline that is based on process model-driven approaches and data mining [34]. The goal of process mining is to discover, monitor and improve real processes by extracting knowledge from event logs readily available in today’s systems, as the Philips iXR machines. The minimal requirements for process mining are that any event can be related to both a case and an activity and that events within a case are ordered. In an event log, as shown in figure 2.1, each row represents an event, and should contain at least a case identifier, the activity name, and a timestamp. The following definitions, which are based on [34], introduce the basic terms that we will use throughout this thesis:

![Figure 2.1: Example event log.](image)

<table>
<thead>
<tr>
<th>CaseId</th>
<th>Activity</th>
<th>Event/Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>B_LE10013</td>
<td>Command: SwitchRoomLight</td>
<td>8/28/2017 8:53:33.037</td>
</tr>
<tr>
<td>B_LE10013</td>
<td>XrayService: acquisition parameters</td>
<td>8/28/2017 8:53:33.918</td>
</tr>
<tr>
<td>B_LE10013</td>
<td>Command: Stop CHANGEPATIENTSUPPorthand.HEIGHT</td>
<td>8/28/2017 8:55:15.985</td>
</tr>
<tr>
<td>B_LE10013</td>
<td>Lab: display mandatory applications</td>
<td>8/28/2017 8:55:15.989</td>
</tr>
<tr>
<td>B_LE10013</td>
<td>XrayService: Fluoroscopy started</td>
<td>8/28/2017 8:55:16.228</td>
</tr>
<tr>
<td>B_LE10013</td>
<td>Command: StartFluoroscopy</td>
<td>8/28/2017 8:55:22.829</td>
</tr>
</tbody>
</table>

Definition 1. A case is a process instance, with an unique identifier.

Definition 2. An activity is a well-defined step in the process.

Definition 3. An event refers to an activity occurring at a specific moment in time, and is related to a particular case.

Definition 4. A trace is the time-ordered sequence of all events that belong to one case.
To give an example, consider the process involved in an operation. One case will be one specific operation performed on a patient. Activities involved in an operation could be: starting the exam, requesting an image, and selecting a radiation protocol. Then, one event will be the instantiation of an activity at a particular moment in time during a case: e.g. selecting protocol for cardiac surgery for operation with case ID $X$. The trace for case $X$ will contain all events related to that case, ordered chronologically.

There are three main “categories” of process mining: process discovery, conformance checking and process model enhancement [34]. The first one focuses on creating a process model by only using event data, and no other knowledge about the process itself. Conformance involves having a process model that can be checked against event data to detect and locate deviations between them. Process model enhancement can be used to repair a process model such that it reflects real-life behaviour, or to extend a process model to include a new perspective. This thesis focuses on process discovery from the control-flow perspective, which looks at the order of the activities, and from the time perspective, looking at execution timing and frequency of events.

### 2.2 Composite State Machines

Traditional process mining approaches are not the most suitable for this project, since as noted in [12], complex process behaviour can result in unstructured process models, which makes them difficult and time-consuming to analyse. Furthermore, there are often multiple views on the same process, and analysts do not always know what they are looking for. An artifact-centric approach was also used in the previous work done on iXR machines usage data for developing a workflow model that incorporates different procedures by using real-life data from one hospital in the Netherlands [16]. Even though for this thesis we focus only on some parts of the workflow, since mining the model for the entire system is too complex, an artifact-centric approach remains the most suitable solution, as we can focus only on specific parts of the system, depending on the usage pattern. Furthermore, the machine is already decomposed into different units (e.g. X-Ray, table) from a design perspective, resulting in a natural artifact-centric structure. An artifact-centric approach implies using the notion of states. According to the IEEE Standard Glossary of Software Engineering Terminology, states are “the values assumed at a given instant by the variables that define the characteristics of a system, component, or simulation”. For example, an artifact can be the status of the radiation, and the state of this artifact would be “on” or “off”.

In this thesis we will use the notion of composite state machine, CSM, which can be used to model the behaviour of a system in terms of its possible states and showing the potential transitions between states. CSM was defined in [12]:

\[
A \text{ process consisting of a number of interacting artifacts is called an artifact system, and we model its behaviour as a Composite State Machine. The state of a CSM is defined as the composition of the states of its artifacts, i.e. it is a vector of states. The set of all possible states of a CSM is a subset of the cartesian product of the sets of states of its artifacts, as not all combinations of artifact states are necessarily possible. Each transition in a CSM represents a change in the state of at least one artifact; we do not allow self loops.}
\]

**Definition 5.** A Composite State Machine $M = (S, T, b, f)$ is a model of a process with $n$ artifacts where $S \subseteq (S_1 \times \cdots \times S_n)$ is a set of states, with $S_1, \ldots, S_n$ the sets of artifact states, $b = (b_1, \ldots, b_n)$ is the initial source state, $f = (f_1, \ldots, f_n)$ is the final sink state, $T \subseteq (S \cup \{b\}) \times (S \cup \{f\})$ is the set of transitions, and $\forall (s, s') \in T : s \neq s'$. We define $\overline{S} = S \cup \{b, f\}$ and $\overline{S}_i = S_i \cup \{b_i, f_i\}$ for $i \in \{1, \ldots, n\}$. 

8 Anti-pattern detection and analysis in data-driven Product Design
2.3 Formalizing patterns and anti-patterns

The term anti-pattern was coined in [19] with the following definition: “An anti-pattern is just like pattern, except that instead of a solution it gives something that looks superficially like a solution, but isn’t one”. Patterns and anti-patterns were defined in the context of process mining and other domains before, and some of the approaches will be discussed next.

One paper that ties pattern mining to process mining is [18], however the patterns addressed here are a means of abstraction for common process model constructs in event logs, and this is not useful for the purpose of this thesis, since the main goal is not to cluster and abstract tasks based on frequency. Incorrect behaviour does not happen frequently, but it is rather sparse.

Other papers, such as [9] define patterns as frequent itemsets, which are useful when mining for association rules, or [26], where architectural and design patterns are investigated to automatically analyze the quality of a software architecture. These aspects do not directly match the goal of this thesis, since we do not focus on the quality of the software design or frequent behaviour, as explained before.

Regarding anti-patterns, paper [17] describes them as poor solutions to design and implementation problems, which are claimed to make object oriented systems hard to maintain. Similarly, in [28] anti-patterns are defined as error prone modeling decisions that can result in the creation of models that allow unintended representations. These definitions refer to poor design decisions, and not to analyzing aspects of the use of a system.

Also related to process mining and process models, [13] introduces anti-patterns as erroneous relations between process models, which can lead to incorrect behaviour such as dead events, lost triggers, or even deadlocks and livelocks of the process architecture as a whole, even though the individual processes do not contain such behaviour. Again, this does not comply with the purpose of this thesis, where the analysis is done for parts of process models, and there is not much focus on the interaction between different models. Moreover, anti-patterns, as we defined them, do not necessarily have to lead to bottlenecks or deadlocks, but rather they represent an incorrect order or use of certain activities in some specific scenario.

As mentioned in Chapter 1, for this project every anti-pattern is defined by the expected undesired use of the system that can lead to an usability, and every pattern by the expected desired use. We base the formalization of each pattern and anti-pattern on Dwyer’s templates [10] [11], where patterns are introduced in the domain of property specifications for finite-state verification. The system under verification is modeled as a transition system with a finite number of states and a set of transitions, labeled with events, between these states. Specification patterns were defined in [10] and [11] as:

Definition 6. A property specification pattern is a generalized description of a commonly occurring requirement on the state sequences in a finite-state model of a system. A pattern comprises:

- A name: describes the nature of the pattern
- A precise statement of the patterns intent: the structure of the behavior described
- Mappings into common specification formalisms for each of the five possible scopes (i.e., “Globally”, “Before R”, “After Q”, “Between Q and R”, “After Q until R”)
- Examples of known uses: describes common situations where the pattern is useful in real-world scenarios
- Relationships to other patterns
We choose to define our patterns and anti-patterns following Dwyer’s templates, because they provide a systematic way of defining patterns, and they have been used in literature extensively. Furthermore, these patterns were extended to include specifying real-time properties, which is a requirement in our project. More will be discussed in section 2.3.2, after introducing MTL in the next section.

### 2.3.1 Metric Temporal Logic

We choose as a specification formalism the state-based temporal logic, which has already been defined in literature. Event-based temporal logic formalism would not be as suitable since we are dealing with a very complex set of events, and not everything needs to be specified. Instead, we focus only on parts (or artifacts) for which we can recreate the data and have the correct mean-

...
• \((\rho, i) \models \neg \varphi \iff (\rho, i) \not\models \varphi\)

• \((\rho, i) \models a \iff I < |\rho| \text{ and } \theta_i = a\)

• \((\rho, i) \models \bigcirc \varphi \iff \text{there exists } j \text{ such that } i \leq j < |\rho|, (\rho, j) \models \varphi \text{ and } \tau_j - \tau_i \in I\)

• \((\rho, i) \models \bigcirc \varphi \iff \text{for all } j \text{ such that } i \leq j < |\rho|, (\rho, j) \models \varphi \text{ and } \tau_j - \tau_i \in I, \text{ and } (\rho, k) \models \varphi_1 \text{ for all } k \text{ with } i \leq k < j\).

One example of using MTL is \(\Box (\text{problem} \rightarrow \bigcirc_{<10} \text{alarm})\), which expresses that each time a problem occurs, within 10 time units, an alarm rings [8]. For a more detailed and comprehensive account of MTL please refer to [21] and [25].

2.3.2 Specification of patterns using templates

Dwyer et al. [10] introduce qualitative specification patterns meant to facilitate the specification of critical properties, such as those that must be satisfied by embedded systems. Specification patterns [11] have been used to guide users of finite state verification tools in expressing system requirements directly in a formal specification language, such as linear time temporal logic. These specification patterns cannot be used to specify real-time properties, since they do not support quantitative reasoning about time, hence, paper [20] defines a collection of “real-time” specification patterns, that can be used to specify real-time properties for embedded systems. In this paper, mappings are provided for the specification patterns in terms of one of the most commonly used temporal logics, MTL. Moreover, it introduces a structured English grammar for capturing the specification in terms of natural language. The paper presents an example real-time specification pattern, the Bounded Recurrence, which contains the complete mappings for MTL. Scopes used in Dwyer et al.’s specification patterns (i.e., “Globally”, “Before R”, “After Q”, “Between Q and R”, “After Q until R”) are also applicable for the real-time specification patterns. A more recent paper, [4], also defines real-time specification patterns based on Dwyer et al., but additionally addresses the complexity of the verification problem. They propose a set of specification pattern templates that can be used to express real-time requirements commonly found in the design of reactive systems.

The pattern specifications templates presented in [20] and [4] are meant to further facilitate the understanding of the meaning of a specification, as formal specification languages are often perceived as difficult to use by practitioners, and are therefore rarely used in industrial software development practices. We will base the formalization of the patterns and anti-patterns needed for the case-study of this project on the templates presented in [4], since it also provides an integrated model checking tool chain for the verification of timed requirements on TTS, an extension of Timed Petri Nets with data variables and priorities. The patterns are described using a hierarchical classification borrowed from Dwyer but adding the notion of “timing modifiers”. Patterns are built from five categories, listed below:

- **Existence patterns**: for conditions that must eventually occur
- **Absence patterns**: for conditions that should not occur
- **Universality patterns**: for conditions that must occur throughout the whole execution
- **Response Patterns**: for (trigger) conditions that must always be followed by a given (response) condition
- **Precedence Patterns**: for (signal) conditions that must always be preceded by a given (trigger) condition.
Timed patterns are obtained using one of two possible kind of timing modifiers that limit the possible dates of events referred in the pattern:

- **Within I**: to constraint the delay between two given events to belong to the time interval $I$
- **Lasting $D$**: to constraint the length of time during which a given condition holds (without interruption) to be greater than $D$

Table 2.1 and figures 2.2, 2.3 present the templates defined in [4] that are needed to formalize the patterns and anti-patters for this project. As shown in the table, the templates have a corresponding MTL formula. When defining patterns, the symbols $A$, $B$, ... stand for predicates. Patterns can be easily combined together using the usual boolean connectives. For example, the pattern “$P_1 \land P_2$” holds for all the traces where $P_1$ and $P_2$ both hold.

Figure 2.2: Overview MTL templates (1).

Figure 2.3: Overview MTL templates (2).
<table>
<thead>
<tr>
<th>Category</th>
<th>Template name</th>
<th>Description</th>
<th>MTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Existence 1</td>
<td>Present $A$ after $B$ within $I = [d_1, d_2]$</td>
<td>Predicate $A$ must hold between $d_1$ and $d_2$ units of time after first occurrence of $B$</td>
<td>$B \land \text{True} \ U I A$</td>
</tr>
<tr>
<td>Existence 2</td>
<td>Present first $A$ before $B$ within $I = [d_1, d_2]$</td>
<td>The first occurrence of $A$ holds between $d_1$ and $d_2$ units of time before the first occurrence of $B$</td>
<td>$(\Diamond B) \rightarrow ((\neg A \land \neg B)U(A \land \neg B \land \neg BU_1B))$</td>
</tr>
<tr>
<td>Existence 3</td>
<td>Present $A$ lasting $D$</td>
<td>Starting from first occurrence when $A$ holds, it remains true for at least duration $D$</td>
<td>$(\neg A)U(\Box[0,D] \neg A)$</td>
</tr>
<tr>
<td>Absence 1</td>
<td>Absent $A$ after $B$ for interval $I$</td>
<td>$A$ must never hold between $d_1$ and $d_2$ units of time after first occurrence of $B$</td>
<td>$B \land \Box I \neg A$</td>
</tr>
<tr>
<td>Absence 2</td>
<td>Absent $A$ before $B$ for duration $D$</td>
<td>No $A$ can occur less than $D$ units of time before the first occurrence of $B$</td>
<td>$\Diamond B \rightarrow (A \rightarrow (\Box[0,D] \neg B)) \ U B$</td>
</tr>
<tr>
<td>Response 1</td>
<td>$A$ leads to first $B$ within $I = [d_1, d_2]$</td>
<td>Every occurrence of $A$ must be followed by an occurrence of $B$ within $d_1$ to $d_2$ units of time</td>
<td>$\Box(A \rightarrow (\neg B)U_1B)$</td>
</tr>
<tr>
<td>Response 2</td>
<td>$A$ leads to first $B$ within $I = [d_1, d_2]$ before $R$</td>
<td>Before first occurrence of $R$, each occurrence of $A$ is followed by a $B$ within $d_1$ to $d_2$ units of time</td>
<td>$\Diamond R \rightarrow (\Box(A \land \neg R \rightarrow (\neg B \land \neg R)U_1B \land \neg R)UR$</td>
</tr>
<tr>
<td>Response 3</td>
<td>$A$ leads to first $B$ within $d_1$ to $d_2$ units of time after $R$</td>
<td>Same as previous, but only considering occurrences of $A$ after the first of $R$</td>
<td>$\Box(R \rightarrow (\Box(A \rightarrow (\neg B)U_1B))))$</td>
</tr>
</tbody>
</table>

Table 2.1: Overview of the templates defined in [4] and adapted to be used for the anti-pattern formalization.
Chapter 3

Patterns and anti-patterns

In this chapter we will refine the concepts introduced in the previous chapter such that they match the requirements of this thesis, and a generic example will be introduced to show how patterns and anti-patterns are defined. Part of our approach is to define the conditions under which an undesired or desired usage pattern occurs by using the state of the system. However, describing the entire state of the system would be again too complex. With the use of artifacts, we define the state of the system as a vector of multiple states, such that we can abstract from the “unknown” or irrelevant parts of the system behaviour, for each pattern. The chapter will end with describing how to detect patterns and anti-patterns in the data, and defining anti-patterns and responses for a different dataset, to show that this approach can be used in various settings.

3.1 Formalization of patterns and anti-patterns

We base the formalization of each pattern and anti-pattern on Dwyer’s templates [10] [11], where patterns can be used to specify properties for finite-state verification, but instead of having one transition system, we have one composite state machine, with a number of artifacts defined. In this thesis we use the notion of composite state machine, CSM, as defined in [12], with an extension for an event set $E$ and a transition label set $L$:

Definition 9. $E = \{e_1, e_2, ..., e_z\}$ is a finite set containing $z$ events, where each event $e_i$ triggers a state change for at least one artifact.

