The novel application of heuristic, machine learning and process discovery algorithms to industrial equipment logs to improve the efficiency of assembly and joining processes

Koehler, Wolfgang

Award date:
2020

Awarding institution:
Coventry University

The Novel Application Of Heuristic, Machine Learning and Process Discovery Algorithms To Industrial Equipment Logs To Improve The Efficiency Of Assembly And Joining Processes

BY
Wolfgang Koehler

A thesis submitted in partial fulfilment of the University's requirements for the Degree of Doctor of Philosophy

March 2020

Content removed on data protection grounds

Certificate of Ethical Approval

Applicant:

Wolfgang Koehler

Project Title:

A Novel Process Mining Approach To Derive Value From Incomplete And Flawed Industrial Equipment Event Logs

This is to certify that the above named applicant has completed the Coventry University Ethical Approval process and their project has been confirmed and approved as Low Risk

Date of approval:

11 August 2019

Project Reference Number:

P93715

Abstract

Today's industry is highly automated and offers an opportunity to log a variety of data for different aspects of the manufacturing process. Research suggests that mining those data can streamline those processes. During the literature review, it became apparent that much research for quality, logistics and chemical process improvements had been done but to the author's knowledge, none of this research addressed improvement suggestions for assembly and joining lines based on already available sensor signals.

The project was split up into the three steps data collection, preprocessing and analysis. The proposed framework requires logging of all of the equipment's sensor data (i.e. proximity switches, lights sensors), which is not a function offered for industrial equipment. Therefore a nomenclature based algorithm was developed to parse the underlying PLC (programmable logic controller) program to extract the data points of interest automatically. These data points then are directly converted into OPC (Open Platform Communications) setup parameters which enable immediate logging.

First the individual events within the log needed to be clustered into cases using novel heuristic algorithms. Evaluation of the resulting cases yielded that the data was flawed. Research suggests that such data can either be repaired, ignored or analysed with fault-tolerant algorithms like the heuristic miner. Since repair and fault-tolerant algorithms bear the risk of additional uncertainty, the deletion of incomplete or flawed cases was chosen. Identification of the faulty cases was addressed by enhancing the data with expert-based engineering features and tags for a fraction of the cases. Machine Learning then was applied to determine the completeness of the remaining cases.

Next expert knowledge was encoded into heuristic algorithms which can pinpoint several process problems within the data. Above described framework was tested with real-life data originating from an automotive body shop and the potential for cycle time improvements of up to 19% was discovered. It was also found that one of the shortcomings of the suggested framework is that the analysis is based on presumed dependencies between the different events. The Process Mining domain suggests algorithms that enable the discovery of the real dependencies by analysing corresponding event logs. Experiments with above algorithms yielded unsatisfactory results which triggered the development of an improved, for industrial assembly and joining equipment suitable Process Discovery algorithm. This algorithm, contrary to the established algorithms, can discover highly accurate models with a minimum number of cases. Based on this finding, rules were formulated to detect the questionable dependencies within a case. Finally, a new process called 'Interactive Trace Induction' was introduced and tested to enable capturing the crucial cases needed to discover such highly accurate models.

Contents

Listing of Figures

Listing of Abbreviations

caseID case identifier
CPMS cyber physical manufacturing system
CPPS cyber physical production system
CPS cyber physical systems
CSM composite state machine

DM data mining
DNS domain name server
DSM design structure matrix

ERP enterprise resource planning

GP gaussian process
GSP generalized sequential pattern

HMIs human-machine interfaces

IMLC inductive miner - life cycle
IoMT internet of manufacturing things
IoT internet of things

KDAM knowledge discovery and analysis in manufacturing
KDD knowledge discovery in databases
KPI key points of interest

LISA line information system architecture

MES manufacturing execution systems
MICE multivariate imputation by chained equations
MQTT message queuing telemetry transport
MVs missing values

OLE Object Linking & Embedding

OPC open platform communications
OPC-DA open platform communications data access
OPC-UA open platform communications unified architecture

PD predicted desirable
PLC programmable logic controller
PP part present
PU predicted undesirable

R&D research and development
Regex regular expressions
RFID radio-frequency identification
RNNs recurrent neural networks

SCADA supervisory control and data acquisition
SCARA selective compliance articulated robot arm
seqNum unique sequence number
SFC sequential function chart
SME medium-sized enterprises
SOA service-oriented architecture
SOO sequence of operation
SPADE sequential pattern discovery using equivalent class
SPARC/E sequential pattern recognition / eleusis

TOVE toronto virtual enterprise

UD unpredicted desirable
USB universal serial bus
UU unpredicted undesirable

VAEs variational autoencoders
VIN vehicle identification number

XIC normally open contact

Symbol Definitions

Table 0.0.1: Definition Of Symbols Used Within Formulas

t_i	incoming event (first event of a case)
t_o	outgoing event (last event of a case)
t	event
t'	equivalent event (same valve)
$\bar{t}$	opposing event
t_s	start event (related to an activity)
t_c	complete event (related to an activity)
t_l	load event
t_u	unload event
t_m	missing events
τ_s	start timestamp
τ_c	complete timestamp
$\Delta\tau$	duration
$\lambda\tau$	mean duration
$a > b$	a is directly followed by b
$b \not> a$	b is not directly followed by a
$a \rightarrow b$	causal relationship between a and b
$a\#b$	a and b have been observed in parallel
$\mathbb{B}$	multi-set (bag)
$\mathscr{A}$	activities
L	event log over A
l	event log limited to one style and one case per trace
σ	trace
init	robot initiation
$\forall$	for all
$\varnothing$	NULL
VIN	vehicle identification number
p_c	probability for collision
n	number of entries
k	the number of bits

Acknowledgments

I'm sincerely grateful for the support of my directors of studies, Dr. Yanguo Jing and Dr. Rahat Iqbal who were available for my questions and concerns almost around the clock, seven days a week.

Special thanks go to Prof. Sharma and Dr. Tickle for their detailed review of my thesis and their much appreciated feedback during the Viva.

Finally I also would like to use this opportunity to thank my loving wife Caroline and our three kids Tonny, Reana and Jeremy for their patience during the time of my studies. Without their continuous support, I would not have been able to balance work, school and family life.

1 Overview

1.1 Introduction

In the realm of manufacturing planning and control the term 'hidden factory', according to George (George 2002), describes processes that 'consume resources and produce nothing of value to the customer'. A more detailed description was provided by Nakajima (Nakajima 1989). He defines hidden factories in terms of breakdown losses, setup and adjustment loses, idling and minor stoppage loses, reduced speed loses, start-up loses as well as quality defects and rework.

The first and most obvious category is breakdown losses. Breakdowns have a major impact on production throughput since they are happening most often at the most inopportune time. Once they occur, the maintenance team needs to be notified, which then comes onsite to assess the problem and develop an action plan. Then spare parts might be needed which may or may not be onsite. Finally, the repair takes place. The number of maintenance personnel and the size of the spare part stock are also a balancing act between effectiveness and cost. Preventive maintenance schemes are often put into place to replace equipment parts based on a predefined schedule. This practice helps to reduce the breakdown losses but at the same time increases the maintenance cost because parts, that might not be at the end of their life cycle, might get replaced. In recent years the trend shifted toward predictive maintenance. Data collected from

the machine is analysed with machine learning algorithms to allow maintenance to shift the time of part replacement closer to its expected point of failure.

The next category is setup and adjustment losses. Modern production equipment requires many adjustments to be made to achieve the best possible balance between product quality and production speed. Some examples are the setting of temperature, air pressure, airflow and drive acceleration/deceleration ramps. Often the equipment manufacturer's only goal is to fulfil the agreed-upon specification which still leaves lots of room for improvement. Typically long-term observation by a person, knowledgeable of the process is required to reach the best possible performance.

Another cost-driving factor is idling and minor stoppage losses. Due to the fixed cost of production facilities, times of no production can be considered waste. The stoppages can be deliberately scheduled, or they can be the result of imbalanced manufacturing lines or in the worst case of inadequate process control. Experiments show that some of these losses are difficult to spot because they might be only fractions of a second adding up because they are reoccurring millions of times within the equipment's life span. In addition to expert knowledge, this issue also requires the use of monitoring equipment.

Another factor is reduced speed losses. These losses can be caused by human intervention, by ageing equipment or due to shared resources. These, often gradual performance reductions are not easily noticed unless a constant, automated observation of the sensor feedback is in place.

An additional issue are start-up losses. Ideally, the manufacturing facility would want to be able to turn on new production equipment and run at a 100% production rate right from the start. Due to the complexity of automated manufacturing lines, often it is not possible to fully validate everything before shipment to the customer. Sometimes the equipment even might get built from the ground up on the customer's shop floor. This type of implementation means that the production lines will undergo a gradual start-up ramp where problems are addressed as they are experienced. Depending on the equipment's complexity, this time frame might stretch up to several months. Equipment manufacturers try to reduce the impact of this phase through modularisation and simulation of the machines before installation.

Quality defects and rework can also become a major cost-driving factor due to the additional time required to rectify the issues found. In some instances, the product might have to go through the production line again, slowing down production. Online quality monitoring helps to spot problems as early as possible and thus contain their impact.

Exploration of the hidden factory promises great opportunities for manufacturing

companies and modern data mining approaches appear to offer all the tools necessary to do so successfully. While approaching such a project, it has to be considered ahead of time that many continuous improvement initiatives fail. McLean et al. (McLean and Antony 2014) summarise the reasons under the following headlines: motives and expectations, organisational culture and environment, management leadership, implementation approach, training, project management and employee involvement levels. Their research shows that more than 60% of improvement projects are considered a failure. These findings led to the conclusion that such an undertaking might be in vain unless there is a common understanding between all the stakeholders regarding the expected outcome, time and cost involved.

When attempting a literature review concerning 'data mining in manufacturing' hundreds of papers can be found. They can be split up into several categories. Searching for the terms 'data mining for industrial processes' results in many works concerning biological and chemical processes. An example would be Charaniya et al. (Charaniya et al. 2010), who proposed the application of machine learning techniques for the evaluation of bio processes. Another issue often considered in the domain of industrial processes is the manufacturing quality. Since many processes are already monitored with the help of analogue sensors the next logical step is to record the values obtained and to apply data mining algorithms to discover patterns which allow conclusions regarding the quality of the part being manufactured. An example can be seen in the work of Gertosio et al. (Gertosio and Dussauchoy 2004) who propose such a scheme for the quality assessment and tuning of truck diesel engines. Application of their methodology resulted in a 27% reduction of the processing time within this production step. Quality also can be monitored and kept at a steady level through equipment diagnostics. Hou et al. (Hou, W. L. Liu, and Lin 2003) proposed a back-propagation neural network to monitor manufacturing processes. They proclaim that a causal relationship can be found between manufacturing parameters and product quality.

Machine learning and data mining strategies are sometimes also recommended for cycle time improvements. Chien et al. (Chien et al. 2005; F et al. 2007) describe the potential for cycle time prediction in semiconductor manufacturing. They mention that this type of manufacturing needs to be highly flexible and that the efficiency of the production line strongly depends on the product being manufactured as well as the equipment being utilised. They, therefore, recommend the clustering of similar manufacturing steps and the comparison of their efficiency to conclude possible process improvements for the currently evaluated process. Park et al. (J. Park, D. Lee, and Bae 2014) use a similar approach to evaluate the efficiency for block building within the shipbuilding industry. Blocks are modules which are later combined to form the

ship. The tasks performed while creating such a block are often similar and thus can be compared. The clustering into similar production steps and the identification of discrepancies can be achieved through machine learning algorithms.

The vast amount of research projects within the manufacturing realm triggered some researchers to perform a comprehensive literature review to derive applicable categories. At the same time, they are also highlighting potential areas which have not received the necessary attention yet. The categories defined by (Harding, Shahbaz, and Kusiak 2006; K. Wang et al. 2007; Choudhary, Harding, and Tiwari 2009; Groeger et al. 2012) are:

- Quality Control
- Job shop scheduling
- Fault diagnostics
- Manufacturing processes
- Manufacturing systems
- Maintenance
- Defect analysis
- Yield improvement

The research for this thesis falls into the category 'yield improvement'. As described before many previous works within this classification gain knowledge from the comparison of similar processes rather than utilising a priori knowledge to identify and rectify the shortcomings based on the data obtained. The data used for this research is another topic of discussion. Process improvements are often based on the cycle time of sub-tasks and the sequence in which they are executed. This thesis argues that the observations used so far are too coarse to allow for a proper process evaluation especially in conjunction with mechanical assembly equipment. Due to the complexity of the individual stations, further investigations typically are performed by production engineers manually through onsite observations and the development of improvements suggestions based on their domain knowledge.

It is perceivable, as Pethig et al. (Pethig et al. 2012) recommend, that the sensors of an existing automation system are logged. Other than a possible methodology on how to achieve such logging, no works have been found that exploit such data to derive improvement suggestions. These sensors are not considered for fault detection and predictive maintenance purposes either. Instead, many researchers suggest, for example, Lee et al. (J. Lee, Lapira, et al. 2013), that additional sensors, such as vibration and pressure sensors should be installed to obtain the data needed. The reason behind that can be seen in the fact, that the existing sensors only yield binary status signals while the proposed sensors provide analogue values which are more suitable for data exploration. A closer look reveals, that even the binary signals, when combined appropriately provide varying values suitable for data mining. Such values could be the time elapsing between

initiating an action and its completion or the sequence of the events. These are the values that build the basis of the research described within this thesis.

When evaluating why industrial data are not more often explored with the help of data mining techniques, Wang et al. (K. Wang et al. 2007) compiled the following list of reasons which seems to confirm onsite observations:

- The majority of researchers in the manufacturing domain area are not familiar with data mining (DM) algorithms and tools.
- The majority of theoretical DM researchers are not familiar with the complex manufacturing domain area.
- The few researchers who are skilled in both DM algorithms and manufacturing domain area are not able to access, often proprietary and sensitive, manufacturing enterprise data.
- It is difficult to evaluate the effectiveness and benefits while DM is implemented in manufacturing
- Industrial data are typically noisy, highly correlated, and very often, they are randomly missing due to various reasons such as faulty sensors and computer communication errors.
- Long time scales and high expense are involved in introducing these new techniques.

Proprietary access to the data, be it because of the bus systems and protocols in use or because of the sensitivity of the production data itself is also one of the main stumbling blocks mentioned by Pething et al. (Pethig et al. 2012). Baier et al. (Baier et al. 2015) add that many data obtained from legacy systems can not be customised. Thus it becomes 'a tedious task to reconstruct a mapping from cryptic names in a database to the activities in a process model'. Wuest et al. (Wuest et al. 2016) also confirm these observations.

All the above reasons are summed up by Windmann et al. (Windmann et al. 2015) stating: 'The high complexity of manufacturing processes and the continuously growing amount of data lead to excessive demands on the users concerning process monitoring, data analysis and fault detection.'

Attempting a manufacturing data mining project should be done in a structured matter. According to Fayyad et al. (Fayyad, Piatetsky-Shapiro, and Smyth 1996), the first workshop concerning Knowledge Discovery in Databases (KDD) took place in 1989. One of the goals of this workshop was to define the necessary steps for a successful data mining project. These steps were summed up by Fayyad et al. as follows:

- Understanding the manufacturing domain
- Collecting the targeted data
- Data cleaning, preprocessing and transformation
- Data integration: multiple data sources
- Choosing the functions of data mining (clustering, classification, prediction, association, regression, summarising)
- Choosing the appropriate data mining algorithm
- Data mining
- Interpretation and visualisation of the results
- Implementation of discovered knowledge

In 2004 Gertosio et al. (Gertosio and Dussauchoy 2004) added to this list the 'economic evaluation' since it has a great impact on the decision whether to put the proposed concept into production or not. Later on, Choudhary et al. (Choudhary, Harding, and Tiwari 2009) also proposed the addition of 'knowledge storage and reuse' when applying the concept to manufacturing systems.

In their paper regarding knowledge discovery and analysis in manufacturing (KDAM) Polczynski et al. (Polczynski and Kochanski 2010) defined the goals of data mining projects within the manufacturing domain as follows:

- Detection of root causes of deteriorating product quality
- Identification of critical and optimal manufacturing process parameters
- Prediction of the effects of manufacturing process changes
- Identification of root causes and prediction of equipment breakdowns

Considering its name, Process Mining seems to be another alternative when attempting to gain knowledge from industrial manufacturing processes. Originally formulated to aid software development, Process Mining was transferred to business processes by van der Aalst (W. V. D. Aalst 2016b) who applied the concept to discover business process models from an event log.

There are several basic terms used in conjunction with said event logs. To record events, there must be defined **activities**. An activity in the manufacturing realm could, for example, be the loading of a work piece into a machine. Associated **events** for such activity would be the initiation of the loading process and its completion. These events are recorded within an **event log** often together with a **timestamp**. The manufacturing process does not only consist of the loading activity. It also, on a high abstraction level, includes processing and unloading activities. The events associated with these activities, when considering an individual part being manufactured, are

considered to be a **case** for which ideally a **case ID** is recorded together with the events for easier identification. An event log, is made up of several events. The order of the events within a case is called a **trace**. For the simple example with three activities, it is likely that the trace for all cases recorded will be equal. If more detailed activities are logged, there are different reasons, which will be explored in the course of this work, that different traces will manifest themselves.

Process Mining uses different algorithms to preprocess such event logs and to extract knowledge from the different traces found. Besides the aforementioned process model discovery, this also includes conformance checking and process enhancements. To most industrial engineers, the research domain of Process Mining is unknown. It only came to the author's attention while searching for approaches that would allow for the clustering of events within a log into related cases. Although the techniques proposed to cluster unstructured event logs into cases did not prove to apply to industrial event logs the concept of Process Discovery as such remained promising for the analysis of industrial processes.

This research proposes the collection of detailed event logs, from automated production equipment to address four of the six big losses which are 'setup and adjustment losses', 'idling and minor stoppage losses', 'reduced speed losses' as well as 'start-up' loses.

The observation during this research was that neither the equipment manufacturers nor the end-users are fully aware of the impact caused by either external factors or the intertwining of the different internal processes. This unawareness is the reason why many machines are not utilised to their full potential. Although today's manufacturing operations are highly automated little data is collected to gauge the performance of such equipment. Therefore this research had to encompass multiple research steps to gather and analyse data.

Figure 1.1.1 shows the initial data collection effort, within the body shop that was made available for this research, highlighted in green and marked with the number 1. The data collection was limited to preselected fault messages and a few event triggers that allow determining the station's cycle time as well as the starved or blocked status.

The proposed collection of detailed event logs from automated production equipment is marked in the graphic in yellow and the number 2. This directly relates to section 3.1. During preprocessing these event logs then need to be grouped into cases and evaluated for completeness. This task is step number 3 within the industry 4.0 vision. The corresponding methodologies can be found in section 3.2. In the final step 4, the data is analysed to discover the big losses described previously. Proposals for such an

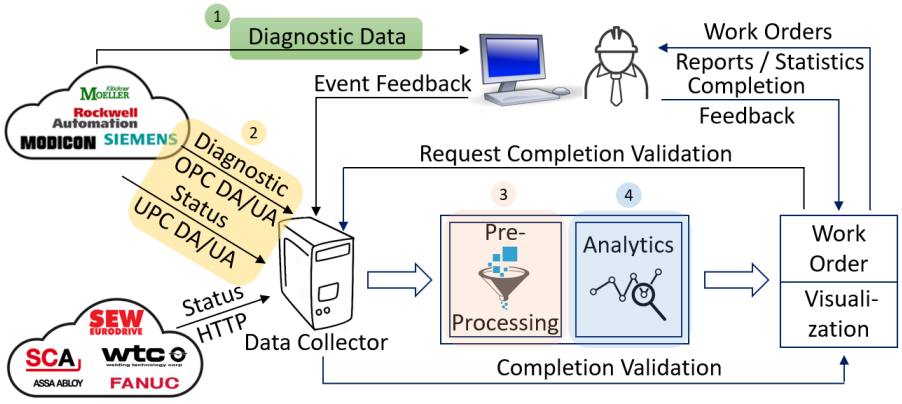

Figure 1.1.1: Industry 4.0 Vision

approach are described in sections 3.3 to 3.5. The system needs to create work orders for the maintenance personnel to allow for rectification of the issues found and the work completion feedback needs to be checked based on new data collected. This part of the process can be handled by existing IT systems and is therefore not considered within this research. It can be assumed that a similar workflow would be desirable for any manufacturing operation and not only for an automotive body shop.

The attempt to create a fully automated system from the data collection through knowledge discovery and the presentation of improvement suggestion goes against the warnings of Holzinger et al. (Holzinger 2013) who argue that an approach diminishing the end-users control and comprehension will lead to modelling artefacts.

1.2 Research Question

How can Heuristic Algorithms, Process Discovery and Machine Learning be applied to industrial equipment logs to improve the efficiency of the assembly and joining processes?

1.3 Motivation

This research has been motivated by the ongoing struggle of the manufacturing industry to improve production processes and to reduce equipment down It is hypothesised that the recording and analysis of readily available equipment data, such as actuator initiation and the corresponding proximity sensor feedback, will yield insight that can help achieve above goals.

1.4 Contributions

- A new novel algorithm to derive tags (PLC addresses) of interest with nomenclature based parsing of the PLC software without a priori knowledge.
- New methodologies to discover faults within an industrial equipment log together with a description of potential root causes and recommendations on how to rectify them.
- Multiple innovative algorithms that allow for the clustering of the event log into cases without a priori knowledge along with a new algorithm to filter noise while determining mean cycle times.
- The formulation of five novel rules to create additional engineered features for industrial equipment logs that allow for the creation of a machine learning model to determine the completeness of the cases within the log.
- A novel framework consisting of eight definitions that aid the discovery of shortcomings that typically can be found within the sequence of automated equipment accompanied by methodologies that allow for their automated annotation within sequence charts.
- The development of an improved Process Discovery method to model automated manufacturing processes based on sparse industrial logs.
- A novel methodology to interactively induce traces, through signal delays, that enable the discovery of a highly accurate process model by revealing the true dependencies.

1.5 Scope

The scope of this research is focused on automated, PLC controlled assembly and joining equipment used for industrial manufacturing processes. Although the presented research is centred around individual manufacturing stations the principals are scalable to suite manufacturing cells or the whole manufacturing process.

1.6 Aim

The aim of this thesis is to investigate new novel approaches to automatically capture, preprocess and analyse low level manufacturing equipment data and identify areas to improve the efficiency of the assembly and joining processes.

1.7 Objectives

- Develop a framework that, based on the PLC code, can extract the data points of interest which are then used for automated OPC setup and data logging.
- Formulate algorithms that can be used to cluster the events stored within the log into cases that describe one cycle of the monitored equipment.
- Develop criteria that can be used to evaluate the quality of the log.
- Utilise machine learning to determine the completeness of above obtained cases.
- Encode expert knowledge into heuristic algorithms that autonomously can generate diagrams pointing out improvement potential within the current process.

1.8 Outline Of Thesis

The project, in its final stage, consists of the five sub-projects data collection, preprocessing, knowledge-based discovery, process model discovery and interactive trace induction. Therefore, each of the following chapters has sections relating to these five sub-projects.

Chapter 2 represents a comprehensive literature review to the above topics, highlighting the fact that this research is addressing gaps identified for each of them. This review is followed by chapter 3, which gives a more in-depth overview of the technologies and concepts available to achieve the objectives outlined previously. Additionally, the corresponding, novel methodologies are introduced and detailed. In chapter 4 these methodologies are put to the test by applying them to equipment within an automotive manufacturing body shop and worst case artificial logs. The results of these case studies are discussed in chapter 5. The work is rounded up by the conclusion and an outlook of future works in chapter 6.

2

Related Works

The related works chapter has been structured to simplify access to the works that relate to the different components of the proposed framework. Section 2.1 reviews research pertaining to data collection for PLC-based equipment. Potential preprocessing methodologies for the obtained log are then discussed in section 2.2. Topics related to knowledge-based discovery are reviewed in section 2.3 prior to Process Model Mining approaches being addressed in section 2.4. Finally, research papers that justify the research into interactive trace induction are cited in section 2.5.

2.1 Data Collection

At the 2011 Hannover trade fair the concept 'Industry 4.0', an initiative of the German Federal Ministry of Education and Research and the German Federal Ministry of Economic Affairs and Energy, was introduced. The term is meant to mark the fourth industrial revolution after mechanisation, mass production and high-level automation. Today Industry 4.0 is a buzz word mostly used in Germany to describe industrial digitisation efforts to improve product quality as well as manufacturing efficiency. In their article Kagermann et al. (Kagermann, Lukas, and Wahlster 2011) referred to cyber physical systems (CPS), a term commonly used in the rest of the world to describe this industrial revolution. This literature review explains the different concepts and

technologies which can be found in CPS. It also mentions the predating concepts of supervisory control and data acquisition (SCADA) and manufacturing execution systems (MES) while showing the need for data acquisition from industrial manufacturing equipment to fuel these developments. Unfortunately, none of the papers reviewed provides any methodologies that allow for automatic, low-level equipment data point selection and event data collection.

Collection of industrial data has been practised well before the announcement of the Industry 4.0 initiative. IEC 62264 describes the automation pyramid, as shown in figure 2.1.1. On the base of the pyramid, the sensors and actuators of the manufacturing equipment can be found. These devices are connected through a field bus system with a programmable logic controller. This controller is responsible for managing the machines sequential cycle as well as alarm processing and communication to peers and the upper-level systems. Although the pyramid shows SCADA, MES and enterprise resource planning (ERP) stacked on top of each other different researchers suggest that today's structure is more intertwined than that. Waschull et al. (Waschull, Wortmann, and Bokhorst 2018) point out that the role of MES is changing. Up to recently low-level data acquisition was mostly the domain of SCADA systems which are responsible for status monitoring, displaying information on human-machine interfaces (HMIs), human-machine interaction and controlling of system-wide parameters like energy consumption. According to Waschull et al. there is no longer a clear separation between ERP & MES although ERP has nothing to do with data collection from the shop floor. Therefore they suggest an adjusted automation pyramid where ERP and MES share the top level.

Low-level data acquisition is unnecessarily complicated due to the significant number of proprietary field bus systems in use. Therefore Ungurean et al. (Ungurean, N. C. Gaitan, and V. G. Gaitan 2014a) propose a SCADA system with different middle-ware technologies which can easily be adapted to new field bus systems. The adaptability is achieved through a newly defined interface that allows the different middle-ware technologies to exchange data based on a publish/subscribe paradigm. The attributes of the data exchanged are an individual ID, the data point value and its type, a timestamp for the time at which the status was recorded and an error code. Abbas et al. (Abbas and Mohamed 2015) wanted to lay the groundwork for an OPC based SCADA system through the internet. Their extensive literature review showed that none of the reviewed methodologies allowed for real-time communication. This led to the conclusion that more research is needed in that area. Vrignat et al. (Vrigant et al. 2018) proposed a framework on how different SCADA applications could be realised. They created an example application using a Cogent DataHub which acts as a gateway and data logger

while allowing for secure remote access. The applications derived are an Excel-based viewer and an email client. For direct data access, Matlab with the OPC- toolbox was used. Although the systems proved to be viable, it still required manual selection of the data points of interest.

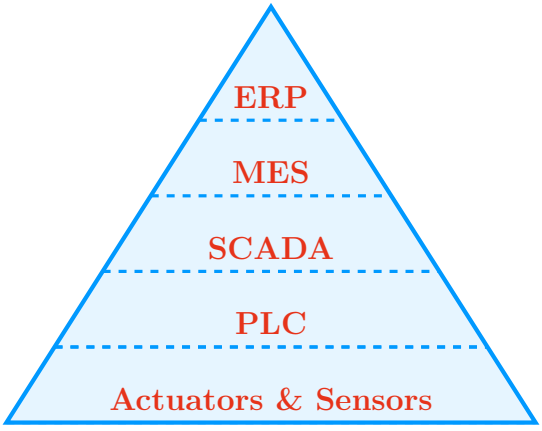

Figure 2.1.1: Automation Pyramid (IEC 62264)

The term cyber physical system can apply to any product or service distributed over a physical and a digital layer. Monostori et al. (Monostori et al. 2016) stated: 'Cyber physical systems (CPS) are systems of collaborating computational entities which are in intensive connection with the surrounding physical world and its on-going processes, providing and using, at the same time, data-accessing and data-processing services available on the Internet.' Lee et al. (J. Lee, Bagheri, and Kao 2015) explain that CPSs are based on the 5C structure. The first 'C' stands for **smart connection**. It is expected that all modules of a CPS describe their interface, which allows for automated integration into the systems communication network to enable data acquisition. During the **data-to-info conversion,** the acquired data is analysed using data mining methodologies and the knowledge gained is made available within a digital hub; hence, the term **cyber**. So far, the process is very similar to the process known from SCADA systems where the knowledge gained is made available for human review. From this point on, the CPS differs from a SCADA system. It offers an automated **cognition** layer instead, which consists of algorithms that recognise problems and potential for improvements. Ideally, this information then is fed back to the physical equipment for adjustment of its **configuration**. Lee et al. demonstrate in their paper how such a system could be used to predict the health of equipment by monitoring system variables as well as additional vibration sensors. To gain knowledge from the data obtained, they created machine learning models for clustering.

According to Rosen et al. (Rosen et al. 2015), the goal of CPS is to develop autonomous machines which are capable to react fast to unforeseen events. This goal can

only be achieved with realistic models and a link to the real machine, which allows obtaining its current state. Rosen et al. point out that within a CPS, simulation is not only needed during the design phase but also production. Abnormal equipment status data can be fed into the simulation model to either predict a future breakdown or to offer alternative processes that enable continuous production of high-quality products. The autonomy of the production process is achieved through the use of RFID (radio-frequency identification) technology to store data that needs to be accessed constantly and contact memory to store required manufacturing steps and recipes for the product. Although the use of models to predict breakdowns seems to be a viable option, the research done for this thesis shows that the models either do not exist at all or they are to corse and incomplete to be of any benefit. Schlechtendahl et al. (Schlechtendahl et al. 2015) see OPC as an enabler for cyber physical systems. Unfortunately, OPC is not capable of automatic server discovery which leads Schlechtendahl et al. to suggest a CPPS (cyber physical production system) information server. Such server provides information on the different communication channels available similar to a DNS (domain name server) known from the internet. Also, Schleipen et al. (Schleipen et al. 2016) see in OPC a good foundation for cyber physical systems. They emphasise that CPS must be aware of and communicate with their partners. This stipulation requires them to possess a self-description and the capability to configure, optimise and heal themselves. As proof of the concept, different OPC use cases are presented.

Zhang et al. (Y. Zhang, G. Zhang, et al. 2015) criticise a 'lack of timely, accurate and consistent information' from the enterprise to the shop floor and machine level, which makes it difficult to make decisions. They see the solution in the internet of manufacturing things (IoMT) where all manufacturing things interact with each other. It is their perception that self-identification of the equipment, as well as the product identification through RFID, is a basic necessity to achieve this goal.

Rosen et al. (Rosen et al. 2015) describe in their paper that the NASA used to create physical duplicates for all of their mission-critical equipment which was considered to be a 'twin'. This twin would remain on earth during their missions, and it would be used to develop solutions for problems experienced in space. Based on technological advances the NASA shifted to a 'digital twin' to replace the physical twin starting in 2010. Qi et al. (Qi and Tao 2018) state that a digital twin 'serves as a bridge between the physical world and the cyber world'. They describe the digital twin as a system that provides insight that can be used for design, manufacturing planning, manufacturing and maintenance rather than a model. A comparison between big data and a digital twin for the analysis of structured, semi-structured and unstructured data is presented. Big data is perceived as being capable of discovering trends rather than

dealing with real-time data. The paper concludes that this difference makes the two methodologies complementary to each other. This thesis argues that a digital twin is a building block of CPS rather than a system in its own. Grieves et al. (Grieves and Vickers 2017) confirm this view by defining a digital twin as a digital copy of real-world equipment. They explain that initially, the digital representation was limited to computer-aided designs which only required a static model. Today's advanced models allow for dynamic simulation of complex systems that not always operate flawlessly and sometimes produce flawed products or fail without warning. According to Grieves et al. system behaviour can be divided into four categories: The first two are predicted desirable (PD) and predicted undesirable (PU) which both have been traditionally considered during the design phase. Unpredicted desirable (UD) system behaviour comes as a pleasant surprise. It is the unpredicted undesirable (UU) events that often caused catastrophic breakdowns in the past. The reason for not addressing these shortcomings earlier is that system testing was done by experienced engineers who intentionally avoided known, problematic situations to prevent costly damage. Thus irrational behaviour was not considered during testing. Simulation with a digital twin eliminates this risk and allows for the discovery of potentially costly flaws. Grieves et al. also point out the difficulties in creating a realistic digital twin because often information is not shared between the different departments involved with the design, build and operation of the equipment. Available computing power is listed as an additional limiting factor.

Boschert et al. (Boschert and Rosen 2016) put even more emphasis on the importance of simulation during the operation phase. They mention that models with different granularity might be needed depending on their purpose. The importance of an ever-evolving model that incorporates data of the products whole life cycle from design to disposal is stressed. Uhlemann et al. (Uhlemann et al. 2017) describe that these data consist of a non-volatile component, including data obtained during the design phase, and a volatile part that needs to be obtained in near real-time during operation of the product. The paper points out the fact that many small and medium-sized enterprises (SME) lack competence concerning matters of Industry 4.0. It is suggested that this problem can only be overcome through the development of learning factories which at the time the paper was written did not incorporate the digital twin concept yet. To enable data collection of automated and non-automated production processes, the usage of locating, such as indoor GPS and image processing systems is proposed. Finally, artificial intelligence could be used to gain insight and to derive optimisation concepts. In a second study, Uhlemann et al. (ibid.) found that few companies collect data and only a fraction of those end up implementing the knowledge gained. As one of the reasons, they mention that optimisation is not a core competence of simulation

and that more research in that field is required.

Modern field bus systems, like IO-Link as described by Heynicke et al. (Heynicke et al. 2017), allow for very detailed status information to be retrieved directly from the sensor through a network connection. The data available are not limited to binary on/off indicators but also include parameters for temperature, sensing range and sensor maintenance requirements. Creating a log of all the data available promises to provide the foundation for predictive maintenance systems. Unfortunately, as Hoffmann et al. (Hoffmann et al. 2016) also point out, the majority of automation systems, installed within manufacturing facilities today, are not based on such technology creating the need for alternative data logging approaches. Felser et al. (Fesler and Sauter 2002) show that the development of field buses started in the early seven and experienced a boom during the eighties. Since there was a lack of standards, many companies attempted to make their own developments standard. At first, this was a problem on a national scale but soon is escalated into a continental and finally a global issue. Unfortunately only the best-marketed systems survived leaving some of the better-suited systems in their wake. Because no suitable compromise could be found 18 wired field bus technologies and several wireless solutions were included in the standard, according to Wilamowski et al. (Wilamovski and Irwin 2016). The successful operation of any production process depends on a well-designed and reliable communication system. Wilamowski et al. postulate that 'the users do not need to know about the internal design of a node; they only have to know about the functions that a node offers'. Sauter et al. (Sauter 2010) sum the field bus development up into three evolutionary steps: field bus systems, industrial Ethernet and industrial wireless networks.

Defining for field bus networks is their relatively small packet size and the requirement for real-time capability. Contrary to office networks, the life expectancy of an industrial network is 10+ years. That fact makes the backward compatibility of new network technologies an essential requirement which leads to a steady progression rather than big leaps.

Hoffmann et al. (Hoffmann et al. 2016) see the need to allow upper-level systems access to field-level data. Such attempts are often hampered because many of the field-level communications are proprietary. As a solution, OPC is offered, which allows mashing of data from different sources. According to Schwarz et al. (Schwarz and Boercsoek 2007) the first OPC- standard was released in 1996, but it has not been very highly regarded by academia. Within OPC, the first version called OPC-DA (data access) was based on the Windows framework as described by Veryha et al. (Veryha 2005) and should be differentiated from its successor OPC-UA (unified architecture), as explained by Reboredo et al. (Reboredo and Keinert 2013) as well as Schleipen et

al. (Schleipen 2008). Hoffmann et al. (Hoffmann et al. 2016) add that there are often problems with the firewalls because of the Windows-based OPC-DA design. This issue has been overcome with OPC-UA. To allow for the inclusion of a legacy OPC-DA based system into an OPC-UA system, Hoffmann et al. present a wrapper framework.

An in-depth data acquisition setup is proposed by Haubeck et al. (Haubeck et al. 2014) that allows for monitoring of PLC in- and outputs through OPC-UA. They conclude: 'To obtain an additional value of the data, the signals must be enriched with semantics to become automatically interpretable.' Gonzalez et al. (Gonzalez et al. 2017) also suggest using OPC for the integration of sensors. Their setup is geared towards R&D (research and development) and education rather than industrial applications. It uses hardware in the loop to control a simulated plant. Oksanen et al. (Oksanen, Piirainen, and Seilonen 2015), on the other hand, provide a framework for accessing OPC based data of mobile systems through the internet.

Mizuya et al. (Mizuya, Okuda, and Nagao 2017) recommend using OPC for control applications only. Monitoring should be done with a lighter weight protocol like MQTT (Message Queuing Telemetry Transport). They demonstrate their hypothesis by developing a custom gateway that includes an OPC client as well as an MQTT publisher to connect to a selective compliance articulated robot arm (SCARA) that only provides a USB (universal serial bus) interface. They fail to prove the benefits of such a setup. Huang et al. (Huang et al. 2017) see the need for a service-oriented architecture (SOA) to be used in industry and consider OPC as a integrate part of it. They point out that the standard function block descriptions in use do not support real-time data monitoring and therefore recommend an extension to achieve this.

Pethig et al. (Pethig et al. 2012) consider the data collection through OPC as rather old-fashioned. They propose instead the use of purpose built data loggers that listen to the network traffic to extract the information required. The main benefits of such a solution are the independence from the proprietary systems, that can be found within the different automation components, and the synchronisation accuracy of less than 10ms. Although Pethig et al. proclaim that their system can work with any field bus system, the solution proposed is limited to Ethernet-based protocols. Another drawback is that the status information is hidden somewhere in the logged protocol. This limitation means that an ontology is needed that allows decoding this protocol and extracting the desired data. The paper does not go into detail about how complex such a task would be.

Based on the literature review it can be concluded that OPC, due to the vast number of application communication modules available for the different proprietary field bus

networks, it is the most suitable out of the box solution currently available. It can be used for monitoring and control of industrial machinery.

IoT (internet of things) and IoMT are the vision of self-aware devices that can, without manual configuration or programming, seamlessly be added to or removed from networks. According to Houyou et al. (Houyou et al. 2012) this allows for rapid integration of a large number of devices because the modules can just be plugged together similarly to the plug and play concept known from computers. Breivold et al. (Breivold and Sandström 2015) point out that IoT for industrial automation has so far not been very well researched. They perceive that there are many open general, automation specific and industry-specific challenges. Through their literature review, they attempt to offer potential solutions for those issues. Ungurean et al. (Ungurean, N. C. Gaitan, and V. G. Gaitan 2014b) see OPC also as an enabling technology for the IoT concept. The resulting perception is that IoT should be considered as a technology applicable to the CPMS (cyber physical manufacturing system) concept.

It is the authors believe that an automated data point selection for equipment monitoring can only be achieved by parsing the program controlling the system. Falcione et al. (Falcione and Krogh 1993) proposed an algorithm to convert Siemens ladder logic to a sequential function chart (SFC) for comprehension and discovery of the design intent. They introduced the concept of simultaneity and dependency graphs but failed to explain the algorithms behind it. Such an explanation would have been beneficial for this thesis because these are the dimensions needed to create a meaningful process model. The SFC resulting from Falcione et al. research still lacks some detail, and it remains unclear if it really is correct. Anand et al. (Anand 2009) proposed using the more powerful context-free grammar instead of regular expressions (Regex) when trying to extract the design intent from a PLC program. They define a grammar that allows for the conversion of the L5K, a textual representation of a RSLogix 5000 program, into a format that can be parsed. Within this format, internal coil relays are replaced with their corresponding physical outputs. This process is taken to a detail which makes the non-graphical result very hard to understand. A potential shortcoming of the approach might be that typically the sequence controlling part of a PLC program only constitutes a fraction of the whole program. It remains unclear how this part could be filtered out of the code. In his book, Levine (Levine 2009) describes Flex & Bison, a Unix text processing tool, which is mainly used to develop compilers. Flex, which is the scanner generator, divides the input into meaningful chunks by associating rules with Regex. The Regex matching is done in parallel, which makes matching thousands of expressions as fast as matching a single one. Bison, which is the parser generator, uses context-free grammar to determine the relation of the tokens and to create a parse

tree. Zhang et al. (Y. Zhang, Lu, and B. Yang 2017) recognised that this approach might be suitable for 'checking the correctness of safety-critical systems and improving the efficiency of software developers'. In their paper, they show the proposed Regex and context-free grammar but do not present the resulting parse tree. They also leave open how that tree can aid software checking.

Feldmann et al. (Feldmann and Colombo 1999) discuss in their paper 'feature-based monitoring using the information contained in the process interface and the logic control structure' without going into details of how to extract the features from the logic control structure and how to access the data within the controller. Also, (J. Lee, Bagheri, and Kao 2015; Palluat, Racoceanu, and Zerhouni 2006; Phaithoonbuathong et al. 2010; Ouelhadj, Hanachi, and Bouzouia 2000) propose in their papers systems for monitoring PLC-based production system. Their focus is mostly on real-time fault detection based on specialised frameworks and algorithms. Fleischmann et al. (Fleischmann et al. 2016) explain that decentralised condition monitoring is standard, but they believe that centralised monitoring is needed for model creation. Nicola et al. (Nicola et al. 2017) demonstrate the data collection through OPC using Labview and MySQL, an open-source relational database management system, as an experimental proof of concept. Their aim is the evaluation of the system response profile.

A more detailed approach is offered by Wan et al. (Wan et al. 2017), who suggest that 'manufacturing big data can be divided into three types: device data, product data, and command data'. They reason that a combination of equipment data and alarm messages can be used for breakdown prediction. Similarly to this thesis, a framework, where knowledge gained is fed back to maintenance, is recommended. As middle-ware OPC, according to Wan et al. is a natural choice. In their experiment they focus, con trary to this research project, on industrial wireless data transmission without going into detail on what data should be monitored how. The experiment is used to estimate the remaining lifetime of a machine tool and shows that the prediction is closer to the real-life expectancy than an average estimate. Theorin et al. (Theorin et al. 2017) also see OPC as a universal solution to access devices over a network. They are introducing the Line Information System Architecture (LISA) which aggregates events based on ID, timestamp and attributes. A key feature of LISA is that it correlates start and stop events which are needed for model creation. It is perceived that such events are fired once per sequence, which is, as this research shows, not necessarily true.

Similarly to Wan et al., Nemeth et al. (Nemeth and Peterkova 2018) also recommend the use of an open-source OPC server in combination with a MySQL database. The recording of alarm messages for the correlation with sensor values is suggested. Their data collector is custom written in C#. The experiment introduced only seems to

analyse the number of different errors that occurred. Common to all above-reviewed papers is that a domain expert manually chooses the data points to be monitored through a software interface.

The above literature review shows that there are many different approaches to collect and analyse industrial equipment data. It also points out that OPC as middleware is the best choice to access field-level data because it eliminates the issues caused by the different field bus systems. The review also cites some researchers that parse PLC code with the help of regular expressions or similar methods to extract the information they require. Finally, the use of a relational database seems to emerge as a favourable solution to store data obtained from a PLC through OPC. Although all these components have previously been applied, there is no evidence for any research bringing these technologies together to automatically extract the data points of interest from a PLC program, set up the corresponding OPC connections and store the events in a SQL database. This work aims to fill this gap by proposing an 'Automated, Nomenclature Based Data Point Selection For Industrial Event Log Generation'.

2.2 Log Preprocessing

Once the data has been obtained the next logical step is preprocessing. During this step related data is grouped together, its quality assessed and its flaws potentially repaired.

Van der Aalst et al. (W. V. D. Aalst, Adriansyah, et al. 2011), who dominantly established the domain of business model discovery, defined the event log, needed for Process Discovery, as a sequential record of events. Each event refers to an activity, which is related to a particular case. Also, the records can include information about the resource, a timestamp for the event and additional data elements. Van der Aalst continues by emphasising that the quality of the event log is directly related to the quality of the Process Discovery result. Because of this relationship, the event log should have the highest priority.

Data evaluation, cleansing and repair, has been a priority of data analysts very early on. There have been many works that address different problems typically experienced when analyzing data like the ones presented by Rahm et al. (Rahm and Do 2000) and Raman et al. (Raman and Hellerstein 2001). Kim et al. (W. Kim et al. 2003) were among the first to compile the possible problems pointed out in different previous papers, together with their proposed solutions, into a single 'Taxonomy of Dirty Data'. This taxonomy provided the data analysts, for the first time, with a comprehensive 'checklist'.

Almost a decade later, Gschwandtner et al. (Gschwandtner et al. 2012) extended this work into 'A Taxonomy of Dirty Time-Oriented Data'. This paper named more potential problems collected from different previous papers, thus providing an more comprehensive checklist. They, contrary to Kim et al. neglected to describe possible solutions to those issues.

With the advance of Process Mining, more and more researchers attempted to adapt the general data mining problem awareness to the Process Mining domain. In their papers, Yang et al. (H. Yang, Hofstede, et al. 2010), Hee et al. (Hee, Z. Liu, and Sidorova 2011) and Yang et al. (H. Yang, Wen, and J. Wang 2012) focused on the completeness of an event log. A log is deemed complete if the number of traces in the log covers all the possible traces within the process model that generated the events.

Rogge-Solti et al. (Rogge-Solti et al. 2013) were first to provide research into the incompleteness of event logs and how to remedy the issues discussed using stochastic and machine learning approaches. They are mainly focusing on compensating for missing events and timestamps by determining the best matching path through a given process model and filling in the voids accordingly.

Bose et al. (Bose, Mans, and W. M. P. Van Der Aalst 2013) followed the example set by Kim et al. and Gschwandtner et al. and compiled a comprehensive checklist of issues that can potentially be experienced when dealing with event logs for Process Mining. The outcome of their research was compressed into a 9 x 4 matrix with a total of 27 different problems to consider. Just like Gschwandtner et al. they also focused only on pointing out the issues without providing solutions for them.

Verhulst (Verhulst 2016) summed up all the potential quality issues once again and added an example and a possible implementation approach. Many of those approaches then also were implemented as plugins for the ProM Process Mining tool. Verhulst sums his work up by stating: 'But there is no 'always-working-solution'-method to ensure data quality. This means that no specific method that works for all cases exist.'

Instead of focusing on potential log problems, Kherbouche et al. (Kherbouche, Laga, and Masse 2017) defined several metrics in the following categories:

- Structural complexity measurements
- Behavioral complexity measurements
- Precision measurements
- Availability measurements

Based on those metrics, they designed a ProM plugin, which will evaluate a given log using predefined thresholds and outputting not only the numeric results but also

recommending a suitable Process Discovery approach. The authors fail to disclose how the thresholds were defined and how the mining algorithm recommendation is derived. Also, contrary to Verhulst, they do not offer any suggestions on preprocessing approaches to improve the log quality metrics.

Suriadi et al. (Suriadi et al. 2017) argue that there 'exists temporal constraints among events, both from a case perspective and a resource perspective'. They conclude that the traditional approaches for data evaluation and cleansing, described previously, do not necessarily hold because of these constraints. This view is contradicted by their use of the Bose et al. (Bose, Mans, and W. M. P. Van Der Aalst 2013) issue matrix for their research work. Based on the assumption that many of the log issues are systematic errors Suriadi et al. propose the definition of generic patterns, which make it easier to identify the problems. They then go on to present possible solutions to remedy those problems without actually offering a possible technical approach.

A fundamental requirement, for Process Discovery, is that the events can be associated with cases. This association often is not part of the event log. As mentioned previously, in production equipment event logs, there is a meagre number of traces. If these are known, it is trivial to correlate events into cases. Essentially a trace is a pattern which could be discovered using algorithms originating in the sequence mining domain. Such algorithms often are used in bioinformatics when deciphering molecules or for discovering typical customer purchasing habits.

As Fournier-Viger et al. (Fournier-Viger et al. 2017) describe in their paper, mining for patterns within sequences has been of interest for decades. Therefore hundreds of papers have been written in this area. It is not intended to present another review of these papers thus only a few of them are mentioned to show the long history as well as the approaches believed to be most applicable to the problem at hand. Dietterich et al. (Dietterich and Michalski 1983) presented early on an inductive learning approach named SPARC/E (Sequential PAttern ReCognition / Eleusis). Its task was to discover rules that describe sequences and thus can predict their continuation. The paper is very self-critical by stating that this approach can only be seen as a first step and that there are many possibilities for improvement. The main downfall for the intended application is that this is a heuristic approach which prunes data deemed irrelevant. This method can lead to flawed results.

Contrary to that, Agrawal et al. (Agrawal and Srikant 1995) use what could be called a reasoning approach. They introduce the AprioriAll, AprioriSome and DynamicSome algorithms. All of them reason that 'if a small pattern does not exist, there can not be a bigger pattern that includes such small pattern'. The goal behind this

method is to reduce the processing time of the algorithms compared to a brute force approach. Srikant et al. (Srikant and Agrawal 1996) improved upon those algorithms by introducing GSP (Generalized Sequential Pattern algorithm), which showed much better performance than the previous three proposals. Zaki (Zaki 2001) developed the SPADE (Sequential PAttern Discovery using Equivalent Class) algorithm which generates possible subsequences and then checks if they can be found within the data set. Again, if a small sequence cannot be found all possible combinations that would include this small sequence are also excluded. The advantage of all those proposals for this application is that if the number of sequences to be discovered is finite, the results will be optimum.

Burattin et al. (Burattin and Vigo 2011) preclude that the case ID must have been uninten ally recorded as an attribute together with the events. They propose different filters to determine the most likely case ID candidates. The selection can be made either automatically by determining which of the attributes has been observed most often or through expert users who possess the required domain knowledge to exclude less promising attributes.

Walicki et al. (Walicki and Ferreira 2011) dissect the log into patterns without repeating symbols which are then arranged to form a case-based event log. The solution with a minimal set of patterns is selected, as it can provide a more compact representation of the generating process.

To address the problem of missing case identifiers Pourmirza et al. (Pourmirza, Dijkman, and Grefen 2015) introduced the correlation miner. It consists of three rules that allow for a probabilistic determination of an event's case-association. First, the orchestration graph rule is shown, which assumes that the number of occurrences of a specific event is equal to the number of cases recorded. Unfortunately for production equipment event logs this assumption not always holds as some motions may occur more than once during a cycle for a specific part. The authors later on also clarify, that in order for their algorithm to work, all loops need to be removed from the event log, which is impracticable. Next the concept of the 'Precede/Succeed matrix' and the 'Duration matrix' is introduced. In both matrices, probabilities are assigned and compared to a chosen threshold. Based on these values, an event's case association then is determined.

Bayomie et al. (Bayomie, Helal, et al. 2016) reason that a causal behavioural profile can be generated from an original process model. Time heuristics are applied to this profile to create a decision tree with the nodes representing events. Each node is annotated with a probability of it belonging to its parent note. The nodes with the

highest-ranking probabilities form a labelled event log. This original approach was extended to become applicable to cyclic process events Bayomie et al. (Bayomie, Awad, and Ezat 2019). Finally, the experiments were repeated with less perfect a priori process models while still achieving an accuracy of 90% by Bayomie et al. (Bayomie, Ciccio, and Rosa 2019). The proposed methodology is very similar to the approach described by Helal et al. (Helal, Awad, and Bastawissi 2015) although they used slightly different heuristics.

Andaloussi et al. (Andaloussi, Burattin, and Weber 2018) also assume that the case ID is one of the attributes or a combination of attributes logged together with the events. Their approach is to choose one of the attributes to become the case ID before performing the process model discovery. The resulting model is then evaluated using the metrics fitness, precision, generalisation and simplicity together with the case ID. This procedure then allows for the best performing model to be chosen.

Djedovic et al. (Djedovic et al. 2019) propose that events, belonging to the same case, poss similarities. They recommend the creation of a similarity matrix based on an F-score across all recorded attributes. After determining the first event within the case, each event is followed by the next, most similar event.

Yang et al. (H. Yang, B. V. Dongen, et al. 2012) recognise that the quality of the Process Discovery result strongly relates to the degree of completeness of the event log. They, remind the reader that the completeness of an event log depends on the type of information expected. A Possible measure for completeness is the number of traces observed in the log versus the possible traces and how well the discovered model covers the behaviour observed. Based on that, an estimator can be used to determine how much of the potential behaviour is captured by the event log.

Ayo et al. (Ayo, Folorunso, and Ibharalu 2017) argue that 'the fitness of the reproduced model is a function of the event log completeness'. To detect and fix cases of incompleteness, they recommend the use of Bayesian Scoring Functions in conjunction with causal matrices before applying genetic mining algorithms. They conclude that this approach can discover complete models where the established algorithms failed.

Another approach to reduce the complexity of the discovered model and to improve Process Discovery results is the clustering of similar cases or traces. Song et al. (Song, Günther, and W. M. P. Van Der Aalst 2008) propose trace clustering, based on a distance calculated for any two cases, to divide the log into homogeneous subsets that are mined independently. This methodology is meant to improve the mining results, especially for highly flexible processes. A similar approach is recommended by Accorsi et al. (Accorsi and Stocker 2011). They, calculate the similarity on the distance

between pairs of activities. Ceravolo et al. (Ceravolo et al. 2017) confirm in their work that 'clustering is considered one of the most relevant preprocessing tasks as grouping similar event logs can radically reduce the complexity of the discovered models'. Their clustering approach computes the probability distribution of the observed activities in specific positions with the help of statistical inference.

Although many researchers in the area of Process Discovery, like van der Aalst (W. V. D. Aalst 2016a) and Yang et al. (H. Yang, Hofstede, et al. 2010), agree that a flawless event log is the best starting point for process model discovery, very little literature about the subject of repairing event logs was found. It appears that there is more focus on repairing the discovered model instead, as described by Fahland et al. (Fahland and W. M. P Van Der Aalst 2015) and Polyvyanyy et al. (Polyvyanyy et al. 2017). The works that were reviewed concentrate mostly on identifying the need for repair rather than the repair process itself. There seem to be two mainstream directions within the papers reviewed. Some researchers like, van Hee et al. (Hee, Z. Liu, and Sidorova 2011) and Yang et al. (H. Yang, Wen, and J. Wang 2012), take a statistics-based approach to determine where within the log repair is required. Rogge-Solti et al. (Rogge-Solti et al. 2013), Wange et al. (J. Wang et al. 2015) and Bertoli et al. (Bertoli et al. 2013) seem to favour a model-based approach, where a model is generated first and based on flaws found within the model conclusions on the faults within the log are drawn.

Luengo et al. (Luengo, Garcia, and Herrera 2012) suggest that the problems most often found in logs are missing values. As the simplest solution to deal with missing values, they propose discarding the sample that contains them. This practice will not be critical as long as only a small percentage of the available samples exhibits missing values. They summarise that 'in most cases, a data set's attributes are not independent of each other. Thus, through the identification of relationships among attributes, MVs (missing values) can be determined.' This process is referred to as imputation. Luengo et al. categorise the available imputation approaches into rule learning algorithms, approximate models and lazy learners. Within their work, they evaluated 'twenty-three classification methods and fourteen different imputation approaches to missing values treatment'.

Ly et al. (L et al. 2012) also seem to favour discarding flawed samples. Their approach of cleaning the logs, based on domain specific constraints, is termed semantic log purging. Ly et al. reason 'that the quality of a mined process is high when an assessment concludes that: (i) Infrequent traces are correctly included in the result. (ii) Parallel branches are correctly identified. Logs, stemming from processes which include late modelling yield meaningful results (a small set of branches). (iii) Ad-hoc

changed instances stemming from manual repair are not incorporated in the process model.' This approach does not discriminate infrequent traces which is intended to improve the mining quality. Dealing with parallel execution is deemed to be one of the most challenging tasks for mining algorithms. It is therefore recommended to use the constraints to separate the branches to mine them individually.

Waljee et al. (Waljee et al. 2013) also examined the performance of existing imputation methods by randomly removing data from a previously complete data set. They were mainly focusing on the four methods MissForest, mean imputation, nearest neighbour imputation and multivariate imputation by chained equations (MICE). Although the 'MissForest had the least imputation errors for both continuous and categorical variables at each frequency of missingness' they warned that any repair approach could introduce systematic errors and thus impact the mining result.

Moritz et al. (Moritz and Bartz-Beielstein 2017) explain that most time series imputation algorithms rely on the correlations that can be found between the event attributes. Univariate time series only possess one attribute which makes such an approach impossible. Instead, Moritz et al. proclaim that time dependencies need to be explored. To address missing values, they recommend the use of the 'imputeTS' package within the R environment.

Che et al. (Che et al. 2018) acknowledge that a variety of statistical solutions have previously been introduced that allow for the imputation of missing values in a time series. With their work, they want to point out that new developments in the field of Recurrent Neural Networks (RNNs) have led to methodologies that achieve state-of-the-art results in many applications with time series or sequential data. Contrary to using artificial intelligence for the log repair Dixit et al. (Dixit et al. 2018) favour the use of domain knowledge and expert involvement. Their algorithm uses 'relevant indicators to detect ordering related problems in an event log to pinpoint those activities that might be incorrectly ordered'. These potential issues are presented to the user who can, based on domain knowledge, make required modifications directly in the log. The algorithm re-evaluates the changes and presents them to the user with the option to keep or discard them.

Martin et al. (Martin et al. 2019) also favour the analysts' involvement in the log cleaning and repair effort. They claim that for most Process Discovery algorithms 'the order of activities is essential, and the exact timestamp values are of secondary importance'. Therefore they are proposing a mixture of data-based and discovery-based data quality assessment which leads to the recommendation of different data cleaning heuristics for the user. Experiments showed that after several iterations between assessment

and cleaning a quality log suitable for Process Discovery is obtained.

Fortuin et al. (Fortuin, Raetsch, and Mandt 2019) propose the use of deep variational autoencoders (VAEs) and Gaussian Process (GP) to map the missing data time series into a latent space without missingness. The dimensional reduction allows for the correlation of their features and thus enables the reconstruction of the missing values.

For meaningful process data mining, it is also paramount that the events belonging together are clustered into cases. The literature offers several solutions to achieve such case clustering. The problem with industrial processes is that a significant number of traces recorded stems from different part types being manufactured after each other, which requires different setup and reset motions to ready the machine. Such complexity is not considered in the existing approaches.

It is a common understanding that the log quality is key to successful data mining. Within the above related works, many approaches to gauge the quality of the log obtained can be found. The quality matrices provided range from data analytics projects in general to Process Mining projects specifically. Process Mining is intended to be used for business processes and the application of the defined quality criteria to industrial processes has not been considered yet.

The quality criteria can be used to determine the flaws within a given log. These findings then trigger the question of what to do with the incomplete data. Within the Process Mining domain, it seems to be mainstream to counteract these problems by implementing a certain degree of fault tolerance into the mining algorithms. Alternatively, some researchers propose different repair techniques to eliminate the issues found. Both approaches bear the potential to introduce additional faults in the data. This consideration leads to the final option of not considering flawed cases while mining the log. Data classification is one of the specialities of supervised machine learning. However, the literature review did not reveal any research where machine learning techniques were used to classify the quality of logged cases. This thesis addresses this gap by proposing engineered features that, in conjunction with tags, enable the creation of a machine learning model that can be used for classification.

2.3 Knowledge-Based Discovery

Plant floor systems, as described by Lee (J. Lee 2003), were the first step towards the autonomous observation of manufacturing processes. They are logging critical parameters of the process which are used to create KPI (key points of interest) charts and to highlight potential bottlenecks. Next cyber physical systems started to emerge. Their goal, to create a digital clone of the real-life production equipment, which can be used

for simulations and deriving predictions, was also documented by Lee (J. Lee, Bagheri, and Jin 2016). Jaber et al. (Jaber and Bicker 2014) showed that predictions regarding required maintenance could also be obtained by applying machine learning techniques to vibration sensor data. The results could help to move the time of preventive maintenance closer to the predicted time of failure, thus realising additional savings. Banerjee et al. (Banerjee and Das 2012) propose a similar approach. Instead of using vibration sensors, which normally are not an integral part of manufacturing equipment, they are utilising the already available sensors for fault detection.

There also have been several machine learning and artificial intelligence-based proposals to extract knowledge from industrial manufacturing data. Hou et al. (Hou, W. L. Liu, and Lin 2003) suggested the use of a propagation neural network to monitor a PLC controlled conveyor belt manufacturing system communicating through Ethernet. The sensor would measure the process temperatures and make predictions regarding the expected product quality. The system is said to be able to reveal the root cause of manufacturing quality problems as well as potential countermeasures.

Ho et al. (Ho et al. 2006) also recommended artificial intelligence as a suitable methodology to derive knowledge regarding the product quality. Their intelligent production workflow mining system uses 'artificial neural networks and fuzzy logic reasoning to form an integrated model'. It consists of three modules for measurement, prediction and improvement. An application within a slider manufacturing facility proved this to be a viable approach. Weiss et al. (Weiss et al. 2010) compare patterns created with the help of binary regression rules. Deviations relative to the overall mean for a particular manufacturing step highlight potential opportunities for yield improvement. They applied their framework to a fab shop for wafer production.

Harding et al. (Harding, Shahbaz, and Kusiak 2006) tried to summarise potential applications for data mining within industrial manufacturing. The areas identified were production processes, operations, fault detection, maintenance, decision support, and product quality improvement. They believe that efforts should be made to develop models that allow for the exploration of all process and material data within a factory or enterprise.

Kim et al. (T. T. T. Kim and Werthner 2011) suggest that enriching an event log with external data sources could help discover previously hidden knowledge. They are proposing 'database-to-ontology mapping techniques to integrate data sources and use semantic reasoning techniques for inferring knowledge'. The ontology used is based on TOronto Virtual Enterprise (TOVE), 'an integrated ontology for supporting enterprise modelling, which contains concepts related to business models'.

Yurin et al. (Yurin 2012) use case-based reasoning to address problems experienced in a manufacturing process. Previous experience has been captured in 'form of examples of solved problems, analogical cases or references'. Group decision-making processes are used to retrieve cases that are closest related to the problem at hand. The previously tried solutions can then be extracted and applied.

Lee et al. (J. Lee, Lapira, et al. 2013) cluster manufacturing problems into two categories. The visible issues include poor cycle time, machine failure and product defects while the invisible issues include things like machine degradation and component wear. To be able to distinguish between process and machine degradation Lee et al. recommend the correlation of controller and quality inspection data with the data obtained from the machines. Correlating could be done within a predictive manufacturing system, also recommended by Windmann et al. (Windmann et al. 2015), which manages all tasks from data acquisition to visualisation of the mining results.

Djenouri et al. (Djenouri, Belhadi, and Fournier-Viger 2018) enable different perspectives on the original event log data by applying transformations and frequent item-set mining methodologies. Since such an approach spans a high number of patterns with potentially useful information, a pruning strategy is applied to derive a small set of patterns useful to decision-makers.

This part of the literature review can be best summed up by quoting Fluxicon, founded by some of the leading researchers in the Process Mining domain, stating on their website: 'Process mining is a discipline and only the process mining analyst can make these distinctions and derive the right actions from the analysis. To think that an AI algorithm can make those decisions for you is an illusion. Don't believe the self-proclaimed "thought leaders" who claim otherwise …'. In short, there have not been any attempts to derive improvement suggestions from industrial equipment log data automatically. This work aims to overcome this preconception by encoding expert knowledge into heuristic algorithms to detect and highlight inefficiencies.

2.4 Process Model Discovery

Processes can not only be found in manufacturing but also for business transactions. Van der Aalst (W. V. D. Aalst 2016b) started at the beginning of this century the development of the research field of Process Mining. The aim is to discover the underlying process model of such business transactions based on logged transaction data. Very early in the history of Process Mining, there was an attempt by Hu et al. (Hu, Z. Li, and A. Wang 2006) to apply Process Discovery techniques to 'flexible manufacturing systems'. Their goal was to provide the means to discover the actual equipment process

and allow for validation, as well as provide a basis for improvements. To achieve this, they expanded upon van der Aalst's α-algorithm by introducing resources through a second matrix. The combination of both then yielded a Petri Net reflecting the manufacturing process. Unfortunately, that seems to be the only recorded attempt to utilise Process Discovery algorithms until 2014. At that time Son et al. (Son et al. 2014) presented their paper 'Process Mining For Manufacturing Processes' in which they utilised established Process Discovery approaches to discover a process model for all steps of product manufacturing and shipping. The model is not fine grained enough for detailed equipment analysis. Yahya (Yahya 2014) considered in his research all of the manufacturing steps a product has to go through. He concluded that the process model to be created needs to be customised to the analysis' goal. At the same time, the granularity needs to be chosen accordingly. Yang et al. (H. Yang, M. Park, et al. 2014) also utilise high-level event data in their research, but they were trying to gain more insight by combining this data with unstructured data like email. Unfortunately, the paper remains unclear about what additional value that approach provides.

Farooqui et al. (Farooqui et al. 2018) chose a more detail-oriented approach. They implemented additional code into industrial robots that allowed them to extract a program pointer position, as well as new signal state events. This data was their basis for generating a log used for model discovery. The purpose of the models was to aid decision making and maintenance efforts. Nowaczyk et al. (Nowaczyk et al. 2018) take a different approach by looking at groups of peers. They propose a framework for unsupervised learning based on several similar systems. The algorithms can detect if the behavior of a single unit deviates from the rest and raises a flag. Brzychczy et al. (Brzychczy and Trzcionkowska 2018) also researched the use of low-level machine data to create logs conforming with Process Discovery requirements. They pointed out that often a key obstacle to overcome is case identification. They proposed a domain knowledge-based approach to identify the beginning and end of a case.

In a first attempt to obtain a process model from an industrial equipment log, the author applied several established algorithms from business process modelling. Namely these were the α-miner (W. V. D. Aalst 2016a), the CSM (Composite State Machine) miner (Eck, Sidorova, and W. M. P Van Der Aalst 2016), the heuristic miner (A. Weijters, W. M. P. Van Der Aalst, and Medeiros 2006), the fuzzy miner (Bose, E. H. M. W. Verbeek, and W. M. P. Van Der Aalst 2011), the inductive miner (S. J. J. Leemans, Fahland, and W. M. P Van Der Aalst 2013) all of which are implemented in the ProM framework (H. M. W. Verbeek, Buijs, and B. F. V. Dongen 2010) and DISCO (Günther and Rozinat 2012), a commercially available Process Mining tool. All of the above-mentioned Process Discovery algorithms have in common that they only consider

one timestamp per event, which in the author's opinion, is the main reason why they were not able to perform as expected. Wen et al. (Wen et al. 2009) express that by stating: 'Existing techniques for Process Mining do not consider event types, i.e., tasks are either considered to be atomic or only the completion of a task is considered. Note that the start and completion of a task can be considered as two atomic tasks when using the classical Process Mining techniques. Unfortunately, such an approach does not detect explicit parallelism.'

Further research showed that there is a second, often ignored or forgotten 'category' of Process Discovery algorithms that consider the activity life cycle. Wen et al. (ibid.) introduced the β-algorithm (to be found in ProM as Tsinghua-α-algorithm) in 2009. It overcame some of the limitations of existing algorithms (e.g. short loop in the original α-algorithm). They proclaimed that the 'information about the start and completion of tasks can be used to detect parallelism explicitly'. At the same time, they point out that the creation of a process model requires a profound knowledge of the process in question. The benefits of discovering a process model are not only seen in the use of Delta analysis but also for a better understanding of the process's execution in reality. The fact that most of the existing Process Discovery algorithms do not consider 'start' and 'complete' timestamps together is regarded as a shortcoming because it does not allow for the detection of parallelism. Within the definitions of the β-algorithm, two rules can be found that support the approach of the proposed α_{LC}-algorithm. The first is definition 10 which states: 'a is succeeded by b if and only if in at least one event trace a is "directly followed" by b, i.e., there is not another complete task occurrence in-between the two task occurrences a(ei,ej) and b(ek,el)'. If tasks intersect, they cannot be considered "directly following". The intersection is defined in definition 11 as follows: 'a intersects with b if and only if in at least one event trace an occurrence of a overlaps with an occurrence of b'. One of the prerequisites of the β-algorithm is the presence of a complete log.