Then, we extend the CSM definition as such, keeping in mind that the transition labels from set $L$ correspond to events from set $E$:

Definition 10. A Composite State Machine $M = (S, T, b, f)$ is a model of a process with $n$ artifacts where $S \subseteq (S_1 \times \cdots \times S_n)$ is a set of states, with $S_1, \ldots, S_n$ the sets of artifact states, $b = (b_1, \ldots, b_n)$ is the initial source state, $f = (f_1, \ldots, f_n)$ is the final sink state, $T \subseteq (S \cup \{b\}) \times L$ is the set of transitions, and $\forall (s, s') \in T : s \neq s'$. We define $S_i = S_i \cup \{b_i, f_i\}$ for $i \in \{1, \ldots, n\}$.

Furthermore, because we define several artifacts, with each artifact having several states, we refine the MTL formalization from Chapter 2, since having an alphabet of atomic events is not sufficient in our case:

Definition 11. $\Sigma$ is a non-finite alphabet of predicates, where every symbol is a label that represents predicates $P$ over artifact states from the state vector $S$.

The predicates are defined over the state vector as a Boolean-valued function $P : S \rightarrow \{\text{True}, \text{False}\}$.

Definition 12. A word over the alphabet $\Sigma$ is a sequence of symbols. A timed word is a pair $\rho = (\theta, \tau)$, where $\theta = \theta_0, \theta_1, ..., \theta_n$ is a word over $\Sigma$ and $\tau$ is a time sequence of the same length. The pair $(\theta_i, \tau_i)$ is a timed symbol, having $\tau_i$ as timestamp of the event that triggers the state change which leads to the predicate becoming true.
According to the generic definition, an anti-pattern is a common practice that leads to quality issues, or a common practice of resolving a problem in a non-optimal way. There are various ways to define anti-patterns. Since none of the definitions found in literature for the term anti-pattern match the requirements of this project, we are going to use our own, having in mind that the user goal represents what the user wants to achieve by performing a sequence of activities and it may or may not be specifically logged in the data.

**Definition 13.** An anti-pattern depicts expected undesired use of the system and is defined as a conjunction or disjunction of one or more MTL templates, each containing different conditions and time-frames, where a specific user goal is not met, and can be replaced by a different, undesired, consequence.

**Definition 14.** A pattern depicts expected desired use of the system and is defined as a conjunction or disjunction of one or more MTL templates, each containing different conditions and time-frames, where a specific user goal is met.

Since MTL templates were defined and validated in previous research work as shown in section 2.3.2, and they further facilitate the understanding of the meaning of a specification, we will use them to define the patterns and anti-patterns investigated for this project, which can be mapped to real-time temporal logic, in form of MTL formulas. In rare cases, it may happen that the MTL templates are not expressive enough, and then new MTL formulas have to be defined. When defining patterns, the symbols stand for predicates that are defined over the state vector, as mentioned before.

**Generic example**
We distinguish patterns and anti-patterns as such: In a pattern, a group of activities are executed to achieve a specific user goal. E.g. when enabling X-ray and then pressing the exposure pedal (=group of activities), the radiation will be on (=goal). In an anti-pattern, the goal is not met, and the reason might be the order of the activities or missing/new activities. E.g. when pressing the exposure pedal and then enabling X-ray, the radiation will not be on. Next, we formalize one anti-pattern, as a generic example. First, we define the artifacts, as shown in table 3.3, with each having a set of possible states.

<table>
<thead>
<tr>
<th>Artifact</th>
<th>Set of possible states</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>{B, C}</td>
</tr>
<tr>
<td>A2</td>
<td>{L, M, N, O, P, Q}</td>
</tr>
<tr>
<td>A3</td>
<td>{X, Y, Z, W}</td>
</tr>
</tbody>
</table>

Table 3.1: Overview of the artifacts and the set of possible states for each artifact; generic example.

Figure 3.1 shows an overview on all the artifacts. Each artifact model contains all the possible states and transitions, which have labels for the events that trigger the state change. As the figure shows, it is possible that the same state change is triggered by multiple events (events e1, e2 for artifact A1), or that the same event triggers multiple state changes (event e0 for artifacts A1 and A3).

One example for state vector S could be $S = (A1.B, A2.O, A3.W)$, and it would satisfy multiple predicates, such as $P_1 = A1.B \land A2.O$, $P_2 = A3.W$. To understand better how patterns and anti-patterns are formalized, we take an example. We define the mapping to real-time temporal logic by using the templates introduced in section 2.3.2, which can be converted to MTL formulas:

**Example Anti-pattern**

**Name:** Anti-pattern 1

**Intent:** Looking for instances of “undesired” behaviour, where artifact A1 is in state B and A2 in state O or P, which leads to A3 being in state W within 5 seconds.
CHAPTER 3. PATTERNS AND ANTI-PATTERNS

Figure 3.1: Overview of all the artifacts modelled for the generic example.

Real-time temporal logic mapping: \((l_1 \land l_2) \leadsto \text{first } l_3 \text{ within } [0, 5]\)

Predicates:
\[
\begin{align*}
l_1 &= A1.B \\
l_2 &= A2.O \lor A2.P \\
l_3 &= A3.W
\end{align*}
\]

MTL Template name: Response 1
Examples of known uses: -
Relationships to other patterns: -

Example Pattern

Name: Pattern 1
Intent: Looking for instances of “correct” behaviour, where artifact A1 is in state C and A2 in state O or P, which leads to A3 being in state X within 10 seconds.

Real-time temporal logic mapping: \((l_1 \land l_2) \leadsto \text{first } l_3 \text{ within } [0, 10]\)

Predicates:
\[
\begin{align*}
l_1 &= A1.C \\
l_2 &= A2.O \lor A2.P \\
l_3 &= A3.X
\end{align*}
\]

MTL Template name: Response 1
Examples of known uses: -
Relationships to other patterns: -

We can see from the generic example that there is no difference when specifying patterns and anti-patterns. The core difference between them stems from a conceptual point of view, since anti-patterns define behaviour which is considered to be undesired, while patterns are the exact opposite. This is translated when specifying them, by having a different set of conditions and predicates.

3.2 Defining responses for anti-pattern occurrences

Next, we focus on investigating what was the response of the user immediately after the anti-pattern occurred. To define them, domain knowledge is used, as there is one ideal scenario where the preemptive functionalities set in place (if any) are effective and efficiently warn the user. The responses are defined using the same specification templates as the anti-patterns.
CHAPTER 3. PATTERNS AND ANTI-PATTERNS

After an anti-pattern occurrence is found, it is interesting to see what are the activities that follow it and whether they can lead to a critical situation. Besides, there are risk control measures set in place in order to prevent from critical situations. Hence, it is desirable to check whether these control measures are actually effective. We define three categories of responses:

1. **Positive**: when the risk control measures are effective, and the user realizes his mistake. This can imply not repeating the mistake, or taking a specific action.

2. **Negative**: when the risk control measures seem ineffective, and the user persists in making the mistake.

3. **Miscellaneous**: cannot be categorized as positive or negative, could be regarded for further analysis if considered. This category of responses is for “surprising” behaviour, which may or may not be undesirable, or could be a valuable work-around for a problem that the designers have not yet identified.

We keep track of the event that enabled the anti-pattern detection, which leads to a separate state we denote *Detected* and is the starting point for the response. It marks the moment the anti-pattern specification becomes true. As with the anti-patterns, we define responses for a generic example.

**Generic example**

Assume there is a risk control measure set in place when the hypothetical anti-pattern defined in 3.1 happens. We need to check if the measure is effective (hence, the user has a positive response).

**Positive response:**

Assume the cause for the anti-pattern was the fact that $A_1$ was in state $B$. A positive response would be if the artifact $A_1$ changes to state $C$ within 10 seconds after anti-pattern was detected.

**Real-time temporal logic mapping:** Detected leadsto first $l_4$ within $[0, 10]$ 

**Predicates:**

$l_4 = A_1.C$

**MTL Template name:** Response 1

**Negative response:**

A negative response is when the user repeats the mistake, in our running example it would mean that $A_1$ does not change to state $C$ and $A_2$ is again in state $O$ or $P$.

**Real-time temporal logic mapping:** Detected leadsto first $l_5$

**Predicates:**

$l_5 = \neg A_1.C \land (A_2.O \lor A_2.P)$

**MTL Template name:** Response 1

**Miscellaneous:**

The user does not repeat the mistake, nor does he repair it. It could be something unexpected, with the user taking a completely different action.

3.3 Detecting patterns, anti-patterns and responses in the data

After specifying the patterns, anti-patterns and responses, the next step is the search for them in the available data. The next two sections will discuss this topic, with the first one introducing verification methods already defined in related work, and the second one will introduce the basis of the framework that was built for this project.
CHAPTER 3. PATTERNS AND ANTI-PATTERNS

3.3.1 Related work

One of the goals of this project was to show that the specifications of the patterns and anti-patterns can be detected. Since “real-time” specifications templates, or patterns, defined in [4] were used for our formalization, they can be validated by using the integrated model checking tool chain mentioned in the paper. The proposed verification approach is based on a set of observers in order to reduce the verification of timed patterns to the verification of LTL formulas[4]. Furthermore, the templates were designed so that checking whether a system satisfies a requirement is a decidable problem. In order to validate the patterns and anti-patterns, one could use the verification framework that has been integrated into a prototype extension of frac, the Fiacre compiler for the TINA toolbox. Fiacre [7] is a formal modelling language for real-time systems, that was extended to support the declaration of real-time patterns. The verification tool chain can be used in conjunction with Topcased [15], an Eclipse based toolkit for system engineering. More information on the software is available at http://projects.laas.fr/tina/ and http://projects.laas.fr/fiacre/. For a more detailed and comprehensive account of the verification framework please refer to [6] and [5].

The systems are modeled using an extension of Time Petri Nets with data variables and priorities, called a TTS (Time Transition System). The behavior of a TTS is defined by the set of all its (timed) traces. Since we analyze finite traces from iXR machine cases and they are timed, these can be translated into TTSs. The only challenge would be the fact that more Time Transition Systems have to be defined, one for each artifact.

Extra verification algorithms

Runtime verification and monitoring have been proposed as lightweight formal verification methods with the explicit goal of checking systems against their formal requirements while they execute. Next to the model checking tool explained previously, there are two more alternative approaches that could be used:

1. Paper [32] investigates monitoring algorithms for MTL without storing execution traces, but rather consuming the events as they are received from the monitored program. In the paper, a monitoring algorithm is presented and based on the idea of transforming the MTL formula as each time-stamped observation, or event, is received from the monitored program: The underlying principle of the algorithm is “resolve the past and derive the future”. By “resolving the past” we mean that the MTL formula is transformed into an equivalent formula with the property that it has no past time operator rooted subformulae which are not guarded by other temporal operators. By “deriving the future” we mean that the MTL formula is transformed into a new MTL formula with the property that the current formula holds before processing the newly received event if and only if the derived formula holds after processing the event.

2. A more recent paper, [14], proposes multi-valued semantics for MTL formulas, which capture not only the usual Boolean satisfiability of the formula, but also topological information regarding the distance from unsatisfiability. This paper introduces both continuous and discrete time semantics of MTL, and presents a monitoring algorithms, similar to [32], that is based on the discrete-time robust semantics of MTL:

We introduce and use timed state sequences (TSSs) as models for the discrete-time representation of signals that also maintain the required timing information. TSSs are a widely accepted model for reasoning about real-time systems. We proceed on to define MTL semantics over timed state sequences. The semantics is defined using a valuation function, which checks whether the TSS satisfies the MTL formula at moment i.

The presented approaches can be used to validate the MTL formulas for the patterns and anti-patterns, as the representation of the behavior of the iXR systems is available to us for analysis in discrete-time. For a more detailed and comprehensive account of the algorithms please refer to...
3.3.2 Detection algorithm

Model checkers only return True/False results. For the case-study that was performed for the project this is not enough, since we want to retrieve and further analyse all instances, and part of the traces where those instances occur. We are dealing with a concrete system, Azurion, so we want to focus on finding the anti-patterns previously defined. MTL is needed to specify formally the patterns and anti-patterns, but Python code is used to mine the available data and instantiate the formulas. To make the mapping between the MTL formulas used in the templates and the code used to search for patterns or anti-patterns more general and easier to reuse in other circumstances, a generic approach is provided in this section.

Algorithm 1 Generic anti-pattern (or pattern) detection

INPUT:
Finite trace of state vectors \( t = <s_1, ..., s_n> \)
Time windows before and after, non-negative numbers
Anti-pattern (or pattern) \( AP \)

OUTPUT:
Set \( T \) of subtraces of \( t \) such that for every \( t' \in T \), \( t' = <s_j, ..., s_k, ...s_l> \) it holds that:
\( AP \) holds for \( <s_j, ..., s_k> \) and there is no smaller subtrace of \( <s_j, ..., s_k> \) for which \( AP \) holds and \( <s_i, ..., s_{j-1}> \) are the state vectors in the time window before \( s_j \) and \( <s_{k+1}, ..., s_l> \) are the state vectors in the time window after \( s_k \)

\( T = \emptyset \)

Step 1: Apply detection strategy (depends on the MTL template) and get the set \( \text{AntipatternStartEndIndexes} \) of pair of indexes for the start and the end of each anti-pattern detection

Step 2: Retrieval of neighboring state vectors for every anti-pattern occurrence

for \( w = 1 \) to \( |\text{AntipatternStartEndIndexes}| \) do
    \((j, k) = \text{AntipatternStartEndIndexes}_w\)
    for \( i = 1 \) to \( j \) do
        if \((i == 1 \text{ or } \text{timestamp}(s_j) - \text{timestamp}(s_{i-1}) > \text{before}) \text{ and } (\text{timestamp}(s_j) - \text{timestamp}(s_i) \leq \text{before})\) then
            startIndex = \( i \)
        for \( l = k \) to \( n \) do
            if \((l == n \text{ or } \text{timestamp}(s_{l+1}) - \text{timestamp}(s_k) > \text{after}) \text{ and } (\text{timestamp}(s_l) - \text{timestamp}(s_{k+1}) \leq \text{after})\) then
                endIndex = \( l \)
            Add \( <s_{\text{startIndex}}, ..., s_{\text{endIndex}}> \) to \( T \)
        return \( T \)

After specifying the patterns and anti-patterns, the next step is to mine the available machine data to find occurrences. Each pattern and anti-pattern will be checked per finite traces of state vectors. Each state vector corresponds to one event. Building the trace of state vectors is a pre-processing step required, before detecting anti-patterns or patterns. When an anti-pattern instance is found in a trace, we retrieve and store the subtrace containing the state vectors for which the anti-pattern holds, together with neighboring state vectors within a certain time window of the moment where the anti-pattern occurs. A post-processing step is needed to retrieve the
corresponding events for the subtrace of state vectors, and store them in a separate log file, such that it can be further analysed. There will be one log per anti-pattern instance found. For this project, the set time window for retrieving neighboring events was of 5 minutes.

Algorithm 1 defines the main logic behind the detection of an anti-pattern (or pattern) specification, and the retrieval of neighboring state vectors based on a set time window before and after the detection. Function \( \text{timestamp}(s_i) \) returns the timestamp of the event corresponding to state vector \( s_i \). Depending on the anti-pattern, and the underlying MTL template, a different strategy will be needed. In this section we present the detection strategy for one of the MTL templates used for the case study, Existence 2, implemented in algorithm 2. Here, the first time predicate \( A \) holds is between \( d_1 \) and \( d_2 \) units of time before the first time predicate \( B \) holds. Furthermore, at the time \( A \) holds, predicate \( B \) must not hold. All other MTL templates can be implemented similarly. Another example is given in Appendix F, where the detection strategy for MTL template Response 1 is shown. When defining patterns, the symbols \( A, B, \ldots \) stand for predicates over the state vector. Function \( \text{checkPredicate}(P, s_i) \) checks whether predicate \( P \) holds on state vector \( s_i \).