In 2015, Burattin (Burattin 2015) proposed the Heuristic++ miner which is a backward compatible extension to the Heuristics Miner. It uses a statistical approach for process model discovery. Just like Wen et al. before, Burattin points out that 'without the notion of duration, it's complex to express parallelism'. Parallelism is given if two activities overlap or contain each other. "Direct following" has a slightly different definition within this work because the 'termination of the first activity must occur before the start of the second and, between the two, no other activity should start'. A benefit of the Heuristic++ algorithm is that, because of its backward compatibility, it can handle logs with a mixture of activities that are expressed as time intervals or instantaneous.

Another algorithm considering the activity life cycle is the IMLC (Inductive Miner - Life Cycle) proposed by Leemans et al. (S. J. J Leemans, Fahland, and W. M. P Van Der Aalst 2016). The goal of this algorithm is not only to discover concurrent (parallel) but also interleaved activities. Interleaving is given if two activities cannot be performed at the same time.

When creating a process discovery algorithm, the choice of tools to be used is of utmost importance. The original α-algorithm, for example, proposes a footprint matrix that is used to mark direct, parallel or non-existing dependencies between activities. Those findings are then processed in the proposed algorithms. In the industrial automation domain, a similar matrix is prevalent. It was made popular in 1981 by Steward (Steward 1981) under the name of design structure matrix (DSM) and is an extension to the precedence matrix introduced in the late 1950s. The basis of a DSM is a square matrix with equal row and column names. In 2001 Browning (Browning 2001) summed up the DSM as a 'simple, compact, and visual representation of a complex system that supports innovative solutions to decomposition and integration problems'. It is interesting to know that Browning (Browning 2002), in his next work stated: 'A process model of what actually flows must be extracted from the existing, implicit way work really gets done'. Browning recognises that there are more accessible ways, like flowcharts or Gantt charts, to visualise processes but at the same time also points out that, because of the added complexity, they often do not represent the full range of interaction among the activities.

In their 2012 book, Eppinger et al. (Eppinger and Browning 2012) also acknowledge that DSM most often is used in engineering management because 'the matrix provides a highly compact, easily scalable, and intuitively readable representation of a system architecture'. Here also the term 'binary DSM' emerges for a matrix in which 'the off-diagonal marks indicate merely the presence or absence of interaction'.

In this section of the related works, it was shown that there had been several attempts to apply Process Discovery techniques to industrial processes. All of the projects reviewed were concentrating on higher-level processes while the evaluation of signals from existing field-level devices was not considered. Since Process Discovery promises to reveal the real dependencies between the events recorded, one of the objectives of this research is to close this gap by applying existing Process Discovery algorithms to such low-level data collections. The resulting process models are then to be evaluated by comparison against the actual models which have been created manually by a subject matter expert.

2.5 Interactive Trace Induction

There have been many attempts by Process Mining Practitioners to quantify the quality of the discovered models. Van der Aalst (W. M. P Van Der Aalst 2018) sums these efforts up by explaining that event logs can only capture example cases which makes it difficult to determine the quality of the model relative to the underlying record. He proposed a set of four quality measures. The metric recall expresses how well the model reflects the behavior found in the log, while precision gauges how good the model discourages unrelated actions. The third metric is generalisation, which is supposed to make the discovered model also suitable for yet unseen cases, thus trying to prevent overfitting. Because in industrial automation, only one sequence of operations is expected for a single part style, the author of this thesis argues that overfitting is desired as long as all of the meaningful traces have been observed. The final quality indicator is the simplicity of the model. Van der Aalst, however, also warns that 'Users of existing conformance measures should be aware of seemingly obvious conformance propositions that are violated by existing approaches'.

Leemans et al. (S. J. J. Leemans, Fahland, and W. M. P. Van Der Aalst 2018) also see the need for 'strong quality guarantees as long as sufficient information is available'. This statement means that the log needs to be nearly complete, which means that most of the behaviors possible should be present. Besides the model quality, Leemans et al. also focus on the scalability of Process Discovery algorithms. Their proposal enables the discovery of models from logs with millions of events from thousands of activities.

DeWeerdt et al. (Weerdt et al. 2011) take a slightly different approach by evaluating the model quality with the help of artificially generated negative events. This aids in establishing the precision measure previously mentioned because event logs usually do not contain cases that are not allowed. The logic behind creating artificial events is that events that have not been seen at a specific position within a case before must be unrelated and thus, should not be allowed by the process model. DeWeerdt et al. point out that this approach allows for a good comparison of process models that have been obtained, based on the same log, by different algorithms. At the same time, they also caution that possible overfitting is not detected. Finally, just like Leemans et al., they want the reader to understand that 'process discovery algorithms generally include the assumption that event logs portray the complete behavior of the underlying process'.

Before continuing, the use of the terms 'variant' and 'trace' needs to be clarified. Lee (C. Li, Reichert, and Wombacher 2008) refers in his work to process variants, which he describes as the result of sequence changes applied to an original process model. His work goes into detail about the difficulties in determining the difference between such

process variants and case variants within an event log. It is these case variants that Guenther (Günther and Rozinat 2012) refers to when using the term variant. Van der Aalst (W. V. D. Aalst 2016a), on the other hand, calls the sequence, found within a case, 'trace'. In the course of this thesis, the author uses the term trace when referring to case variants or traces.

The literature review shows that there has been much concern about the quality of the discovered process model in the Process Discovery domain. Process Discovery postulates that an increased number of recorded cases, and thus hopefully an increased number of traces, holds the potential to improve the quality of the Process Discovery model. None of the papers reported any attempts to define which traces are needed to discover a high accuracy model. This research closes said gap by attempting to define rules to pinpoint the needed traces. In addition it offers a methodology that allows for the recording of these traces within a few machine cycles.

3
Methodologies

This chapter is meant as an introduction into the technologies, methodologies and algorithms required for the proposed automated framework. It encompasses all the aspects from determining the data points that need be logged to the traces that need to be captured to allow for the discovery of a highly accurate industrial process model. A description of the methodologies applicable to the proposed data collection is given in section 3.1. Criteria to evaluate the quality of the obtained log, together with methodologies that allow for the clustering of the individual events into cases and traces are introduced in section 3.2. Section 3.3 explains novel rules that can be applied to an industrial process Gantt chart to identify potential areas of improvement. Process Discovery concepts are explained in section 3.4 because they promise the discovery of the actual dependencies between the different events. Variations to one of those algorithms are proposed to reduce the impact of generalisation on the resulting process model. The capability of the altered Process Discovery algorithm, to discover complete process models from minimalistic logs, finally prompted the development of the 'interactive trace induction' concept described in section 3.5.

3.1 Data Collection

Advances in industrial automation often go with the increased complexity of the manufacturing equipment. Today an automotive manufacturing cell, consisting of several work stations, is controlled by a PLC which interfaces with dozens of process-specific controllers for robots, welding and sealing systems. Many of these process-specific controllers already have a built-in function, that collects process-related data. This data may or may not be accessible to the end-user and is not the subject of this thesis.

The work stations often have dozens of actuators and sensors. Besides, ever-increasing health and safety stipulations require that such equipment is contained, which makes the simple observation of the equipment's sequence difficult or impossible. Lean manufacturing efforts require that equipment status information is made available in a central location. To achieve this, many manufacturing facilities have implemented so-called 'plant floor systems' that collect a range of status information which then is broadcast, through display boards and radios, to the maintenance personal. They collect a predetermined set of alarm messages to display them on maintenance screens and also preserve them for statistical analysis. Also, these systems receive triggers for some predetermined events, such as the machine being blocked or starved. These signals are also predominantly used for statistics and dashboards. The main issue with both data streams is that the data has to be made available by the programmer in a defined format. Often the data generation is either neglected altogether, is error-prone or even subject to intentional manipulation.

Besides the 'plant floor systems' described above, there are also software packages available, that allow the user to manually select any of the data points for monitoring, that are accessible through OPC. This approach is not only time consuming but also requires in-depth knowledge of the system to choose the tags suitable for the intended analysis. One of the leading providers of such software was invited to implement it on a system, unknown to him, with one PLC. The vendor spent roughly 150 hours until he was able to start the data collection. Similar results could be expected for any equipment that is previously unknown to the data collecting entity since there are no global naming conven or programming standards that could simplify the task.

3.1.1 Communications Structure

The communications network within a production facility has different levels (see figure 3.1.1. The highest level on the shop floor would be the plant floor communication which links the different manufacturing cells to a supervisory control and data acquisition system. Typically this communication uses Ethernet and can also be accessed by other,

authorised computers. The cell level communication usually is within a private Ethernet network which can only be externally accessed through bridges. Besides the Ethernet connection which mainly interlinks the PLC with the robots and the HMI there also is a proprietary field level network that allows the PLC to exchange data with I/O devices. For this project, the field bus used is Devicenet. Since all the I/O status information is gathered within the PLC, external access can be achieved via OPC.

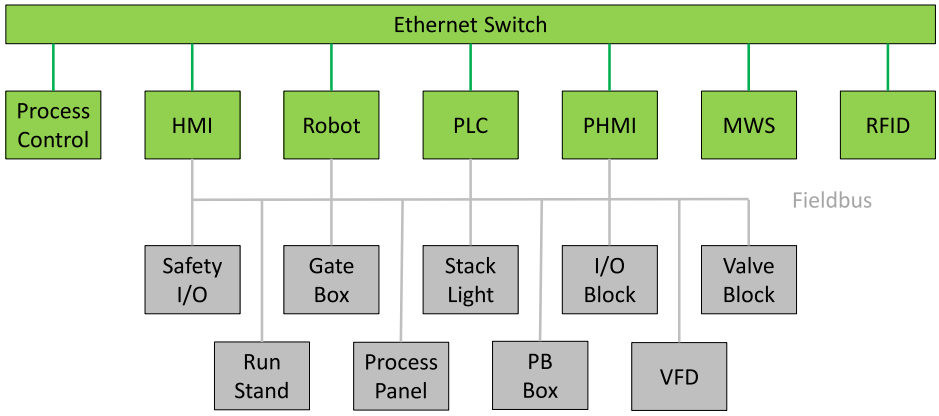

Figure 3.1.1: Cell Communications Structure

3.1.2 The Concept of OPC

Historically the vendors of automation equipment often developed their own, proprietary network architecture. This practice became a roadblock when trying to link dissimilar systems together. In the early nineties several controls equipment manufacturers established the OPC foundation to lay the groundwork for a standardised communication structure which would improve interoperability. Since then, there have been several improvements to the standards. The version used in the course of this project is OPC-DA because it is the standard provided with the Controllogix controllers used at the test site. The general structure on which the OPC communication is based can be seen in figure 3.1.2.

Within the hierarchy, the target device, shown on the right, still features its proprietary communication structure. It is the task of the OPC server to interpret the requests received through the OPC interface and translate them into the devices native protocol and vice versa. The standardised OPC protocol is then used for communication to the OPC client, which essentially is the reversal of the OPC server. It interprets and translates the requests of the second proprietary device and makes them available to the OPC interface. The OPC client also receives and translates the responses of the OPC server.

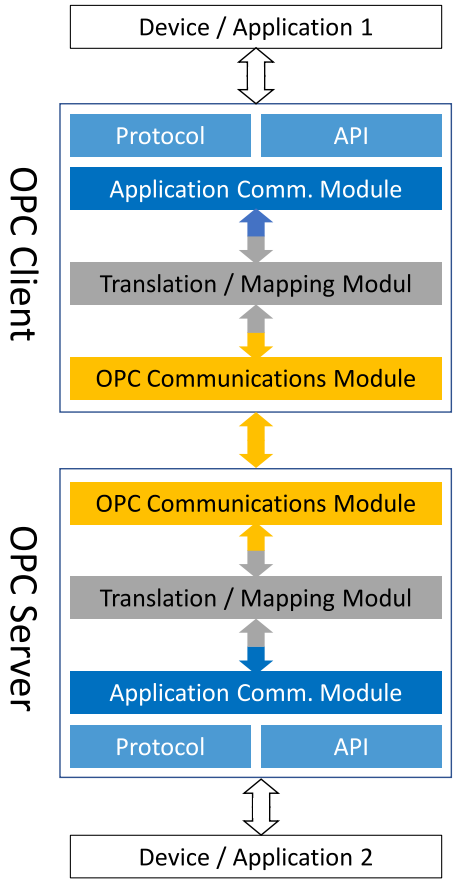

Figure 3.1.2: The OPC Structure

OPC-DA and its newer version OPC-UA are available for many of today's controls systems. Keeping the potential application of this work to different automation systems in mind OPC became the natural choice. The project originates from the need to record the sequence of operations of a wholly encapsulated laser welding cell which had cycle time problems. The housing around the cell prevented observation by the engineers, which made the deployment of a monitoring system necessary. The Rockwell family of controllers requires for the use of their programming software an additional communication software called RSLinx, which also comes with a built-in OLE (Object Linking & Embedding) and OPC server. Since OLE is a technology developed by Microsoft, it can be linked to the Microsoft Office products. Therefore the first feasibility studies were done using Excel spreadsheets. The success of this initial application prompted this research for a completely automated framework. Visual Basic was chosen for the data acquisition because the plant's maintenance personal has basic knowledge of it and therefore should be able to maintain and enhance the system later on. Advosol, an OPC Foundation member, provides a suitable .net OPC library.

3.1.3 OPC Setup

When setting up OPC communication, several choices have to be made:

RSLinx Topic: To set up the communication to a Rockwell programmable logic controller, which is the main brand of PLCs used at Opel, a topic needs to be defined within RSLinx. The topic, therefore, is the name that is associated with the communication. First, the communications driver needs to be chosen. For this project, it is Ethernet IP. Linking of the driver to the device is accomplished through its IP address. Once the connection is established, the topic can be created and settings like the processor type and the message pull rate defined. All that is now needed is the entry of a simple string following the structure

=application|topic!'item'

into one of the Excel cells to retrieve current status information on the tag selected. RSLinx even offers a browser to choose the desired tag and generate the necessary string for it.

OPC Items: An OPC item is the definition of the data point to be monitored within the OPC server. Its format for Controllogix data is:

[*TopicName*]Program:*ProgramName.TagName*

Each item can be set to active or inactive.

OPC Groups: OPC items can be grouped into OPC groups. Besides providing a structure, these groups also enable the joint manipulation of the items settings. These settings include the acquisition mode, the data source, the update rate and the dead band.

Push vs Pull: The OPC client can request the status of a given data point which then is returned by the OPC server. Since this means that even unchanged status information will be relayed, excessive communication can be expected. The OPC server can also be set up to raise an event which includes the status information, quality and timestamp only for data points that have changed. This approach was chosen since it results in less burden on the communication infrastructure.

Synchronous vs Asynchronous: This setting only applies if the data are pulled from the OPC server. It determines if the client waits for the request to be completed (synchronous) or if the client can issue other requests before receiving a response (asynchronous). For this research, it was chosen that the server raises an event upon status change. Therefore the synchronous or asynchronous setting is of no concern.

From Device vs. From Cache: Another option, when pulling data from the OPC server, is the source of the data. Usually, the OPC server keeps the data in a cache and renews them based on a predefined refresh rate. The OPC client can also define that the cache should be ignored and that the data should be read from the device directly at the time of the request. This data access is noticeably more time-consuming than reading from the cache. For this research, this option can be ignored because the push option was previously chosen.

Deadband: The deadband defines the degree of status change required to trigger a change event. Since all changes should be logged the deadband was set to 0 for this work.

3.1.4 Data Collection Procedure

In the realm of automotive manufacturing equipment, the process is equal to the sequence of operation (SOO). Therefore, timestamps are required for every actuator and its corresponding sensors within a station. Robots can be working in more than one station which can lead to unidentified gaps within a stations sequence. This effect only became obvious during the course of this research. Gaps would also be shown if the robot performed an automated maintenance task such as tip dressing or sealer system purging. The frequency of these interruptions strongly relates to the process being performed. In some instances such main nance tasks can be necessary after each process cycle. This problem can only be avoided if the robots' working segments are also logged. The standard of the equipment used for the case study requires the robot to feed back a numeric segment number which identifies in which station the robot is currently working and what it is doing. A typical sequence of segments would be home position - pounce - unload station - clear station - load next station - clear next station - return to home. This is just a rough example. In reality there are even more segments defined.

An identifier needs to be recorded together with the events to allow for grouping of the events into cases. In a first attempt, recording the parts sequence number was chosen. Application of the concept then revealed that sub-assembly parts are not assigned a sequence number which hinders the tracking of a part through the production cell.

When loading a part into a cell, it is associated with a job data package which defines its style and options. The job data also allow for build status information and details about the carrier being used. Since carriers are mainly used for major assemblies, the field often remains unused. Therefore the collection algorithm was extended to take advantage of the OPCs writing capability, and a unique 'tracking number' was stored into that data field to be used for part tracking and case clustering. The job data is only transferred to a station once the part has been loaded which leaves motions required for station setup, loading and station reset still without a suitable identifier.

For future predictive analysis, the data needs to be correlated with fault events. That could be achieved either by tagging the data manually a posteriori or by logging the events (alarms) concurrently.

Data collection could be achieved with multiple local, decentralised data collectors, running on the maintenance work stations, that periodically submit the collection results to a centralised location. Here the main advantage is that the traffic between the data collector and the PLCs stays in the local subnet. The lower data volume also allows for a less expensive database version to be used, and at the same time, the error rate might decrease. Besides, a single point of failure only impacts the data collection for one of the cells rather than the whole body shop. Also, the current version of the PLC software is readily available on the maintenance work station so that updates to the collection algorithm are possible.

On the other hand, such an approach requires multiple licenses for the OPC server software along with an increased effort to manage and maintain the system. The alignment of the data from different sources can become, due to timestamp inaccuracies, challenging.

The second option is a centralised setup, where the collection algorithm, the OPC server and the database are located on only one computer. For this scenario, the above described advantages and disadvantages can be reversed. Because of manageability, the centralised setup was chosen, for the proof of this concept. It is believed that a decentralised setup would be more beneficial for a production solution.

Table 3.1.1 shows how the PLC program aligns with the actual equipment for this project. Also, it shows what keywords can be found within the PLC logic's text file and what regular expressions can be used to locate them.

Successful parsing requires an understanding of the structure to be parsed. A schematic representation of the software structure within a ControlLogix controller can be seen in figure 3.1.3

Most noticeable is that the Rockwell PLCs work with two levels of tags. There

Table 3.1.1: Relationship Between Equipment And Software

actual equipment	software equivalent	tags within L5K	Regular Expression (Regex)
cell	controller	CONTROLLER	^CONTROLLER\s\w+\s\(
station	program	PROGRAM	^(\t+|\s+)PROGRAM\s\w+\s\(
advancing motion	(advance) routine	ROUTINE	ROUTINE S
returning motion	(return) routine	ROUTINE	ROUTINE S
action feedback	rung	.comp(UDT za_Action)	(OTL\(|OTE\()[A-Z,a-z,0-9]+\.Comp\)
sensor	contact	XIC(...)	XIC(

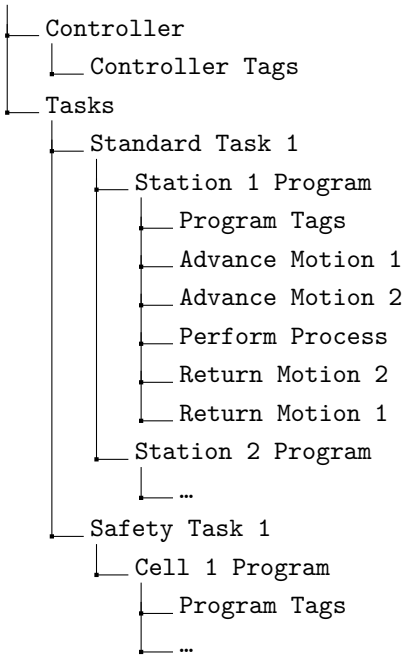

Figure 3.1.3: PLC Code Structure

are controller tags, which can be accessed from every program, and program tags, which can only be accessed by the program in which they were defined. Addressing the tags through OPC also requires different nomenclatures. The first part of the algorithm, therefore, needs to locate the controller tags and store them in an array for later referencing.

Program tags are identified by the name of the program in which they have been defined. This fact requires keeping track of the current program name while parsing the logic and associating it with the data point of interest, as long as that tag is not a controller tag.

For the body shop observed, the standard defines that every motion is to be programmed within its own routine. This routine has to be structured according to the

motion structure shown in figure 3.1.4. The reason why the complete rung is one of the first rungs within the routine is that one can spot the status of the motion directly upon opening the routine in the viewer without having to scroll down in the program structure. This structure not only simplifies parsing, but it also allows for automatic generation of the code during the design phase because it applies to most motions encountered for an automation project.

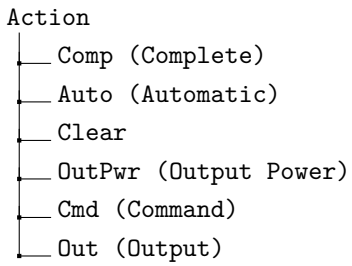

Figure 3.1.4: Motion UDT Syntax

Once the controller tags have been identified, parsing can be done line by line from the beginning to the end. Since all the data to be gathered, belong to sequence routines the parsing can be improved by skipping all lines that are not located within a section which starts with 'ROUTINE S' (where the 'S' stands for sequence) and 'END_ROUTINE'. Once a program rung that ends with a '.Comp' tag is encountered (figures figs. 3.1.5 and 3.1.6), the tag name is extracted and the extension is replaced with '.Out'. For this project, that is the name of the tag that initiates a motion (see figure 3.1.4). The resulting sensor inputs can also be found within the same rung, programmed as normally open (XIC) contacts. All the sensor addresses can be obtained by extracting all the tags that have been programmed with the XIC instruction (Figure 3.1.7).

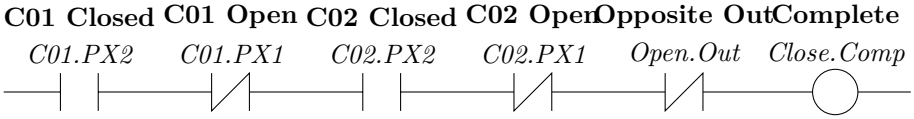

Figure 3.1.5: Ladder Example Motion Complete Rung

XIC C01.PX2 XIO C01.PX1 XIC C02.PX2 XIO C02.PX1 XIO Open.Out OTE Close.Comp

Figure 3.1.6: L5K Example Motion Complete Rung

If no such apparent relation between actuator and sensors is to be found within the PLC code, it would also be imaginable to discover the relations a posteriori with the help of some tag-name-based reasoning algorithms. Alternatively, the software could 'learn' the relation through a 'teach in' process, where the machine is manually cycled through its sequence step by step.

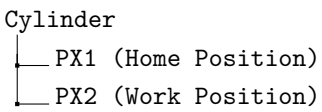

Figure 3.1.7: Cylinder UDT Syntax

The current robot segment number also can be found within a defined controller tag (e.g. RA400R01.SegNum). While parsing the program, all that needs to be extracted, are the names of the robots associated with that station (e.g. RA400R01).

For this project, the actual alarm messages are included within the software as rung comments, with an alarm-number, which represents the last three digits of the actual alarm. The leading one or two digits of the alarm are a program related offset. To decode an occurring alarm, the alarm messages need to be extracted as well as the offsets associated with the program currently being parsed.

To summarise, there are three basic requirements for nomenclature based data point selection. The most important is a standardised tag naming or data type structure of the outputs initiating the actuators (e.g. Clamps1Close.Out; za_Action). The same applies to the naming and data type structure of the sensors (e.g. C01.PX1; zp_Cylinder). Ideally, there is also an identifiable rung that sums up the sensors associated with the actuators to allow deriving the connection between the inputs and outputs.

3.1.5 Database Structure

The database structure established is shown if figure 3.1.8. It is set up to mirror the automated production equipment. As previously explained an automotive body shop consists of different zones in which the different sections of the car's body are manufactured. Therefore the first table holds the names of the zones being observed. A zone itself consists of many cells which are controlled by a PLC. The PLCs table is related to the zones table and stores information about the location of its offline program, its version and the processor name. Cells consist of stations and robots. Their tables include, besides the relationship to the PLC, fields for their names as well as temporary storage areas for the current style, option and sequence number. Within a station, many motions can be found. These motions are defined by their relation to the station, a 'start tag' (an output initiating the actuator) and an 'end tag' (a sensor that indicates motion completion). This table is the foundation for the actual event log which is stored as raw records with a timestamp, style, option and sequence number information. A station can be in manual or automatic mode. This state is additionally captured in the cycle log table. More detailed information about the robots whereabouts, like the current segment number and the interference zone

occupied, are stored in the robot log table. The details about the styles and options, which are referenced in the previous tables, are stored separately. Since it seems possible to use the data obtained for fault prediction, it was decided to extract all the alarms encountered while parsing the PLC file. The alarms are stored in the alarms table, represented by an alarm-number and a string definition. The alarm events themselves are logged in the alarm log. Many of the previously detailed tables also have a field called 'outdated'. Its purpose is to enable or disable records depending on which of the PLCs was chosen for monitoring. This solution helps to prevent double entries in the tables if monitoring is switched off and on again.

3.2 Log Preprocessing

Process Mining, as proposed by Cook et al. (Cook and Wolf 1995), as well as Agrawall (Agrawal, Gunopulos, and Leymann 1998), was meant to support the development and maintenance of software projects. Van der Aalst (W. V. D. Aalst, T. Weijters, and Maruster 2004) extended its usage to the discovery of models for business processes while developing it into a research field. The methodology behind Process Discovery is to use an event log and apply a discovery algorithm to it to derive a model in a well-established notation for the area of its application. Depending on its notation the model can be used to replay simulations, using the event log, to identify bottlenecks as well as for making predictions on the remaining time to complete a case. If the model is based on enough cases, it also might be suitable to create recommendations on the best actions to take, to complete a case as soon as possible.

It appears that this approach might be suitable for industrial production processes, which yield detailed event process logs. Contrary to business processes, industrial equipment only has a finite number of traces, which all can potentially be discovered with a minimal amount of log data. This feature in turn means that generalisation is not desired for such models, which can only be accomplished using near 'perfect logs' when applying Process Discovery techniques. The equipment logs often do not have the degree of completeness required to derive an accurate process model, which may lead to the discovery of causalities which do not reflect reality. Therefore evaluation and repair of the log are of the utmost importance. This observation is shared by many of the researchers in this field (Rogge-Solti et al. 2013; Bose, Mans, and W. M. P. Van Der Aalst 2013; Verhulst 2016; Suriadi et al. 2017).

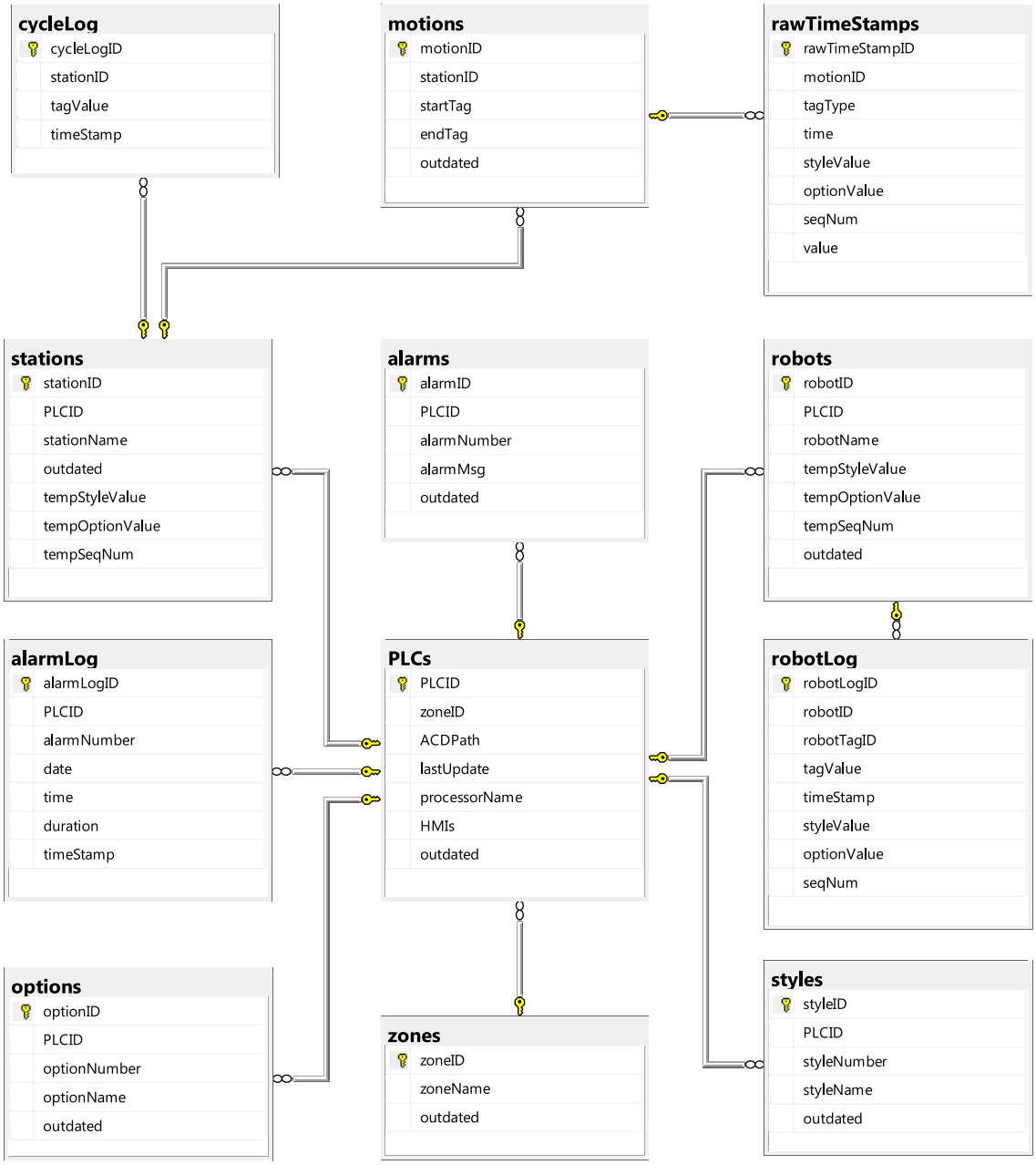

Figure 3.1.8: Database Structure

3.2.1 Definitions

Since most technical data scientists are not familiar with the Process Discovery paradigm, first a brief overview of some of the most important terms is given. Considering the production equipment event log shown in figure 3.2.1 the following terms need to be explained according to van der Aalst (W. V. D. Aalst 2016b):

Figure 3.2.1: Example Production Equipment Event Log

serialNum	startTag	endTag	startTime	endTime	style
65524	Clamp1.Close	C01.PX2	2017-10-05 08:28:12	217-10-05 08:28:14	1
65524	Clamp2.Close	C02.PX2	2017-10-05 08:28:14	2017-10-05 08:28:16	1
70013	Clamp1.Close	C01.PX2	2017-10-05 08:29:37	2017-10-05 08:29:39	2

Event Log

An event log is a log of activities, recorded together with a timestamp and some additional attributes. For industrial production equipment, an event log will hold records of all the motions occurring within the machine, including, but not limited to, robot activities.

Activity

In a production equipment event log, any motion occurring is considered to be an activity. In the example figure 3.2.1 that refers to the column 'start tag' which is the signal that triggers the motion.

Case

The term case refers to a business case which could be, for example, all the tasks relating to an order with a specific purchase order number. For an equipment event log a case refers to one manufacturing cycle which starts with the machine being ready to accept the next part style followed by the machine being loaded. Once the parts are present in the station the process cycle starts which concludes with the unload event. Finally, depending on the station, there also might be a reset event which moves equipment back into position to receive the next part. Process Discovery requires that all the before mentioned activities are associated with a numerical case ID.

Case Attributes

Case attributes apply to the whole case. For this research the style and option information of the part would be considered the case attributes. The option was dropped for this research as including the options would lead to more differentiation, but the approach for cleansing and preprocessing the data would remain the same.

Trace

Some papers use trace and case interchangeably, while others use trace and variant as synonyms for each other. A trace, as referred to in this thesis, is a generic pattern of events following each other which can be used to classify a case found in an event log. In the context of a production event log, a trace depends on few internal and external factors. Therefore this equipment only has a few traces, although there might be thousands of cases.

Event

An event is a row within the event log, consisting of activities, which are associated with cases. The example figure 3.2.1 consists, therefore of three events.

3.2.2 Reference Quality Matrix

According to Bose et al. (Bose, Mans, and W. M. P. Van Der Aalst 2013) when evaluating event logs, besides event granularity, voluminous data, case heterogeneity, process flexibility and concept drifts, the data quality issues shown in figure 3.2.2 need to be considered.

Figure 3.2.2: Quality Matrix Proposed By Bose Et Al.

	case	event	belongs to	c_attribute	position	activity name	timestamp	resource	e_attribute
missing data									
incorrect data									
imprecise data									
irrelevant data									

Since equipment logs are not resource bound, resources have been omitted. The proposed framework also does not record event attributes, which also makes them not applicable. The position of the events is not essential either, as it can be determined by the sequence of the timestamps. The activity names are automatically generated while parsing the PLC program. Since the PLC program is in production, it can be assumed that the tags retrieved are correct and therefore, do not have to be evaluated. Applying these restrictions leads to the compressed matrix shown in figure 3.2.3.

Figure 3.2.3: Data Quality Issues In Production Equipment Event Logs

	cases	events	relationships	case attributes	timestamps
missing					
incorrect					
imprecise					
irrelevant					

In the remainder of this section, all the issues that potentially are applicable will be discussed.

3.2.3 Missing Cases

Cause And Impact

The equipment event logs are created with the help of a system external to the manufacturing equipment. The equipment is monitored using OPC-DA. Upon a tag change event, a timestamp is written, along with the activity and case identifier, into a SQL database. Multiple scenarios can cause the loss of cases:

- loss of network connection
- failing monitoring system
- failing database connection
- equipment switched to manual mode

Since the proposed methodology is not intended to analyse real-time events or to replay the cases, missing cases can be compensated by increasing the sample size used for process model discovery. No other interventions are required.

Detection

Within an automotive body shop, there are two types of manufacturing processes. One is the sub-assembly manufacturing, where the individual parts of the car body, like the fenders, are manufactured. Typically those parts cannot be associated with the car to which they will be mounted, at the time of their manufacture. Therefore a unique number is assigned to the part at the beginning of the manufacturing line for tracking

purposes. Since that number includes a sequential component, a missing case could be detected based on a deviation within the sequence.

The second process is the assembly of the car body from the sub-assemblies. At this point, the VIN (vehicle identification number) will be assigned to the car. A missing case only could be identified when comparing the event log with the production manifest. Therefor

$$\forall\ VIN \notin production\ schedule \tag{3.1}$$

could be considered missing cases.

3.2.4 Missing Events

Cause And impact

Missing events can be caused by the same issues described for missing cases in 3.2.3. Missing events can cause several problems. Since not all the events in the raw log file have a case identifier, the remaining events, necessary for production, could either be associated with another case or they could wrongly be declared as a different trace. If the issue of missing events is not addressed during preprocessing, it also could cause the mining algorithm to discover wrong activity dependencies.

Detection

Missing events can be detected during the process of correlating them into cases. A case consists of setup/reset, load/unload and process events. All events depend on the style of the part being manufactured. The setup/reset event additionally depends on the previous and next part style. A comparison of cases, therefore, can be made if these factors are taken into consideration. Logs of the same trace will include the same station events t. Therefore missing events are defined as:

$$\forall\ t_{(trace)} \notin t_{(case)} \tag{3.2}$$

Methodology

The simplest approach is to count the number of events and just delete the cases, which don't have the required number of events. Therefore no events t are missing if:

$$\sum t_{(trace)} = \sum t_{(case)} \tag{3.3}$$

Besides the number of events, a 1:1 comparison of all events could be an indicator to which case to delete. Then

$$\forall t_{(trace)} = \forall t_{(case)} \tag{3.4}$$

would hold.

Alternatively, the missing event also could be replaced by a copy from a previous case since it can be assumed, that within a short time frame only negligible or no differences in the motion of the monitored components will be seen. With t_x being a certain event this assumption is expressed by:

$$t_{x(case)} \approx t_{x(case-n)} \tag{3.5}$$

Copying the missing data will help to preserve the remaining data and allow for a continuous simulation within the discovered process model if desired.

3.2.5 Missing Relationships

Cause And impact

As previously described, cases are associated with the parts being manufactured. Every production equipment also has setup events and reset activities that are happening when there is no part being processed. These events will be recorded without a clear relationship indicator. One of the necessary stipulations when applying Process Discovery to an event log is that the events need to be case-based. This requirement means that all the events belonging together need to be marked with a unique identifier. This case ID enables process model discovery and the replay of the cases on that model.

Detection

The evaluation of the equipment log showed that there were many events which were missing the intended case attribute 'VIN'. Since the case ID needs to be a unique number and there is no additional benefit in utilising the VIN it was decided to assume for all events that no case identifier is present and that a new, unique number has to be assigned.

Methodology

A machine sequence can be split up into five sections, as indicated by the curly brackets on the bottom of figure 3.2.4.

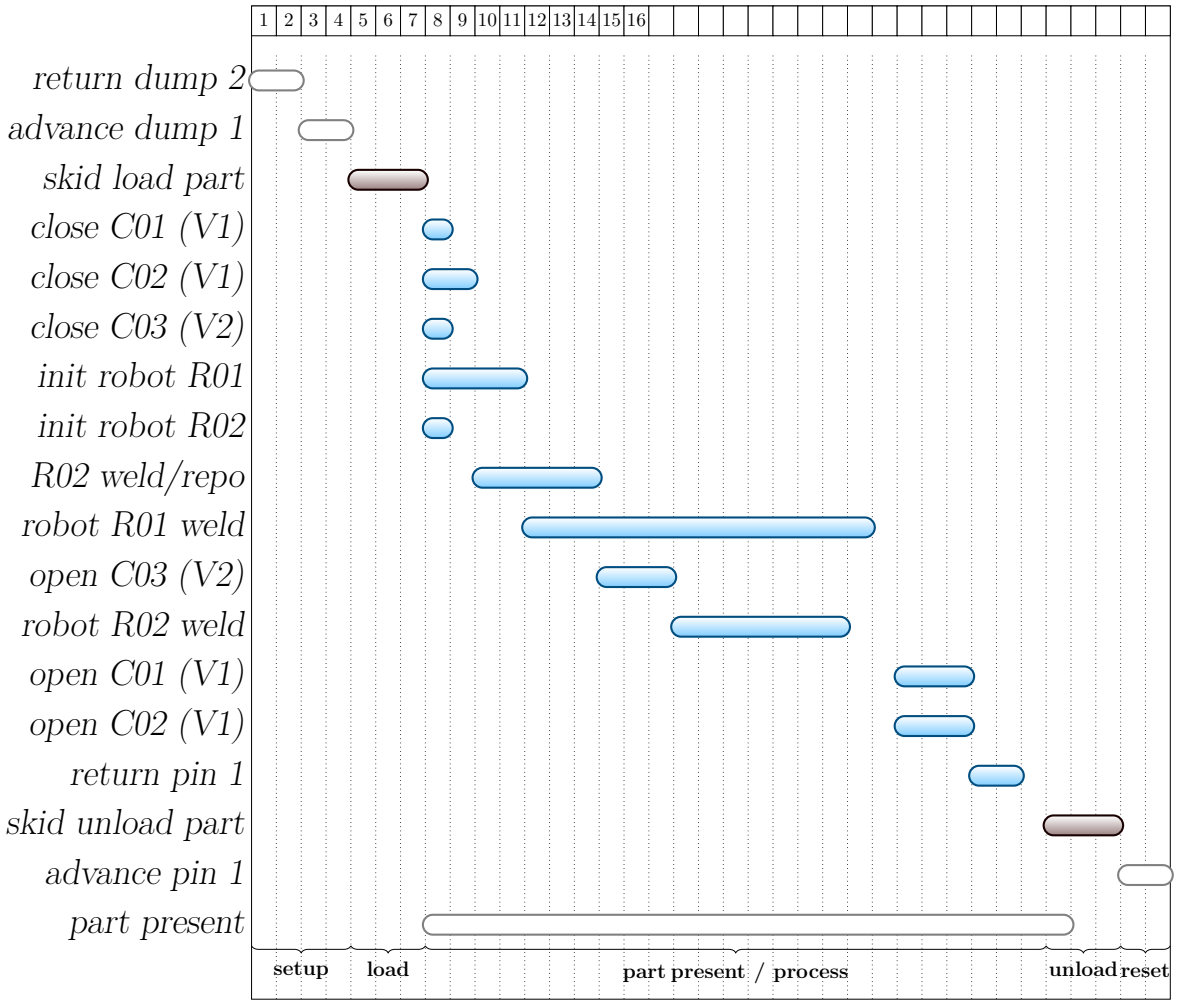

Figure 3.2.4: Gantt Chart Of Example Machine Sequence

The one signal, that is common to all manufacturing cells, is the part being present in the tooling. Therefore the data acquisition was extended to include the part present signals, although they are, for the sequence itself, of no interest. The Opel global controls standard defines two tags for that purpose: 'no parts present' and 'all parts present'. If just a single part is loaded into the station, the two signals are just opposites of each other. With multiple parts the 'no parts present' signal will change its state, once the first part is loaded. The 'all parts present' signal will not change its state before all parts are loaded. It has been found that the 'no parts present' signal often is not used when working with a single part. Therefore both signals need to be observed and mashed into a single indicator that shows if any part or no part is present in the station. This indicator needs to be '1' if the 'no parts present' signal changes state from '1' to '0' or the 'all parts present' changes from '0' to '1'. It needs to be reset to '0' if the 'no parts present' signal changes state from '0' to '1' or the 'all parts present'

changes from '1' to '0'. The **process section** of the sequence starts with a part being present in the station and ends one event prior to the unload step. The unload event is the last step for which the presence of a part has been recorded. This leads to the following theorems:

Theorem 1 *All events recorded while a part is present, with the exception of the unload step, will be considered a process step.*

Loading steps and unload steps can be defined as follows:

Theorem 2 *The event recorded just prior to the part being present within the station must then be the first loading step while the last step with a part being present must be considered the unload step.*

If the part unload-position differs from the loading-position, the station finally needs to have **reset events** to ready it for the next part being loaded. These events might be style related which leads to the following definition:

Theorem 3 *An event is an station reset event if it occurs always following a certain style x followed by any style.*

Alternatively the following definition would hold as well:

Theorem 4 *Reset events are all events that occur between the unload step of a part with the style x and the load event of the following part with the identical style x.*

While transitioning from one style to another station **setup events** might be required to ready the station for the new part style being loaded. Setup events can be defined as follows:

Theorem 5 *Setup events are events that are recorded after the final reset event of style x and prior to the load event of style y.*

3.2.6 Missing Case Attributes

Cause And impact

For this research, the case attributes are recorded every time a motion is logged that can be directly related to the part being manufactured. This fact means that all the events that can not directly be associated with a part are missing the case attributes. These attributes are essential to this work as specific manufacturing sequences could also be the cause for delays.

Detection

NULL values in any of the attribute columns can identify missing case attributes.

Methodology

Since every part related motion is recorded together with the associated case attributes, missing case attributes can be copied to the remaining events once their case association has been discovered.

3.2.7 Missing Timestamps

Cause And Impact

Missing timestamps could be caused by the same issues described in 3.2.3. The intended Process Discovery approach is based on timestamps. Each event has two timestamps. The first timestamp marks the time a motion was triggered and the second one the time the unit reached its end position. If the first timestamp is missing, the position of the event within the cases sequence can no longer be determined. The second timestamp missing will prevent determining activity dependencies as a dependency only can be assumed if activity 'b' starts after activity 'a' has been completed.

Detection

Missing timestamps can be identified by determining which motion has only one timestamp and therefore is missing either the 'start' or 'end time' stamp.

Methodology

The missing start timestamp $\tau_s(t)$ or complete timestamp $\tau_c(t)$ can be calculated based on the mean duration $\lambda\tau(t)$ of the event t:

$$\tau_s(t) \approx \tau_c(t) \; - \; \lambda\tau(t) \tag{3.6}$$

and

$$\tau_c(t) \approx \tau_s(t) \; + \; \lambda\tau(t) \tag{3.7}$$

Alternatively, the incomplete event could just be deleted and treated like a missing event, as shown in 3.2.4.

3.2.8 Incorrect Cases

Cause And impact

Bose et al. (Bose, Mans, and W. M. P. Van Der Aalst 2013) describe incorrectness as the entity, relation, or value provided in a log being incorrect. In the context of cases, this could only be the wrong data format or a wrong case number. Bose et al. describe the most common cause for such problems being manual entry by humans. This problem also would lead to the discovery of activity connections, within the process model, which do not exist in reality.

The formatting issue, for this approach, is eliminated through format definitions within the database. The database, therefore, would reject entries in the wrong format. Also, the case numbers are generated automatically by the preprocessing algorithm. As long as it is ensured that duplicate case identifiers are not possible, this quality issue should not be a concern when evaluating automatically generated equipment event logs.

Detection

A wrong entry could only be identified by either finding duplicate events, associated to the same case, or by detecting a case identifier (caseID) which is not used in any of the other event t of the case:

$$for\ n = 1, 2, 3 \ldots \backslash \{x\};\ caseID_{t_{x(case)}} \neq caseID_{t_{n(case)}} \tag{3.8}$$

3.2.9 Incorrect Events

Cause And impact

As mentioned previously, incorrectness can mainly be attributed to manual data entry. Such failures would lead to the discovery of invalid activity connections within the process model.

Algorithms were designed that parse the proven PLC programs controlling the equipment's activities. All the events are automatically correlated to the controls tags found in that program. Also, the logbook is automatically generated based on those tags. Therefore the possibility of incorrect events, due to the data recording process, can be neglected.

The experiments showed that the following two scenarios lead to duplicated entries. It is standard, that the end positions of some motions are coded into the logic triggering another motion to prevent collisions. If this programming is done wrong, this can lead

to intermittent interruption of the motion. The event then has to be triggered again to come to completion. The second reason is based on mechanical failure. A motion is initiated and reaches its end position. Therefore the event is completed. At the end position, the unit bounces back again and leaves the end position. The PLC controller then needs to re-trigger the motion once again.

Detection

An incorrect event could only be detected if it is an unique event or if the event t_x within a case is duplicated:

$$for\ n = 1, 2, 3 \ldots \backslash \{x\};\ t_x(case) = t_n(case) \tag{3.9}$$

Methodology

For Process Discovery, the existence of such duplicated events would not cause a problem. It would lead to the discovery of another trace, and the reviewer would be able to discover the issue. It was decided to note and remove such duplicates because it allows the creation of an issue list for the maintenance personal at an early stage during the data analysis and at the same time prevents wrong traces from being discovered.

3.2.10 Incorrect Relationships

Cause And impact

As explained earlier, the raw event log is missing many of the relationships, and those relationships need to be defined during preprocessing (see 3.2.5). Incorrect relationships can only be attributed to this preprocessing step since it relates the events to the cases. This problem will lead to the discovery of process models, which do not reflect reality.

Detection

For each style there typically is only one event path. Therefore the discovery of a second model for any given style would be suspicious. This research shows that programming errors, within the controls system, can lead to such an instance. If this problem is random, it can be detected as an outlier. If it is systematic, it could only be discovered by the expert reviewing the generated process model.

Methodology

Since incorrect relationships are mostly the result of preprocessing, they can only be avoided through careful testing of the preprocessing steps and expert evaluation of the resulting model, which should lead to the revision of the corresponding algorithms.

3.2.11 Incorrect Case Attributes

Cause And impact

The case attributes help to discover the repetitive traces. Incorrect case attributes will lead to the discovery of a trace, which does not exist in the process. Faulty production data could be the only cause for incorrectness within the case attributes.

Detection

Incorrect case attributes can only be discovered if there is a variation within the events of a case with an identical identifier by simple comparison. For example if the style of one event t_x deviates from the remaining events within the same case:

$$for\ n = 1, 2, 3 \ldots \backslash\{x\};\ style_{t_{x(case)}} \neq style_{t_{n(case)}} \tag{3.10}$$

Methodology

If there is a variation of attributes, for events with the same case identifier, then the faulty attributes can be replaced by the attributes shown for the majority of the events with that case identifier. The proposed algorithm already corrects this problem while clustering the cases in 3.2.5.

3.2.12 Incorrect Timestamps

Cause And impact

There are multiple possible causes for incorrect timestamps. The events could be logged on different devices, not time-synchronised, before being merged or there is a manual intervention into the manufacturing process, causing untypical delays. Also, signal processing and logging consume some time. For example, the OPC server evaluates the tag status every 50ms. Once a change is detected, the logging algorithm needs to process that change, which adds another 20ms of uncertainty. Signals triggered twice can also be a cause for incorrect timestamps. If the 'start tag' is triggered twice, two records will be added to the database. Once the corresponding 'end tag' is triggered,

one of the records will be completed, leaving the 'end time' for the other record open. If at a later time, the 'end tag' should be triggered twice, this void will be filled by a wrong timestamp.

Detection

The proposed framework also records the system status to detect manual intervention. Therefore any event logged, while the machine is not in automatic mode, should be treated as suspicious. Processing time, during manual operation, should not be of any concern while mining the log. This logic also applies to the remaining data of that case. Typically the total cycle time, for a station being evaluated within this research, is in the range of 60 seconds. Therefore any event duration $\Delta\tau_t$ exceeding 60 seconds between 'start time' and 'end time' must be incorrect. The same holds on the other end of the spectrum. The fastest motions recorded are in the range of 400 milliseconds which leads to the conclusion that any motion faster than that is recorded wrongly:

$$\begin{aligned} \Delta\tau_t &> 60\ sec \vee \\ \Delta\tau_t &< 400\ ms \end{aligned} \tag{3.11}$$

Methodology

The timestamps are based on the device recording the events. Since all the events of one production line are collected by one device, it can be assumed that the timestamps are consistent, which is key for the mining process. Truly incorrect timestamps only come into play, when data from different recording devices are merged, as correlation will become more difficult. In such a case, all the devices should be connected to an enterprise time server to avoid the problem. If that is not possible, markers within the different event logs need to be found that allows for synchronisation of the logs.

If a manual intervention was discovered for any event within a case, the whole case could be dropped, or the suspect timestamps could be replaced with approximations based on previous cycles. Besides, all the remaining timestamps of the same case would have to be adjusted accordingly.

Uncertainty, due to signal processing times, can not be avoided but needs to be taken into consideration when deriving Process Mining results from the data. Some of the motions recorded happen within $4/10^{\text{th}}$ of a second. A deviation of a 10^{th} of a second, which here is equal to 25%, could easily be interpreted as speed reduction due to wear. This deviation is not always constant. Therefore the mechanical evaluation of a device

needs to be done on a median over several timestamps rather than just based on a single measurement.

Timestamps that cause duration outliers, as described above, could be removed and the record treated like any event with missing timestamp described in 3.2.7.

3.2.13 Imprecise Relationships

Cause And Impact

Bose et al. (Bose, Mans, and W. M. P. Van Der Aalst 2013) define impreciseness as data being too coarse, thus missing the needed precision. Imprecise relationships would make the association of the events to cases difficult or impossible causing the Process Discovery algorithms to discover models that are not true to reality. For the equipment event logs, there are only three possibilities. A case identifier can be present, missing or wrong. There is no other granularity which would allow for imprecision. Since the proposed framework assumes that all relationships are missing (see 3.2.5) imprecise relationships are of no concern for this work.

3.2.14 Imprecise Case Attributes

Cause And impact

What holds for imprecise relationships also applies to imprecise case attributes for such event logs (see 3.2.13). Therefore impreciseness of case attributes is not a concern either.

3.2.15 Imprecise Timestamps

Cause And impact

The precision of the timestamps is solely based on the logging algorithm and the database settings. Exporting data from a data table into a different format, for example '.csv', can also cause a loss of precision. Manipulation of the data within Python causes a loss of precision when the data is written back to the SQL Server as fractions of seconds are dropped. Since some of the motions, recorded in an equipment event log, might be as fast as 0.4 seconds, the timestamp needs to be recorded down to at least a 10^{th} of a second.

Detection

Imprecision of the timestamp can be detected by evaluating the timestamp values.

Methodology

Once the precision of a timestamp is lost, there is nothing that can be done to recover it again. Therefore it needs to be made sure that the logging algorithm writes the data with the desired precision into the database and that the database is set up to store the values with the desired precision. Also, when creating preprocessing or transformation algorithms, it needs to specifically made sure that there is no implication to the precision of the values. In the case of Python, this means, that timestamps should be written back to the database as strings rather than date/time values.

3.2.16 Irrelevant Cases

Cause And impact

A case only becomes irrelevant if some of the data are missing or incorrect. The methodology for dealing with such problems has been described above.

3.2.17 Irrelevant Events

Cause And impact

In a production equipment event log, some events are value-added, e.g. the welding cycle of a robot and some events that are non-value added, like the maintenance cycle of a robot. The value-added events are of interest to determine the time needed to bring a case to completion. The non-value added cases need to be considered because they are potentially causing delays in the manufacturing process. Initiating them at a different time within the process may yield substantial improvements. This fact leads the author to believe that there are not any irrelevant events within a production equipment log.

3.2.18 Adjusted Quality Matrix

Based on the above findings the applicable data quality issues, for production equipment event logs, can be summarised as shown in figure 3.2.5

3.2.19 Correlating Event Data

Every event consists of an output, that initiates a motion and a feedback signal, that acknowledges that the motion was completed. The previously described data collection algorithm records each of the two signals separately. Therefore a view had to be written, within the SQL Server, that brings the two signals and their corresponding

Figure 3.2.5: Data Quality Issues To Be Addressed

	events	relationships	timestamps
missing	X	X	X
incorrect	X	X	X
imprecise			X
irrelevant	X		

timestamps together into a single record, which equals an event needed for Process Discovery. The relation between the output triggering the activity and the sensor input is captured in the table that was originally created by the PLC code parsing algorithm 3.1.4. Correlating can be achieved by finding a combination of these two signals with the least time difference that does not exceed the stations standard cycle time of 60 seconds.

3.2.20 Purging Not Needed Data

Removal Of Part Present Events

The part present switches are not part of the sequence and were only added to group the motions to cases. After this grouping is completed this information can be removed.

Removal Of Events With 'End Time' Equaling NULL

As described in section 3.2.7 it is not the aim of this research to simulate all the cycles of a station. Since model discovery doesn't require seamless records, events with missing timestamps should be removed. Presented research shows that such events mainly stem from interrupted data acquisition attempts.

Removal Of Cases With Station In Manual Mode

If a station is in manual mode, all units can be moved by the operator any way desired. Therefore the recorded sequence has no guarantee to resemble the automatic sequence. Also the motion's duration will not represent the duration recorded during automatic cycle. Therefore deleting such cases is the best option, with the final goal of Process

Discovery in mind.

Combining Of 'Double Triggers'

As described more detailed in 3.3.4 double triggers are caused by programming mistakes. The best way to handle them is to combine the earliest found 'start time' within the case with the corresponding latest 'complete time'. The remaining timestamps for that activity and case then can be deleted. Since these issues cause excessive mechanical wear and increased cycle times, maintenance needs to be made aware of them. Therefore the data is logged into a table prior to removal.

Combining Of 'Bouncing' Motions

As described in detail in 3.3.5 bouncing motions are caused by mechanical problems and do not represent the typical sequence of the station. Just as described for the double triggers above their earliest 'start time' and latest 'complete time' can be combined into a single event while erasing the remainder of the data related to that activity. Since maintenance needs to be aware of those issues also they are stored in a separate table prior to deletion.

3.2.21 Trace Identification

Using Checksum As Equality Measure

To evaluate the sequences it is paramount to have a simple way to identify if sequences are identical. All the events in the available log are marked with a motion ID, which stands for the combination of initiating 'start tag' and resulting 'end tag'. A sequence consists of multiple such motions. If those motion IDs are sorted by start timestamps and compiled into a comma delimited string, a checksum algorithm can be used to create a 'fingerprint' of each of the sequences. If the checksum of two cases is equal, it can be assumed that the sequences match. According to Wolper et al. (Wolper and Leroy 1993) the probability for collision p_c for a hash is

$$p_c \approx 1 - e^{-\frac{n^2}{2^k}} \tag{3.12}$$

Where n is the number of entries and k the number of bits used for encoding. Based on a example of 50 events (entries) in a sequence this results in a probability for CRC32 of:

$$p_c \approx 1 - e^{-\frac{50^2}{2^{32}}} \approx 5.8 * 10^{-7} \tag{3.13}$$

This is sufficiently low to make CRC32 suitable for the intended purpose.

Determining The Number Of Occurrences

To be able to compare the sequences the events are grouped by the case identifier. At the same time the CRC32 over the motion IDs is calculated. The result then is grouped by the CRC32 values to get the number of occurrences. It is assumed that the sequence, that occurs most often, is complete. Complete, in reference to cases and traces, refers to all events of the process being captured in the log.

Determining The Mean Cycle Time

While grouping by the CRC32 values it is also desirable to get the mean cycle time for each of the groups. Plotting the cycle times for one of the most occurring sequences yields a distribution like figure 3.2.6:

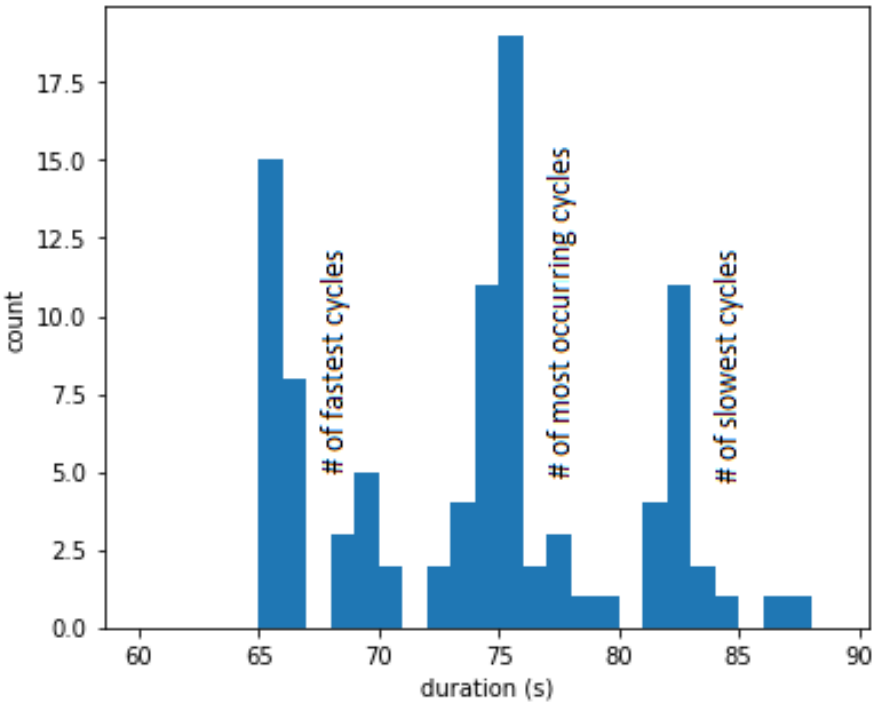

Figure 3.2.6: Sample Cycle Time Distribution

The figure 3.2.6 shows that there are multiple peaks within the distribution. Of those peaks the one for the shortest cycle time, the most occurring one and the one with the longest cycle time are of most interest. In a first approach the two unsupervised machine learning algorithms DBScan and mean-shift were applied to calculate the mean cycle time. Both algorithms do not provide a parameter to limit the bandwidth of the search.

Since the distribution approximates a normal distribution, fitting a normal distribution curve, which does not allow for limiting the bandwidth either, was tested. The final solution is the algorithm 1. Within this code a window is moved along the time axis and the mean is calculated for all the members within the window as long as the number of members is at the maximum and exceeds a given threshold.

```
Result: An array that includes all means with more the minimum cluster
        members within the window size
input parameters: data, windowSize, slidingDistance, minClusterMembers;
set cluster center to minimum duration;
while cluster center less then maximum duration do
    group the data within the window;
    if member count exceeds minClusterMembers then
        calculate the mean;
        store mean and member count in array;
    end
    move the cluster center by slidingDistance;
end
return clusters array;
```

Algorithm 1: Sliding Window Means Sub Routine

3.2.22 Feature Creation Through Expert Knowledge

The only features associated with the grouped traces at this point are the number of events within the trace and the number of occurrences for the same trace. This is not enough information to reliably classify the traces as being complete.

Most Occurring Trace

The data of identical cycles, that are observed most often, are assumed be complete.

Least Occurring Traces

Identical cycles, that have been observed less than five times, are not of interest. This helps to eliminate randomly incomplete cases from being evaluated as valid cycles. The threshold was chosen based on observations and is purposely not expressed as a percentage since some styles are very low runners and therefore a percentage still would potentially allow random 'one of' cycles to be considered.

Station In Bypass Mode

In some finishing stations, where typically no additional parts are added to the assembly, a part could just pass through. This might happen if the part was designated as scrap. In such an instance the only events recorded will be the motions through which the part enters and leaves the station. The section, within the sequence, prior to a part being present and the last event for which the part still is present must be the load and unload events. These two motion groups can be extracted and their motion IDs hashed with CRC32. If the checksum of a sample matches this checksum the sample can be classified as valid sequence.

Opposing Motion Present Fraction

Most motions, within a sequence, have two opposing events. For example a clamp closes at the beginning of a cycle and opens towards the end of a cycle. If any motion doesn't have an opposing match there is a good chance that one of the events has not been recorded. There are also situations where this doesn't hold. For example for cylinders that change position based on the part style being manufactured. Based on this knowledge this indicator is not binary but instead needs to be expressed as a fraction.

Robot Initiate Followed By Process

This feature describes the fact that whenever a robot gets initiated within a sequence it must be followed by a process. The same holds the other way around. If there is a robotic process it must be preceded by an initiate event unless the robot has a different station number. It therefore can be concluded that, if this feature is not true, the case must be incomplete.

Load / Unload Events Present

As explained in 3.2.5 every case must have load and unload events. If these, previously identified events are not present it is certain that the cycle is incomplete.

3.2.23 Machine Learning To Identify Valid Traces

To automatically determine which traces are complete and which are incomplete, different machine learning algorithms can be applied after engineering the additional features. The application of the different algorithms is explained more in detail in section 4.2.2.

3.3 Knowledge-Based Discovery

This research aims to devise an automated framework that will, based on the code of the programmable logic controller (PLC), monitor the desired production equipment and generate a Gantt chart of its actual sequence while highlighting areas of improvement. For the issue identification the following eight rules were defined.

3.3.1 Excessive Manual Cycles

During production, the equipment typically is in automatic mode unless a problem occurs that requires manual intervention. The machines within an automotive body shop often are specified to provide an uptime of 80+%. If excessive manual cycles are recorded daily, it can be concluded that there is a systematical problem which needs to be addressed.

3.3.2 Identical Units

Several pneumatic cylinders are often connected to a single solenoid valve. The grouped cylinders typically have the same bore and stroke and therefore should require the same time to advance and return. Setup can impact the synchronous movement of the units. The impact can be spotted in the event log. An example is shown in figure 3.3.1, where cylinders C01 and C02 are attached to the same valve, but their closing time is different. This improvement potential is marked in red and labelled with (a). With $\Delta\tau_t$ being the duration of a station event, $\Delta\tau_{t'}$ being the duration of an equivalent event triggered by the same solenoid and $\lambda\tau_t$ the mean duration of an identical reference event, setup problems are present if

$$\Delta\tau_t \neq \Delta\tau_{t'} \ \vee \ \Delta\tau_t \neq \lambda\tau_t \tag{3.14}$$

3.3.3 Opposing Motions

If a motion in one direction takes longer than into the opposing direction, another setup problem is present. A nomenclature based algorithm can identify which activities are opposing motions. The open events for C01 and C02 in figure 3.3.1 take longer than their corresponding closing events. Therefore the potential improvement is labelled with (e). Let $\Delta\tau_t$ be the duration of station event t and $\Delta\tau_{\bar{t}}$ the duration of the events opposing motion $\bar{t}$ then the setup is correct if:

$$\Delta\tau_t = \Delta\tau_{\bar{t}} \tag{3.15}$$

3.3.4 Double Triggers

Programming errors may lead to an equipment motion being started, interrupted and restarted again. Such behaviour causes increased cycle time and is responsible for excessive mechanical wear. In the log this manifests as two records for the same motion with different start time stamps and identical complete stamps or with one of the complete timestamps missing. Since events can happen twice within a case, the detection algorithm has to consider that the opposing motion did not happen in between these two events. If the start timestamp of a station event t is defined as $\tau_s(t_n)$, the complete event as $\tau_c(t_n)$ and the opposing motion of that event as $\overline{t_n}$ then a double trigger is present if

$$\tau_s(t_n) < \tau_s(t_{n+x}) < \tau_c(t_n) \ \wedge \ t_n = t_{n+x} \tag{3.16}$$

as long as

$$t_{n+1} \dots t_{n+(x-1)} \ \neq \ \overline{t_n} \tag{3.17}$$

3.3.5 Bouncing Motions

The term 'bouncing motion' was coined for a motion that reaches its end position but, due to the mechanical setup, bounces back so that it needs to be triggered once again to arrive at the stop position. In the event log, this can be identified by an event with start and complete timestamps followed shortly after by again the same event with start and complete timestamps without the opposing motion being recorded in between. Double triggers and bouncing motions manifest themselves in figure 3.3.1 similar to the opposing motion rule mentioned previously (figure 3.3.1 (c)). Only the underlying data allows the discovery of the actual root cause of the flaw highlighted within the Gantt chart. Based on above definitions a bouncing motion can be detected if

$$\tau_s(t_n) \neq \varnothing \ \wedge \ \tau_c(t_n) \neq \varnothing \tag{3.18}$$

is followed by an identical event $t_{n+x} = t_n$ with

$$\tau_s(t_{n+x}) \neq \varnothing \ \wedge \ \tau_c(t_{n+x}) \neq \varnothing \tag{3.19}$$

as long as

$$t_{n+1} \dots t_{n+(x-1)} \ \neq \ \overline{t_n} \tag{3.20}$$

3.3.6 Gaps

In the automotive body shop domain, there should be no period within a sequence, where there is no motion occurring. Considering that for this experiment, a variance of ∼100ms within the timestamps was found, it can be concluded that any gap >200ms marks an area of possible improvement. Gaps can be caused either by programming errors or by external circumstances which are not recorded. A typical example of a gap is marked with the letter (d) within figure 3.3.1. Gaps can be detected by splitting up all events (t) in a cases Gantt chart into 200ms bins defined by n. A gap is present in a normalised Gantt chart starting at 0 if

$$for\ n = 0ms, 200ms, 400ms \ldots; \sum_{x_n}^{t_n+199ms} \forall t(x) = 0 \tag{3.21}$$

3.3.7 Station Blocked

A particular case of the above described external circumstances is the station being blocked. A blockage is caused by the next station not being ready to receive the completed part. In that case, the event data will show a gap before the unload event. A blocked condition has been highlighted within figure 3.3.1 with the letter (f). Let $\tau_s(t_u)$ be the start time of the unload event and $\tau_c(t_{u-1})$ the complete time of preceding event then a blocked condition exists if

$$\tau_s(t_u) - \tau_c(t_{u-1}) > 200ms \tag{3.22}$$

3.3.8 Special Event - Robot Initiation

The duration of the robot initiation event was found to be varying substantially. During this time-frame, the robot receives its program number and a start signal which triggers it to move to a pounce position. Typically this routine takes a maximum of two seconds. This value allows for the assumption that a robot initiation lasting more than two seconds is suspicious. Such a situation is shown in figure 3.3.1 with the letter (b). With $\Delta\tau_{init}$ being the duration of a robot initiation event a reason for suspicion is present if

$$\Delta\tau_{init} > 2sec. \tag{3.23}$$

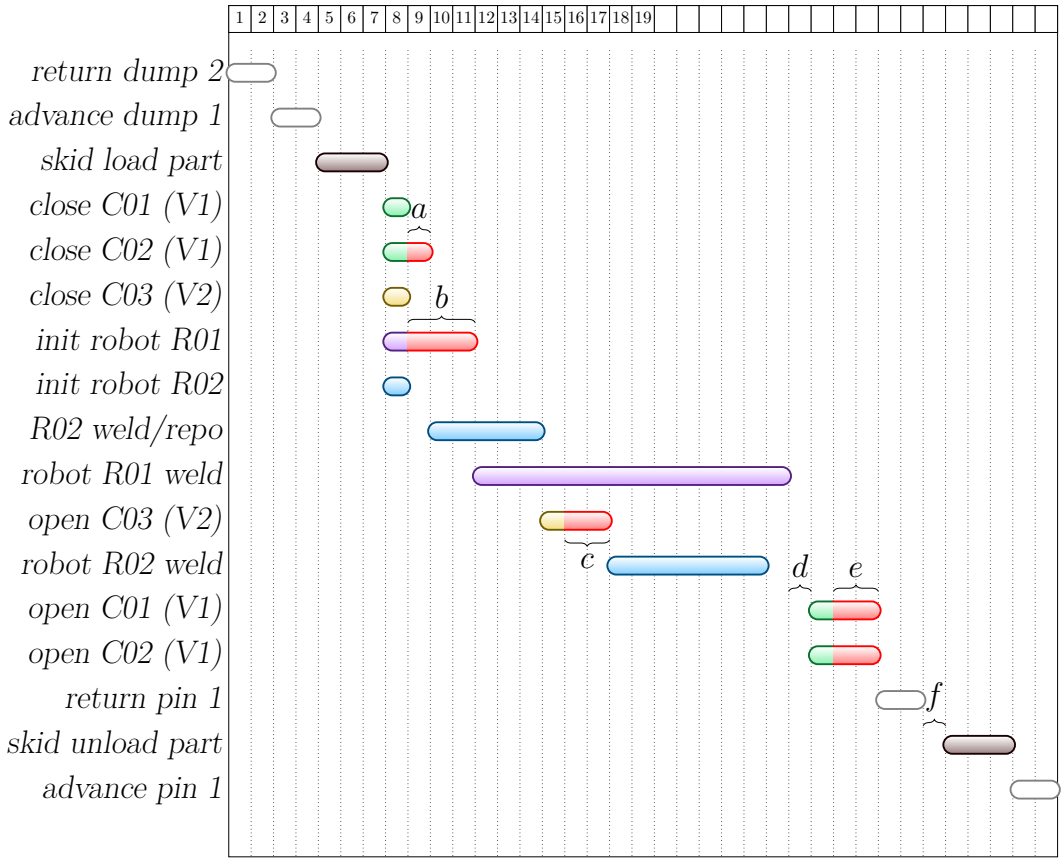

Figure 3.3.1: Gantt Chart Of Example Machine Sequence (20% Improvement Possible)

3.4 Process Model Discovery

Real world logs have shown that the actual process of industrial equipment does not always mirror the design intent. Therefore, the models developed during the design phase cannot be used to derive additional value. Contributing factors are late changes in design, leading to changes within the process. These changes need to be provided to manufacturing as quickly as possible to limit the impact on timing and cost. This rush reflects on the quality of the associated documentation. Often also mechanical interferences and cycle time issues are found during the integration phase, as described by Koehler et al. (Koehler and Jing 2018), which are addressed through, mostly undocumented, sequence changes by the integration engineers.