**Algorithm 2** Detection Strategy for MTL template Existence 2

**INPUT:** Anti-pattern (or pattern) Present \( A \) before \( B \) within \( I = [d_1, d_2] \)

**OUTPUT:** Set \( \text{AntipatternStartEndIndexes} \) of pairs \((x, y)\) of indexes for the start and the end of each anti-pattern detection

\[
\text{AntipatternStartEndIndexes} = \emptyset \\
\text{found}_A = \text{False} \\
x = 1 \\
\text{while } x \leq n \text{ and } \text{found}_A == \text{False} \text{ do} \\
\quad \text{if } \text{checkPredicate}(A, s_x) == \text{True} \text{ and } \text{checkPredicate}(B, s_x) == \text{False} \text{ then} \\
\quad \quad \text{found}_A = \text{True} \\
\quad \quad \text{time}_A = \text{timestamp}(s_x) \\
\quad \quad \text{found}_B = \text{False} \\
\quad \quad y = x + 1 \\
\quad \text{while } y \leq n \text{ and } \text{found}_B == \text{False} \text{ do} \\
\quad \quad \text{if } \text{checkPredicate}(B, s_y) == \text{True} \text{ then} \\
\quad \quad \quad \text{time}_B = \text{timestamp}(s_y) \\
\quad \quad \quad u = \text{time}_B - \text{time}_A \\
\quad \quad \quad \text{if } (d_1 \leq u \leq d_2) \text{ then} \\
\quad \quad \quad \quad \text{found}_B = \text{True} \\
\quad \quad \quad \quad \text{add } (x, y) \text{ to } \text{AntipatternStartEndIndexes} \\
\quad \quad \quad y = y + 1 \\
\quad \quad x = x + 1 \\
\quad \quad \text{return } \text{AntipatternStartEndIndexes}
\]

The Python code for every pattern and anti-pattern specification is tested by using artificially created event logs as shown in figure 3.2. Here we see that the artificial log has 3 fields: timestamp, activity name, and additional information. The events here show the anti-pattern where during an exam, the pedal is pressed while X-ray is disabled, leading to a user guidance message. When the anti-pattern is detected, a new trace log will be generated, as shown in figure 3.3. Here, we also see 2 additional fields: the case id, which is irrelevant for now, and \textit{pattern}, which marks the events which lead to the anti-pattern detection. We see that only events in a specific time-span of 5 minutes before and after the detection are logged.
CHAPTER 3. PATTERNS AND ANTI-PATTERNS

3.4 Case study 1: Investigating data from Philips iXR machines

This section will describe part of the case-study that was performed for the project. We begin by introducing the available data, followed by examples on what patterns and anti-patterns were defined. Then, an example for specifying responses is given, ending with an overview on the results that were obtained.

3.4.1 Available data

The data logged by the iXR machines is uploaded to a Philips server on a daily basis and stored in a relational database. There are numerous events that are being logged, however, the most important categories for this project are:

1. User commands: there are several types of command a user can initiate, but the most important can involve enabling and disabling the X-Ray, starting and stopping an acquisition. There are two main types of acquisitions: exposure and fluoroscopy. Fluoroscopy uses the least radiation and is usually used to guide wires and catheters through the body. Exposure provides the best images, but also uses the most radiation. Depending on the procedure, different built-in protocols can be selected, which can vary the dose and area of radiation.

2. User guidance messages: messages that appear on the User Interface to help guide or inform the user.

3. Start of a lab exam: Marking the beginning of a procedure, starting from the moment the first acquisition is initiated.

4. End of a lab exam: Marking the end of a procedure, which happens when the user initiates the end or suspends the current examination.

5. Reporting different metrics after every acquisition: These metrics include the protocol that was selected, the intensity of the radiation, total exposure time, total number of images taken and many others.

6. Movement related events: information regarding the positions and geometry values of the machine.

There are hundreds, if not thousands, event types that are being logged which makes the data very fine-grained and difficult to interpret for process mining purposes. That is why anti-patterns are first defined and then recreated on a test machine. This enables a better correlation between the high-level activities that are performed by a user of the system and the low-level events that are being logged by the machine, and narrows down the search.
3.4.2 Pattern and anti-pattern specification

Based on customer complaints and hypothetical hazardous situations, six anti-patterns and six corresponding patterns were defined. Each was first defined as a list of tasks, which was recreated using a test machine. An example of such a list can be found in Appendix A. After recreating them, the tasks were correlated to logged events, and then formally defined. The patterns and anti-patterns defined for this project can be found in Appendix B.

The artifacts defined for the case study are shown in Table 3.2. Each artifact has a set of possible states. XRayStatus refers to the X-ray being enabled or disabled, ExamStatus refers to the status of the exam, PedalStatus refers to the status of the pedals, UserGuidanceStatus refers to the messages that are being shown to the user, RadiationStatus refers to the radiation being active, ConfiguredPedalStatus refers to the status of a special pedal, that is configured only for some machines, RestartStatus shows whether the system is restarted, GeometryStatus refers to locking geometry such that the table cannot be moved anymore, TableMovementStatus refers to the table movements.

<table>
<thead>
<tr>
<th>Artifact</th>
<th>Set of possible states</th>
</tr>
</thead>
<tbody>
<tr>
<td>XRayStatus</td>
<td>{Enabled, Disabled}</td>
</tr>
<tr>
<td>ExamStatus</td>
<td>{ExamStarted, Idle}</td>
</tr>
<tr>
<td>PedalStatus</td>
<td>{FluoroPressed, ExposurePressed, BothPressed, None}</td>
</tr>
<tr>
<td>UserGuidanceStatus</td>
<td>{XRayDisabled, DoorOpened, GeoLocked, None}</td>
</tr>
<tr>
<td>RadiationStatus</td>
<td>{On, Off}</td>
</tr>
<tr>
<td>ConfiguredPedalStatus</td>
<td>{None, SingleShotPressed}</td>
</tr>
<tr>
<td>RestartStatus</td>
<td>{None, ColdRestart, WarmRestart}</td>
</tr>
<tr>
<td>GeometryStatus</td>
<td>{Locked, Unlocked}</td>
</tr>
<tr>
<td>TableMovementStatus</td>
<td>{None, ChangeHeight, Cradle, Tilt, Executed}</td>
</tr>
</tbody>
</table>

Table 3.2: Overview of the artifacts and the set of possible states for each artifact, case-study 1.

Figure 3.4 shows an overview on all the artifacts defined so far. Each artifact model contains all the possible states and transitions, which have labels for the events that trigger the state change.

We present one running example from the case study performed for this project, focusing on the anti-pattern where exposure or fluoroscopy pedal is pressed while X-ray is disabled, leading to no image shown on the monitor, which would be the expected user goal. The safety-related risk here is that the user is unaware that X-ray is disabled and that he is looking at a recorded image instead of live image. There are one pattern and one anti-pattern defined:

**Pattern**

**Name:** “Positive Press pedal while X-ray enabled”

**Intent:** Looking for instances of “correct” behaviour, where one pedal (exposure or fluoroscopy) is pressed when X-ray is enabled and during a lab exam. The expected consequence is a state change of the artifact RadiationStatus, meaning that the radiation will be turned on.

**Real-time temporal logic mapping:**

\((\text{EnabledExamStarted} \land \text{PedalPress}) \implies \text{first RadiationOn within } [0, 5])

**Predicates:**

\[
\begin{align*}
\text{EnabledExamStarted} &= \text{XRayStatus.\text{Enabled}} \land \text{ExamStatus.\text{ExamStarted}} \\
\text{PedalPress} &= \text{PedalStatus.\text{ fluoroscopyPedal}} \lor \text{PedalStatus.\text{exposurePedal}} \\
\text{RadiationOn} &= \text{RadiationStatus.\text{On}}
\end{align*}
\]

**MTL Template name:** Response 1

**Examples of known uses:** regular, intended use

**Relationships to other patterns:** related to “Accidental X-ray” anti-pattern
CHAPTER 3. PATTERNS AND ANTI-PATTERNS

Anti-pattern

Name: “Negative Press pedal while X-ray disabled”

Intent: Looking for instances of “undesired” behaviour, where one pedal (exposure or fluoroscopy) is pressed when X-ray is disabled and during a lab exam. The expected consequence is a state change of the artifact UserGuidanceStatus, meaning that the radiation will remain turned off, and that the user will get a message informing him of the fact that X-ray is disabled.

Real-time temporal logic mapping:
\((\text{DisabledExamStarted} \land \text{PedalPress}) \leadsto \text{first} (\text{UserMsgDisabled} \land \text{RadiationOff}) \text{ within } [0, 5]\)

Predicates:
\begin{align*}
\text{DisabledExamStarted} &= \text{XRayStatus.Disabled} \land \text{ExamStatus.ExamStarted} \\
\text{PedalPress} &= \text{PedalStatus.FluoroPressed} \lor \text{PedalStatus.ExposurePressed} \\
\text{UserMsgDisabled} &= \text{UserGuidanceStatus.XRayDisabled} \lor \text{UserGuidanceStatus.DoorOpened} \\
\text{RadiationOff} &= \text{RadiationStatus.Off}
\end{align*}

MTL Template name: Response 1

Examples of known uses: could be user error, or intended to check whether X-ray is disabled

Relationships to other patterns: related to “Accidental X-ray” pattern

3.4.3 Responses specification

We define responses for the anti-pattern defined in 3.4.2 for the case-study. The others can be found in Appendix C. The definitions are based on domain knowledge, and depend on the anti-pattern. The risk control measure set in place when a pedal is pressed while X-ray is disabled, is to show a user guidance message. We need to check if this message is effective and the user enables X-ray.

Positive response:

The user enables the radiation within 5 seconds after the user guidance message is shown, stating that the X-ray is disabled.

Real-time temporal logic mapping: Detected \(\leadsto\) first Enabled within \([0, 5]\)

Predicates:
\(\text{Enabled} = \text{XRayStatus.Enabled}\)

MTL Template name: Response 1

Negative response:

The user continues to try and presses the pedal, failing to realize that radiation is disabled. We must check that the lab exam does not end. As soon as it ends, the formula will not hold anymore.

Real-time temporal logic mapping: Detected \(\leadsto\) first (DisabledExamStarted\land\text{PedalPress})

Predicates:
\begin{align*}
\text{DisabledExamStarted} &= \text{XRayStatus.Disabled} \land \neg \text{ExamStatus.Idle} \\
\text{PedalPress} &= \text{PedalStatus.FluoroPressed} \lor \text{PedalStatus.ExposurePressed}
\end{align*}

MTL Template name: Response 1

Miscellaneous:

Other: one example could be ending a lab exam - to double check that X-Ray is indeed disabled, or performing other type of activities after the pedal press.
3.4.4 Results

After the anti-patterns are formalized as shown in section 3.4.2, code is instantiated for the templates based on the algorithm from section 3.3.2 to mine the available data for an initial set of 14 Azurion machines, for the year 2017. Then, the same is done to mine and categorize the anti-pattern responses, as defined in section 3.2. Overall, events from 16230 exams are retrieved and analyzed, covering approximately 9964 hours of machine working time. Working time is defined as the time a machine is used, hence the time during an exam, from the first image taken up to the last one. The results can be divided into two main categories:

1. **A few occurrences found:** for some anti-patterns, there are only a few (between 10 and 150) instances found. Out of those instances, only a fraction result in a negative response, which is the one that we put most emphasis on. This amount of data is insufficient for further analysis, such as mining for process models or mining to find correlations and trends. Hence, each instance will be investigated individually to find possible explanations and causes.

2. **Many occurrences found:** for other anti-patterns, there are a lot more ($\geq 3000$) instances found. This data cannot be investigated individually and is sufficient for further analysis and different strategies are applied and will be explained in the chapters 4 and 5.

![Figure 3.5](image1)

**Figure 3.5:** Overview results for anti-pattern where fluoroscopy or exposure pedal is pressed while X-ray is disabled; many occurrences.

![Figure 3.6](image2)

**Figure 3.6:** Overview results for an anti-pattern that lead to finding only a few occurrences.

The results for the anti-pattern presented in the running example (see figure 3.5), where X-ray is enabled and the user requests fluoroscopy or exposure by pressing the pedal, fall in the second category, and will be used as input for the analysis done in the next chapters.

Other anti-patterns that are defined in Appendix B have results that fall in the first category. One example is shown in figure 3.6, without stating what the anti-pattern and the responses are, due to confidentiality requirements. However, we can see that there are far less instances found, and to understand more about the individual situations that lead to this cases, more actions are taken to talk to users or system experts and enable root cause analysis. This is described in section 5.4.
CHAPTER 3. PATTERNS AND ANTI-PATTERNS

Figure 3.4: Overview of all the artifacts modelled for the case-study 1.
3.5 Case study 2: Investigating data from Philips smart baby bottles

To show that this approach and way of formalizing anti-patterns can be used for other datasets, we take another case study, presented in [35] and [36], where data is retrieved during a study where Philips worked on the design of a smart baby bottle equipped with various sensors. There were 9 families participating in the study, who were each provided with a prototype of the smart bottle for a 3 week period. 358 instances of baby feedings are contained in the sensor data gathered during the study. There are two main perspectives defined in [35]: a temperature sensor and an accelerometer measuring bottle movement, which will be used as artifacts in this thesis as well. The states of the artifacts correspond to the state of the sensor signals of these two sensors and their product-specific interpretation. They were obtained by clustering the sensor measurement values and labelling the cluster centroids [35].

Let us define a set of possible states for each of the two artifacts:

<table>
<thead>
<tr>
<th>Artifact</th>
<th>Set of possible states</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature</td>
<td>{Critical, WarmConstant, WarmIncreasing, WarmDecreasing, ColdConstant, ColdIncreasing, ColdDecreasing}</td>
</tr>
<tr>
<td>Accelerometer</td>
<td>{UprightBigMove, DownwardBigMove, UprightNotMoving, DownwardNotMoving}</td>
</tr>
</tbody>
</table>

Table 3.3: Overview of the artifacts and the set of possible states for each artifact; case-study 2.

After defining the artifacts and their corresponding possible states, patterns and anti-patterns can be defined, and two examples are provided:

1. *Anti-pattern* where the bottle is getting critically warm, and this is a safety-risk that can potentially lead to burns:
   \((\text{Temperature.WarmConstant} \land \text{Temperature.WarmIncreasing}) \leadsto \text{first} (\text{Temperature.Critical} \land \neg (\text{Temperature.WarmDecreasing} \lor \text{Temperature.WarmConstant}))\)

   Then, the *positive response* would be to observe that the bottle is too warm and to decrease its temperature within 1 minute:
   \(\text{Detected} \leadsto \text{first} (\text{Temperature.WarmDecreasing} \lor \text{Temperature.WarmConstant}) \text{ within } [0, 60]\)

   The *negative response* would be to feed the baby anyway, leading to serious harm. We assume the baby is fed when the bottle is hold downwards and not moving:
   \(\text{Detected} \leadsto \text{first} \text{Accelerometer.DownwardNotMoving for } 60 \text{ before} (\text{Temperature.WarmDecreasing} \lor \text{Temperature.WarmConstant})\)

2. *Anti-pattern* where the bottle is getting cold while feeding the baby:
   \((\text{Temperature.WarmDecreasing} \land \text{Accelerometer.DownwardNotMoving}) \leadsto \text{first} (\neg \text{Accelerometer.UprightNotMoving} \land (\text{Temperature.ColdDecreasing} \lor \text{Temperature.ColdConstant}))\)

   Then, the *positive response* would be to observe that the bottle is cold and to increase its temperature within 1 minute:
   \(\text{Detected} \leadsto \text{first} (\text{Temperature.WarmIncreasing} \lor \text{Temperature.WarmConstant}) \text{ within } [0, 60]\)

   The *negative response* would be to continue to feed the baby anyway, leading to misusing the bottle. We assume that if the bottle is not turned upright for 5 minutes, it means the baby was fed continuously:
   \(\text{absent} (\text{Accelerometer.UprightNotMoving} \land (\text{Temperature.WarmIncreasing} \lor \text{Temperature.WarmConstant})) \text{ after} \text{Detected for } 300\)
Similarly, other properties can be defined by using the approach presented in this chapter for different kind of datasets, as long as states and artifacts can be defined based on the data. This type of formalizing patterns can be used for numerous case-studies, and can be used to define much more complex usage patterns. They are not limited by the formulas presented in this thesis, as you can extend the patterns to also include calculations on the event data, by enriching the artifacts and their states. For this project, we proceeded with further investigation only for Case study 1, due to time limitations. This will be discussed in the next two chapters.
Chapter 4

Data reduction and mining for Process Models

Mining for process models for responses using the original trace logs is not useful since the models would be too complex to understand. The original logs contain large amounts of freedom and variety in behaviour, combined with a very large number of activities, which leads to very complex models as shown in figure 4.1. That is why we came up with several strategies to help us reduce the logs by focusing on relevant data. These will be discussed next, followed by an overview of process mining techniques and results. But first we describe how the data was prepared to serve as input for process mining.

4.1 Preparation for event data

Before the data could be used as input for mining for process models, it had to be converted into event data and cleaned. The following two sections discuss these steps.

Mapping relational data to event data

The traces retrieved for each anti-pattern occurrence come from database queries, hence they are relational data. Process mining algorithms require event data as input. In event data, each row represents an event that refers to only one case, a single instance of a process.

As mentioned in chapter 2, the minimal requirements for process mining are that any event can be related to both a case and an activity and that events within a case are ordered. Each event should contain the activity name, a timestamp, and a case identifier, as presented in table 4.1.
**CHAPTER 4. DATA REDUCTION AND MINING FOR PROCESS MODELS**

<table>
<thead>
<tr>
<th>Event data column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CaseID</td>
<td>artificially build it</td>
</tr>
<tr>
<td>Concept:Name</td>
<td>name of the activity</td>
</tr>
<tr>
<td>Time</td>
<td>timestamp of the event</td>
</tr>
</tbody>
</table>

Table 4.1: Mapping relational data to event data by columns

The events that are retrieved from the database per day and per machine will be sorted based on their timestamps.

We need to define what a case, or process instance, is. The first approach was to define a case to be a lab exam, or procedure. Two types of cases were defined: during lab exam (D_LE), and in between lab exams (B_LE). The case ID was assigned based on three events:

- **Lab Exam Started**: every time this event is found the counter for D_LE is increased and current event and all events following it are assigned this case id. The procedure key found in the additional information of this event will be stored such that the matching end can be found later on.

- **Lab Exam Ended**: every time this event is found and the procedure key in the additional information matches the one found for the Lab Exam Start, the counter for B_LE is increased and all events following it are assigned this case id.

- **Lab Exam Suspended**: same as Lab Exam Ended.

However, this approach was only used for the initial trace logs as they were retrieved from the database per day and per machine. It turned out to be challenging because doctors can have two exams opened at the same time, and the start of a lab exam is logged only when the first acquisition takes place. Hence, there can be cases in which the lab end is logged without having a corresponding lab start; e.g. a patient is added, an examination is (re)selected, but no acquisition was made yet. Furthermore, when detecting an anti-pattern occurrence, we retrieve events in a time-span before and after. If in one trace log there are \( x \) anti-pattern instances, then \( x \) different trace logs will be created. Hence, some events may be duplicated, and the case ID based on the exam is not unique anymore. See figure 4.2 for such an example. Therefore, a second approach was chosen when retrieving and storing traces when an anti-pattern is detected, and it defines a case to be an anti-pattern instance, or occurrence. The CaseID is an unique combination of anti-pattern name, date, machine identifier and occurrence number for that date/machine. For example, suppose we investigate anti-pattern \( A \), for date 2017-01-01, machine \( B \), and we find 3 occurrences. Then, there will be 3 different cases with 3 different IDs:

- antipatternA_2017-01-01_machineB_1
- antipatternA_2017-01-01_machineB_2
- antipatternA_2017-01-01_machineB_3

Lastly, in order to make process mining easier and faster, several data cleaning steps were performed:

1. Remove time and identifiers from event names - to significantly reduce the number of activities.
2. Remove multi-line events - can lead to issues later on when loading and analyzing the log.
3. Remove events with empty name - they are not relevant in the process model, since we do not know what they stand for (can be a result of the previous point).
4. Shorten long event names - to prevent the mining algorithms from breaking.
4.2 Data reduction strategies

Due to time constraints, it was decided to focus only on analyzing the user responses after an anti-pattern occurs. The logs were clustered per response type and further investigated such that a smaller subset of relevant activities can be obtained for each category. In order to come up with a reduced set of activities for each anti-pattern, and for each response category, four different strategies were used:

1. Defining system variables, and keeping only those activities that are correlated to variables that are influenced for a specific anti-pattern.
2. Using direct entropy-based activity filtering.
3. Computing relative frequency per response category for each activity and each anti-pattern. In this way, we can immediately see whether an activity tends to appear a lot more often in one category compared to the others.
4. Defining three more measures: support, confidence and lift to help choose the most representative set of activities for each response category, for each anti-pattern.

Each strategy will be elaborated in the following sections.

4.2.1 Mapping activities to system variables

For the iXR systems, several system variables can be defined, such as Radiation, Movement, Dose, Intensity. The approach then is to reduce the number of activities such that we only keep those who influence a relevant variable. In other words, for every anti-pattern, only the activities that influence the state of the system such that the goal can be met are kept.

The mapping between variables and activities are defined at a conceptual level, as shown in figure 4.3. Domain knowledge is used to derive the mapping, without implementing a general method that would do it, due to the time limitation of this project.

Conceptually, we define several steps for this approach:

1. Create a list of variables that are relevant for the anti-pattern and response category under investigation.
2. For each variable define a list of activities that are relevant.
3. Filter event logs, by removing all other events that are not in the list previously defined.

Let us take an example, an anti-pattern where an acquisition is requested although X-Ray is disabled. The only variable that is connected to the goal is Radiation. The activities that influence it are:
CHAPTER 4. DATA REDUCTION AND MINING FOR PROCESS MODELS

Figure 4.3: Mapping between anti-patterns, MTL templates, activities and system variables.

- Activities related to pressing exposure or fluoroscopy pedal (requesting an acquisition).
- Activities related to enabling or disabling X-ray.
- Activities related to starting radiation.
- User guidance messages related to the inability to perform X-ray.

After trying the previously mentioned approach, we observed that it can happen that this type of mapping cannot be differentiated between response categories, since most of the mappings could be very similar. Furthermore, not many variables can be defined since most of them are out of the scope of logging (e.g., knowing whether a patient is on the table). Lastly, by only using these mappings, important events might be missed and the obtained process models could be misleading. This strategy led to no useful results for the case-study.

4.2.2 Direct entropy-based activity filtering

A different approach is mentioned in [31], where event mappings are used to distinguish between events that are part of the mainstream behavior of a process and outlier events: Event mappings compute similar behavior and dissimilar behavior between each two executions of the process as a mapping. Paper [31] considers a chaotic activity to be an activity that can occur at any point in the process and that complicates the discovery of the rest of the process by obfuscating the directly-follows relations of the event log. It proposes a technique to detect chaotic activities in event logs and to filter them, called Direct Entropy-based Activity Filtering. The entropy of an activity is calculated based on its directly-follows ratio vector and the directly-precedes ratio vector by using the usual definition of function for the categorical probability distribution.

The authors of the paper implemented this filtering approach in ProM [2], which outputs several log files, by removing one activity at a time. For example, if you provide as input an event log with 100 activities, the output will consist of 99 events logs: one with 99 activities, one with 98, until we reach one with 1 activity. To depict how this approach was used, we take the running example defined in section 3.4.2, where X-ray is enabled and the user requests fluoroscopy or...
exposure by pressing the pedal, and we investigate the miscellaneous responses, where the user
does not enable X-ray immediately but also does not repeat the same request. Table 4.2 depicts
some of the statistics of the event log before and after filtering, taking the event logs with 15
and with 100 activities left after filtering. We see that the number of events and activities are
significantly reduced. The filtered logs will be used as input for process mining and the results
will be discussed in section 4.3.

<table>
<thead>
<tr>
<th></th>
<th>Cases</th>
<th>Events</th>
<th>Event classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before filter</td>
<td>1177</td>
<td>40872</td>
<td>471</td>
</tr>
<tr>
<td>After filter keeping 15 act.</td>
<td>1175</td>
<td>12620</td>
<td>15</td>
</tr>
<tr>
<td>After filter keeping 100 act.</td>
<td>1176</td>
<td>21764</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 4.2: Overview of the number of cases, events and activities left before and after filtering
based on entropy; pedal press while X-ray disabled, miscellaneous responses.

### 4.2.3 Relative frequency of activities

This approach aims at defining a smaller subset of activities that are more representative for
each response category, for each anti-pattern. Therefore, for each activity its relative frequency
is computed. This frequency is split per the three response categories, and sums up to 1. The
following steps were performed, for each anti-pattern:

1. Create a list of activities based on the previously obtained traces.

2. For each activity count its occurrences after the pattern is detected per response category:
   $Count_P, Count_N, Count_M$.

3. Get the total number of traces per response category: $Traces_P, Traces_N, Traces_M$.

4. For each activity, compute the average number of occurrences per response type: $Avg_P = Count_P/Traces_P$.

5. For each activity, compute relative frequency per response type: $RelFreq_P = Avg_P/(Avg_P + Avg_N + Avg_M)$.

6. Make a plot to have an overview of the results.

Figure 4.4: Example of relative frequency plot; divided per response category: green-positive, red-
negative, yellow-miscellaneous; all sum up to 1 such that it is clear which activities are correlated
more with one of the response categories.
CHAPTER 4. DATA REDUCTION AND MINING FOR PROCESS MODELS

Figure 4.4 shows an example of a relative frequency plot, where the frequency is calculated and shown per response category and per activity for one anti-pattern. We can clearly see that there are activities that only belong to one response category, such as F, G for positive response. This indicates that these activities only appear in traces categorized as positive, and should be kept when mining for process models.

However, since the number of responses for each anti-pattern is highly unbalanced, with often many more traces for one response category, the results obtained by using this approach were not enough to filter on. Furthermore, now every activity has a frequency that sum up to 1, but there are activities which are very infrequent and this will not be visible in the plot. Hence, a new approach was found, and is explained in the next section.

4.2.4 Support, Confidence, Lift

The results of the previous strategy were not enough to filter on. In order to select interesting rules and do not miss out on infrequent behaviour that might be relevant, various measurements of significance and interest are used. The best-known measurements are minimum thresholds on support, confidence and lift [23].

**Support** indicates how frequently an itemset appears in the dataset. In our case the dataset is the set of responses, and an itemset can either contain activities or response categories. Let us take activity \( A \) as an example. The support of \( A \) is the probability of \( A \) appearing in the responses:

\[
\text{supp}(A) = \frac{\text{number of responses containing } A}{\text{total number of responses}}
\]

**Confidence** measures the strength of a rule or how often the rule is found to be true. Let us take rule \( A \rightarrow Pos \), where \( A \) is an activity and Pos is positive response. Then:

\[
\text{conf}(A \rightarrow Pos) = \frac{\text{number of positive responses containing } A}{\text{total number of positive responses}}
\]

**Lift** is a measure of performance of a rule, and when the value is close to 1 it implies the itemsets are independent of each other. The higher the lift is, the more it shows how dependent on one another the itemsets are. Let us take again rule \( A \rightarrow Pos \). Then:

\[
\text{lift}(A \rightarrow Pos) = \frac{\text{number of positive responses containing } A}{\text{number of positive responses}}
\]

In order to filter and keep only relevant activities for each response category, the following approach was used for every anti-pattern:

*Set a minimum threshold for support, confidence and lift for each category. Next, for each category, construct a set of filtered activities that are above that threshold. Then, filter events that are not in the set from the traces in each corresponding response category.*

Log projection was used to filter events in the trace logs such that they only contain a subset of activities, given as input. The subset of activities are obtained after filtering on support, confidence and lift, for each response type and anti-pattern. We define log projection based on [33]:

**Definition 15.** Sequence projection: Let \( X \) be a set and \( Q \subseteq X \) one of its subsets. \( [Q] \in X^{*} \rightarrow Q^{*} \) is a projection function and is defined recursively: (1) \( [Q = \emptyset] \) and (2) \( \forall \sigma \in X^{*} \) and \( x \in X \):

- \( [\langle x \rangle \cdot \sigma][Q] = \sigma[Q] \), if \( x \notin Q \)
- \( [\langle x \rangle \cdot \sigma][Q] = \langle x \rangle \cdot \sigma[Q] \), if \( x \in Q \)

So, \( \langle y, z, y \rangle \cdot [\langle x, y \rangle] = \langle y, y \rangle \).

The projection function \( [\_] \) is generalized to event logs, i.e. for some event log \( L \in B(A^{*}) \) and set \( X \subseteq A : L|_X = [\sigma \cdot x | \sigma \in L] \), where \( A \) is a set of activities.

To depict how this approach was used, we take the running example defined in section 3.4.2, where X-ray is enabled and the user requests fluoroscopy or exposure by pressing the pedal. The following thresholds are set: \( \text{Confidence} \geq 0.3, \text{Support} \geq 0.2, \text{Lift}_{\text{positive}} \geq 2, \text{Lift}_{\text{negative}} \geq 1.3, \)
Lift_{miscellaneous} \geq 1.1. Different lift thresholds are required due to the imbalance of the data for each response category. After applying the filtering, 15 activities are left for the miscellaneous category, 33 activities for positive category and 3 activities for the negative category. An overview of the results after filtering is shown in table 4.3 and it shows a big reduction compared to the original numbers shown in table 4.4. The filtered logs will be used as input for process mining and the results will be discussed in section 4.3.

<table>
<thead>
<tr>
<th>Response</th>
<th>Cases</th>
<th>Events</th>
<th>Event classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negative</td>
<td>1920</td>
<td>47008</td>
<td>3</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>1171</td>
<td>11970</td>
<td>15</td>
</tr>
<tr>
<td>Positive</td>
<td>608</td>
<td>93739</td>
<td>33</td>
</tr>
</tbody>
</table>

Table 4.3: Overview of the number of cases, events and activities left after filtering based on support, lift and confidence; pedal press while X-ray disabled.

<table>
<thead>
<tr>
<th>Response</th>
<th>Cases</th>
<th>Events</th>
<th>Event classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negative</td>
<td>1942</td>
<td>228804</td>
<td>532</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>1177</td>
<td>40872</td>
<td>471</td>
</tr>
<tr>
<td>Positive</td>
<td>614</td>
<td>121247</td>
<td>506</td>
</tr>
</tbody>
</table>

Table 4.4: Overview of the number of cases, events and activities left before filtering; pedal press while X-ray disabled.

Validating reduction approach

At this point we have a set of events related to anti-patterns and responses that is concise enough for analysis. The strategies presented before are evaluated based on the degree of usefulness of the results, and one is selected to be optimal for the case-study performed within this project. A different case-study can lead to a different optimal strategy, depending on the product analyzed and the nature of the data. Several ProM mining algorithms are used with different approaches, such that the results could be compared. The next section will describe the different mining algorithms that were used, followed by showcasing the most relevant findings together with their limitations.

4.3 Creating process models

After having reduced the event logs by applying entropy-filtering and filtering based on support, lift and confidence, the next step is to mine for process models. Several ProM mining algorithms were used with different approaches, such that the results could be compared, and two of them will be discussed more in detail next.

4.3.1 Inductive Visual miner

After data reduction, we believe we have a smaller, less complex event log with only a subset of relevant activities, which is concise enough to create a single process model. The Inductive Visual Miner (IvM) is described in paper [22], which identifies four aspects that are crucial for process exploration: zoomability, evaluation, semantics, and speed. IvM provides features similar to those of commercial tools and aims to be as user-friendly, while providing maps with semantics. It also provides a user-friendly, easy to follow way to visualize deviations, by immediately discovering an initial model, then computing deviations and showing animation of the traces of the log. IvM uses process trees to ensure sound models. However, the results are shown to the user using a directly-follows based representation, extended with a start and end state, and Petri net places.
CHAPTER 4. DATA REDUCTION AND MINING FOR PROCESS MODELS

to provide semantics. To support parallelism, BPMN parallel gateways are used. The complete representation is easily translatable to both BPMN and Petri nets. IvM provides log animation and immediate parameter feedback such that exploring a process becomes easier, and this is shown in the paper by using a case study and comparing it with different algorithms.

Results

We present in this section the results after mining the event logs obtained in section 4.2.4 after filtering based on support, confidence and lift, for requesting an image while X-ray is disabled, and the three response categories. For response categories negative and miscellaneous the models are small enough to show a clear and concise set of actions. In case of miscellaneous responses, we see in figure 4.5 that the user either performs no action or decides to end the exam, which leads us to believe that users disable the X-ray and check that indeed it is disabled by pressing the pedal before ending the exam. However, the order of the activities that follow ending a lab exam remains unclear, as the process model has loops which allow many different sequences. For negative responses the actions are clear, the users press the fluoroscopy pedal again requesting an image while X-ray remains disabled as shown in figure 4.6. The event log containing positive responses still contains a high number of activities, as it is defined as the user enabling X-ray, which can be followed by any steps involved in an exam, such as movements or requesting images. Hence, the traces were further reduced by only keeping events within 1 minute from the detection of the anti-pattern, and preprocessing some activities labels, by renaming them and even removing some activities for which we did not know the exact meaning. Figure 4.7 shows the process model before the further reduction, while figure 4.8 shows the model obtained after. Even though the model has a high degree of freedom and allows for a lot of behaviour, we still can still distinguish the main activities which are performed during an operation, such as requesting exposure or fluoroscopy, moving the patient table or moving different components of system, all of which are normal activities during a regular operation.