Besides the apparent compliance checking of the actual sequence against the design intent, there are several potential use cases for a process model. Replaying the equipment log, within the process model, can highlight unintended delays and bottlenecks. At the same time, an accurate model allows the prediction of idle times within a production cell. This idle time can then be used for automated maintenance tasks like

robot brake checks, weld tip dressing or sealer nozzle purging, without impacting the throughput of the cell. It is also imaginable that the model could be applied to operator guidance by identifying the next task to be executed to achieve maximum performance. Finally, the model can be used for diagnostics. Highlighting the completed edges allows maintenance to pinpoint what steps did not complete on time.

3.4.1 Reasons For Traces

While exploring an actual automotive equipment log, using the Process Mining software DISCO, a surprisingly high number of recorded traces were discovered. Before using these traces for process model discovery, the reason for their manifestation had to be understood. The author's research led to the following three contributors:

1. **Log Issues:** As mentioned in the introduction, there have been many works discussing reasons for incomplete logs. Since the model discovery does not require consecutive cases within the log, the simplest approach to deal with flawed logs is to remove the problematic events. Evaluating the remaining cases with the help of machine learning allows for the use of complete cases for the model discovery.
2. **Style Dependencies:** The research also showed that there is a high probability that the sequence of a production station will vary based on the part style being manufactured. Such variations are called 'process variants' in the referenced works. Bolt et al. (Bolt, W. M. P. Van Der Aalst, and Leoni 2017) recommend splitting the log into sub-logs 'to reduce the variability and complexity'. This leads to the conclusion that a simple and correct model can only be discovered for one style at a time.
3. **Partially Asynchronous Concurrent Processes:** In the software, domain concurrency is given if two processes are executed in parallel within two different threads. When carrying this concept over to industrial automation, it can be seen that many concurrent processes have a common predecessor, and after their execution lead to a single succeeding process step. Their execution is asynchronous. There are multiple reasons how this can lead to different traces being recorded. Malfunctions or utility fluctuations can cause extended activity life cycles (e.g., a robot weld gun getting stuck to the part). The log typically spans a single manufacturing station. Sometimes the robots are utilised in multiple stations, which might cause the robot not to always be available at the same time. Automated maintenance tasks, like weld tip dressing, can also cause unrecorded delays.

It is mainly the third category of reasons that contributes to traces that can aid the model discovery. It, therefore, is concluded that the traces originating from the first

two factors need to be eliminated before attempting model discovery.

3.4.2 Assumptions

Within the Process Mining community it is generally assumed that the discovery of a complete process model is more likely if it is based on a log with complete cases. Therefore, only complete cases will be used for model discovery. Further, it can be assumed that within a complete case, every 'complete' timestamp is the predecessor of another activity's 'start' timestamp.

Typically, a case starts with only one activity e.g., the part being loaded and ends with a single activity, which often is the part being unloaded.

Very rarely, it is found within automated equipment sequences that the same activity occurs twice within one cycle. In such a case, it would be necessary to differentiate the two occurrences with the help of an index in the design structure matrix to avoid confusion. Within a parsing algorithm, this would require some additional counters. It was concluded that this does not require any further investigation. A similar approach has been described by Li et al. (J. Li, D. Liu, and B. Yang 2007) who temporarily renamed duplicate task names until the process model discovery is completed.

One of the shortcomings of the α-algorithm is that it cannot discover short loops. These are instances where an activity is the predecessor of itself, which corresponds to the diagonal within the design structure matrix. It is assumed that such behavior is not typical to automated production equipment and it is not considered within this research.

3.4.3 Dependency Definitions

For industrial automation process model discovery, the author proposes an extension to the rules established for the α-algorithm by Van der Aalst (W. V. D. Aalst 2016b). In the below definitions '$a >_L b$' should be read as 'a is directly followed by b' while '$b \not> _L a$' means 'b is not directly followed by a'. '$a \rightarrow_L b$' expresses the causal relationship between a and b. Contrary '$a\#_L b$' shows that there is no relation between a and b. Finally, '$a \parallel_L b$' denotes that a and b have been observed in parallel.

Original α-algorithm definitions: Let L be an event log over $\mathscr{A}$; i.e. $L \in \mathbb{B}(\mathscr{A}^*)$. Let a, b $\in \mathscr{A}$.

1. $a >_L b$ if and only if there is a trace $\sigma = \langle t_1, t_2, t_3, ..., t_n \rangle$ and $i \in \{1, ..., n-1\}$ such that $\sigma \in L$ and $t_i = a$ and $t_{i+1} = b$;

2. $a \rightarrow_L b$ if and only if $a >_L b$ and $b \not> _L a$;
3. $a \#_L b$ if and only if $a \not> _L b$ and $b \not> _L a$;
4. $a \parallel_L b$ if and only if $a >_L b$ and $b >_L a$;

Proposed α_{LC}-algorithm definitions: Let L be an event log over $\mathscr{A}$; i.e. $L \in \mathbb{B}(\mathscr{A}^*)$ Let sub-log $l \in L$ be limited to one-part style and one case per trace. Let a, b, c $\in \mathscr{A}$, t_s a start event and t_c a complete event.

1. $a >_l b$ if and only if there is a trace $\sigma = \langle t_1, t_2, t_3, ..., t_n \rangle$ and $i \in \{1, ..., n-1\}$ where the timestamp ($\tau(t_i) < \tau(t_{i+1})$ or $\tau(t_i) = \tau(t_{i+1})$ as long as not ($t_i = t_c$ and $t_{i+1} = t_s$) and $k \in \{i+1, ..., n\}$ such that $\sigma \in l$ and $t_i = a_c$ and $t_k = b_s$ and there is no j such that $i < j < k$ and $t_j = c_c$;
2. $a \rightarrow_l b$ if and only if $a_c >_l b_s$ and $b_s \not>_l a_c$;
3. $a \#_l b$ if and only if $a_c \not>_l b_s$ and $b_c \not>_l a_s$;
4. $a \parallel_l b$ if and only if $a_c >_l b_s$ and $b_s >_l a_c$;

Although α_{LC}-definition #3 & #4 holds, they are of no importance for the proposed algorithm. Gantt charts are best suited to explain the two other definitions more in detail. According to Sommer (Sommer 2012) they are typically used in the automation domain to visualise the sequence of operation for a machine. The columns represent time increments while bars, displayed in the rows, stand for the duration of an activity where the beginning of the bar marks its 'start' time and the end the 'complete' time. For completeness, it needs to be mentioned that Gantt charts often also include links to express the dependencies between the different activities. It is precisely those dependencies that Process Discovery aims to discover. Therefore, the Gantt charts shown within this thesis are drawn without the links: α_{LC}-definition # 1 stipulates that $a >_l b$ if the timestamp $\tau_s(b) \geq \tau_c(a)$. Also, there cannot be a complete-timestamp of another activity in between unless $\tau_s(b)$ is the first start-timestamp following $\tau_c(a)$. Figure 3.4.1 shows that the activities b and c, contrary to activity d, fulfill these requirements concerning activity a. If activity c, is recorded a little earlier, as shown in figure 3.4.2, then the α_{LC}-definition #1 leads to the conclusion that $a, c >_l b, d$.

α_{LC}-definition #2 excludes a dependency found according to the α_{LC}-definition #1 if any case is observed where the timestamp $\tau_c(a) > \tau_s(b)$. Possible such scenarios are shown in figure 3.4.3 in red for activity a in relation to activity b.

3.4.4 DEPENDENCIES MATRICES

As described in the related works, dependencies matrices are a tool often used in the Process Mining domain. They consist of a square matrix with equal column and row

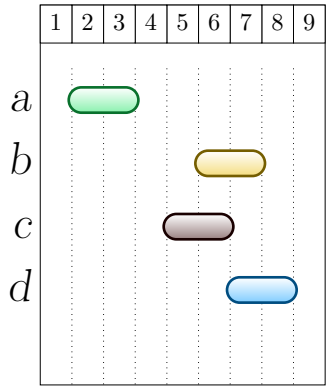

Figure 3.4.1: Dependency Example 1

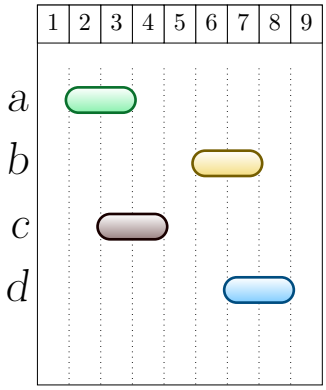

Figure 3.4.2: Dependency Example 2

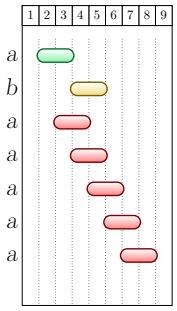

Figure 3.4.3: Criteria To Exclude Dependencies

names. For this research, the column/row names are the names of the activities within a case sorted by their start timestamps. An example can be seen in figure 3.4.4. Here the rows are related to the start timestamp while the columns represent the complete timestamp. The diagonal of this matrix represent instances where an activity depends on itself. These cases are referred to as short loops in the Process Mining domain and do not exist for industrial assembly processes. A '1' marks a discovered dependency while a '0' marks a not existing dependency. The lower triangle of the matrix is used to mark situations where a start timestamp, according to the definitions of section 3.4.3 directly follows the complete timestamp of its previous activity ($a_c >_L b_s$). The upper triangle is not used because the proposed algorithm considers life cycle information which can not be represented in a single dependencies matrix. The opposing dependencies matrix, representing complete timestamps (rows) following start timestamps (columns) ($a_s >_L b_c$), uses the upper triangle only. Inverting and transposing the opposing matrix onto the initial matrix using a logical 'AND' function creates the final matrix from which a flowchart can be constructed.

Figure 3.4.4: Sample Dependencies Matrix

Motion (ID)		complete										
		load part (1)	close clamps 1 (2)	close clamps 3 (4)	initiate R01 (5)	initiate R02 (6)	R01 weld / clear (7)	R02 weld / reposition (8)	open clamps 3 (9)	R02 weld / clear (10)	open clamps 1 (11)	unload part (13)
start	load part (1)	0	0	0	0	0	0	0	0	0	0	0
	close clamps 1 (2)	1	0	0	0	0	0	0	0	0	0	0
	close clamps 3 (4)	1	0	0	0	0	0	0	0	0	0	0
	initiate R01 (5)	1	1	1	0	0	1	0	1	0	0	0
	initiate R02 (6)	1	1	1	1	0	0	1	0	0	0	0
	R01 weld / clear (7)	0	1	1	1	1	0	0	1	1	1	0
	R02 weld / reposition (8)	0	1	1	1	1	1	0	0	0	0	0
	open clamps 3 (9)	0	0	0	1	0	1	1	0	0	0	0
	R02 weld / clear (10)	0	0	0	1	0	1	0	1	0	0	0
	open clamps 1 (11)	0	0	0	0	0	1	0	0	1	0	0
	unload part (13)	0	0	0	0	0	0	0	0	0	1	0

3.4.5 Simplified Flowchart

Flowcharts often are the tool of choice in the industrial automation domain. Although they can contain logical splits and joins as well as decisions, the flowcharts used for this research consist of activities and dependencies only. Because the models are discovered for each style individually, there should be only one, ever repeating sequence of operations. Based on this assumption, it can be defined that multiple dependencies shown for an activity must be an 'AND' joint or split. An example of such a flow chart can be seen in figure 3.4.5.

3.5 Interactive Trace Induction

One of the main concerns in the Process Discovery domain is the quality of the discovered model. The metrics required to express confidence in these approximations are not satisfactory. In his 2002 paper, Browning (Browning 2002) makes an important observation by stating: 'Some people confuse the real process (how work is really done) with the process model or description, which is only an abstract representation of the real process. ... Thus, a prerequisite to process improvement (changing the real process) is increasing the adequacy and accuracy of the process model'.

3.5.1 Traces and Their Usage

Traces found in business processes are primarily caused by either faulty/incomplete logs or human intervention. Traces in industrial automation processes, on the other hand, are also caused by different part styles and parallel processes. Since these concurrent processes are often independent of each other, it has to be expected that the position of their activities within a log will vary. Also, the duration of the activities can vary,

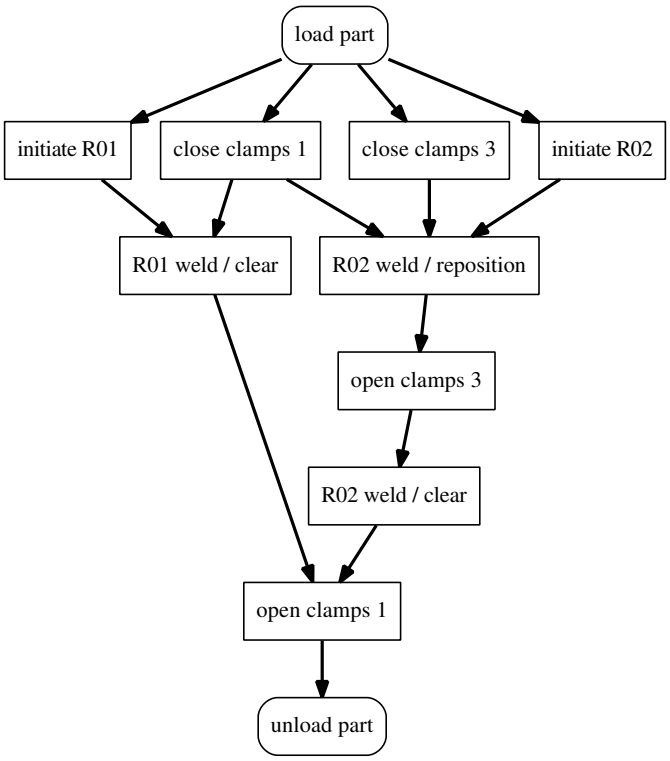

Figure 3.4.5: Sample Flow Chart

which mostly is caused by fault conditions. For example, a clamp might close slower, because of weld slack collecting on it, which in turn will lead to the depending activities to show up later within the log.

A casual observer typically concludes that the sequence of activities within a case clearly describes the process. This observation is a misconception. Figure 3.5.1 shows an example log representing the sequence of activities for a single part style. Applying the α_{LC}-algorithm to one case only will yield the flowchart shown in figure 3.5.2, which differs from the complete model shown in figure 3.5.3. The reason for these discrepancies is the lack of dependencies which lets the onlooker infer that the start of a new activity depends on all of the recently completed activities. This confusion is best seen in the associated Gantt chart figure 3.5.4.

The discovery of dependencies is only possible with the help of additional traces being recorded. Figure 3.5.5, for example, is a case of an equipment where activity $a, b \rightarrow_l c$ and activity $d \rightarrow_l e$. Looking at this one case only will lead to the conclusion that activity $a, d >_l e$ and $b, e >_l c$. Only when the second case (shown in figure 3.5.6) is considered it will become apparent that $a \rightarrow_l c$ and that $a \#_l e$. Therefore traces aid the model discovery by breaking up falsely perceived links and revealing previously unseen dependencies.

Figure 3.5.1: Log Example Style 1

caseID	style	motionID	startTime	completeTime
1	1	1	00:00:00	00:00:04
1	1	2	00:00:04	00:00:05
1	1	4	00:00:04	00:00:05
1	1	5	00:00:04	00:00:05
1	1	7	00:00:04	00:00:05
1	1	6	00:00:05	00:00:10
1	1	8	00:00:05	00:00:12
1	1	9	00:00:13	00:00:14
1	1	10	00:00:14	00:00:18
1	1	11	00:00:18	00:00:19
1	1	13	00:00:19	00:00:23

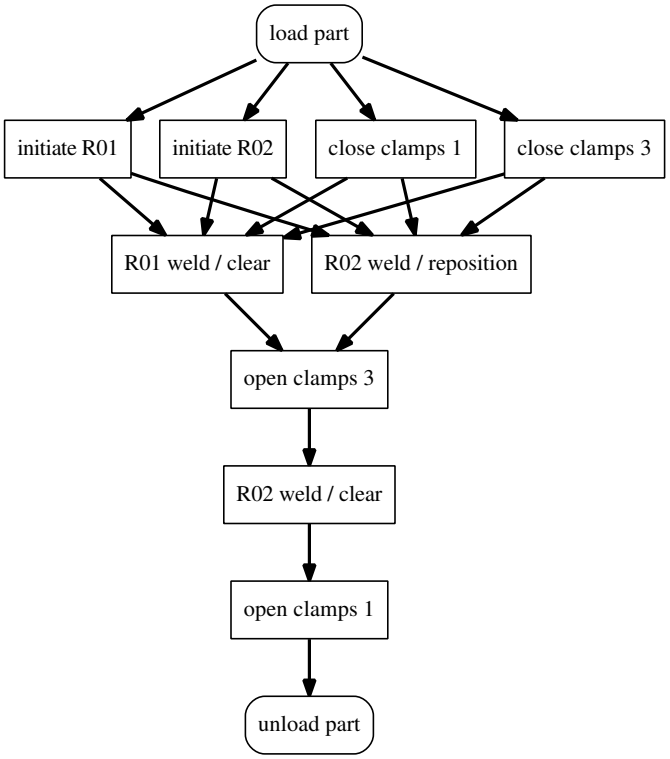

Figure 3.5.2: The One Trace Model

3.5.2 Required Traces

During previous experiments, it was found that the α_{LC}-algorithm is capable of discovering a highly accurate process model from a log comprised of just three, purposely selected cases. This led to the following question: 'Can an algorithm determine which minimum traces need to be present to allow for a highly accurate model discovery?' Taking the Gantt chart figure 3.5.4 as an example, the potential causes of confusion,

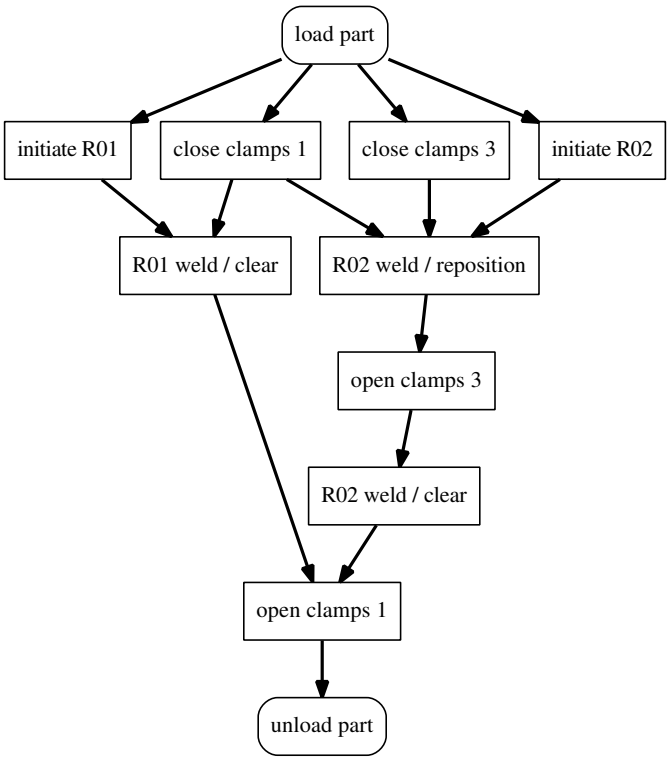

Figure 3.5.3: The Complete One Style Model

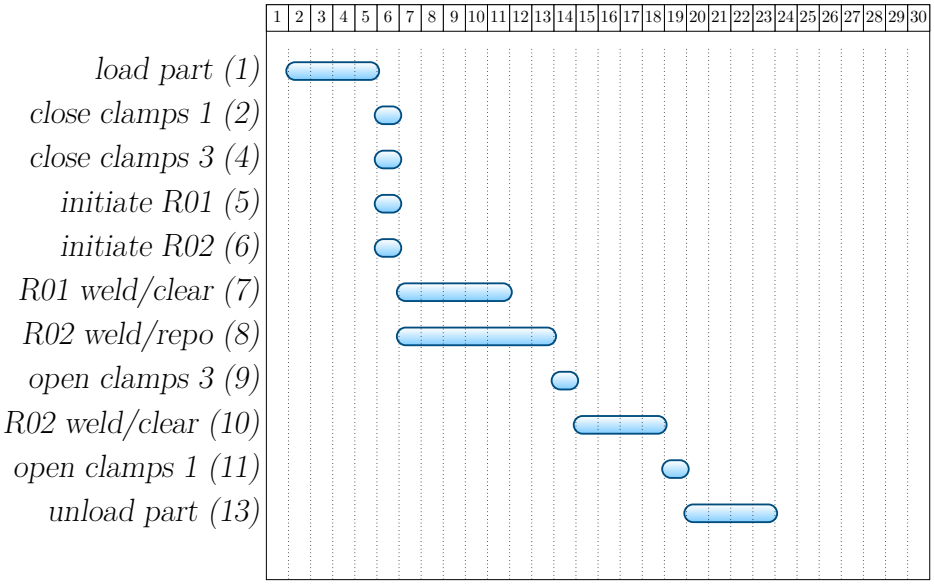

Figure 3.5.4: Random Case

due to the lack of links, can be pinpointed. The 'start' and 'complete' events of activities 2, 4, 5 and 6 are identical. Also, the 'start' events of activities 7 and 8 follow directly after that. Therefore, it is not possible to determine which activities depend on each other. This confusion can only be solved if one of the four previously men-

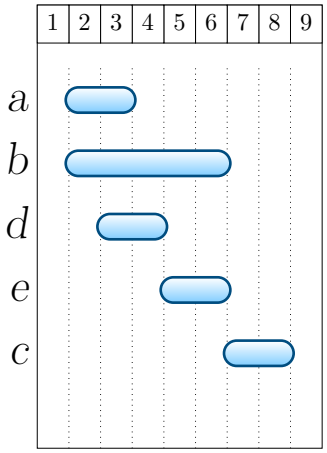

Figure 3.5.5: Example Case 1

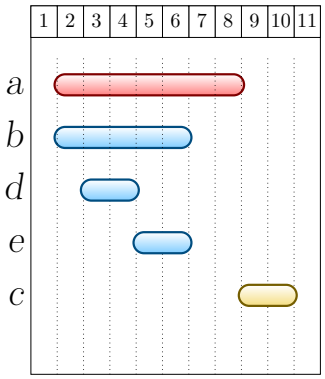

Figure 3.5.6: Example Case 2

tioned activities completes later than the other three. Only then it will become clear which other activities depend on it. All dependencies, therefore, will be known if such a scenario has been observed for all four activities. The same reasoning can be applied to activities 7 and 8 since they are executed in parallel. The following theorems have been developed to limit the traces to be recorded to a minimum:

Theorem 6 *Contrary to business processes the edge in a flowchart representing an industrial automation process can be considered instantaneous, which means that an activity starts immediately after the last preceding activity has been completed. If there is only a single preceding activity the dependency can be considered as correct and does not need further checking.*

Theorem 7 *If activity 'b' directly follows activity 'a' ($a >_l b$) and activity 'c' directly follows activity 'b' ($b >_l c$) then there is no explicit dependency between activity 'a' and 'c' ($a \#_l c$). This also holds for longer dependency chains.*

Theorem 8 *Finally definition two of the α_{LC}-algorithm (see section 3.4.3) holds which states that b_s can not depend on a_c if a_c has been observed later than b_s as well.*

The dependencies matrices introduced in chapter 3.4.4 also can be helpful to determine which minimum traces are required to discover a highly accurate process model. Parsing the random case, depicted in figure 3.5.4 will yield the dependencies matrix shown in figure 3.5.7. In this initial matrix all direct following dependencies are marked by an 'X' because none of the above definitions have been applied and therefore the dependencies have not been confirmed.

Applying theorem 6 allows the values of the cells highlighted in green in figure 3.5.8 to be changed to '1' because there is only one preceding activity. According to theorem

Figure 3.5.7: Initial Depend. Matrix

Motion (ID)		complete										
		load part (1)	close clamps 1 (2)	close clamps 3 (4)	initiate R01 (5)	initiate R02 (6)	R01 weld / clear (7)	R02 weld / reposition (8)	open clamps 3 (9)	R02 weld / clear (10)	open clamps 1 (11)	unload part (13)
start	load part (1)											
	close clamps 1 (2)	X										
	close clamps 3 (4)	X										
	initiate R01 (5)	X										
	initiate R02 (6)	X										
	R01 weld / clear (7)		X	X	X	X						
	R02 weld / reposition (8)		X	X	X	X						
	open clamps 3 (9)							X				
	R02 weld / clear (10)								X			
	open clamps 1 (11)									X		
	unload part (13)										X	

Figure 3.5.8: Depend. Matrix 1st Iteration

Motion (ID)		complete										
		load part (1)	close clamps 1 (2)	close clamps 3 (4)	initiate R01 (5)	initiate R02 (6)	R01 weld / clear (7)	R02 weld / reposition (8)	open clamps 3 (9)	R02 weld / clear (10)	open clamps 1 (11)	unload part (13)
start	load part (1)											
	close clamps 1 (2)	1										
	close clamps 3 (4)	1	0									
	initiate R01 (5)	1	0	0								
	initiate R02 (6)	1	0	0	0							
	R01 weld / clear (7)	0	X	X	X	X						
	R02 weld / reposition (8)	0	X	X	X	X	0					
	open clamps 3 (9)	0						1				
	R02 weld / clear (10)	0						0	1			
	open clamps 1 (11)	0						0	0	1		
	unload part (13)	0						0	0	0	1	

7 all the values of the red shaded cells can be set to '0' since they would be part of chained dependencies. Finally, based on theorem 8 the cells marked in blue can be set to '0' as well because parallel activities can not depend on each other.

Above example shows that the evaluation of a single trace already lead to the discovery of 50% of the existing or non-existing dependencies. The cells still marked with an 'X' and the blank cells are yet to be evaluated. Now it is possible to repeat this process by interactively adding, purposely chosen traces that can explain the remaining dependencies. The presence or absence of these traces within the log allows for a better judgement regarding the quality of the resulting process model.

3.5.3 Interactive Trace Induction

In section 3.5.2, the criteria for a minimum set of traces needed for a highly accurate model discovery, have been laid out. Fortunately, the automation domain offers the opportunity to influence the process. This thesis, therefore, proposes the concept of 'interactive trace induction'. In many of today's PLC programs, physical sensor inputs are mapped to meaningful tags. Instead of hoping for weld slack, that slows down the clamp, to build up an additional condition can be added to that mapping, which allows the sensor tag to be delayed (see figure 3.5.9). Initiating that delay, manually or through OPC, will result in a new trace being recorded.

The rules as stated in section 3.5.2, first can be used to determine which of the activities are suspect. Once identified, the corresponding sensor tags can be delayed one by one, until all unrelated activities are evaluated. This process needs to be repeated for every style. The outcome is a event log of all traces, which are required to describe the process.

If the inclusion of this additional tag becomes standard, there will be no extra time required for its implementation into new production equipment. Modifying legacy equip-

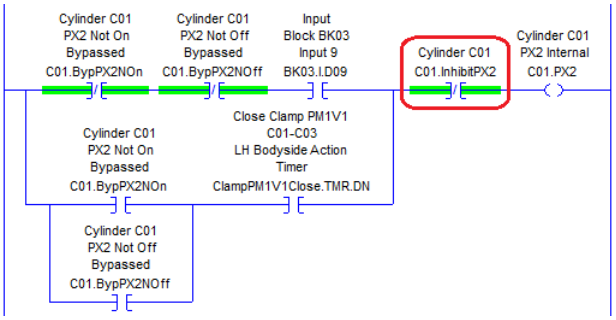

Figure 3.5.9: Required Logic Modification

ment will require only little effort, as long as its control logic adheres to some tag standardisation.

The main advantage of 'interactive trace induction', is that the process of model discovery no longer depends on chance. Instead, the required traces are induced on demand. Since this can be done at a convenient time, it has minimum impact on production. At the same time, external processes impacting the cases recorded, can be kept to a minimum, thus increasing the quality of the log. Because 'interactive trace induction' can discover the desired model with a minimum of traces, the time for data logging can be reduced from weeks, or even months to just a few hours. Besides, the processing time for such a minimalistic log decreases dramatically.

3.6 Summary

In the course of this chapter a methodology was introduced that allows for the automated discovery and logging of manufacturing equipment status data (see section 3.1). Since the data obtained can be flawed, rules were established in section 3.2 to gauge the log quality and to potentially correct the identified faults. Clustering the events into cases proved to be the main requirement for any data analysis activity. The clustering was achieved by splitting the events into five categories which then were combined into cases. It has been hypothesized that the completeness of those cases can be evaluated by applying additional engineering features combined with machine learning algorithms.

The pre-processed event logs are the foundation of the knowledge-based improvement discovery algorithms described in section 3.3. These algorithms aim to identify previously unknown irregularities that slow down the manufacturing process. Business process mining has a similar aim and it therefore was reasoned that it could be applicable for manufacturing processes as well. This paper introduces in section 3.4 the α_{LC}-algorithm which enables the discovery of a highly accurate process model based on the previously captured production equipment event log. The work with the α_{LC}-

algorithm triggered the question if it is possible to determine which cases had to be observed to discover a highly accurate process model. This thought finally led to the development of the iterative trace induction described in section 3.5. This algorithm represents a reversal of the discovery process by making assumptions regarding the process at first and then validating each of those assumptions by delaying potentially related activities.

In the following chapter the above described framework is put to the test utilising an extensive data set obtained from an automotive body shop and some artificial data sets used to simulate behaviour that was not observed within the real life data.

4

Case Study And Experiments

The majority of experiments were done with data from an automotive body shop at the Opel manufacturing plant in Eisenach/Germany. At the time of the study this plant manufactured, the now obsolete, Opel Adam at a rate of approximately 600 - 1000 units a day. The output range strongly depended on the number of production shifts.

Often a body shop is divided into several areas. Typically zones for the manufacturing of the underbody, the body sides, the roof and the hang-on parts (doors, hood ...) can be found. The underbody, the body sides and the roof are brought together in geometric fixtures, so-called framing stations, where they are tagged together. This process is followed by the re-spot stations where the remaining weld spots are placed on the body of the car. Finally, the hang-on parts are attached to the car body on the fitting line. The slowest station dictates the output of the body shop since all these manufacturing stations are interconnected.

At the Eisenach plant manufacturing engineering had identified four re-spot stations as the bottleneck with the help of the existing plant floor systems. A systematic review of theses stations, however, did not lead to any improvement suggestions. For that reason, it was decided that for this project, the data of all the Rockwell based cells within this body shop should be collected but the focus of the analysis should be on said four stations.

One of the stipulations for the project was to use, wherever possible, existing tech-

nology and infrastructure. This restriction would keep the budget to a minimum and eliminate the need for additional training of the onsite personnel.

As previously mentioned, the project was split up into five sub-projects. Therefore the remainder of this chapter is structured as follows: Section 4.1 explains the details of the automated data collection followed by the preprocessing of the recorded data in section 4.2. Section 4.3 shows the application of the previously introduced heuristics to generate Gantt charts that are marked up with improvement suggestions. Section 4.4 documents the application of Process Discovery algorithms since the previously obtained charts are based on perceived dependencies between the events rather than the real dependencies. The methodology chapter 3.5 introduced the concept of 'interactive trace induction' which allows for the recording of the traces needed for a proper model discovery. The matching experimental results are presented in Section 4.5.