Taking the event log with 15 activities obtained in section 4.2.2 after filtering based on entropy, for the same anti-pattern but only taking into account miscellaneous responses, the results are not as useful, as we do not see activities that are clearly related to ending an exam (see figure 4.9). We see that activities vary a lot, and even though they do not relate to specific actions during surgery such as taking images, they do not bring a lot more information on what is happening. We see that a new patient can be added, which does not necessarily mean the end of the previous exam, or that system startup is completed, which again does not say much about what is happening in the exam room.

4.3.2 CSM miner

The models obtained with the Inductive Visual Miner are more compact, but they still allow for a high degree of freedom, and it is very difficult to find the main control flow. Since we used artifacts to specify anti-patterns and responses, it would be natural to use an artifact-centric approach also when mining for process models. Paper [12] shows that from event data we can automatically discover composite state machines representing artifact-centric processes. The CSM miner provides ways to quantify and visualize also interactions among artifacts, and interesting correlations can be analysed. The paper shows that the proposed method can help a process analyst to find possible explanations for performance issues by applying it on real life process data.

Results

We present in this section the results after mining the event logs obtained in section 4.2.4 after filtering based on support, confidence and lift, for requesting an image while X-ray is disabled,
and the three response categories. We divided the event data into a number of artifacts, depending on the type of the event. For response categories negative and miscellaneous the models are small enough to show a clear and concise set of actions. In case of negative responses, we see in figure 4.10 that when the User guidance artifact is in the state that shows the message that X-ray is disabled, most often the artifact Command is in the state that stops requesting fluoroscopy, followed by the state that requests it again. For miscellaneous responses (see figures 4.11 and 4.12) we differentiate several things happening when the user guidance message is shown: first, we see that when the message is shown the exam is not ended yet, but in most cases ending is the activity that follows. Then, we see some states related to Connectivity, DataHandler and SessionManager, where some processes are terminated, or data is exported after the end of the exam is initiated. Lastly, we see that the artifact Command is in the state that stops requesting fluoroscopy, followed by some states that reset certain components or the machine, or change the orientation of the patient, all regular steps at the end of an exam. What can also be seen is that it can happen that all other artifacts are in state Notstarted when the user guidance message is shown, which implies the other scenario: the user does not perform any action in the time span that is analysed after the detection of the anti-pattern.

As before, the traces for positive responses were further reduced by only keeping events within 1 minute from the detection of the anti-pattern, and preprocessing some activities labels. The process models obtained are shown in figures 4.13 and 4.14. Again, we look at other artifacts when the user guidance message is shown to the user. For the Command artifact, we see the states for requesting fluoroscopy and exposure co-occur the most, following by stopping the request. Next to that, we also see table movements, and less machine movements, which is logical since the machine does not move that much while taking images. For artifact X-ray we see that the enable button is pressed, which is the positive response itself, and fluoroscopy and exposure are started now that X-ray is enabled. As mentioned before, these are all normal steps during a procedure, but in the case of the CSM miner we see a more structured flow of activities.

Using the log filtered based on entropy did not bring any useful results as with the Inductive Miner, and will be left out of discussion. Overall, we see that the models obtained with CSM miner are more structured that the ones from Inductive Visual Miner, and do not allow that much freedom. Moreover, the models obtained with the CSM miner also show lift and confidence values for the correlation between two states, which can be very useful when investigating what is happening during a particular state.

4.3.3 Additional miners

Several heuristic process discovery methods have been proposed to cope with less structured processes and the presence of noise in the event log. [24] presents the interactive Data-aware Heuristics Miner (iDHM), which enables quick interactive exploration of the parameter space and several heuristics by offering an interactive UI. It uses data attributes of the event log to reveal infrequent conditional dependencies. The paper shows that iDHM has reached a high degree of maturity, by applying it to an event log of a hospital billing process with more than 450,000 events. This miner was also used, however its compact process models are depicted as Directly Follows Graphs, which leads to an increase in the number of connections and harder to read models. The results can also be depicted as Petri nets, however the results we obtained from the CSM Miner seem to be more useful.

Another approach is to search for local process models in the anti-pattern responses. Local process models (LPM) describe only parts of a process model, capturing frequent patterns and only on a subset of activities. [30] describes a method to extract local process models, allowing for: choice, concurrency, loops, and sequence relations. Five quality dimensions are defined and express the degree of representativeness of a local process model with regard to an event log: support, confidence, language fit, coverage, and determinism. The case studies presented in the paper show that
CHAPTER 4. DATA REDUCTION AND MINING FOR PROCESS MODELS

LPM miner revealed process insights in the form of valuable patterns when the degree of variance of the event data prevented traditional process discovery techniques to discover a structured start-to-end process model.

To depict some of the results, we take again the anti-pattern presented in defined in section 3.4.2, where X-ray is enabled and the user requests fluoroscopy or exposure by pressing the pedal, and take only the miscellaneous responses. Using the reduced log with from section 4.2.4 after filtering based on support, confidence and lift with 15 activities left for miscellaneous responses, we obtain no local process models. Taking the event log with 15 activities obtained in section 4.2.2 after filtering based on entropy, we obtained many local process models, and two of them are shown in figure 4.15. The conclusion is that too many LPMs are obtained, and the activities depicted are not useful for our case-study. It became clear that this approach makes it too difficult to find interesting models, and is not suitable for the event data under analysis. For other case-studies the results might be different.

4.3.4 Limitations of the models

The first limitation stems from the nature of the event data in our case-study. It is very fine-grained, and most of the time it is difficult to map it to a higher-level action that a user might perform or request. Also, the high number of events that are being logged makes it harder to differentiate between types of events, hence states and artifacts. For example, it would be useful to tag logging such that it is clear whether it is related to movement commands, requesting and performing images, or other type of user actions. In that sense, it would be helpful to know if an event is the result of external user action or is an automated internal event. By doing so, it would be easier to distinguish between them, and focus on the user input for creating process models.

The second limitation is that a very large number of events are logged within a very small time window (e.g. 1 second), and the order in which those events are logged is not consistent, but rather random. This leads to an explosion of connections in the process models. Moreover, some of the events are duplicates, since some activities are logged multiple time for the same occurrence. As we seen with the Inductive Visual Miner, the models obtained allow for numerous paths and we cannot clearly detect a main control-flow of activities. The original logs contain large amounts of freedom and variety in behaviour, combined with a very large number of activities, so even after consistently reducing the number of activities, the process models obtained show a high degree of freedom. This is not necessarily a limitation of the miners, but rather a limitation of the data available for our case-study.

Lastly, we were expecting to uncover activities in the process models that can bring more insight into what happens after an anti-pattern occurs. In reality, most of the time, the activities left in the process models are the ones used to define the response or very closely related to them. That is why a different approach was implemented to uncover unknown trends, or correlations between the occurrence of the anti-patterns, the response categories and other system variables such that the results can be used as input for root-cause analysis. This approach, which is discussed in the next chapter, does not consider just the event dimension, but also others such as time or trace dimensions.
Figure 4.5: Inductive Miner - Process model after reduction based on support, lift, confidence; running example case-study, miscellaneous responses

Figure 4.6: Inductive Miner - Process model after reduction based on support, lift, confidence; running example case-study, negative responses
Figure 4.7: Inductive Miner - Process model after reduction based on support, lift, confidence; running example case-study, positive responses before further reduction.

Figure 4.8: Inductive Miner - Process model after reduction based on support, lift, confidence; running example case-study, positive responses after further reduction.
Figure 4.9: Inductive Miner - Process model after entropy-based filtering; running example case-study, miscellaneous responses

Figure 4.10: CSM Miner - Process model after reduction based on support, lift, confidence; running example case-study, negative responses
Figure 4.11: CSM Miner - Process model after reduction based on support, lift, confidence; running example case-study, miscellaneous responses (1)

Figure 4.12: CSM Miner - Process model after reduction based on support, lift, confidence; running example case-study, miscellaneous responses (2)
Figure 4.13: CSM Miner - Process model after reduction based on support, lift, confidence; running example case-study, positive responses (1)

Figure 4.14: CSM Miner - Process model after reduction based on support, lift, confidence; running example case-study, positive responses (2)
Figure 4.15: Two Local Process Models after entropy-based filtering; running example case-study, miscellaneous responses
Chapter 5

Correlation and causal analysis

The data was retrieved as explained in chapter 3 and reduced as described in 4. This chapter describes what analysis was further performed in order to uncover unknown trends, or correlations between the occurrence of the anti-patterns, the response categories and other system variables. Finally, a validation plan using the results will be discussed, also showing how to make the design process more data-driven, and make user interviews and observations more precise.

In order to investigate correlations and learn more about the occurrence of anti-patterns, several dimensions were considered:

1. **Time dimension:** investigating where an anti-pattern tends to occur within an examination, and the time between occurrences.

2. **Trace dimension:** investigating trace variables such as such as the country of the machine, the protocol used for radiation when the incident happened, or the software release number.

3. **Event dimension:** mining for process models (discussed in chapter 4).

The dimension set is not complete, as more dimensions can be added later on, depending on the needs and on the anti-patterns.

5.1 Analysing distribution of occurrences

An aspect that was further investigated was the distribution of anti-pattern occurrences, per response category. Density plots were created to help find trends and answer the following question - *When does the anti-pattern occur within a lab exam?* Note that the moment an anti-pattern occurs is the moment the anti-pattern is detected (i.e. its end). Three measures were used:

1. Difference to lab start: $\text{Detected} - \text{Start}$

2. Difference to lab end: $\text{End} - \text{Detected}$

3. Normalizing: $\frac{\text{Detected} - \text{Start}}{\text{End} - \text{Start}}$. Because cases vary in length (see figure 5.1), we use scaling to see when the anti-pattern occurs along an exam (0 meaning right at the start, and 1 right at the end).

Two different plots were used to show results: histogram and kernel density plot. A density plot visualises the distribution of data over a continuous interval or time period. This chart is a variation of a histogram that uses kernel smoothing to plot values, allowing for smoother distributions. The peaks of a density plot help display where values are concentrated over the interval. An advantage density plots have over histograms is that they are better at determining the distribution shape because they are not affected by the number of bins used.
Chapter 5. Correlation and Causal Analysis

Figure 5.1: Computing difference between the time the anti-pattern occurs and lab start, end, and scaling it.

KDE (Kernel Density Estimation) [29] is a nonparametric technique for probability density estimation in which a known density function (the kernel) is averaged across the observed data points to create a smooth approximation. Also, KDE is a non-parametric density estimators, this means that the estimator has not a fixed functional form but only it depends upon all the data points we used to reach an estimate and the result of the procedure has no meaningful associated parameters. We use Gaussian kernel, and the estimation function includes automatic bandwidth determination. The number of bins has to be provided as input, and as we want to have 5-minutes bins for the differences to exam start and end, we take the difference between the maximum and minimum of the values and divide it by 300 (since we work in seconds):

\[ \text{bins} = \frac{\text{max} - \text{min}}{300}. \]

For the normalized values, we use the default of 100 bins.

The histogram method divides the range of possible values into distinct non-overlapping bins, then counts how many samples are in each bin. For the histogram we use the same number of bins as for the KDE plot.

For some anti-pattern responses, the KDE and histogram plots clearly show a tendency. Looking at the running example defined in section 3.4.2, where X-ray is enabled and the user requests fluoroscopy or exposure by pressing the pedal, we can see that the miscellaneous category tends to happen towards the end of a procedure. Both in figure 5.2 and 5.3 there is a blue (-miscellaneous) spike towards the end, which leads us to believe that users disable the X-ray and check by pressing the pedal before ending the lab exam. This would imply that miscellaneous traces have a different goal, i.e. ending the exam right after the pedal press while X-ray disabled implies that the goal of the user was not to make an image, but rather to check that X-ray is indeed disabled.

Furthermore, in order to see how far apart anti-pattern instances happen within a procedure, the framework is extended to add the time since last occurrence in seconds for each anti-pattern detection. The results are used to:

- Plot the distribution of frequency per exam (i.e. how many times the anti-pattern occurs per procedure) - figure 5.4 depicts a histogram where Y-axis shows the number of procedures and X-axis shows the number of anti-pattern instances per procedure. We can see that most procedures have 1 or 2 occurrences. There is only one procedure which has 43 occurrences, and further investigation showed that the user enabled and disabled the X-ray several times in a short time-span during the beginning of that procedure.

- Plot the distribution over the time since last occurrence for each anti-pattern - figure 5.5
CHAPTER 5. CORRELATION AND CAUSAL ANALYSIS

Figure 5.2: Histogram after scaling AP occurrences for running example; red is negative response, green is positive and blue is miscellaneous.

Figure 5.3: KDE plot after scaling AP occurrences for running example; red is negative response, green is positive and blue is miscellaneous.

depicts a histogram where X-axis has 5 intervals ("0-5 min", "5-30 min", "30-60 min", "1-6 hours", "6-12 hours") and Y-axis shows the number of anti-pattern instances. Most occurrences happen close to each other (within 5 minutes). There are a few instances which happen more than 6 hours apart, which is very unusual since a procedure is not supposed to last this long. After further investigation, it was concluded that this was not the case, and some events that indicate the end of a procedure were missing from the data. Looking at the data it was assumed that one procedure lasted more than 12 hours, when in reality 4 different procedures happened. The reason behind this remains unknown, but the best guess is that the users simply did not end the lab exam before proceeding to the next one.

Figure 5.4: Distribution of frequency per exam - X-axis show the number of instances during procedure, Y-axis shows the number of procedures with that many instances.

Figure 5.5: Distribution over the time since last occurrence, where X-axis has 5 intervals ("0-5 min", "5-30 min", "30-60 min", "1-6 hours", "6-12 hours") and Y-axis shows the number of anti-pattern instances.

5.2 Association rule mining

There are several variables that can be defined for each incident, or anti-pattern occurrence, such as country of the machine, month, day and hour of the incident, the protocol used for radiation when the incident happened, or the software release number. Storing these variables in a vector for each anti-pattern instance, enables association rule mining, to see if we can find interesting and relevant relationships between the variables and the response category, for each anti-pattern.
Association rule mining searches for association, correlation, or causal structures among sets of items [23]. An association rule takes the following form: \( LHS \rightarrow RHS \), where \( LHS \) and \( RHS \) are an item, set of items or set of attributes, and \( LHS \) (left-hand-side) implies \( RHS \) (right-hand-side). A concrete example of an association rule could be: \( \{milk, bread\} \rightarrow \{eggs\} \). This can be read as “if the customer buys milk and bread, then it is highly likely that eggs will be bought too”.

In order to select interesting rules from the dataset, again, some important metrics are used: support, confidence and lift. Support is a number between 0 and 1 and indicates how frequently that particular rule is true in the dataset. Confidence is also a number between 0 and 1 and represents how many times the rule has been found to be true. Another important concept is the concept of lift. Lift represents the degree to which the attributes occur independently from each other, and that a value \( > 1 \) implies that there is some sort of dependency. A value of 1 implies that these attributes are independent and no rule can be created between the two attributes or items.

Conceptually, we define several steps for each anti-pattern:

1. For each occurrence, retrieve the variables that will be investigated for finding association rules. In our case, those are: machine identifier, country, release number, month of occurrence, day of occurrence, hour of occurrence, protocol category (a doctor can select from several pre-defined protocols, to use the right dose of radiation depending on the area under surgery; e.g. vascular, cardiac), name of anti-pattern, and the response category.

2. Apply FP-growth algorithm, which is used to obtain all frequent itemsets from the given dataset. Frequent itemsets are groups of items that often appear together in the data, which can be used to derive association rules by setting threshold for support and confidence. One example of the FP-growth algorithm can be found in [37]. For this project, we use RapidMiner [3], a visual workflow designer for machine learning, data mining and analytics. The workflow created in RapidMiner can be found in Appendix E.