4.1 Data Collection

The methodology described in chapter 3.1 was implemented using VB.net and the Advosol OPC DA .NET framework. It has been installed on a single workstation PC, along with an OEM version of RSLinx, as OPC server and a developers version of SQL Server. The chosen OPC settings have been previously explained in chapter 3.1.3. At the time of the experiment, 193 stations, controlled by 27 Rockwell PLCs, were monitored, while logging ~1.000.000 events a day. The monitoring architecture is shown in figure 4.1.1.

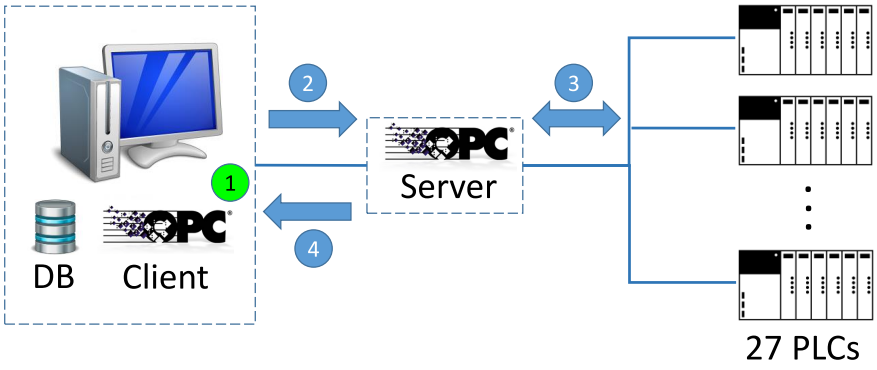

Figure 4.1.1: Monitoring Architecture

It consists of a PC on which a SQL Developer Server is installed together with the developed collection algorithms which also act as an OPC client (1). This OPC client communicates with RSLinx, which represents the OPC server. Depending on the license used RSLinx can either be installed on the same computer on which the OPC

client resides, or it could be on a remote computer allowing access to multiple clients. Through the OPC client, OPC groups with the items to be monitored are defined and submitted to the server (2). Within the OPC server, 27 topics linking to 27 PLCs have been configured to enable communication. The OPC server monitors the requested items (3) and upon detecting a change raises an event within the OPC client. This event leads to its tag being stored within the SQL database together with its current state and a timestamp.

One of the main concerns, within IT, was the increased network load due to the data collection efforts. The system was implemented in a plant which has a 100 Mbit/s Ethernet system for the production floor level. It was found that communication to the PLCs increased by 3.8%; from 371.164 packets/s to 385.171 packets/s. Since this was a minor increase of overall network traffic, the work was cleared to proceed by the IT department.

The controls group, on the other hand, was concerned that the constant monitoring of the PLC through OPC would put an additional burden on the overhead time slice. Therefore some code was created that could measure the overhead time slice within the PLC, and it was found that, although the monitoring caused an relative increase of 3.89%, from 7.7% to 8%, it was still well within the preset boundaries of maximum 10%.

Additional code was implemented within the PLC program that measured 200 cycle times of a particular motion in parallel to the OPC data acquisition to evaluate it the external monitoring would introduce any errors. The result is shown in figure 4.1.2 and can be summed up by stating that the OPC measurements were within $\pm 1.5\%$ of the measurements obtained directly within the PLC. What can not be seen in the figure is that the timeline of the OPC curve is shifted by an average of 100ms to the right. These errors were deemed acceptable for this research.

The data for these experiments were collected over a two day period. The data logging resulted in approximately 330,000 events being recorded for the four said stations. A more in detail evaluation of these records is given in the following chapter.

4.2 Log Preprocessing

Before going into the preprocessing details it needs to be mentioned that the sequences of operation, for the four chosen stations, were unknown to the author of this thesis. This underlines that the proposed methodologies are not customised to this problem but instead are applicable to assembly and joining processes in general. A typical sequence for such a station is shown in figure 4.2.1:

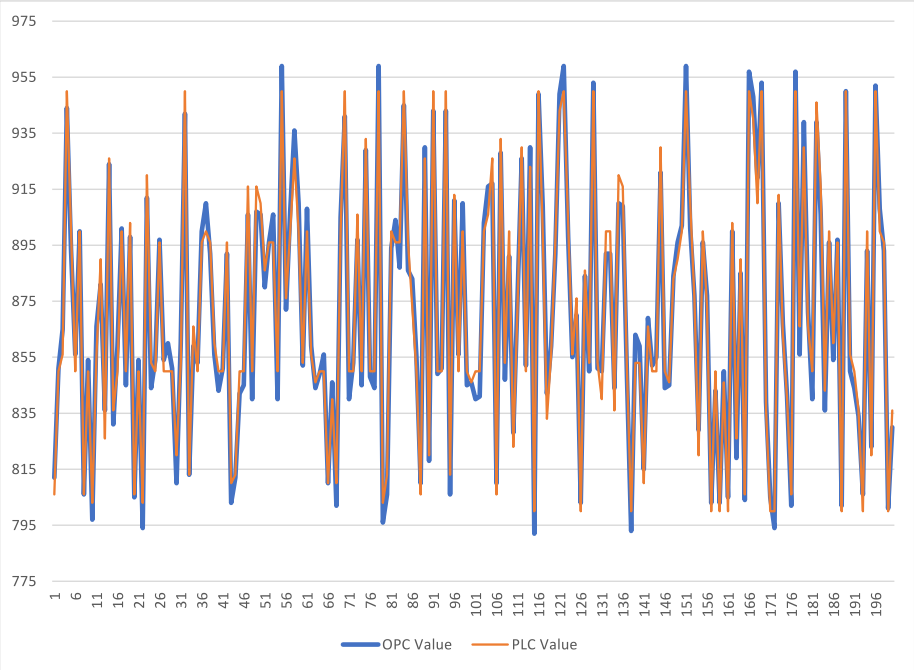

Figure 4.1.2: OPC vs. PLC Cycle Values

4.2.1 Quality Evaluation

The methodologies to evaluate the different quality criteria introduced in sections 3.2.2 - 3.2.18 were encoded with Python and applied to the data obtained from the Eisenach body shop. The results have been combined in table 4.2.1. It has to be noted that the rows marked in gray do no impact the logs quality as these faults can be corrected during pre-processing.

When calculating the correctness of the recorded events, missing relationships were not considered, because they can be correctly discovered with the algorithm proposed in section 3.2.5. The incorrect events and incorrect relationships were also not included in the calculation because they are caused by additional events due to programming and mechanical errors, which can be removed. The counts shown for incorrect timestamps are based on the station in manual mode for which the whole case was removed. Let t_m be the missing events in % and τ_m the missing time stamps in percent then the correctness c can be expressed as:

$$c = 100\% \ - \ (t_m + \tau_m) \tag{4.1}$$

The missing timestamps could also be estimated using various imputation methods. It, however, is presumed that imputation introduces an error within the data as well.

The table 4.2.1 shows the highest error rate was recorded for station 2. The error of

Table 4.2.1: Evaluation Summary

description	station 1		station 2		station 3		station 4	
	count	%	count	%	count	%	count	%
events recorded	59724	100	89682	100	104064	100	76004	100
correctness of records	99.22%		96.42%		99.61%		99%	
missing / unclear events	222	0.37	774	0.86	371	0.36	589	0.77
missing relationship	1198	2	2424	2.7	1224	1.18	1229	1.62
missing timestamps	242	0.41	2438	2.72	35	0.03	176	0.23
incorrect events	88	0.15	1137	1.27	203	0.2	2046	2.69
incorrect relationships	216	0.36	21	0.02	0	0	61	0.08
incorrect timestamps	127	0.21	315	0.35	251	0.24	165	0.22
imprecise timestamps	0	0	0	0	0	0	0	0
irrelevant events	0	0	0	0	0	0	0	0
cleaned cases obtained	1157		1181		1185		1184	

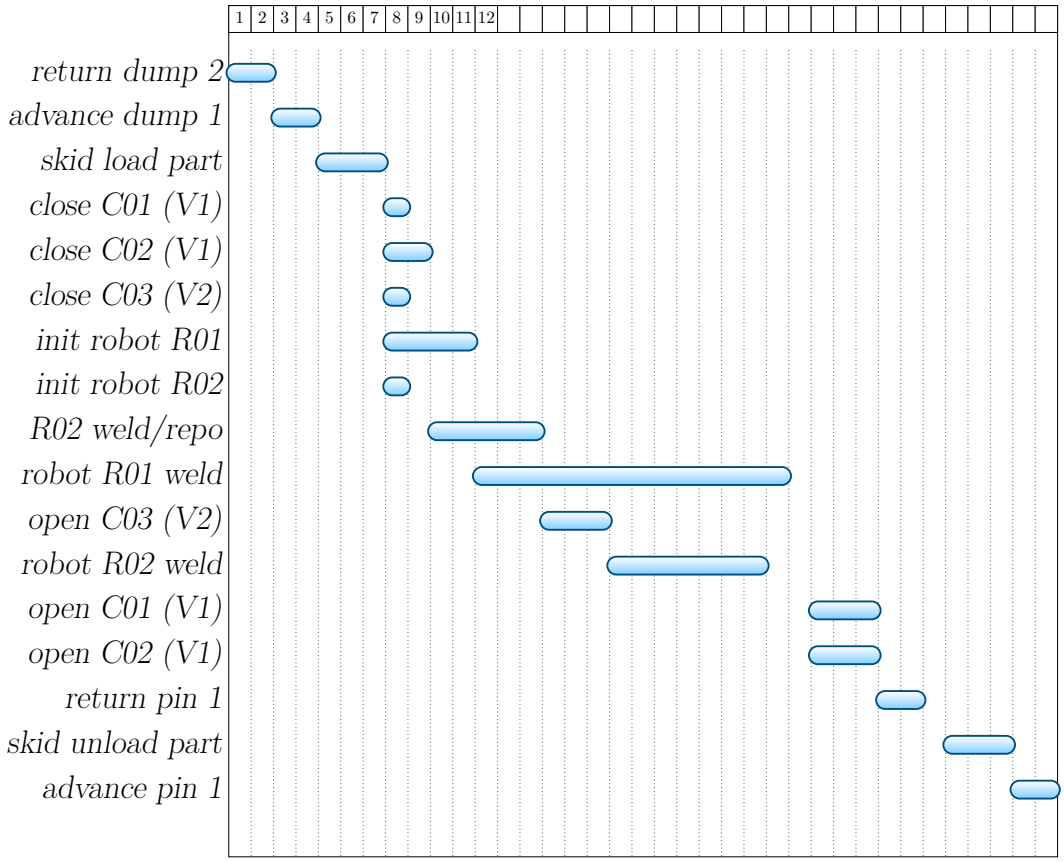

Figure 4.2.1: Gantt Chart Of Example Machine Sequence

less than 4% could be considered marginal, and because the model discovery does not require a seamless log, it was decided to delete all events with missing or unclear events as well as missing timestamps as discussed in section 3.2.20.

4.2.2 Case Clustering

Missing relationships, as explained in section 3.2.5, need to be resolved because the analysis described later on require that the events are clustered into cases which each represent one individual machine cycle. As previously mentioned the process activities can be identified based on a part being present in the station. Algorithm 2 offers a methodology that allows to encode this concept in Python:

In the next step, the load and unload events are identified. Load events are events that have been initiated together just previous to the first event with a part being present. The unload event, according to the definitions from section 3.2.5, is the last event for which the presence of a part has been recorded. Remaining are the transition events. To differentiate between reset and setup events, fields are added that capture the previous and next style. A query then is used to determine whether the event

```
Result: All steps belonging together are identified with a unique sequence
        number (seqNum)
set seqNum = number of part present signals;
loop through events from the bottom up;
while not first recorded event do
    if part present signal then
        store reverse of signal as part present (PP);
        if PP then
            reset paste mode;
            set copy mode;
        else
            set paste first step mode;
        end
    else
        if paste mode then
            if PP then
                paste style & seqNum;
            else
                if paste first step mode then
                    paste style & seqNum;
                    memorise first step tag;
                    reset paste first step mode;
                end
                if tag = first step tag then
                    paste style & seqNum;
                else
                    reset paste mode;
                end
            end
        end
        if copy mode then
            copy style & seqNum;
            decrement seqNum;
            set paste mode;
            reset copy mode;
        end
        if neither copy nor paste mode then
            set style & seqNum to '0';
        end
    end
end
```

Algorithm 2: Process Cycle Identification

always follows a style x independent of the next part style y. If that is the case, the event can be considered a reset event. Once all reset events have been identified, it can be concluded that the remaining transition events must be set up events needed to prepare the station for the next part style. This assumption can be confirmed by double checking that the setup events are recorded the majority of the time for a given previous/next style combination.

The clustering of the data, associated with the four body shop stations, resulted in the discovery of 4640 cases. These cases were grouped together into 175 traces using a CRC32 algorithms described in section 3.2.21.

Feature Engineering

Instead of log repair this thesis applies machine learning techniques to identify flawed traces. Manual expert evaluation of the identified traces yielded that there is too little information available to make a judgement if the trace is complete or not. Therefore, additional columns for engineered features were added to the traces by applying the rules defined in section 3.2.22.

Tagging The Data

Supervised machine learning specifies that a portion of the data is tagged. This requirement lead to the tagging of 10% randomly selected traces for each of the stations. During that process, it was essential to go thoroughly through every cycle to determine its completeness. This effort proved to be more cumbersome then expected because, based on the available features, it was not apparent if a case was complete or not. Experts had to cross-reference the PLC code to determine if all events had been captured correctly. This also explains, why simple reasoning algorithms could not be used instead of machine learning. As a result of this review, a column 'isPlossible' was added to the grouped traces.

Choice Of Validation Method

The two validation methods considered where split, train and test and stratified cross validation. First the split, train and test validation was implemented but it was found that this technique reduced the already small set of tagged data even further resulting in an under-performing model. The analysis was repeated using stratified cross validation. This has the benefit that the training set, through the iterations, sees all the tagged data which in turn leads to a better model.

Choice Of Tuning Method

Most machine learning algorithms have a set of parameters which allow for fine tuning. As described by Mueller et al. (Müller and Guido 2016), finding the correct parameters is an iterative process. Mueller recommends utilising the grid search and cross validation implemented in scikit-learn over explicitly looping through all the possible parameters and doing a cross validation for each combination. During the experiments it was found that the second approach yields better results than the scikit-learn implementation. Therefore this approach was used for fine tuning in conjunction with stratified cross validation.

Machine Learning Algorithms Applied

Because of the engineered features a decision tree seemed to be the best choice to start with. Since, according to Mueller et. al (ibid.), a random forest typically outperforms a single tree, a random forest classifier was applied to the data for comparison purposes. Other classification algorithms provided by sci-kit learn and applied to the problem are: Gradient boosting classifier, k-nearest neighbors classifier, support vector classifier, logistic regression and Gaussian naive Bayes.

Summarising the experiment it can be said that all classifier algorithms performed reasonably well on the data set (see table 4.2.2). The decision tree not only showed one of the best accuracies with 99% but also one of the smallest tolerances with +/-1% considering a 95% probability. The confusion matrix (figure 4.2.2) also shows, that the distribution between the two classifications is almost equal, which is desirable for machine learning. When adjusting the hyper parameters focus was placed on avoiding false positives, which would lead to false discoveries during Process Discovery. Since a decision tree is the simplest approach, which also is human readable (figure 4.2.3), it was chosen as the processing model.

4.3 Knowledge-Based Discovery

For evaluation purposes, reasoning based algorithms for all of the above criteria (section 3.3) were implemented using Python and applied to event logs of the four real-life framing re-spot stations.

4.3.1 Excessive Manual Cycles

The station status has been recorded together with the events. Within Python, the number of cycles, interrupted through manual mode, can be counted by grouping the

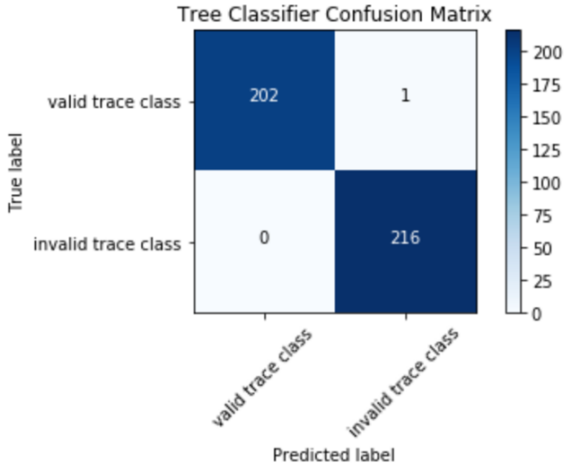

Figure 4.2.2: The Confusion Matrix

Table 4.2.2: Accuracy Of Classifier Models

	accuracy without hyper parameter tuning	accuracy with hyper parameter tuning
gradient boosting classifier	94% +/- 8%	99% +/- 1%
random forest classifier	93% +/- 8%	99% +/- 1%
decision tree	93% +/- 8%	99% +/- 1%
gaussian naive bayes	87% +/- 5%	87% +/- 5%
support vector classifier	91% +/- 8%	95% +/- 3%
logistic regression	91% +/- 12%	95% +/- 2%
k-nearest neighbors	90% +/- 7%	99% +/- 2%

case summaries based on the mode. This count can be expressed as a percentage to be evaluated based on a predetermined threshold. This evaluation is done during the preprocessing phase and does not result in a mark-up within the Gantt chart. Instead the number of manual cycles in comparison to the total number of cycles is provided to the maintenance group to act upon if required.

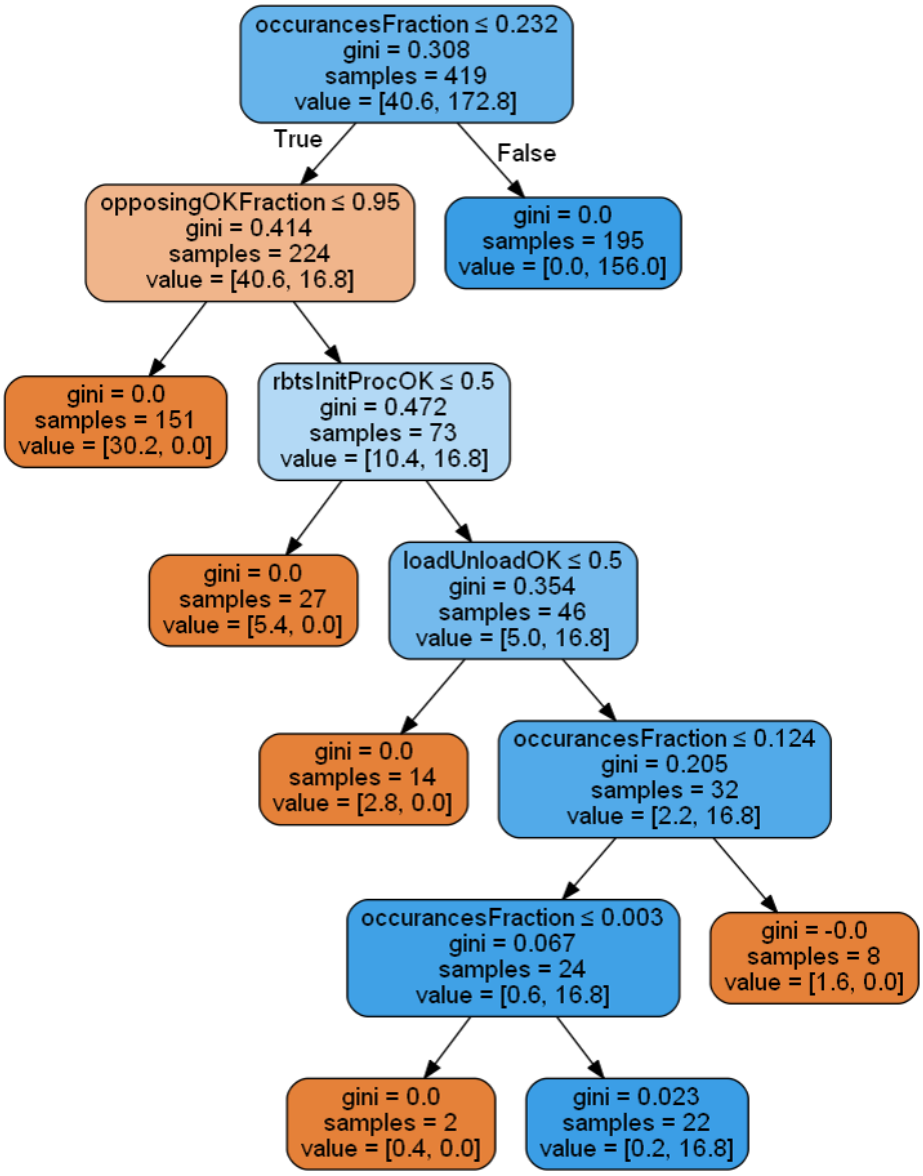

Figure 4.2.3: The Decision Tree

4.3.2 Identical Units

It can be assumed that units sharing the same pneumatic manifold are sized identically. This fact allows the assumption that the cycle time for advancing and returning those units should be identical. Within the proof of concept, the duration for every motion has previously been calculated. By grouping the motions based on their start-tag, the minimum duration can be determined and used as a reference for the remaining units. This methodology was implemented as described in the following algorithm 3.

Result: minimum cycle time for identical units

input parameters: sequence matrix;

add column duration calculating endTime - startTime;
group sequence by startTag appending the minimum duration;
rename duration column to minDuration;
merge minDuration with sequence matrix based on startTag;

Algorithm 3: Minimum Duration Of Identical Units

4.3.3 Opposing Motions

Within the log used for the experiments, opposing motions can be detected based on their unit naming. The units name can be extracted from the end tag and added in a separate column to the log. Now for each of the unit names the minimum cycle time can be determined. If the opposite motion exceeds this minimum, it can be marked up accordingly within the Gantt chart. The algorithm can be summed up as shown below in algorithm 4.

Result: minimum cycle time for opposing motions

input parameters: sequence matrix;

add column extracting unit name from motionLabel;
add column duration calculating endTime - startTime;
group sequence by unit name appending the minimum duration;
rename duration column to minMotionDuration;
merge minMotionDuration with sequence matrix based on startTag;

Algorithm 4: Minimum Duration Of Opposing Motions

4.3.4 Double Triggers

Double triggers can be identified if the same activity with an identical end timestamp exists twice in the log. This can be checked within Python with the help of the 'duplicate' function evaluating the 'motionID' and the 'endTime'. Double triggers could also be detected if two events are found of which the one with the earlier start timestamp is missing the complete time stamp.

4.3.5 Bouncing Motions

As described for the opposing motions, it is possible to identify the advance and return motion of a unit based on the naming. A bouncing motion can be identified if two activities for the same motion have been recorded within the same machine cycle

without the opposing motion being present in between. A possible approach is parsing through a case and writing every activity, with the exception of robot related activities, into a temporary matrix. Only if the opposing motion is found, the record is removed from that matrix. If the motion already is present while attempting to write it into the temporary matrix, it must be an indicator for a bouncing motion. The implementation has been done according to the following pseudo-code shown in algorithm 5.

```
Result: matrix of bouncing motions
input parameters: sequence and temp matrix;
while not last activity in sequence do
    extract the unit and action name from the startTag;
    if activity not robot related then
        if unit not in temp matrix then
            add activity to temp matrix;
        end
        if unit in temp matrix and this is opposing activity then
            remove unit from temp matrix;
        end
        if unit and same activity are in temp matrix then
            add unit to list of bouncing motions;
        end
    end
end
```

Algorithm 5: Identifying Bouncing Motions

4.3.6 Gaps

Gaps describe times where there is no motion occurring within a station. An expert reviewer can spot them by reviewing the Gantt chart. For automation purposes, the bars within the Gantt chart can be split up into 0.2s columns. The first column that does not intersect with any of the bars marks the beginning of a gap. This gap ends with the next column, that is intersected. For the evaluation, the area is marked with a red rectangle and annotated with the duration of the gap. The logic was implemented according to algorithm 6:

4.3.7 Station Blocked

As previously described the station being blocked is a special case of a gap. Therefore the same methodologies, as described for the gap detection above, apply. The difference is that this gap is detected just prior to the unload event.

Result: list of gaps

input parameters: matrices with bar start and end times;

while *not last value in start matrix* **do**

  create values between start and end time with 0.1s increments;

  append list to checkpoints list;

end

bin the checkpoints into 0.1s bins using histogram function;

while *not end of bin* **do**

  if *if bin empty and no gap start found yet* **then**

    record gap start;

  end

  if *if bin empty and gap start found yet* **then**

    record gap end;

  end

end

Algorithm 6: Gap Detection

4.3.8 Special Event - Robot Initiation

The robot initiation typically takes 1 to 2 seconds. If the actual duration, shown in the Gantt chart exceeds those limits, the activity can be marked up accordingly. Because of this simplicity, the methodology was not implemented within the experimental evaluation.

The issues automatically discovered for the four stations are summed up in table 4.3.1.

Table 4.3.1: Evaluation Results For Four Body Shop Re-spot Stations

	station 1	station 2	station 3	station 4
number of cycles recorded	1157	1181	1185	1184
excessive manual cycles	$< 0.5\%$	$< 0.5\%$	$< 0.5\%$	$< 0.5\%$
time differences for identical units	0 sec.	0.2 sec.	0.2 sec.	0 sec.
time differences for opposing motions	1 sec.	2.3 sec.	1.2 sec.	1.9 sec.
double triggers events (# of times)	7 (88)	13 (1033)	9 (203)	11 (2045)
bouncing motions events (# of times)	0	2 (104)	0	0
gaps	0.8 sec.	6.4 sec.	10.7 sec.	0.4 sec.
station blocked	0.3 sec.	5.5 sec.	0.4 sec.	0.2 sec.
difference typical to fastest trace	0 sec.	0 sec.	0 sec.	10 sec.

4.4 Process Model Discovery

4.4.1 Model Discovery

For the evaluation of the existing Process Discovery algorithms, as well as the proposed α_{LC}-algorithm, an event log was constructed artificially because the logs obtained from the four 'respot' stations did not include all the possible pitfalls often found within manufacturing stations. This became apparent when, unexpectedly, a perfect model was discovered from the real life log with a earlier version of the α_{LC}-algorithm. An artificial log also allows for better control over the number of traces contained within the derived record and also enables comparison of the discovered model to the model underlying the generated log. The model used incorporated three different styles of parts to be manufactured. For style 1, the clamps 1 are used; for style 2, the clamps 2 are actuated instead, while in style 3 both clamps are being utilised. An example of the log data is shown in figs. 4.4.1 to 4.4.3 with the above-described differences marked in yellow. The traces for these sequences were generated by randomly varying the life cycle of the different events. In total 10000 cases for all three styles were generated using an Excel spreadsheet and randomising formulas. The underlying model for style 1 is shown in figure 4.4.4 as a reference.

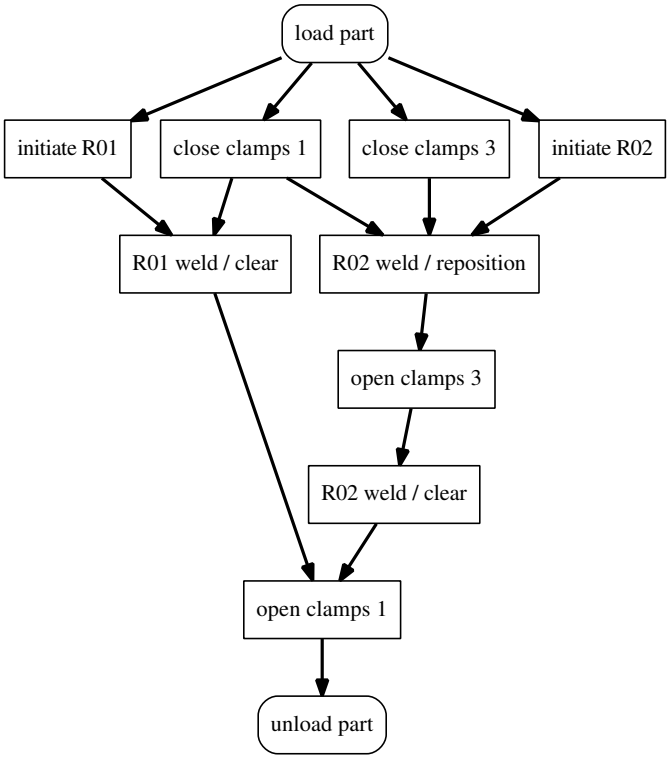

Figure 4.4.4: The One Style Model - High Trace Count

Figure 4.4.1: Log Example Style 1

caseID	style	motionID	startTime	completeTime
4	1	1	00:01:38	00:01:42
4	1	2	00:01:42	00:01:44
4	1	4	00:01:42	00:01:44
4	1	5	00:01:44	00:01:45
4	1	7	00:01:45	00:01:50
4	1	6	00:01:49	00:01:54
4	1	8	00:01:54	00:02:01
4	1	9	00:02:01	00:02:03
4	1	10	00:02:03	00:02:08
4	1	11	00:02:08	00:02:09
4	1	13	00:02:09	00:02:13

Figure 4.4.2: Log Example Style 2

caseID	style	motionID	startTime	completeTime
2	2	1	00:00:30	00:00:34
2	2	3	00:00:34	00:00:35
2	2	4	00:00:34	00:00:37
2	2	5	00:00:35	00:00:36
2	2	7	00:00:36	00:00:42
2	2	6	00:00:37	00:00:45
2	2	8	00:00:45	00:00:52
2	2	9	00:00:52	00:00:54
2	2	10	00:00:54	00:00:58
2	2	12	00:00:58	00:01:00
2	2	13	00:01:00	00:01:04

Figure 4.4.3: Log Example Style 3

caseID	style	motionID	startTime	completeTime
1	3	1	00:00:00	00:00:04
1	3	2	00:00:04	00:00:05
1	3	3	00:00:04	00:00:05
1	3	4	00:00:04	00:00:07
1	3	6	00:00:06	00:00:12
1	3	5	00:00:07	00:00:16
1	3	8	00:00:12	00:00:19
1	3	7	00:00:16	00:00:22
1	3	9	00:00:19	00:00:20
1	3	10	00:00:20	00:00:24
1	3	11	00:00:24	00:00:26
1	3	12	00:00:24	00:00:26
1	3	13	00:00:26	00:00:30

Starting with a life-cycle-based log, the α_{LC}-algorithms discovery process is executed in the following steps:

1. Cluster log into cases: (see paragraph 3.2.5). This step was not necessary for the artificial log because the generated cases already had a reliable case identifier.
2. Use decision tree learner to filter out incomplete cases: (see paragraph 4.2.2). The artificial log did not contain any flaws since it was created for model discovery, and not the evaluation of the classifier algorithms. Therefore, this processing step did not have to be applied.
3. **Filter log for one-part style only:** (see paragraph 3.4.1)

4. **Create a design structure matrix layout:** This is achieved by sorting the events within the log for one style by the start timestamp. The same order is applied for row and column headers.
5. **Split log into 'start' and 'complete' events:** The log samples figs. 4.4.1 to 4.4.3 show only one entry per activity which includes a 'start' and a 'complete' timestamp. To discover, if a 'complete' timestamp is before another activities' 'start' timestamp, it is more efficient to split the record into two events; one for the 'start' timestamp, and one for the 'complete' timestamp. The process involves duplicating the log, removing the 'complete' timestamp column from the original, and the 'start' timestamp column from the duplicate. Merging the two records, while adding a type column that indicates if the event is a 'start' or 'complete' event, will yield the desired result.
6. **Sort log by timestamp, event type, and motion ID:** According to the definition 3.4.3, an activity follows another even if its 'start' timestamp is equal to the previous events 'complete' timestamp ($\tau_c(a) = \tau_s(b)$). Sorting the log by the above added event type will put the 'complete' events in front of the 'start' events to aid this discovery.
7. **Use hashing techniques to determine traces within sub-log:** (see paragraph 3.2.21)
8. **Create sub-log $l \in L$ with just one representative case for each trace and style:** Since the proposed α_{LC}-algorithm is based on the traces available, the author proposes a reduction of the log to one representative example per trace and style only. For the sample, this means that instead of 74,074 events for style 1, only 8,140 events need to be evaluated. This approach has also previously been described by Schimm (Shimm 2004).
9. **Parse sub-log l for the opposing relation $b_{start} >_l a_{complete}$ into the binary DSM $\overleftarrow{\mathbf{D}}$:** Adhering to the definitions of 3.4.3, the algorithm shown in listing 4.1 was used to parse the sub-log into the $\overleftarrow{\mathbf{D}}$ matrix figure 4.4.5. For better visibility, the marks found have been highlighted in red.
10. **Transpose and invert $\overleftarrow{\mathbf{D}}$ to create $\overleftarrow{\mathbf{D}'^{\intercal}}$:** The result can be seen in the matrix figure 4.4.6 in which the red markings, highlighting the opposing dependencies found, persist.
11. **Parse sub-log l into the binary DSM $\overrightarrow{\mathbf{D}}$:** The markings within this matrix are placed according to the α_{LC}-definition #1 using the algorithm shown in listing 4.2. The result of this operation can be seen in matrix figure 4.4.7. The marks found have been accentuated in light green.
12. **Combine the two matrices following $\mathbf{D}_R = \overrightarrow{\mathbf{D}} \wedge \overleftarrow{\mathbf{D}'^{\intercal}}$:** This yields the result

design structure matrix $\mathbf{D}_R$ as shown in figure 4.4.8 for the high trace count log, figure 4.4.9 for the low trace count log and figure 4.4.10 for the complete high trace count log that has not been reduced to one style only. Here, the dark green markings represent the resulting, direct following dependencies.

13. **Graphic representation:** The lower triangle of the DSM $\mathbf{D}_R$ now shows all the causal relations satisfying the α_{LC}-definition #2. Since the log for one-part style only exhibits possible AND connections, a simple flowchart can easily be created from $\mathbf{D}_R$ by converting the row headers into nodes and creating edges between those nodes for all the marks within $\mathbf{D}_R$. The result for the matrices figs. 4.4.1 to 4.4.3 is shown in figs. 4.4.11 to 4.4.13. Note: the red edge shown in figure 4.4.12 represents a causality that was discovered but does not exist in reality. For the model, across all styles, in figure 4.4.13, needs to be explained that all the dependencies shown are correct but the model lacks the OR and XOR connections that allow for differentiation between the different styles. These connections could be discovered by extracting the model for each style and then mashing them afterwards. The author, however, can not think of any use cases for such a unnecessarily complex model within the industrial automation domain.

Listing 4.1: Parse sLog Into oDSM

```
for index, row in sLog.iterrows():
 if sLog.loc[index]['type'] == 'start':
  column = sLog.loc[index]['motionID']
  case = sLog.loc[index]['caseID']
  pointer = index
  while sLog.loc[pointer]['caseID'] == case and
    pointer < (len(sLog)-1):
   pointer += 1
   if sLog.loc[pointer]['type'] == 'complete':
    row = sLog.loc[pointer]['motionID']
    C4_Ref22.loc[row, column] = True
```

The whole α_{LC}-Process has been summed up in figure 4.4.14 below to offer a better overview.

4.4.2 Comparison With Process Discovery Algorithms

Testing of the established Process Discovery algorithms vs. the newly proposed α_{LC}-algorithm was done using the artificial logs as described previously, one with a low and the second with a high number of traces. As previously stated, one of the goals of this research was to discover a highly accurate process model using a framework that can be used by tradesmen who possess basic process knowledge but have no understanding

Figure 4.4.5: Opposing Dependencies Matrix - High Trace Count

Motion (ID)		start										
		load part (1)	close clamps 1 (2)	close clamps 3 (4)	initiate R01 (5)	initiate R02 (6)	R01 weld / clear (7)	R02 weld / reposition (8)	open clamps 3 (9)	R02 weld / clear (10)	open clamps 1 (11)	unload part (13)
complete	load part (1)	1	0	0	0	0	0	0	0	0	0	0
	close clamps 1 (2)	1	1	1	1	1	0	0	0	0	0	0
	close clamps 3 (4)	1	1	1	1	1	1	0	0	0	0	0
	initiate R01 (5)	1	1	1	1	1	0	1	1	1	0	0
	initiate R02 (6)	1	1	1	1	1	1	0	0	0	0	0
	R01 weld / clear (7)	1	1	1	1	1	1	1	1	1	0	0
	R02 weld / reposition(8)	1	1	1	1	1	1	1	0	0	0	0
	open clamps 3 (9)	1	1	1	1	1	1	1	1	0	0	0
	R02 weld / clear (10)	1	1	1	1	1	1	1	1	1	0	0
	open clamps 1 (11)	1	1	1	1	1	1	1	1	1	1	0
	unload part (13)	1	1	1	1	1	1	1	1	1	1	1

Figure 4.4.6: Opposing Dependencies Matrix Transformed - High Trace Count

Motion (ID)		complete										
		load part (1)	close clamps 1 (2)	close clamps 3 (4)	initiate R01 (5)	initiate R02 (6)	R01 weld / clear (7)	R02 weld / reposition (8)	open clamps 3 (9)	R02 weld / clear (10)	open clamps 1 (11)	unload part (13)
start	load part (1)	0	0	0	0	0	0	0	0	0	0	0
	close clamps 1 (2)	1	0	0	0	0	0	0	0	0	0	0
	close clamps 3 (4)	1	0	0	0	0	0	0	0	0	0	0
	initiate R01 (5)	1	0	0	0	0	0	0	0	0	0	0
	initiate R02 (6)	1	0	0	0	0	0	0	0	0	0	0
	R01 weld / clear (7)	1	1	0	1	0	0	0	0	0	0	0
	R02 weld / reposition (8)	1	1	1	0	1	0	0	0	0	0	0
	open clamps 3 (9)	1	1	1	0	1	0	1	0	0	0	0
	R02 weld / clear (10)	1	1	1	0	1	0	1	1	0	0	0
	open clamps 1 (11)	1	1	1	1	1	1	1	1	1	0	0
	unload part (13)	1	1	1	1	1	1	1	1	1	1	0

Figure 4.4.7: Dependencies Matrix - High Trace Count

Motion (ID)		complete										
		load part (1)	close clamps 1 (2)	close clamps 3 (4)	initiate R01 (5)	initiate R02 (6)	R01 weld / clear (7)	R02 weld / reposition (8)	open clamps 3 (9)	R02 weld / clear (10)	open clamps 1 (11)	unload part (13)
start	load part (1)	0	0	0	0	0	0	0	0	0	0	0
	close clamps 1 (2)	1	0	0	0	0	0	0	0	0	0	0
	close clamps 3 (4)	1	0	0	0	0	0	0	0	0	0	0
	initiate R01 (5)	1	1	1	0	0	1	0	1	0	0	0
	initiate R02 (6)	1	1	1	1	0	0	1	0	0	0	0
	R01 weld / clear (7)	0	1	1	1	1	0	0	1	1	1	0
	R02 weld / reposition (8)	0	1	1	1	1	1	0	0	0	0	0
	open clamps 3 (9)	0	0	0	1	0	1	1	0	0	0	0
	R02 weld / clear (10)	0	0	0	1	0	1	0	1	0	0	0
	open clamps 1 (11)	0	0	0	0	0	1	0	0	1	0	0
	unload part (13)	0	0	0	0	0	0	0	0	0	1	0

Figure 4.4.8: α_{LC} Result: One Style - High Trace Count

Motion (ID)		complete										
		load part (1)	close clamps 1 (2)	close clamps 3 (4)	initiate R01 (5)	initiate R02 (6)	R01 weld / clear (7)	R02 weld / reposition (8)	open clamps 3 (9)	R02 weld / clear (10)	open clamps 1 (11)	unload part (13)
start	load part (1)	0	0	0	0	0	0	0	0	0	0	0
	close clamps 1 (2)	1	0	0	0	0	0	0	0	0	0	0
	close clamps 3 (4)	1	0	0	0	0	0	0	0	0	0	0
	initiate R01 (5)	1	0	0	0	0	0	0	0	0	0	0
	initiate R02 (6)	1	0	0	0	0	0	0	0	0	0	0
	R01 weld / clear (7)	0	1	0	1	0	0	0	0	0	0	0
	R02 weld / reposition (8)	0	1	1	0	1	0	0	0	0	0	0
	open clamps 3 (9)	0	0	0	0	0	0	1	0	0	0	0
	R02 weld / clear (10)	0	0	0	0	0	0	0	1	0	0	0
	open clamps 1 (11)	0	0	0	0	0	1	0	0	1	0	0
	unload part (13)	0	0	0	0	0	0	0	0	0	1	0

Figure 4.4.9: α_{LC} Result: One Style - Low Trace Count

Motion (ID)		complete										
		load part (1)	close clamps 1 (2)	close clamps 3 (4)	initiate R01 (5)	initiate R02 (6)	R01 weld / clear (7)	R02 weld / reposition (8)	open clamps 3 (9)	R02 weld / clear (10)	open clamps 1 (11)	unload part (13)
start	load part (1)	0	0	0	0	0	0	0	0	0	0	0
	close clamps 1 (2)	1	0	0	0	0	0	0	0	0	0	0
	close clamps 3 (4)	1	0	0	0	0	0	0	0	0	0	0
	initiate R01 (5)	1	0	0	0	0	0	0	0	0	0	0
	initiate R02 (6)	1	0	0	0	0	0	0	0	0	0	0
	R01 weld / clear (7)	0	1	1	1	0	0	0	0	0	0	0
	R02 weld / reposition (8)	0	1	1	0	1	0	0	0	0	0	0
	open clamps 3 (9)	0	0	0	0	0	0	1	0	0	0	0
	R02 weld / clear (10)	0	0	0	0	0	0	0	1	0	0	0
	open clamps 1 (11)	0	0	0	0	0	1	0	0	1	0	0
	unload part (13)	0	0	0	0	0	0	0	0	0	1	0

Figure 4.4.10: α_{LC} Result: All Styles - High Trace Count

Motion (ID)		complete											
		load part (1)	close clamps 1 (2)	close clamps 2 (3)	close clamps 3 (4)	initiate R01 (5)	initiate R02 (6)	R01 weld / clear (7)	R02 weld / reposition(8)	open clamps 3 (9)	R02 weld / clear (10)	open clamps 1 (11)	open clamps 2 (12)	unload part (13)
start	load part (1)	0	0	0	0	0	0	0	0	0	0	0	0	0
	close clamps 1 (2)	1	0	0	0	0	0	0	0	0	0	0	0	0
	close clamps 2 (3)	1	0	0	0	0	0	0	0	0	0	0	0	0
	close clamps 3 (4)	1	0	0	0	0	0	0	0	0	0	0	0	0
	initiate R01 (5)	1	0	0	0	0	0	0	0	0	0	0	0	0
	initiate R02 (6)	1	0	0	0	0	0	0	0	0	0	0	0	0
	R01 weld / clear (7)	0	1	1	0	1	0	0	0	0	0	0	0	0
	R02 weld / reposition (8)	0	1	1	1	0	1	0	0	0	0	0	0	0
	open clamps 3 (9)	0	0	0	0	0	0	0	1	0	0	0	0	0
	R02 weld / clear (10)	0	0	0	0	0	0	0	0	1	0	0	0	0
	open clamps 1 (11)	0	0	0	0	0	0	0	0	0	1	0	0	0
	open clamps 2 (12)	0	0	0	0	0	0	1	0	0	1	0	0	0
	unload part (13)	0	0	0	0	0	0	0	0	0	0	1	1	0

Listing 4.2: Parse sLog Into DSM

```
for index, row in sLog.iterrows():
 if sLog.loc[index]['type'] == 'complete':
  column = sLog.loc[index]['motionID']
  case = sLog.loc[index]['caseID']
  pointer = index
  found = 0
  while ((sLog.loc[pointer]['type'] == 'start') or
    (found == 0)) and pointer < (len(sLog)-1) and
    sLog.loc[pointer]['caseID'] == case:
   pointer += 1
   if sLog.loc[pointer]['type'] == 'start':
    found = 1
    row = sLog.loc[pointer]['motionID']
    DSM.loc[row, column] = True
```

of parameter tuning. Therefore, all the algorithms were tested using their default parameters. Typically, automated production equipment, when considering one-part style only, has one fixed sequence. This sequence hereafter, will be referred to as 'complete sequence'. In table 4.4.1, the column 'wrong node' refers to a node that is present in the discovered model but is not part of the complete model. A missing node on the other side is a node that is present in the complete model but is not present in the discovered model. The same holds for edges. Missing edges are edges that have not been found, where wrong edges are edges that are shown in the discovered model but do not exist in the complete model.

Table 4.4.1 shows that the β-algorithm and Heuristic++ miner yield the same results as the α_{LC}-algorithm. Although more complicated, this was expected for the β-algorithm because of its similarity to the α_{LC}-algorithm. The Heuristic++ miner seems to have discovered the correct model by chance only. It is based on thresholds and does not discover edges for which the number of occurrences was not above that threshold. This shortcoming becomes especially evident when limiting the number of traces to eight selected versions that include all information needed for a proper discovery. The β-algorithm and the α_{LC}-algorithm are still able to discover a highly accurate model under such circumstances while the Heuristic++ miner no longer can (see figure 4.4.2). Limiting the traces further to only three traces, which still contain the required information, also prevents the β-algorithm from discovering the complete model, while the α_{LC}-algorithm still finds the proper dependencies. The IMLC, although it considers the activities life cycle information, did not perform as expected. The problem arises because the algorithm is trying to discover interleaved dependencies according to Leemans (S. J. J Leemans, Fahland, and W. M. P Van Der Aalst 2016), which results in a diagram with lots of edges. For the Fuzzy miner, it needs to be mentioned,

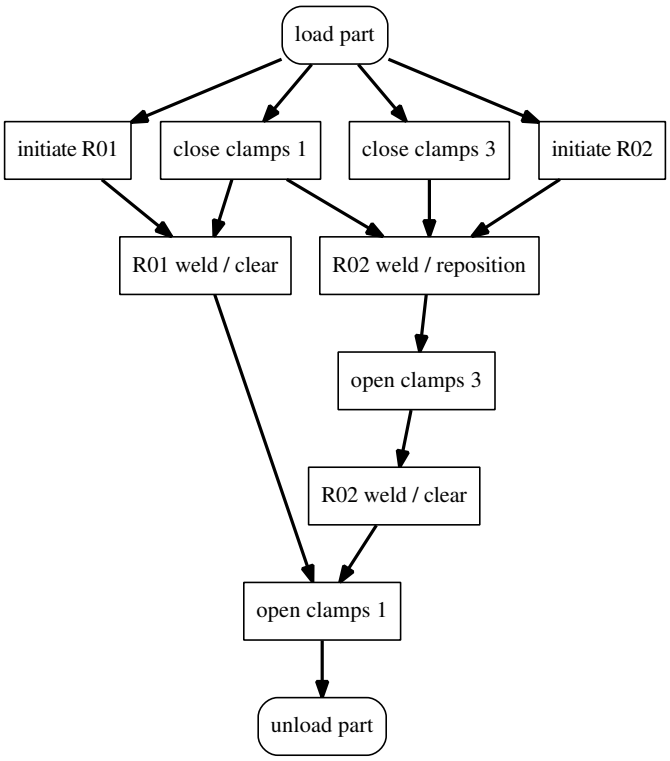

Figure 4.4.11: The One Style Model - High Trace Count

that the discovered model showed two nodes for each activity because the 'start' and 'complete' timestamps were treated as two separate events. The occasions were counted and shown in table 4.4.1 in grey but not added to the total.

Table 4.4.1: Discovered Model Evaluation (One Style)

	high number of traces					low number of traces				
	nodes		edges		sum	nodes		edges		sum
	wrong	missing	wrong	missing		wrong	missing	wrong	missing	
α Miner	0	0	0	15	15	0	0	0	15	15
α_{LC} Miner	0	0	0	0	0	0	0	1	0	1
β Miner	0	0	0	0	0	0	0	1	0	1
CSM Miner	0	0	33	4	37	0	0	26	5	31
DISCO	0	0	5	6	11	0	0	5	6	11
Fuzzy Miner	11	0	4	6	10	11	0	7	5	12
Heuristic Miner	0	0	3	3	6	0	0	3	5	8
Heuristic++ Miner	0	0	0	0	0	0	0	1	0	1
Inductive Miner	0	0	8	4	12	0	0	3	8	11
Inductive (LC) Miner	0	0	7	7	14	0	0	30	8	38

Table 4.4.2: Discovered Model Evaluation (Limited Traces)

	Eight Traces					Three Traces				
	wrong	missing	wrong	missing	sum	wrong	missing	wrong	missing	sum
α_{LC} Miner	0	0	0	0	0	0	0	0	0	0
β Miner	0	0	0	0	0	0	0	7	0	7
Heuristic++ Miner	0	0	0	1	1	0	0	1	1	2

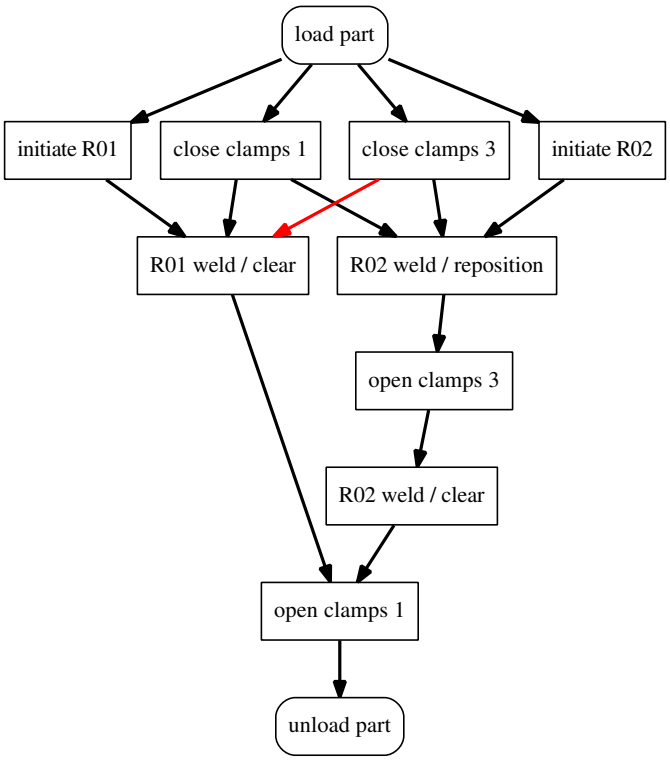

Figure 4.4.12: The One Style Model - Low Trace Count

4.5 Interactive Trace Induction

4.5.1 Viability Study

As a first step, it had to be proven that it is possible to delay the sensor inputs of the different motions, as suggested in section 3.5.3. This delay was achieved by implementing the change shown in figure 3.5.9 within a select number of devices in a small, standalone robotic cell. At first, the newly created tags were toggled one by one manually, and the resulting equipment behavior was recorded. Reviewing the log proved that the concept worked as expected. Afterward, the same test was repeated, controlling the tags through an OPC connection. Using OPC required that the tags are enabled for remote writing, which is the standard setting for the RSLogix controller available for the test. The resulting log again, matched expectations.

4.5.2 Artificial Log

For the remaining experiments the artificial log introduced in section 4.4 is reused because its limitation to just a few important events should aid the understanding by the reader. A random case for that artificial log is shown in figure 4.5.1. (Note the random case is for a specific style only which does not require clamps 2. Consequently

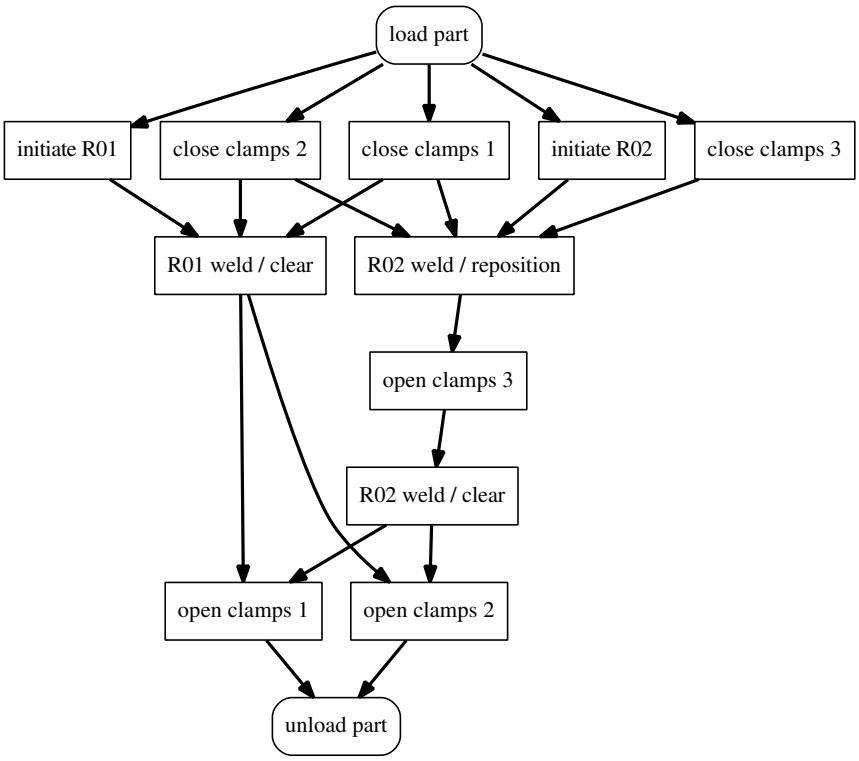

Figure 4.4.13: The All Styles Model - High Trace Count

the motions 3 and 12 are not shown in below figures.)

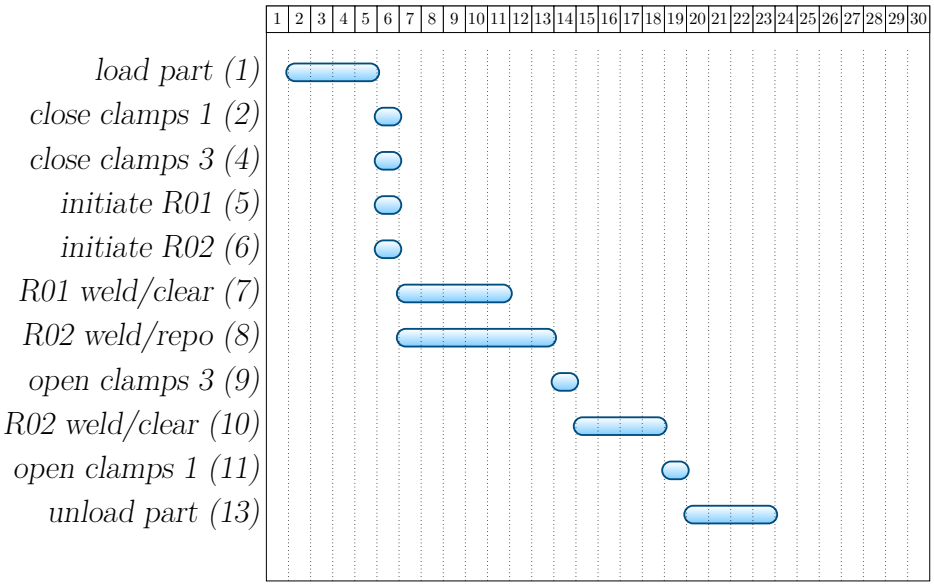

Figure 4.5.1: Random Case

Figure 4.5.2: Initial Depend. Matrix

	Motion (ID)	complete										
		load part (1)	close clamps 1 (2)	close clamps 3 (4)	initiate R01 (5)	initiate R02 (6)	R01 weld / clear (7)	R02 weld / reposition (8)	open clamps 3 (9)	R02 weld / clear (10)	open clamps 1 (11)	unload part (13)
start	load part (1)											
	close clamps 1 (2)	X										
	close clamps 3 (4)	X										
	initiate R01 (5)	X										
	initiate R02 (6)	X										
	R01 weld / clear (7)		X	X	X	X						
	R02 weld / reposition (8)		X	X	X	X						
	open clamps 3 (9)							X				
	R02 weld / clear (10)								X			
	open clamps 1 (11)									X		
	unload part (13)										X	

Figure 4.5.3: Depend. Matrix 1st Iteration

	Motion (ID)	complete										
		load part (1)	close clamps 1 (2)	close clamps 3 (4)	initiate R01 (5)	initiate R02 (6)	R01 weld / clear (7)	R02 weld / reposition (8)	open clamps 3 (9)	R02 weld / clear (10)	open clamps 1 (11)	unload part (13)
start	load part (1)											
	close clamps 1 (2)	1										
	close clamps 3 (4)	1	0									
	initiate R01 (5)	1	0	0								
	initiate R02 (6)	1	0	0	0							
	R01 weld / clear (7)	0	X	X	X	X						
	R02 weld / reposition (8)	0	X	X	X	X						
	open clamps 3 (9)	0						1				
	R02 weld / clear (10)	0						0	1			
	open clamps 1 (11)	0						0	0	1		
	unload part (13)	0						0	0	0	1	

4.5.3 Interactive Trace Induction

Interactive trace induction is proposed to solve the dilemma of unknown dependencies. As previously described, this is achieved by artificially delaying the 'complete' events, of questionable activities, one at a time. This forces all of the dependent, downstream activities to be isolated, and thus, reveal their actual dependencies. Note: The below example depicts, for better understanding, only one dependencies matrix for each of the traces to be evaluated. All the theorems mentioned refer to the previous chapter 3.5.2. To discovery the parallel activities according to theorem 8 (marked in blue) with a software algorithm it would be necessary to work with a second dependencies matrix in which the opposing dependencies are recorded as shown for the α_{LC} algorithm in chapter 4.4.

Gantt chart figure 4.5.1 represents a random case example. It is provided without any links because the dependencies between the activities are not yet known. The data is parsed into a decision matrix, marking potential dependencies with 'x', as shown in figure 4.5.2. Applying theorem 6 allows the values of the cells highlighted in green in figure 4.5.3 to be changed to '1' because there is only one preceding activity. According to theorem 7 all the values of the red shaded cells can be set to '0' since they would be part of chained dependencies. Finally, based on theorem 8 the cells marked in blue can be set to '0' as well because parallel activities can not depend on each other.

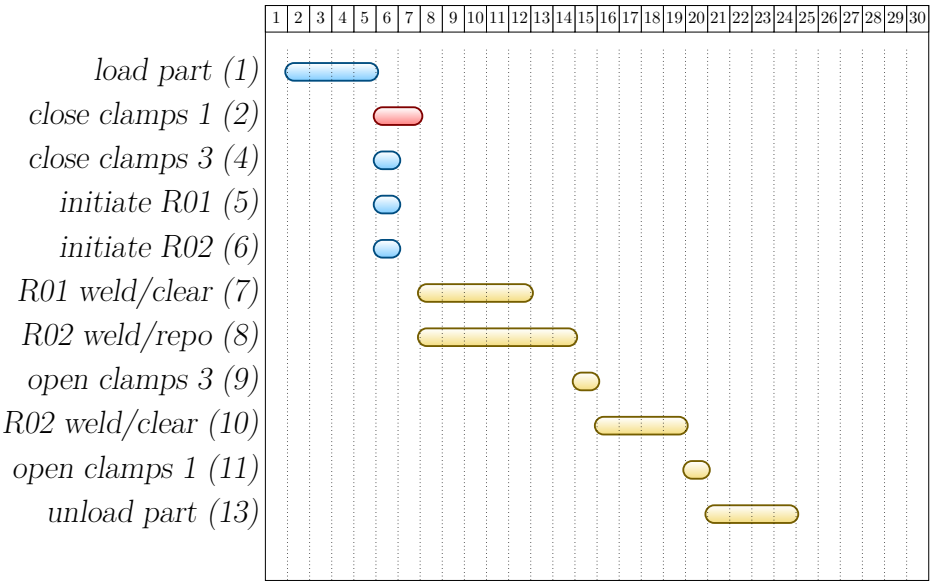

Figure 4.5.4: Activity 2 Delayed

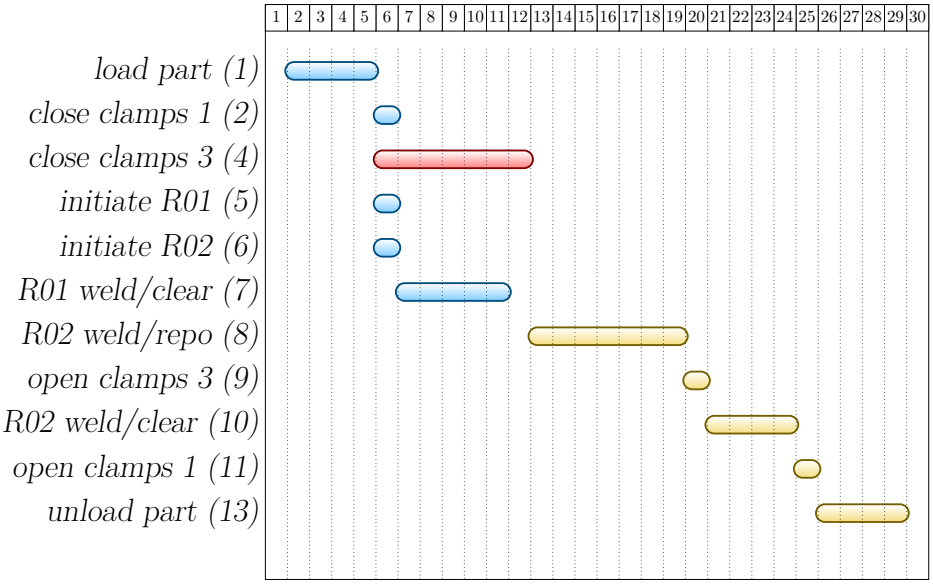

Figure 4.5.5: Activity 4 Delayed

The unknown dependencies are addressed one by one. The first question is if 'R01 weld/clear (7)' depends on 'close clamps 1 (2)'. To determine that activity 2 needs to be artificially delayed and the corresponding machine cycle recorded. For the example this would result in the trace shown in figure 4.5.4. This Gantt chart clearly shows that not only activity 7 but also activity 8 directly follow activity 2 ($2 > 7, 8$) as postulated in theorem 6. These two dependencies are therefore marked with '1' (highlighted in

Figure 4.5.6: Depend. Matrix 2nd Iteration

	Motion (ID)	complete										
		load part (1)	close clamps 1 (2)	close clamps 3 (4)	initiate R01 (5)	initiate R02 (6)	R01 weld / clear (7)	R02 weld / reposition (8)	open clamps 3 (9)	R02 weld / clear (10)	open clamps 1 (11)	unload part (13)
start	load part (1)											
	close clamps 1 (2)	1										
	close clamps 3 (4)	1	0									
	initiate R01 (5)	1	0	0								
	initiate R02 (6)	1	0	0	0							
	R01 weld / clear (7)	0	1	X	X	X						
	R02 weld / reposition (8)	0	1	X	X	X	0					
	open clamps 3 (9)	0	0					1				
	R02 weld / clear (10)	0	0					0	1			
	open clamps 1 (11)	0	0					0	0	1		
	unload part (13)	0	0					0	0	0	1	

Figure 4.5.7: Depend. Matrix 3rd Iteration

	Motion (ID)	complete										
		load part (1)	close clamps 1 (2)	close clamps 3 (4)	initiate R01 (5)	initiate R02 (6)	R01 weld / clear (7)	R02 weld / reposition (8)	open clamps 3 (9)	R02 weld / clear (10)	open clamps 1 (11)	unload part (13)
start	load part (1)											
	close clamps 1 (2)	1										
	close clamps 3 (4)	1	0									
	initiate R01 (5)	1	0	0								
	initiate R02 (6)	1	0	0	0							
	R01 weld / clear (7)	0	1	0	X	X						
	R02 weld / reposition (8)	0	1	1	X	X	0					
	open clamps 3 (9)	0	0	0				1				
	R02 weld / clear (10)	0	0	0				0	1			
	open clamps 1 (11)	0	0	0				0	0	1		
	unload part (13)	0	0	0				0	0	0	1	

green) within the decision matrix figure 4.5.6. Now since a parallelism between the events 7 and 8 has been established theorem 8 allows for the blue marked cell to be set to 0. Based on theorem 7 it can also be concluded that the activities $9, 10, 11, 13\#2$ (do not depend on 2). Therefore the value '0' can be assigned (marked in red).

Looking at the decision matrix 4.5.6 the next question is if activity 7 depends on activity 4. Delaying activity 4 will produce the Gantt chart plotted in figure 4.5.5. It shows activity 4 and 7 in parallel which, according to theorem 8 means that there is no dependency. Consequently the value of the corresponding cell can be set to '0' in figure 4.5.7 (shown in blue). Activity 8 however directly follows activity 4 and therefore that value can be set to '1'. Based on theorem 7 the activities $9, 10, 11, 13\#3$ and their cells are set to '0' as highlighted in red.

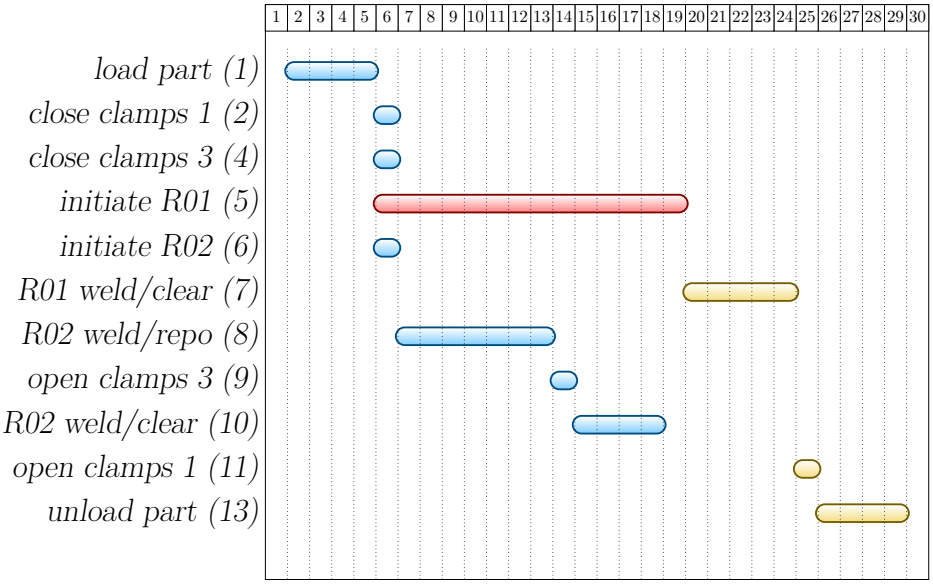

Figure 4.5.8: Activity 5 Delayed

Figure 4.5.10: Depend. Matrix 4th Iteration

	Motion (ID)	complete										
		load part (1)	close clamps 1 (2)	close clamps 3 (4)	initiate R01 (5)	initiate R02 (6)	R01 weld / clear (7)	R02 weld / reposition (8)	open clamps 3 (9)	R02 weld / clear (10)	open clamps 1 (11)	unload part (13)
start	load part (1)											
	close clamps 1 (2)	1										
	close clamps 3 (4)	1	0									
	initiate R01 (5)	1	0	0								
	initiate R02 (6)	1	0	0	0							
	R01 weld / clear (7)	0	1	0	1	X						
	R02 weld / reposition (8)	0	1	1	0	X	0					
	open clamps 3 (9)	0	0	0	0		0	1				
	R02 weld / clear (10)	0	0	0	0		0	0	1			
	open clamps 1 (11)	0	0	0	0		1	0	0	1		
	unload part (13)	0	0	0	0		0	0	0	0	1	

Figure 4.5.11: Depend. Matrix 5th Iteration

	Motion (ID)	complete										
		load part (1)	close clamps 1 (2)	close clamps 3 (4)	initiate R01 (5)	initiate R02 (6)	R01 weld / clear (7)	R02 weld / reposition (8)	open clamps 3 (9)	R02 weld / clear (10)	open clamps 1 (11)	unload part (13)
start	load part (1)											
	close clamps 1 (2)	1										
	close clamps 3 (4)	1	0									
	initiate R01 (5)	1	0	0								
	initiate R02 (6)	1	0	0	0							
	R01 weld / clear (7)	0	1	0	1	0						
	R02 weld / reposition (8)	0	1	1	0	1	0					
	open clamps 3 (9)	0	0	0	0	0	0	1				
	R02 weld / clear (10)	0	0	0	0	0	0	0	1			
	open clamps 1 (11)	0	0	0	0	0	1	0	0	1		
	unload part (13)	0	0	0	0	0	0	0	0	0	1	

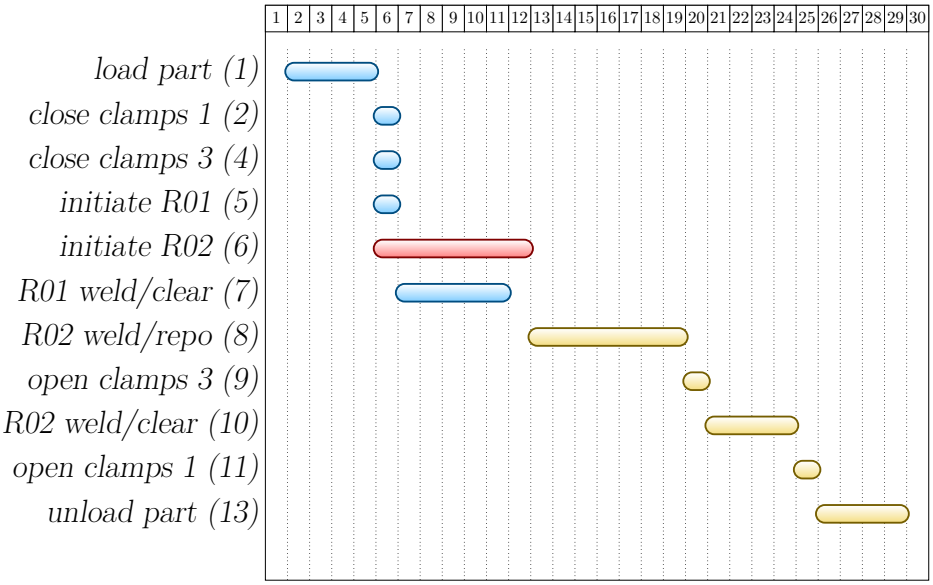

Figure 4.5.9: Activity 6 Delayed

Continuing on, the relation between activity 7 and activity 5 is questioned. Delaying activity 5 will result in the Gantt chart 4.5.8. Based on theorem 6 it can be concluded that a dependency exists between activity 7 and activity 5 as well as between activity 11 and activity 7 (both marked in green in figure 4.5.10) while the theorems 7 and 8 exclude the dependencies shown in red and blue.