3. Filter out results based on support, confidence and lift to select the most interesting rules. Taking the running example again, defined in section 3.4.2, where X-ray is enabled and the user requests fluoroscopy or exposure by pressing the pedal, we set the following thresholds for the rules found: \( \text{support} \geq 0.3 \) and \( \text{confidence} \geq 0.5 \). Then we filter the rules such that they contain the response category, since finding rules of the type \( \{\text{Country} = \text{France}\} \rightarrow \{\text{Release} = 9.0.1\} \) are not interesting for our investigation. We are trying to see if there is any strong correlation between the defined variables and a response category. The results are shown in 5.6 and the most interesting ones are highlighted. We see that month being March, or country being Netherlands can imply a negative response for the anti-pattern. So it was observed more often then and there that the user tended to press the pedals again, failing to realize the X-ray is disabled. The same applies for protocol being Vascular. The rules have a small support, which means the items are not that frequent, however the confidence is high enough (over half of cases where the rule has found to be true). Also, the lift is not very high, perhaps more data would be needed to find better correlations. The rules that contain the release number are not interesting for now necessarily, since only one release was under investigation. This would be interesting to investigate later on, if more machines are added to the analysis.
5.3 Anti-pattern frequencies

This last section covers how the pattern and anti-pattern occurrences were summarised and visualized. Plots are created to aggregate the frequencies for each anti-pattern and response category:

- Per month (figure 5.7) - to see whether there are months with more “undesired behaviour”.
- Per machine (figure 5.8) - to see whether there are machines for which it happens frequently, this may imply that the users have a habit of using the system in a non-desirable way.
- Per protocol (figure 5.9) - doctors can select different protocols, depending on the area that is under surgery, and on the dosage and intensity of the radiation needed. This will reveal if there is any correlation between the protocol and the anti-pattern.
- Per country (figure 5.10) - similar to month and machine.

It is not enough to summarize frequencies, since it is also important to take into account how much the machine is being used. The more a machine is used, the more chances are that an anti-pattern occurs. Because of this, the frequencies are normalized over the number of procedures and over the time in which the machine was working (in minutes): for each date and machine, we count

<table>
<thead>
<tr>
<th>Premises</th>
<th>Conclusion</th>
<th>Support</th>
<th>Confidence</th>
<th>Lift</th>
</tr>
</thead>
<tbody>
<tr>
<td>Month = 3</td>
<td>Response = negative</td>
<td>0.072</td>
<td>0.573</td>
<td>1.085</td>
</tr>
<tr>
<td>Release = 0.0.2.0, ProtocolCategory = Vascular of...</td>
<td>Response = negative</td>
<td>0.077</td>
<td>0.575</td>
<td>1.089</td>
</tr>
<tr>
<td>Country = THE NETHERLANDS</td>
<td>Response = negative</td>
<td>0.099</td>
<td>0.597</td>
<td>1.130</td>
</tr>
<tr>
<td>Country = THE NETHERLANDS</td>
<td>Response = negative, Release = 6.0.2.0</td>
<td>0.099</td>
<td>0.597</td>
<td>2.295</td>
</tr>
<tr>
<td>Release = 0.0.2.0, Country = THE NETHERLANDS</td>
<td>Response = negative</td>
<td>0.999</td>
<td>0.597</td>
<td>1.130</td>
</tr>
<tr>
<td>ProtocolCategory = Vascular of...</td>
<td>Response = negative</td>
<td>0.125</td>
<td>0.649</td>
<td>1.213</td>
</tr>
</tbody>
</table>

Figure 5.6: Association rules found for running example case-study.
how many anti-pattern instances were found, then how many out of those has positive/negative/miscellaneous responses. Next, we retrieve the number of procedures and the total working time in minutes for that date and machine. Then, a ratio will be computed for the number of procedures, and another one for the working time. Figure 5.11 depicts a heatmap with the mean ratio of anti-pattern occurrences over the number of exams, where X-axis contains months, and Y-axis contains machines. The heatmap depicts results for the running example defined in section 3.4.2, where X-ray is enabled and the user requests fluoroscopy or exposure by pressing the pedal, without differentiating between response types, as splitting the data per response category did not bring much different results. The results show that there are 2 machines for which the anti-pattern occurrence seems to be consistent. This indicates that this behaviour is already a set routine for the staff, and the users should be asked to clarify why they do it.

Figure 5.11: Heatmap anti-pattern occurrences over number of exams per machine (Y-axis) per month (X-axis); running example case-study

Lastly, averages of anti-pattern occurrences have to be computed such that they can be compared to the estimations previously made in risk management. There are two ways Philips computes the likelihood of a risk happening:

1. Based on field data: all service calls and complaints receive a code from which it can be deduced how many hazardous situations have occurred in one time interval $T$: $N_{hs-T}$. It is important to note that one exam can lead to 0 or 1 hazardous situations, but not more. Then, the formula for computing the probability of the occurrence of the hazardous situation, denoted by $P_1$ is: $P_1 = \frac{N_{hs-T}}{IB \times ExamT}$, where $IB$ is the number of installed machines and $ExamT$ is the number of examinations per system in the time interval.

2. Other techniques, when no field data is available: worst case estimates, dividing the sequence of events into several parts and estimate per part. These are also known as Fermi estimates, where the challenge is to make reasonable assumptions, use what you know, and estimate as well as you can what you do not know.

Figure 5.12 depicts how causes (or anti-patterns) that can lead to hazardous situations are identified, and then potential damages are defined. It is important to note that Philips groups the
potential damage, or harm, into 4 different severity categories and then estimates the likelihood for each one using the two approaches mentioned above. The 4 categories are: less than moderate or no injury, moderate injury harm (e.g. bruise), serious injury (e.g. severe burns), immediate death. Our data-driven approach cannot categorize results the same way, as the data does not record what happens to the patient, or the damage. Hence, we can only estimate the light-grey part of the figure, by looking at how often certain anti-patterns happen on average, taking into consideration that they do not necessarily lead to harm, although we do not know if that is the case.

Figure 5.12: Overview input and output for hazardous situations.

In order to compare the results with the estimations Philips had for the hazardous situations, a similar average to the first category \( P_1 \) is computed, per month:

1. Set number of machines investigated: \( N_M \).
2. Get average number of exams (average by summing up all exams from that month and dividing by number of machines investigated): \( N_E \).
3. Get number of exams where there is at least one occurrence for one anti-pattern: \( N_{EX} \). This is needed due to the way hazardous situations are counted by Philips. Even if a situation happens several times during an exam, it will be counted just as one hazardous situation. Hence, counting the number of exams that have at least one anti-pattern occurrence will help compare the values from a closer perspective.
4. Get number of anti-pattern occurrences for one anti-pattern: \( N_{AP} \). The average obtained using \( N_{AP} \) will be higher compared to \( P_1 \), due to the way hazardous situations are counter, however, it will offer a different perspective, different from using \( N_{EX} \).
5. Compute monthly average: \( P_{AP} = N_{AP} / (N_M \times N_E) \) and \( P_{EX} = N_{EX} / (N_M \times N_E) \).
6. Aggregate \( P_{AP} \) and \( P_{EX} \) per year.
7. Convert average into parts per million (ppm): \( PPM_{AP} = P_{AP} \times 10000 \) and \( PPM_{EX} = P_{EX} \times 10000 \).

The same approach is used to compute averages for positive, negative, or miscellaneous responses per anti-pattern. The average is converted into parts per million, since this measure in used to estimate the likelihood of risks happening within Philips. One part per million denotes one part per 1000000 parts, so it implies that the hazardous situation has the chance of happening once in every million cases. The data-driven averages cannot be mapped one-on-one with the estimations,
since we did not mine for all the possible anti-patterns that can potentially cause a hazardous situation. Furthermore, we have to take into consideration that for most occurrences found in the machine data, the anti-pattern led to no harm most probably. This category is not even considered in the estimations. Hence, it is best if the obtained averages are used as another source of input when updating the estimations.

Taking another perspective, we use the number of corresponding pattern occurrences for each anti-pattern. In this way, we take into account how many times the user behavior was done as intended, instead of taking the average number of exams:

1. Get number of anti-pattern occurrences: \( N_{AP} \)
2. Get number of pattern occurrences: \( N_P \)
3. Compute monthly average for the anti-pattern: \( P_{ap} = \frac{N_{AP}}{N_{P} + N_{AP}} \)
4. Compute monthly average for the pattern: \( P_p = \frac{N_P}{N_{P} + N_{AP}} \)

An example is shown in figure 5.13, where the average percentage of positive versus negative instances is shown for the running example. We can see that out of all instances, the patterns are preponderant, consisting of more than 99%.

![Figure 5.13: Percentage of pattern instances (green) versus anti-pattern instances (red); running example case-study.](image)

5.4 From correlations to causalities

The results presented so far have to be evaluated and validated in some way. Because we are investigating user actions based on the data from machine logs, the most suitable approach is to discuss and check the results with relevant and knowledgeable people involved in the process. Furthermore, the project resulted in unexpected findings, which is normal, considering that we used data and process mining techniques, and not statistical analysis. In statistical analysis we try to answer questions which are already known, in data and process mining we find answers for questions we did not even know we have to ask. These findings can be validated either by observing or talking with people that use or know the system.

The main category of people that know the system is, as expected, the users of the machines, who can be doctors, technicians or other people involved in operations. Next, employees from clinical marketing are considered, since they are in close interaction with hospitals and medical staff, and can report back on what they typically see. The final category considered covers system experts, and more specifically system designers. These are people that are very knowledgeable about the functionalities of the system and also the intended use and expected user “mistakes”.

To summarize the findings, and speed up the process, a questionnaire was created and passed to
several people from the aforementioned categories. The questions have to be general and unbiased, such that the respondents do not have the tendency to respond affirmatively. Taking as example one of the investigated anti-patterns, where two pedals are pressed simultaneously (fluoroscopy and exposure, or fluoroscopy and single shot), which is an undesired use of the system, some questions that were asked are:

- *Do you ever press both pedals simultaneously? If yes, in what situation?*
- *Do you ever see someone press both pedals simultaneously? If yes, in what situation? Was it on purpose or accidentally?*
- *Do you ever take single shots while having fluoro pressed? Are you using two feet?*
- *Have you noticed that you accidentally press one pedal while trying to press/release the other?*

As expected, when asking these questions in the field, the users’ answers were vague, and they did not recall doing things in an “unexpected” way. To give an example, the questions above were asked in a hospital that uses a machine which was investigated and where this anti-pattern occurred, and the doctor said the he does not press two pedals at the same time. However, when the usability engineer that was asking the questions observed the doctors, she noticed something unusual: two doctors were working on the same set of pedals (see figure 5.14), and two different set of pedals were used in the same exam room (see figure 5.15). These two ways of operating can increase the chance of having both pedals pressed simultaneously, and system designers and usability engineers did not know about them beforehand. It is an example of unexpected finding that the analysis approach presented in this thesis can uncover, which would have been missed otherwise, because the users did not realize that working together could lead to both pedals pressed simultaneously, and when answering questions, they just considered themselves and said that they do not do it.

![Figure 5.14: Two doctors working on the same set of pedals.](image1)

![Figure 5.15: Two different pedals sets used in the same exam room.](image2)

Regarding clinical marketing and system experts, the answers were not conclusive, since the feedback received was that the questions seem to be too detailed, and cannot be answered. Hence, the main source remains system users. However, in order to get enough answers from the field, numerous doctors have to be interviewed and observed. By applying the analysis techniques explained in this chapter, the scope can be narrowed down. We can learn from data and several questions can be answered by this type of analysis before going to talk to the users:

1. For semi-structured interviews, the analysis helps answer the following questions: *Who to ask* - what hospital, country; *What to ask?* - what tasks we mainly see in the occurrences, or in the responses.
2. For observations, the analysis helps answer the following questions: *Who to observe; What to observe; Where to observe - what machine; When to observe - what hour of the day, what day, month.*

It is up to system experts to decide on the candidate machines, or hospitals. To have a better overview on how the results of this project and, more specifically this chapter, can be used, a validation plan is created and can be used by system experts when deciding who and what to ask and investigate:

1. Check that logging was accurate, and the results are not the outcome of software bugs - this should be the first check, before taking any other action, although it is very unlikely that this will be the case, since the definitions of the patterns and anti-patterns are based on recreated data on a test machine. We assume that the logging is consistent throughout systems of the same type and with the same software version.

2. Identify machines, countries, months, day or hours that have higher frequencies for a certain anti-pattern.

3. Create a set of relevant questions.

4. Interview users.

5. Correlate answers with results.

6. Observe users to find unusual tasks that can be related to the anti-pattern.

7. Take further action or draw conclusions.

A last step was to check customer complaints to see if we can trace back some of the things that were observed in the analysis process, or if we can find explanations, or possible causes. This also turned out to be a challenging task, and no conclusive answers were found. A list of the anti-patterns investigated, and a summary of the results can be found in Appendix D.
Chapter 6

Conclusions

The main driver of this project was the data, and the main goal was to improve the design process by facilitating the understanding of usability-related issues through analysis of system usage data. This thesis presented a way of formalizing usage patterns and anti-patterns, and after applying it on a case-study which involved analyzing product usage data from a highly complex interventional system we shown that this approach can lead to discovering new ways the system is being used, that the experts were unaware of. One example is presented, where based on the investigation results and the following actions taken by a system expert, we have uncovered a new way of using the system, with two users working simultaneously on the same or two different pedal sets, which is something experts were not aware of. When one user was asked about it, he did not recall it since interaction is not always done consciously. This is a clear example that shows how traditional techniques of studying user behaviour cover a restricted view of a small set of users, by taking only one perspective into account, and can benefit from analysing product usage data and making product design more data-driven.

To achieve goal 1, patterns and anti-patterns were defined based on domain knowledge and the two main input sources: customer complaints and hypothetical hazardous situations. Then, the main tasks were recreated using a test machine such that the machine data can be correlated better to higher level activities. A formalism was chosen to define these patterns and anti-patterns in a way that is easier to read and understand by everyone, and not just experts. The specifications were then mapped to a detection algorithm, which acted as the basis for building a Python framework that connects to the database where the machine data is stored, and mines, retrieves, and stores relevant data for anti-patterns.

Research goal 2 was met by using the same formalism to define responses after an anti-pattern occurs, and the same Python framework was used to categorize as positive, negative or miscellaneous every response of an anti-pattern occurrence by looking at the events that were logged after the anti-pattern was detected. The case-study showed some interesting results, as presented in chapter 3. Furthermore, to show that this approach is suitable for a broader scope of datasets, another example was given, with sensor data logged from a smart baby bottle. Because of the nature of the logging, data had to be reduced, and several strategies were implemented, to find the most optimal one. Then, different process mining techniques were applied to obtain clearer, more concise process models. Data was cleaned and converted into event data beforehand. The results revealed a deeper issue, stemming from the logging of the events: within a very short time span numerous events are being logged in a random order, which leads to an explosion of connections in process models, since there is no clear logic in ordering some of the events.

For research goal 3, we have used different techniques to enable insights and input for root-cause analysis. Further analysis was performed in order to uncover unknown trends, or correlations between the occurrence of the anti-patterns, the response categories and other system variables.
CHAPTER 6. CONCLUSIONS

Different variables and dimensions were considered, while focusing on frequencies of anti-patterns, but also the time difference between consecutive occurrences, or when an anti-pattern tends to occur within an exam. In order to further investigate the anti-patterns, the system experts have to get enough answers from the field, and numerous doctors have to be interviewed and observed. By applying the analysis techniques explained in this thesis, the scope was narrowed down. We showed that we can learn from data and several questions can be answered by this type of analysis before going to talk to the users, such as what to ask, when to ask, at what time to observe, what to look for. This makes the old methods used by Philips usability engineers more precise when talking to or observing users. Overall, the project resulted in unexpected findings, which were considered to be very valuable by people within Philips. Lastly, our approach lead to having data-driven results, that should be used as another source of input when updating the estimations of Philips for risk management and analysis.

To summarize, there are three main novelties presented in this thesis, and they will be briefly discussed next:

1. **Formalization of patterns and anti-patterns by using artifacts:** the way anti-patterns were defined and mined is a novelty, as from our knowledge this has not been done before. The specifications rely on temporal logic templates from previous work done in literature, however, instead of having only one transition system, we use a Composite State Machine to model the Azurion system, allowing us to define and focus only on a subset of artifacts, or system components, depending on each anti-pattern's needs. In general, if the design of a system is object-oriented, it is more natural to use an artifact-centric approach and keep the same logic when defining patterns and anti-patterns.

2. **Making the design process more data-driven:** the methodology presented in this thesis allows to discover the right questions to ask and what to observe, while making old methods, such as semi-structured interviews, more precise. It can be used by usability engineers and system designers, because it allows them to gain information on what features, what hospitals and what time periods need more of their attention.

3. **Combining conformance checking with mining:** we are using original trace logs which are retrieved per day and machine to search for specific anti-patterns, which leads to generating new trace logs. Basically, we redefine the original traces, depending on the number of anti-pattern instances that are found. Hence, one event log can lead to the generation of 0, 1 or more event logs. Also, the definition of a case is changed, as we retrieve only a subset of events in a time span close to the anti-pattern detection. A case is no longer an exam, but rather an anti-pattern instance.

**Limitations and Future work**

The amount of available time and data for executing this project have lead to several limitations, or challenges that remain open for future work.

First of all, the biggest set-back was the lack of documentation available for all the fields, and events that are being logged. The documentation available for the database and the database tables needs to be more elaborated, as it does not clarify the meaning of each table or column. As an example, this is the description for the field `EquipmentNumber`, which is meant to uniquely identify a machine: “VARCHAR(32), NOT NULL, Extracted from the x folder name: string followed by E”. Clearly this description does not say much about the actual field, or its possible values and their meaning. Moreover, it is not very clear what time-zone is used when logging event timestamps. For now, it is assumed that all timestamps are in the time-zone of the hospital that is using that machine. Next, sometimes there are missing or duplicate events which make the analysis more difficult. The cause of missing events remains unclear, as it could be a software bug, or a user skipping important tasks that should not be skipped, like ending a lab exam. As for duplicates, the most probable cause is the fact that several blocks of code log the same event,
causing it to appear multiple times in the data. Also, some events are unclear, and cannot be correlated. For example, when a movement request is executed, the event that is logged contains an unique movement identifier. However, the corresponding command which comes as user input before executing the command, does not contain such information. In general, the following list of good practices should be followed when logging the machine data:

- Do not use time in events names. This leads to thousands of unique event names referring to the same event.
- Do not use numbers, such as identifiers, software versions, in events name.
- Do not use multiple-line events. Having a “line-break” will cause issues when exporting the rows of events.
- Use consistent names throughout releases. Currently, there are cases where the event name or format change from one release to another, which requires extra efforts when analyzing the data.

Secondly, the logged data is missing some events or functionality. As an example, it would be very useful to know whether a patient is on the table. That would reduce assumptions, and make clear the cases when the patient is indeed in danger to be hurt. The machines are equipped with a weight sensor, so it would be an extra step to log that information. Another important aspect that is missing from most events is the location, or where do commands come from. It would be very useful to know and separate commands that come from the exam room versus the control room, or to know which exact set of pedals they come from. Missing events is the main reason why some hazardous situation anti-patterns could not be reproduced and mined, together with anti-patterns that were physically impossible to reproduce, such as collisions or patient falling from the table.

Thirdly, the fact that the logged data is so fine-grained makes mining for process models a very difficult task. The first issue is the fact that a very large number of events are logged within a very small time window (e.g. 1 second), and the order in which those events are logged is not consistent, but rather is random. This leads to an explosion of connections in the process model, and make the main control-flow very hard to detect and follow. Moreover, it would be very useful to know if an event is the result of external user action or is an automated internal event. By doing so, it would be easier to distinguish between them, and focus on the user input for creating process models. It would be even more helpful if the logged events would have a well-defined tag attached to them, such that it can be easy to differentiate between different type of activities, such as patient-movement, table-movement, user-command etc.

Another limitation is the fact that for some anti-patterns we did not retrieve a sufficient amount of traces to build a process model or apply other data mining techniques. For this project, a set of 14 machines was investigated, looking at data from the year 2017. Therefore, mining data from other machines, and other periods of time is possible, and even recommended. By getting more data and more anti-pattern instances, the analysis results will be more robust and reliable. Also, until now, we only focused on the events that are logged after an anti-pattern is detected, by investigating what was the response immediately after. For future work, it would also be interesting to investigate the events that are logged before, and possibly extending the time-window that currently is set to 5 minutes before and after a detection. By looking at the events leading to the anti-pattern being detected, more hypotheses and potential causes can be discovered. Another task for future work would be to compare patterns to anti-patterns instances to find key differences between them. This approach would be a different way of trying to find causes and answers that can explain why and under what circumstances these anti-patterns occur.

To describe how the approach presented in this thesis could be used in practice in the future, several usage scenarios are defined:
• **Compare releases** - Check one anti-pattern for 2 different releases to find key differences, and check to see if there are improvements. New releases are supposed to be more efficient, safe, and less likely to impose problems.

• **Investigate new functionalities** - When designers or system experts want one functionality to be removed or replaced, one can analyse the way it is used and its frequency and use the results to form solid, data-driven arguments.

• **Check scale of impact when receiving a complaint** - When one customer complains, it is good to check how are the others experiencing that specific anti-pattern.

• **Trend analysis** - If this analysis is continuously done throughout the years, such that sufficient data is analysed, one could see if and how things change over time.

• **Automate anti-pattern detection** - This scenario requires the most amount of work, but if the approach presented in this thesis can be put in place such that data from the machines is continuously streamed and analysed, Philips could see anti-patterns as they are happening in real time, and inform timely its clients such that the hazard can be avoided.

Lastly, sometimes it is a good idea to at one point, when there are enough results obtained, stop looking further in the data, and instead go and talk to the user of the system. This will save time, and help clarify wrong assumptions. One final recommendation would be to take the user in the loop, and get him involved early on in the process. This will guarantee improved results, and will decrease the chance of searching for the wrong patterns and anti-patterns.
Bibliography


[34] W.M.P. van der Aalst. *Data Science in Action*. Springer, 2016. 7, 8


Appendix A

Example scenario for recreating machine data

This is an example for the usage patterns and anti-patterns we have recreated using a test machine. Every pattern involves a set of specific tasks which were times, such that later on, the events from the machine data could be correlated. This example focuses on the anti-pattern where both pedals are pressed simultaneously, resulting in fluoroscopy not starting. First the corresponding pattern is reproduced, where exposure is not pressed when fluoroscopy is pressed, and fluoroscopy happens. Then, the anti-pattern is reproduced where exposure is pressed when fluoroscopy is pressed, and is ignored. The tasks that were performed, in this order, are:

1. Add patient.
   Observed system response: none

2. Select single shot protocol.
   Observed system response: none

3. Press exposure pedal.
   Observed system response: exposure starts

   Observed system response: exposure stops

5. Press fluoroscopy pedal.
   Observed system response: fluoroscopy starts

   Observed system response: fluoroscopy stops

7. Press exposure pedal.
   Observed system response: exposure starts

8. Press fluoroscopy pedal.
   Observed system response: none

   Observed system response: none

    Observed system response: exposure stops

11. End procedure.
    Observed system response: none
Appendix B

Anti-patterns and patterns formalized for the case-study

MTL templates were used for defining six anti-patterns and their corresponding patterns on a case-study involving Philips Azurion machines. The definitions are presented next.

B.0.1 Formalization for “Press pedal while X-ray disabled”

The safety-related risk here is that the user is unaware that X-ray is disabled and that he is looking at a recorded image instead of live image.

Pattern

Name: “Positive Press pedal while X-ray enabled”
Intent: Looking for instances of “correct” behaviour, where one pedal is pressed when X-ray is enabled and during a lab exam. The expected consequence is a state change of the artifact RadiationStatus, meaning that the radiation will be turned on.
Real-time temporal logic mapping:

\[
(\text{EnabledExamStarted} \land \text{PedalPress}) \implies \text{first RadiationOn within } [0, 5]
\]

Predicates:

\[
\text{EnabledExamStarted} = \text{XRayStatus.Enabled} \land \text{ExamStatus.ExamStarted} \\
\text{PedalPress} = \text{PedalStatus.FluoroPressed} \lor \text{PedalStatus.ExposurePressed} \\
\text{RadiationOn} = \text{RadiationStatus.On}
\]

MTL Template name: Response 1
Examples of known uses: regular, intended use
Relationships to other patterns: related to anti-pattern “Accidental X-ray” anti-pattern

Anti-pattern

Name: “Negative Press pedal while X-ray disabled”
Intent: Looking for instances of “undesired” behaviour, where one pedal is pressed when X-ray is disabled and during a lab exam. The expected consequence is a state change of the artifact UserGuidanceStatus, meaning that the radiation will remain turned off, and that the user will get a message informing him of the fact that X-ray is disabled.
Real-time temporal logic mapping:

\[
(\text{DisabledExamStarted} \land \text{PedalPress}) \implies \text{to (UserMsgDisabled} \land \text{RadiationOff) within } [0, 5]
\]

Predicates:

\[
\text{DisabledExamStarted} = \text{XRayStatus.Disabled} \land \text{ExamStatus.ExamStarted} \\
\text{PedalPress} = \text{PedalStatus.FluoroPressed} \lor \text{PedalStatus.ExposurePressed} \\
\text{UserMsgDisabled} = \text{UserGuidanceStatus.XRayDisabled} \lor \text{UserGuidanceStatus.DoorOpened}
\]
APPENDIX B. ANTI-PATTERNS AND PATTERNS FORMALIZED FOR THE CASE-STUDY

RadiationOff = RadiationStatus.Off

MTL Template name: Response 1

Examples of known uses: could be user error, or intended to check whether X-ray is disabled

Relationships to other patterns: related to pattern “Accidental X-ray” anti-pattern

B.0.2 Formalization for “Accidental X-ray”

The safety-related risk here is that the user is unaware that X-ray is enabled and that he requests X-ray accidentally, by pressing a pedal.

Pattern

Name: “Positive Accidental X-ray”

Intent: Looking for instances of “correct” behaviour, where X-ray is disabled, so pressing a pedal does not lead to radiation being turned on, but rather to a state change of the artifact UserGuidanceStatus, meaning that the user will get a message informing him of the fact that X-ray is disabled.

Real-time temporal logic mapping:

\[(\text{Disabled} \land \text{PedalPress}) \leadsto \text{first} \ (\text{UserMsgDisabled} \land \text{RadiationOff}) \text{ within } [0, 5]\]

Predicates:

\[
\begin{align*}
\text{Disabled} &= XRayStatus.Disabled \\
\text{PedalPress} &= \text{PedalStatus.FluoroPressed} \lor \text{PedalStatus.ExposurePressed} \\
\text{UserMsgDisabled} &= \text{UserGuidanceStatus.XRayDisabled} \lor \text{UserGuidanceStatus.DoorOpened} \\
\text{RadiationOff} &= \text{RadiationStatus.Off}
\end{align*}
\]

MTL Template name: Response 1

Examples of known uses: should not happen, still better than the negative scenario

Relationships to other patterns: related to anti-pattern “Press pedal while X-ray disabled” anti-pattern

Anti-pattern

Name: “Negative Accidental X-ray”

Intent: Looking for instances of “undesired” behaviour, where X-ray is enabled, so pressing a pedal will lead to a state change of the artifact RadiationStatus, meaning that the radiation will be turned on. However, since this is a normal occurring practice in exams, in order for the “incorrect” behaviour to happen, the pedal must be pressed for a longer period of time (radiation on for more than 1 minute for exposure, and 5 minutes for fluoroscopy), if it is during a lab exam. Otherwise, the radiation should not be on at all.

Real-time temporal logic mapping:

\[(\text{absent} \ \text{RadiationOff} \text{ after} \ (\text{EnabledIdle} \land \text{PedalPress} \land \text{RadiationOn}) \text{ for } 1) \lor \]
\[(\text{absent} \ \text{RadiationOff} \text{ after} \ (\text{EnabledExamStarted} \land \text{FluoroPress} \land \text{RadiationOn}) \text{ for } 300) \lor \]
\[(\text{absent} \ \text{RadiationOff} \text{ after} \ (\text{EnabledExamStarted} \land \text{ExposurePress} \land \text{RadiationOn}) \text{ for } 60) \lor \]

Predicates:

\[
\begin{align*}
\text{EnabledExamStarted} &= XRayStatus.Enabled \land \text{ExamStatus.ExamStarted} \\
\text{EnabledIdle} &= XRayStatus.Enabled \land \text{ExamStatus.Idle} \\
\text{PedalPress} &= \text{PedalStatus.FluoroPressed} \lor \text{PedalStatus.ExposurePressed} \\
\text{FluoroPress} &= \text{PedalStatus.FluoroPressed} \\
\text{ExposurePress} &= \text{PedalStatus.ExposurePressed} \\
\text{RadiationOn} &= \text{RadiationStatus.On} \\
\text{RadiationOff} &= \text{RadiationStatus.Off}
\end{align*}
\]

MTL Template name: Absence 2

Examples of known uses: first one is “cleaner’s case”, when there is no exam but someone presses the pedal by mistake, not knowing X-ray is enabled. The second two are too long exposure or fluoroscopy times, which should not happen normally.

Anti-pattern detection and analysis in data-driven Product Design 65
**APPENDIX B. ANTI-PATTERNS AND PATTERNS FORMALIZED FOR THE CASE-STUDY**

**Relationships to other patterns:** related to pattern “Press pedal while X-ray disabled” anti-pattern

**B.0.3 Formalization for “Pressing fluoro while exposure is pressed”**

The safety-related risk here is that the user is unaware that exposure pedal is already pressed, so he presses fluoro and is under the impression that he is looking at a live image, when in reality he is looking at a recorded image (exposure has priority), if exposure is set to single-shot. Otherwise, the risk is to increase the radiation dose too much, and unnecessarily.

**Pattern**

**Name:** “Positive Pressing fluoro while exposure is pressed”

**Intent:** Looking for instances of “correct” behaviour, where no pedal is pressed and radiation is off, and enabled. The expected consequence is that pressing fluoroscopy pedal will lead to a state change of the artifact RadiationStatus, meaning that the radiation will be turned on.

**Real-time temporal logic mapping:**

\[
\text{present first } (\text{NoPedalPress} \land \text{EnabledExamStarted}) \text{ before } \text{FluoroPress within } [0.001, 87000]
\]

**Predicates:**

\[
\begin{align*}
\text{EnabledExamStarted} &= \text{XRayStatus.Enabled} \land \text{ExamStatus.ExamStarted} \\
\text{NoPedalPress} &= \text{PedalStatus.None} \\
\text{FluoroPress} &= \neg\text{XRayStatus.Disabled} \land \neg\text{ExamStatus.Idle} \land \text{PedalStatus.FluoroPressed}
\end{align*}
\]

**MTL Template name:** Existence 2

**Examples of known uses:** regular, intended use

**Relationships to other patterns:** -

**Anti-pattern**

**Name:** “Negative Pressing fluoro while exposure is pressed”

**Intent:** Looking for instances of “undesired” behaviour, where exposure pedal is already pressed, so when fluoroscopy pedal is also pressed it will be ignored, and radiation will continue to be on for exposure.

**Real-time temporal logic mapping:**

\[
\text{present first } (\text{EnabledExamStarted} \land \text{ExposurePress} \land \text{RadiationOn}) \text{ before } \text{BothPress within } [0.001, 87000]
\]

**Predicates:**

\[
\begin{align*}
\text{EnabledExamStarted} &= \text{XRayStatus.Enabled} \land \text{ExamStatus.ExamStarted} \\
\text{ExposurePress} &= \neg\text{XRayStatus.Disabled} \land \neg\text{ExamStatus.Idle} \land \text{PedalStatus.BothPressed} \\
\text{RadiationOn} &= \text{RadiationStatus.On}
\end{align*}
\]

**MTL Template name:** Existence 2

**Examples of known uses:** user error or pedal stuck

**Relationships to other patterns:** -

**B.0.4 Formalization for “Pressing fluoro while single shot is pressed”**

**Pattern**

**Name:** “Positive Pressing fluoro while single shot is pressed”

**Intent:** Looking for instances of “correct” behaviour, where single shot pedal is not pressed and radiation is off, and enabled. The expected consequence is that pressing fluoroscopy pedal will lead to a state change of the artifact RadiationStatus, meaning that the radiation will be turned on.

**Real-time temporal logic mapping:**

\[
(\text{EnabledExamStarted} \land \text{SingleShotReleasedFluoroPressed}) \text{ leadsto first } \text{RadiationOn within } [0, 5]
\]
APPENDIX B. ANTI-PATTERNS AND PATTERNS FORMALIZED FOR THE CASE-STUDY

Predicates:
EnabledExamStarted = XRayStatus.Enabled ∧ ExamStatus.ExamStarted
SingleShotReleasedFluoroPressed = PedalStatus.FluoroPressed ∧ ConfiguredPedalStatus.None
RadiationOn = RadiationStatus.On
MTL Template name: Response 1
Examples of known uses: regular, intended use
Relationships to other patterns: -

Anti-pattern

Name: “Negative Pressing fluoro while exposure is pressed”
Intent: Looking for instances of “undesired” behaviour, where single shot pedal is already pressed, so when fluoroscopy pedal is also pressed it will be ignored.
Real-time temporal logic mapping:
present first (EnabledExamStarted ∧ SingleShotOnly ∧ RadiationOn) before FluoroAndSingleShot within [0.001, 87000]
Predicates:
EnabledExamStarted = XRayStatus.Enabled ∧ ExamStatus.ExamStarted
SingleShotOnly = PedalStatus.None ∧ ConfiguredPedalStatus.SingleShotPressed
RadiationOn = RadiationStatus.On
FluoroAndSingleShot = ¬XRayStatus.Disabled ∧ ¬ExamStatus.Idle ∧ ¬ConfiguredPedalStatus.None ∧ PedalStatus.FluoroPressed
MTL Template name: Existence 2
Examples of known uses: user error or pedal stuck
Relationships to other patterns: -

B.0.5 Formalization for “Movements while geo is locked”

Pattern

Name: “Positive Movements while geo is locked”
Intent: Looking for instances of “correct” behaviour, where the user gives a movement command, which will be executed, meaning the geo is unlocked.
Real-time temporal logic mapping:
(ExamStarted ∧ MoveCommand) leadsto first MoveExecuted within [0, 30]
Predicates:
ExamStarted = ExamStatus.ExamStarted
MoveCommand = TableMovementStatus.ChangeHeight ∨ TableMovementStatus.Cradle ∨ TableMovementStatus.Tilt
MoveExecuted = TableMovementStatus.Executed
MTL Template name: Response 1
Examples of known uses: regular, intended use
Relationships to other patterns: -

Anti-pattern

Name: “Negative Movements while geo is locked”
Intent: Looking for instances of “undesired” behaviour, where geo is locked and the user gives a movement command, which will be ignored, and a user guidance message will be displayed.
Real-time temporal logic mapping:
(ExamStartedGeoLock ∧ MoveCommand) leadsto first UserMsgGeoLocked within [0, 5]
Predicates:
ExamStartedGeoLock = ExamStatus.ExamStarted ∧ GeometryStatus.Locked
MoveCommand = TableMovementStatus.ChangeHeight ∨ TableMovementStatus.Cradle ∨ TableMovementStatus.Tilt
UserMsgGeoLocked = UserGuidanceStatus.GeoLocked
MTL Template name: Response 1

Anti-pattern detection and analysis in data-driven Product Design 67
APPENDIX B. ANTI-PATTERNS AND PATTERNS FORMALIZED FOR THE
CASE-STUDY

Examples of known uses: -
Relationships to other patterns: -

B.0.6 Formalization for “Restart during lab exam”

Pattern
Name: “Positive no Restart during lab exam”
Intent: Looking for instances of “correct” behaviour, where there is no restart (cold or warm) during a lab exam.
Real-time temporal logic mapping:
present IdleNoRestart after ExamStarted
Predicates:
ExamStarted = ExamStatus.ExamStarted
IdleNoRestart = ExamStatus.Idle ∧ RestartStatus.None
MTL Template name: Existence 1
Examples of known uses: regular, intended use
Relationships to other patterns: -

Anti-pattern
Name: “Negative Restart during lab exam”
Intent: Looking for instances of “undesired” behaviour, where there is a restart (cold or warm) during a lab exam.
Real-time temporal logic mapping:
present Restart after ExamStarted
Predicates:
ExamStarted = ExamStatus.ExamStarted
Restart = (RestartStatus.ColdRestart ∨ RestartStatus.WarmRestart) ∧ ¬ExamStatus.Idle
MTL Template name: Existence 1
Examples of known uses: system malfunction, leading to an unresponsive system
Relationships to other patterns: -
Appendix C

Anti-patterns responses formalized for the case-study

We define responses for the anti-patterns defined previously in the case-study.

C.0.1 “Press pedal while X-ray disabled”

The risk control measure set in place when a pedal is pressed while X-ray is disabled, is to show a user guidance message. We need to check if this message is effective and the user enables X-ray.

Positive response:
The user enables the radiation within 5 seconds after the user guidance message is shown, stating that the X-ray is disabled.

Real-time temporal logic mapping:
Detected leadsto first Enabled within [0,5]

Predicates:
Enabled = XRayStatus.Enabled

MTL Template name: Response 1

Negative response:
The user continues to try and presses the pedal, failing to realize that radiation is disabled. We must check that the lab exam does not end. As soon as it ends, the formula will not hold anymore.

Real-time temporal logic mapping:
Detected leadsto first (DisabledExamStarted ∧ PedalPress)

Predicates:
DisabledExamStarted = XRayStatus.Disabled ∧ ¬ExamStatus.Idle
PedalPress = PedalStatus.FluoroPressed ∨ PedalStatus.ExposurePressed

MTL Template name: Response 1

Miscellaneous:
Other
(One example could be ending a lab exam - to double check that X-Ray is indeed disabled, or performing other type of activities after the pedal press.)

C.0.2 “Accidental X-ray”

The risk control measure set in place for long radiation is a buzzer that goes off after 5 minutes of fluoroscopy. However, for exposure there is no buzzer and in previous step, no instance could
APPENDIX C. ANTI-PATTERNS RESPONSES FORMALIZED FOR THE CASE-STUDY

be found of long fluoroscopy. On the other hand, when exposure is on there is a specific sound (different buzzer) that goes off (if it was not disabled by the hospital). Hence, one check to see whether the buzzer if effective is to see if the user continues to do long exposures.

**Positive response:**

The user does not continue to press the pedal for too long, i.e. when negative response does not happen within 5 minutes (time window taken when logging scenario traces).

**Negative response:**

The user continues to press the pedal for too long. Here the condition and goal are the same as for the scenario:

**Real-time temporal logic mapping:**

\[
\neg \text{RadiationOff} \quad \text{after} \quad (\text{Detected} \land \text{FluoroPress} \land \text{RadiationOn}) \quad \text{for} \quad 300 \\
\lor \quad (\neg \text{RadiationOff} \quad \text{after} \quad (\text{Detected} \land \text{ExposurePress} \land \text{RadiationOn}) \quad \text{for} \quad 60)
\]

**Predicates:**

\[
\text{RadiationOff} = \text{RadiationStatus.Off} \\
\text{RadiationOn} = \text{RadiationStatus.On} \\
\text{FluoroPress} = \neg \text{ExamStatus.Idle} \land \text{PedalStatus.FluoroPressed} \\
\text{ExposurePress} = \neg \text{ExamStatus.Idle} \land \text{PedalStatus.ExposurePressed}
\]

**MTL Template name:** Absence 1

**Miscellaneous:**

It is still not clear how it is possible to have cases where exposure or fluoro happen without being during a lab exam (lab exam should start after first acquisition starts). Hence, all instances found where the event for starting a lab exam is missing will be moved to this category.

C.0.3 “Pressing fluoro while exposure is pressed”

There is no risk control measure set in place to inform the user that both pedals are pressed. However, when exposure is pressed there is a buzzer going on that should help the user realize exposure is on. Hence, a positive response would be to release the exposure pedal immediately (within 2 seconds). It does not matter how much time passes until fluoro is released because it is the intended use (and should work once exposure pedal is released). A negative response happens when fluoro is released and then more than 2 seconds pass until exposure is released. It implies that the user was unaware that exposure was on while trying to make fluoroscopy.

**Positive response:**

**Real-time temporal logic mapping:**

Detected leadsto first FluoroPress within [0, 2]

**Predicates:**

\[
\text{FluoroPress} = \text{PedalStatus.FluoroPressed}
\]

**MTL Template name:** Response 1

**Negative response:**

**Real-time temporal logic mapping:**

\[(\text{Detected} \land \text{ExposurePress}) \quad \text{leadsto first} \quad \text{ExposureReleased} \quad \text{within} \quad (2, 87000)\]

**Predicates:**

\[
\text{ExposurePress} = \text{PedalStatus.ExposurePressed} \\
\text{ExposureReleased} = \text{PedalStatus.None}
\]

**MTL Template name:** Response 1
APPENDIX C. ANTI-PATTERNS RESPONSES FORMALIZED FOR THE CASE-STUDY

Miscellaneous:
Other - One example could be the case when one pedal is pressed and the other one is also touched by accident, and then both pedals will be released approximately simultaneously. Releasing the pedals simultaneously cannot be considered a positive response: e.g. a doctor presses both pedals by mistake for 10 seconds without realizing that exposure is on and thinking that fluoroscopy is working, then he would normally release the exposure pedal while releasing the intended fluoroscopy pedal.

C.0.4 “Pressing fluoro while single shot is pressed”
There is no risk control measure set in place to inform the user that both pedals are pressed. However, when exposure is pressed there is a buzzer going on that should help the user realize exposure is on. Hence, a positive response would be to release the single shot pedal immediately (within 2 seconds). It does not matter how much time passes until fluoro is released because it is the intended use (and should work once exposure pedal is released). A negative response happens when fluoro is released and then more than 2 seconds pass until single shot is released. It implies that the user was unaware that exposure was on while trying to make fluoroscopy.

Positive response:
Real-time temporal logic mapping:
Detected leadsto first SingleShotReleased within [0, 2]
Predicates:
SingleShotReleased = ConfiguredPedalStatus.None
MTL Template name: Response 1

Negative response:
Real-time temporal logic mapping:
(Detected ∨ SingleShotPressed ∧ ¬FluoroPress) leadsto first SingleShotReleased within (2, 87000]
Predicates:
SingleShotPressed = ConfiguredPedalStatus.SingleShotPressed
FluoroPress = PedalStatus.FluoroPressed
SingleShotReleased = ConfiguredPedalStatus.None
MTL Template name: Response 1

Miscellaneous:
Other - One example could be the case when one pedal is pressed and the other one is also touched by accident, and then both pedals will be released approximately simultaneously. Releasing the pedals simultaneously cannot be considered a positive response: e.g. a doctor presses both pedals by mistake for 10 seconds without realizing that exposure is on and thinking that fluoroscopy is working, then he would normally release the exposure pedal while releasing the intended fluoroscopy pedal.

C.0.5 “Movements while geo is locked”
The risk control measure set in place when a movement command is made while geometry is locked, is to show a user guidance message. We need to check if this message is effective and the user unlocks geometry.

Positive response:
Real-time temporal logic mapping:
Detected leadsto first GeometryUnlocked within [0, 10]
APPENDIX C. ANTI-PATTERNS RESPONSES FORMALIZED FOR THE CASE-STUDY

Predicates:
GeometryUnlocked = GeometryStatus.Unlocked \land \neg ExamStatus.Idle

MTL Template name:  Response 1

Negative response:

Real-time temporal logic mapping:
Detected leads to first UserMsgGeoLocked

Predicates:
UserMsgGeoLocked = UserGuidanceStatus.GeoLocked \land \neg ExamStatus.Idle

MTL Template name:  Response 1

Miscellaneous:

Other

C.0.6 “Restart during lab exam”

There is no positive response. We categorize responses as miscellaneous if the restart happens in between cases, or the restart event is logged within 60 second of the same event (to deal with duplicate messages being logged 6-7 times for same event). The first occurrence of the logged restart event will be in the negative response category.
Appendix D

Results summary of the investigated anti-patterns

We present a table with a list of the anti-pattern investigated, and their results. We have the following categories for results: expected (EX) or unexpected (UEX), critical (C) or user issue (UI), cause determined (YC) or no clear cause (NC).

<table>
<thead>
<tr>
<th>anti-pattern</th>
<th>Response</th>
<th>EX/UEX</th>
<th>C/UI</th>
<th>YC/NC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pedal press X-ray disabled</td>
<td>Pos</td>
<td>EX</td>
<td>-</td>
<td>YC</td>
</tr>
<tr>
<td>Pedal press X-ray disabled</td>
<td>Neg</td>
<td>EX</td>
<td>UI</td>
<td>NC</td>
</tr>
<tr>
<td>Pedal press X-ray disabled</td>
<td>Misc</td>
<td>UEX</td>
<td>UI</td>
<td>NC</td>
</tr>
<tr>
<td>Long X-Ray</td>
<td>Pos</td>
<td>EX</td>
<td>-</td>
<td>YC</td>
</tr>
<tr>
<td>Long X-Ray</td>
<td>Neg</td>
<td>UEX</td>
<td>C</td>
<td>NC</td>
</tr>
<tr>
<td>Long X-Ray</td>
<td>Misc</td>
<td>UEX</td>
<td>-</td>
<td>NC</td>
</tr>
<tr>
<td>Pressing both pedals</td>
<td>Pos</td>
<td>EX</td>
<td>-</td>
<td>YC</td>
</tr>
<tr>
<td>Pressing both pedals</td>
<td>Neg</td>
<td>EX</td>
<td>UI</td>
<td>YC</td>
</tr>
<tr>
<td>Pressing both pedals</td>
<td>Misc</td>
<td>UEX</td>
<td>UI</td>
<td>YC</td>
</tr>
<tr>
<td>Pressing both pedals - single shot</td>
<td>Pos</td>
<td>EX</td>
<td>-</td>
<td>YC</td>
</tr>
<tr>
<td>Pressing both pedals - single shot</td>
<td>Neg</td>
<td>EX</td>
<td>C</td>
<td>YC</td>
</tr>
<tr>
<td>Pressing both pedals - single shot</td>
<td>Misc</td>
<td>UEX</td>
<td>UI</td>
<td>YC</td>
</tr>
<tr>
<td>Movement while geo locked</td>
<td>Pos</td>
<td>EX</td>
<td>-</td>
<td>YC</td>
</tr>
<tr>
<td>Movement while geo locked</td>
<td>Neg</td>
<td>EX</td>
<td>UI</td>
<td>NC</td>
</tr>
<tr>
<td>Movement while geo locked</td>
<td>Misc</td>
<td>UEX</td>
<td>UI</td>
<td>NC</td>
</tr>
<tr>
<td>System restart during procedure</td>
<td>-</td>
<td>EX</td>
<td>C</td>
<td>NC</td>
</tr>
</tbody>
</table>

Table D.1: Summary of the results of investigating the anti-patterns and their responses.
Appendix E

RapidMiner workflow for mining association rules

A workflow was created in order to find association rules between the variables that can were stored for each anti-pattern instance.

![Figure E.1: Overview RapidMiner workflow for mining association rules.](image)

Two following operators are used:

- **Retrieve dataset**: we load the data where every row represents an anti-pattern occurrence, and has all the variables previously defined, such as machine identifier, country, release etc.

- **Filter examples**: we match the anti-pattern name to just one, such that we can search rules only for that particular anti-pattern.

- **Select attributes**: in case we want to leave out some variables, such as anti-pattern name.

- **Numerical to polynomial and Nominal to binomial**: to covert all fields into binomial, since the FP-growth algorithms requires only binomial input.

- **FP-growth**: to search and find frequent itemsets. Here, the minimum support is set, to filter out infrequent combinations.

- **Create association rules**: to generate a set of association rules from the given set of frequent itemsets. Here, minimum confidence is set, to filter out infrequent rules.
Appendix F

Detection strategy for MTL template

Depending on the anti-pattern, and the underlying MTL template, a different search strategy will be needed. In this section we will discuss the strategy for the MTL template most used for the case study, Response 1, implemented in algorithm 3. Here, every time predicate $A$ holds, predicate $B$ must hold within $d_1$ to $d_2$ units of time. All other MTL templates can be implemented similarly. When defining patterns, the symbols $A, B, \ldots$ stand for predicates over the state vector. Function $checkPredicate(P, s_i)$ checks whether predicate $P$ holds on state vector $s_i$.

**Algorithm 3 Detection Strategy for MTL template Response 1**

**INPUT:** Anti-pattern (or pattern) $A$ leads to first $B$ within $I = [d_1, d_2]$

**OUTPUT:** Set $AntipatternStartEndIndexes$ of pairs $(x, y)$ of indexes for the start and the end of each anti-pattern detection

$$\begin{align*}
  AntipatternStartEndIndexes &= \emptyset \\
  found_A &= False \\
  found_B &= False \\
  for \ x = 1 \ to \ n \ do \\
    if checkPredicate(A, s_x) == True then \\
      found_A &= True \\
      found_B &= False \\
      time_A &= timestamp(s_x) \\
    for \ y = x \ to \ n \ do \\
      if checkPredicate(B, s_y) == True and found_B == False then \\
        found_B &= True \\
        time_B &= timestamp(s_y) \\
        u &= time_B - time_A \\
        if !(d_1 \leq u \leq d_2) then \\
          found_B &= False \\
        else \\
          add \ (x, y) \ to \ AntipatternStartEndIndexes \\
    if found_A == True and found_B == True then \\
      return AntipatternStartEndIndexes \\
    else \\
      return \emptyset
\end{align*}$$

Anti-pattern detection and analysis in data-driven Product Design