Now the dependency between activity 7 and activity 6 needs to be evaluated by delaying activity 6 as shown in Gantt chart 4.5.9. As marked in the decision matrix 4.5.11, theorem 6 determines that there is a relation between the activities 8 and 6 (marked in green) while the remaining theorems allow for the conclusion that there aren't any dependencies for the cells highlighted in red and blue.

At this point all dependencies and non-dependencies are clear and no further traces

Figure 4.5.12: α_{LC} Result: Interactive Trace Induction Validation

	Motion (ID)	complete										
		load part (1)	close clamps 1 (2)	close clamps 3 (4)	initiate R01 (5)	initiate R02 (6)	R01 weld / clear (7)	R02 weld / reposition (8)	open clamps 3 (9)	R02 weld / clear (10)	open clamps 1 (11)	unload part (13)
start	load part (1)	0	0	0	0	0	0	0	0	0	0	0
	close clamps 1 (2)	1	0	0	0	0	0	0	0	0	0	0
	close clamps 3 (4)	1	0	0	0	0	0	0	0	0	0	0
	initiate R01 (5)	1	0	0	0	0	0	0	0	0	0	0
	initiate R02 (6)	1	0	0	0	0	0	0	0	0	0	0
	R01 weld / clear (7)	0	1	0	1	0	0	0	0	0	0	0
	R02 weld / reposition (8)	0	1	1	0	1	0	0	0	0	0	0
	open clamps 3 (9)	0	0	0	0	0	0	1	0	0	0	0
	R02 weld / clear (10)	0	0	0	0	0	0	0	1	0	0	0
	open clamps 1 (11)	0	0	0	0	0	1	0	0	1	0	0
	unload part (13)	0	0	0	0	0	0	0	0	0	1	0

are required. As a nice side effect it needs to be mentioned that the final dependencies matrix 4.5.11 also equals the result matrix of the α_{LC} algorithm applied to the above recorded traces. This in turn means that the desired process model, in form of a flow chart, can be generated based on matrix 4.5.11 without any further mining effort.

To prove this the α_{LC}-algorithm is applied to the four induced cases, generating the dependencies matrix shown in figure 4.5.12.

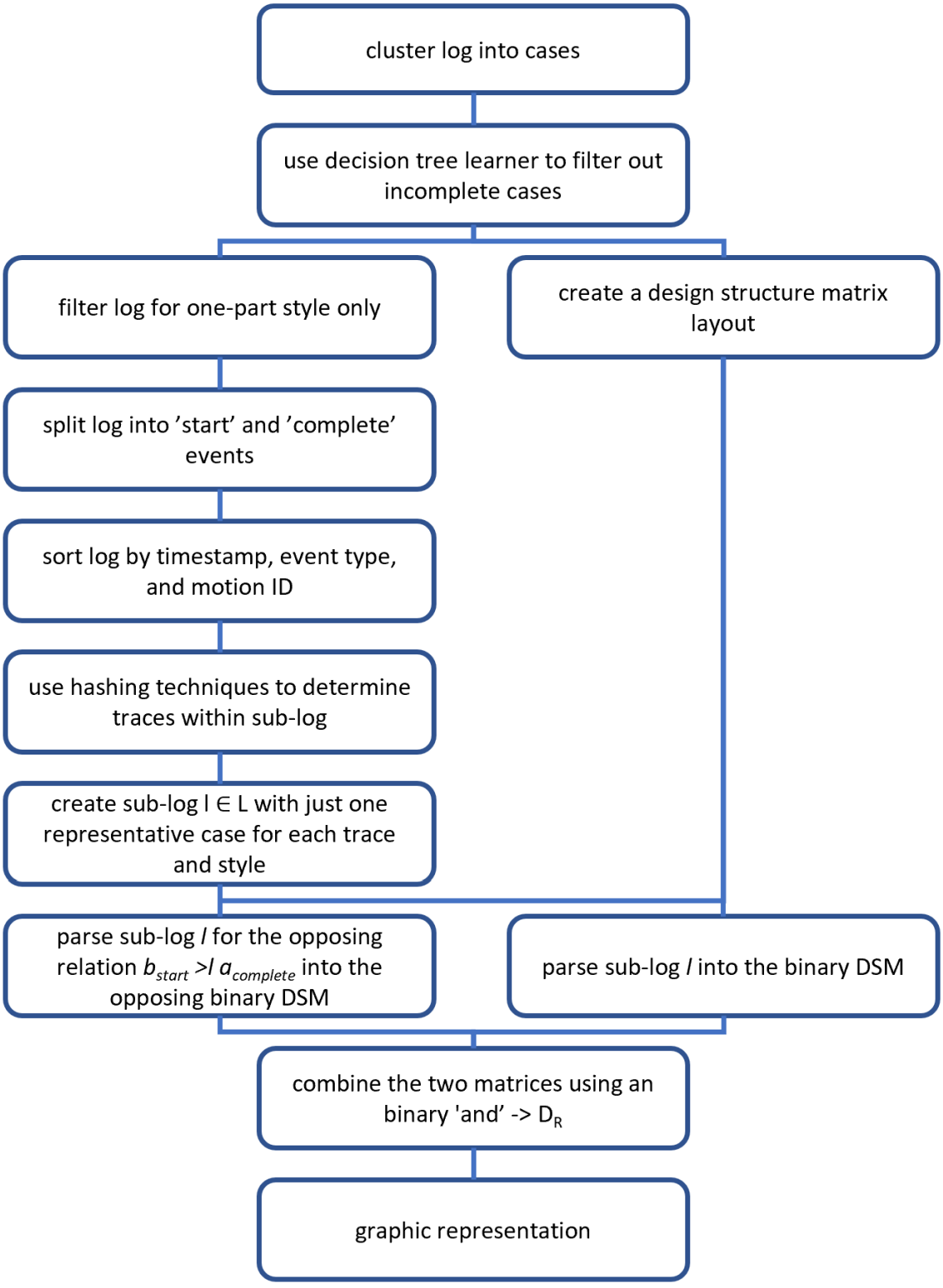

Figure 4.4.14: The α_{LC}-Process

5 Discussion

The purpose of this chapter is to discuss the results from the case studies described in chapter 4. To better relate the results and their evaluation, the same structure of the sub-chapters previously used was applied. In section 5.1, the findings in conjunction with the data collection experiments are reviewed. The preprocessing trials are then analysed in section 5.2 before the knowledge-discovery-based findings are summed up in section 5.3. Section 5.4 takes a look at possible limitations of the new α_{LC} algorithm. The chapter concludes in section 5.5 with the discussion of the advantages of 'interactive trace induction' over discovered Process Models that require quality metrics.

5.1 Data Collection

The deployment of this methodology to the 27, unknown PLCs took approximately 30 minutes (compared to 150hrs for 1 PLC as described previously). From these 30 minutes, 26 minutes were spent converting the PLC programs to text-based files, which had to be done with keyboard macros, since the Rockwell software does not provide a command-line instruction to do so. The remaining 4 minutes were needed to parse the text files and set up the OPC groups and items for monitoring.

It also needs to be mentioned that hardware changes within the monitored stations will trigger changes within the PLC logic. These changes also need to be reflected in the

data collection system. The maintenance work station holds typically the up to date PLC programs. During the experiments, logic was added to the data acquisition software that would periodically check if the date of the associated .ACD files has changed. Upon a detected change, a new L5K file would be created and the system automatically updated. The functionality was finally deactivated because it was triggered for every change made within the PLC software. Hardware changes are not a common occurrence which makes the manual triggering of PLC code re-parsing a viable option.

The event notification of the OPC server is based on numeric handles. These handles have to be provided while setting up the OPC items. Once an event notification arrives, it only includes the handle and the status of the event. The logging logic needs to correctly map the handle to the PLC tag to add meaning to the event. It seems it would have been more effective if the OPC server would provide the name of the tag together with its status rather then just the handle because there would not be the need to keep a mapping table in memory. During the creation of an OPC item, the server returns a so-called server handle. This handle, which is different to the client handle, has to be used for write access prompting the need to capture the relationship between server handle and the tag also.

A comparison of the times obtained through OPC with times recorded for the same motions within the PLC, unfortunately, showed that there is a slight difference. The OPC standard defines that all data exchanged has to be annotated with a timestamp, but it does not specify from where the timestamp comes. Some PLCs provide a timestamp where in other cases the timestamp is provided by the OPC server. Unfortunately, the Controllogix PLCs monitored for this research do not provide a timestamp with the data. This shortcoming required the creation of a timestamp within the logging logic which does not consider any time delays occurring between the time the event takes place and the time the event is reported to the logging logic.

One such delay is the refresh rate. The OPC standard allows for a refresh rate setting of 0, which equals a near-instant event notification. Contrary, the minimum setting that could be achieved with the proposed system was 50ms. This value means that the times recorded are between 0 and 50ms seconds delayed. On top of that comes a measured system latency of 100ms. These inaccuracies could be avoided if the PLC provided the event timestamp.

One of the goals of the data collection was to collect the life cycle information of the different motions. The life cycle refers to the time it takes a motion from the point it is triggered until it reaches the end position. The events recorded are just individual events. This means that the log will contain an entry for the motion being triggered and

later on a second event for the motion reaching its end position. A view was written within the SQL server that merges the events based on the tag name and the distance between them. While applying this view, it was found that there were motions that were triggered but never reached the end position. This fact led to the conclusion that the recorded log is flawed and therefore requires preprocessing to handle those flaws accordingly.

5.2 Preprocessing

It was found that the classifier models made some mistakes that could have been avoided based on the engineered features. Like stated in 3.2.22, the classification needs to be '1' if the sample cycle is a bypass cycle. On the other hand, the classification needs to be '0' if not all load/unload events are present or if there is not a set of robot initiate/process events. Post-processing of the classification results thus would be recommended.

The impact of the flaws found within the activity log depends on the use case for the data collected. For the discovery of the workflow of a whole production line, such a fine-grained log is not required. Therefore many of the faults experienced might not have an impact at all. On a station level, the impact is much bigger. Process Discovery works based on the traces found within the log. Since flaws essentially are responsible for new traces, it can be concluded that the discovery result will be distorted. Many of the established algorithms use statistical approaches to eliminate such impact, which could also lead to important traces being discredited, thus leading to wrong discoveries. Therefore the author believes that the best starting point for a process model discovery is a complete log free of flaws. Incomplete and imprecise cases are best removed from the log before model discovery because there is no need for utilising consecutive cases.

Another use case for the data would be predictive maintenance. The data obtained during this research showed that, due to the system used, a variance of 50 milliseconds was recorded for the activity life cycle. Putting that value in relation with a clamps cycle time of 400 milliseconds this equals a 12.5% error which might not be acceptable. The use of the data for predictive purposes also requires the recording over a longer period. Only then it can be determined if it still is possible to extract any trending information from the data which allow for conclusions on the wear of the equipment. Other fault predictions additionally require the logging of the actual fault case because otherwise no patterns leading up to it can be detected. This correlation currently is not given within the proposed framework although the data has been collected.

Improvements can only be achieved through changes made to the data collection effort. One of the issues using the RSLinx OPC server was that the minimum refresh

rate could not be set to less than 50 milliseconds which already introduces uncertainty. The second contributing factor was the communication over multiple switches and networks. This issue could be overcome through local data collection within the cells local network. The collected data then could be periodically uploaded to a central database. The final option would be collecting the data right in the PLC. Such data collection would require that the data collection intent becomes part of the equipment's control standard. Depending on the size of the expected local data storage, this might come at an additional cost due to the PLC controllers comparably small internal storage.

The most challenging part of this section, if not the whole thesis, was the clustering of the activities into cases based on their relation. Initially, it was reasoned that the parts, as they are being manufactured, are associated with a unique part identifier. Recording that identifier, along with the events, would allow for clustering. The assumption that every part is associated with a unique ID proved wrong. Therefore an attempt was made to create a unique number for each part on the fly. The problem with both approaches is that they do not consider that this data is only available once the part is present within the station because that is when the jobs data is transferred. Based on the previously shown case this time frame would equal the process phase only, leaving the setup, load, unload and reset phase unconsidered.

Machine learning offers several clustering algorithms. Before developing the methodology shown within this work, several months were spent trying to use these algorithms for the required case clustering. These attempts failed to yield the expected results. One lesson learned was the choice of software tools to use. The first options were R and Python, which both were unknown to the author. It soon became clear that Python is the more commonly used language within the research community, although R might be more powerful. Besides these text-based programming languages, there are also several machine learning tools with graphic user interfaces. Tools tried for this research were Matlab and Rapid Miner, which require a commercial but for students affordable license. On the other end of the spectrum is the open-source KNIME project which, based on its functionality, could be compared to Rapid Miner. All three of these options allow easy access to the required machine learning algorithms, but the projects quickly become complicated due to the use of predefined processing blocks. Besides, some of the projects created took hours or even days to complete processing. These issues finally prompted the use of Python in conjunction with the Jupyter notebook. It is understood that Python is an interpreted language and that languages like Java and C# offer a far more superior processing power. However, Python comes with all the tools needed to create any desired experiment quickly.

5.3 Knowledge-Based Discovery

The examples for automated knowledge discovery shown within this thesis are based on a log containing start and complete timestamps for all the units and processes within a production station. Besides, the log contains the part present status. It is perceivable that, with additional information available, more suggestions can be generated automatically. Initiating a robot typically takes one to two seconds. Often this event is only triggered once the station is ready for the robot to execute its process. The initiation could also happen as soon as the parts style information is available, which also could be marked up within the Gantt chart.

Another cause for delays often is the robots waiting for each other. Robots avoiding each other is accomplished with the help of zones for shared areas within the PLC logic. Recording these zone requests would show when a robot is actually delayed, and improvement suggestions could be given accordingly.

The original design documentation could also be considered as a source of information. Besides the sequence of operations, it also contains the physical specification of the cylinders used as well as indicators which point at potential interferences between the different units. The cylinders bore and stroke information would allow for the comparison of identical units since the system air pressure is equal for all units. The interference information would allow suggestions regarding the execution of units in parallel. Such a proposal currently would have to be subject to an experts review because the process itself might demand specific sequencing to ensure the desired quality. It is imaginable that such critical dependencies are also marked within the design documentation. The same holds for the dependency of unit operations on a part being present or not present within the station. If this would be marked within the design documentation, some unit's activation could be pulled ahead.

Previously it was mentioned that, besides the most occurring cycle, the slowest and the fastest trace are also of interest because they can give the expert reviewer hints to what is possible and what sometimes goes wrong. Robot process duration could also be considered. If there is an extreme imbalance rethinking the distribution of process tasks between the different robots might lead to improved cycle times. Such decision requires an expert reviewer because based on the available data, it remains unknown for the mining algorithm, which process can potentially be done by which robot.

5.4 Process Model Discovery

The main limitation of the α_{LC}-algorithm is, as with all Process Discovery algorithms, the number of traces available within the log. An increased number of traces often result in a better process model because of the increased chance that critical traces have been observed. Another potential approach to improve the discovered process model would be the inclusion of domain knowledge. For the equipment used in an automotive body shop, it might be advantageous to group all events with the same trigger together. For example, all cylinders, which are actuated by one valve become one artificial event with the shared 'start' timestamp and the 'complete' timestamp of the slowest unit within the group. This simplification reduces the number of dependencies to the next activity and, therefore, increases the likelihood of its discovery. It can also be argued that a robot initiation can only be followed by a robot process. A non-robotic dependency could be automatically disqualified. For non-metal-forming processes, it is also a given that there is no causality between opposing motions. In case of a clamp, this would mean that the closing of the clamp cannot directly depend on it being opened in the preceding event.

Many of the traces found in industrial equipment logs, used for this research, were caused by the robotic processes. For example, robots were initiated but did not immediately respond because they were still busy on a tip maintenance task, which is not part of the regular equipment sequence and, therefore, is not logged. Some robots also interact with multiple stations, which causes the robot not to be ready for the station in question when needed. Since these activities are assigned to another station, they cannot be found in the log of the station for which a process model needs to be discovered. Enriching the record with this information might lead to better model discovery.

Depending on the size of the log, processing time can become an issue during process model discovery. One of the solutions presented in this work was the use of the traces only instead of all the cases. This already dramatically reduces the processing time. Another issue during the trials might have been the use of Python which is an interpreted language. Choosing a compiled language instead might improve the performance. The most time consuming task is parsing the log into the matrices. Comparing the performance of the Pythons algorithms with the Process Discovery algorithm within ProM or DISCO leads to the conclusion that there must be a more efficient methodology to do so which is unknown to the author.

It could be argued that, given the standardised programming requirements above, it would be feasible to discover the equipment's process model just by parsing the controlling PLC program. Such an approach certainly is possible, although the effort

required remains unclear. Changing the way the model is discovered would render the α_{LC} algorithm obsolete. The remaining tasks still would be required to derive value since the model discovery through code parsing would not yield any performance data.

5.5 Interactive Trace Induction

It is often presumed that the quality of the discovered model is likely to increase with the number of cases recorded for that process. Based on the findings of this thesis, it can be concluded that the quality of the discovered model depends on the necessary traces being recorded, and a log spanning a more extended period increases the chance of doing so. Unfortunately, the reasons for traces within business processes seem to differ from industrial automation processes. Therefore, the concept of 'interactive trace induction' might only be partially applicable for the discovery of concurrent processes within the business domain.

When applying 'interactive trace induction' to real-life processes, one should be aware that parallel processes are not the only reason leading to the manifestation of multiple traces. Another, often more dominant cause, is that the logs are flawed due to wrongly recorded or missing events. Therefore, while executing 'interactive trace induction', it is still necessary to validate the completeness of the log obtained. Such verification could be as simple as comparing the number of events recorded for each of the cases. Previous research, with real-life logs in chapter 4, showed that the majority of cases are indeed flawless. Therefore, it can be concluded that the number of events found in the majority of cases is correct. Alternatively, the 'interactive trace induction' could be executed multiple times to ensure that the log is accurate. Deviating cases could be discarded.

This thesis purposely does not gauge the discovery results with the help of any of the established quality metrics. The reason is that the α_{LC}-algorithm is not based on any statistical methods and also does not strive for generalisation. Therefore, the discovered model fits the log data available. Some practitioners might term that as over-fitting, but the author argues that within the industrial automation realm, this is preferred over generalisation. This view is supported by Schimm (Shimm 2004) who states: 'In many cases, the benefit of workflow mining depends on the exactness of the mined models'. Also, confidence metrics can be misleading. Looking at example figs. 4.5.4, 4.5.5, 4.5.8 and 4.5.9, it can be seen that a highly accurate model also can be discovered with just three of the four traces, because the first trace (figure 4.5.4) does not lead to any information gain. At a first glance it could be assumed that four traces need to be discovered for the artificial example process to obtain all the required

information. If the record does not contain the non value-added trace described above, confidence would be lowered to 75%, although the discovered model will still be highly accurate.

'Interactive trace induction' has at least two easily overlooked side effects. When delaying the completion of an activity, one inadvertently triggers the watchdog mechanism that should be present in any PLC logic. This behavior can be seen as a positive benefit because it enables a simple, automated procedure for testing the PLCs fault logic. At the same time, 'interactive trace induction' might force the equipment into a status that will never occur during regular operation. Therefore, its implications might not have been considered by the programmer. In a worst-case scenario, this could lead to a collision within the machine. It is recommended that the responsible control personnel is engaged before executing 'interactive trace induction' to minimise that risk.

5.6 Related Topics

When selecting an OPC Server one has to choose between several products available on the market. One of the best known is Kepware. This software offers drivers for dozens of different OPC devices. Kepwares advantage is that the setup remains the same no matter which device it interfaces. It also offers features that allow for logging of the obtained data directly into a database. The downfall is that it requires licenses to be purchased.

Rockwell offers for their products the software package RSLinx. Unfortunately, it is not as easy to set up as Kepware, and it also does not provide any automated logging features. Since the OPC concept is based on translating a proprietary communication protocol into the OPC-standard, it can be assumed that the PLC manufacturer knows his product best. For this reason and because it was readily available RSLinx was chosen as the OPC Server.

Besides the OPC Server also the choice of database has an impact on the performance of the proposed framework. Today there are two main categories of competing database system - relational and NoSQL databases. In the traditional, relational database, the data is well structured in tables that are connected to each other. Contrary NoSQL databases do not rely on such a structure what makes them flexible and easily scalable. It is commonly acknowledged that NoSQL databases are best suited for significant quantities of data.

Within the facility, where this project was implemented, the prevailing database is Microsoft SQL Server. At first, the free Express version was used. Trials showed that

for monitoring a single PLC, this version was sufficient. The database was not able to store the data coming from multiple PLCs fast enough and ultimately failed. As a remedy, the developer version, which offers much better performance, was used for the remainder of the project.

The acquisition algorithm has to deal with a massive amount of data when monitoring multiple PLCs and storing the events raised into the database. This issue can be solved using multi threading. For the data collection, one thread is used for every group of events raised by the OPC server while another thread is responsible for the storage of those data into the database.

As previously described the data point selection is based on the text-based L5K file. The generation of that file is a feature of the Controllogix programming software. Unfortunately, the API does not allow for any parameters to convert a native '.ACD' file into an L5K file. Instead, a keyboard buffer script had to be written that opens a '.ACD' file within the Controllogix software and safes it as .L5K. It was found that this procedure varies with version changes of the programming software, which makes this approach less reliable. The L5K files may be generated manually as long as they are stored in the same folder as the '.ACD' file.

Since a distributed acquisition system would be preferable to increase the log quality while reducing the network load, efficient ways to update the distributed software packages had to be found. The proposed framework includes a table in the database which stores the latest acquisition software version and the names of the computers on which they are installed. Once the software is started, it first checks if the version is up to date and if the computer is authorised to run it. Is the software outdated the new version is downloaded from an FTP server, and a new windows task is initiated which kills the old software, installs the new software version and restarts it again.

The user interface of the data acquisition algorithm can be seen in figure 5.6.1.

The collection software always starts in the background and is represented by an icon within the windows toolbar. A right-click on that icon reveals the setup window which allows for the systems set up in three steps. At first, the user needs to choose the SQL server used for data storage. If that server does not host the required database yet, the user will be asked if he wants to have the database created. In the next step, the user chooses the folder in which the PLC programs of the PLCs to be monitored are located. Finally, he places check marks for the desired PLCs. Clicking on OK will trigger the conversions to L5K, parsing into the DB, creation of the OPC groups and items and eventually the start of monitoring.

The framework introduced can be applied to any PLC controlled manufacturing

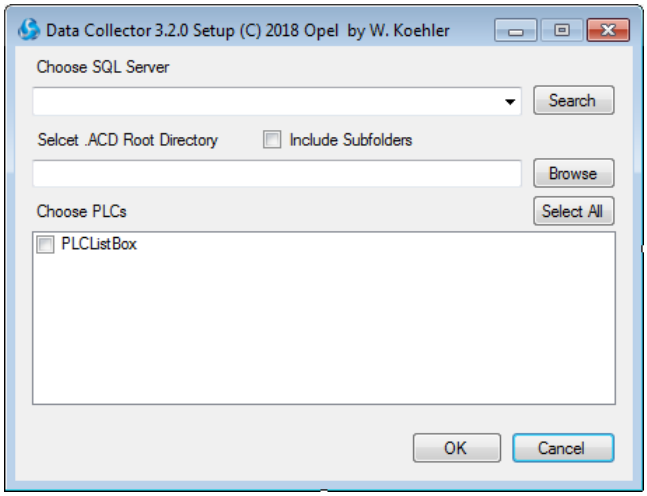

Figure 5.6.1: Simplified User Interface

equipment as long as two fundamental rules apply. First, the controller software needs to be modular to allow a regular expression based algorithm to locate the data points of interest. Besides, the naming within the program needs to be standardised to allow choosing the correct tags. If these two requirements are fulfilled, only minor modifications are required to adapt the introduced methodology to PLC systems from different manufacturers.

Pedestal processes are processes that are executed after the robot unloads the station. For example spot, nut or stud welding as well as sealing applications. These processes are considered to be part of the previous station for the production lines that were monitored for this project. This definition results in overlapping cases because the previous station gets reloaded as soon as the last part has been unloaded, although the last part is still being processed. For such a scenario, the proposed case split into setup, load, part present, unload and reset events no longer applies. This problem can be overcome by adding an ID to the part data so that the pedestal process can be appended to the station process. Alternatively, the pedestal process could become a distinct station which would eliminate the problem.

During data preprocessing, one of the tasks was to cluster the recorded events into cases that allow for the discovery of the process models. It needs to be understood that this clustering does not allow tracing of the parts through the production line. Traceability requires an ID that moves along with the parts. During this research, such an ID was available as part of the data stored on an RFID tag. Because the RFID only can be read with the part in the station, the setup and the load events were always associated with the previous ID causing wrong case associations. If traceability is desired, an algorithm needs to be added to correct this flaw.

The perception of processes is that there is a defined beginning and end. The author argues that all processes are just sub-processes because reduced to its start and end timestamp, it can become an event within a higher level process. Automotive body shops, for example, consist of zones which encompass multiple cells. Each cell by itself consists of multiple stations. This research was based on the motions on the station level with a case consisting of dozens of motions. If such a case is reduced to the start timestamp of the first motion and the complete timestamp of the last motion, then it represents an event on the cell level. If cases on the cell level are reduced to their start and complete timestamps, then they become events on the zone level. This cycle can only end if a top-level is defined on purpose. Contrary the observations does not allow the conclusion that every event represents a sub-process. For example a single motion within a detailed event log does not represent a sequence of other events. This understanding is essential for the selection of the appropriate data acquisition, preprocessing and discovery algorithms.

One of the biggest stumbling blocks encountered during this project was the lack of willingness to act upon the discovered improvement suggestions. Although this framework allows automating the process of identifying the shortcomings of a production line, their rectification still requires human resources, material and time.

6

Conclusion And Future Works

6.1 Conclusion

Nowhere is the phrase 'time is money' more fitting than in the manufacturing industry. Because of the high production volume seconds or even fractions of seconds translate into millions of Euros gained or wasted within the lifetime of a production line. Within the research area of operations management, the term 'hidden factory' was coined to describe the unidentified potential lingering within existing production equipment. At the same time, industrial maintenance related studies point out the cost resulting from inadequate equipment setup and undetected delays. Contrary to popular belief, often neither the manufacturer nor the end-user of the production equipment is fully aware of the current production process due to undocumented changes made in the equipment start-up and production phases. These considerations culminated in the research question: 'How can Heuristic Algorithms, Process Discovery and Machine Learning be applied to industrial equipment logs to improve the efficiency of the assembly and joining processes?'

Several issues need to be overcome to answer this question. First, there needs to be a detailed log that provides enough information so that such conclusions can be drawn. Currently, there are mainly fault logs available which might be helpful for maintenance predictions as suggested by some researchers. More detailed monitoring

often is achieved by adding additional sensors, e.g. vibration sensors, to critical parts of the equipment. The resulting data is also mainly used for breakdown predictions. Also, throughput data are collected which are so coarse that they only allow pinpointing bottleneck stations but do not support formulating improvement suggestions. Improving the throughput of the few bottleneck stations is the key to increasing the output of the whole production line. Today it is customary that experts visually observe the equipment in question to provide improvement suggestions based on their experience. Visual observation becomes more and more difficult due to the complexity of the machines and safety measures which prevent access to the manufacturing processes.

Literature reviews show that there has been only limited work done to answer the aforementioned research question. One of the key hindrances seems to be the accessibility to appropriate data for researchers with interest in the subject and the data owner's lack of imagination to recognise the potential hidden within their data. Additionally, efforts might be hampered by the perception that custom production equipment is not suitable for any universally applicable methodology. As previously mentioned, the logs typically available are very coarse, prompting the need for more in-depth logging which, up to now, requires experts to identify the data points of interest manually. A domain that seems to be perfectly suited to identify shortcomings within a manufacturing process is Process Mining. It focuses on the discovery of process models from activity logs created by business processes. Process Discovery relies on traces found within the log to extract the underlying model. Most practitioners expect that the log obtained from automated production equipment would not yield such traces because the machine follows an ever-repeating, predetermined sequence. The Process Mining community also restricted their possibilities for a long time by proclaiming that the resulting models do not lend themselves to automatic improvement suggestion discovery.

Above mentioned stumbling blocks could be overcome with the help of the authors thirty years of experience within the industrial automation domain and the willingness of Opel Automobile GmbH to think outside the box by supporting this research. To make this overwhelming project manageable first several stages were defined. Initially, an in-depth log needed to be obtained. The focus was to define a framework to allow for automated data point selection and logging without a priori knowledge. Next, the quality of the obtained log had to be assessed, and data preprocessing and cleaning methodologies found to prepare the data for the mining efforts. At that time also the potential to rectify any logging issues encountered had to be considered. Again the expectation on such a system was that it should be highly automated. The resulting data then could be graphically represented in the form of a Gantt chart which should give insight into potential improvement areas within the system being evaluated. Since de-

pendencies between the activities within such an initial Gantt chart are only presumed, Process Discovery techniques should be applied to determine the true dependencies. The expectation was that the discovered dependencies are 100% correct.

As previously described the data typically available from industrial production equipment is too coarse and therefore does not allow concluding the inner workings of the equipment. In the automation domain, it is standard that the workings of a machine are described with the help of a Sequence Of Operations (SOO). The SOO lists all the units within the equipment together with their expected cycle time and their dependencies. One of the aims of this work was to compare the 'as is' SOO with the design SOO. This goal defined the need to record the sensor signals of all motions. Most automation equipment is controlled by Programmable Logic Controllers which are interconnected through an Ethernet network. Open Platform Communication is a protocol built into many PLCs that allows external access to its data. Signal monitoring can be done through an OPC server. Here a topic is defined to establish communication from a PC to that PLC. Now OPC items can be defined which symbolise the addresses to be logged. Commercial products available for such monitoring require the user to either enter those addresses one by one or to choose them from a drop-down menu. Either option demands that the user has a good understanding of the machines controls concept. Besides, such an effort is very time-consuming.

Analysing the controls concept, it became clear that a PLC offers two potentially helpful features. Knowledgeable programmers tend to create a structured program because the different languages used are in to be used modular. Many of the equipment's end users even specify programming standards to ensure the modularity of the programs and the addresses used. The second feature is that most PLCs allow exporting the program in a text format which lends itself to being parsed. It is, therefore, possible to create a parser that, with the help of regular expressions, scans the PLC software and automatically extracts the addresses of the units to be monitored. During this research, it was determined that it would also be beneficial to know if a part was present within the machine or not which prompted the formulation of an additional RegEx to extract their addresses also.

The final framework uses a relational database to store the addresses extracted from the PLC program as well as the events logged. The software created allows the user to select the PLC program of the desired equipment, which is then parsed to extract the data points of interest. The algorithm then encodes these data points into OPC items and monitoring is started.

Evaluation of the framework was done within an automotive body shop in Eisenach,

Germany. At the time this body shop was managed by 27 Rockwell PLCs controlling a total of 193 work stations. The software developed was able to parse the logic of all the PLCs and set up the necessary communication through OPC within 30 minutes without any additional user intervention. Currently, the system is logging approximately 1.000.000 events per day.

Besides the development of the parsing and monitoring framework, this research focused on the potential impact of such a fine-grained monitoring effort on the communication infrastructure. Tests showed the impact on the Ethernet network as well as the PLC were negligible (see section 4.1). In conjunction with this analysis the pros and cons of a centralised vs decentralised monitoring system were considered. Finally, the accuracy of the data recorded was evaluated by comparison with parallel records from the PLC. It was shown that there was only a variation of up to 1.5%.

Based on above findings the contribution of this work can be summed up as a new novel algorithm to derive tags (PLC addresses) of interested with nomenclature based parsing of the PLC software without a priori knowledge.

Before being able to preprocess and clean the log, an understanding of potential quality problems is required. There have been several works on quality issues that can be found within data logs. For the domain of Process Mining, these generalised issues were adapted to suit the need of process logs. In the course of this work, the elements of the said matrix were evaluated one by one to determine their applicability to industrial process logs. Because not all issues apply to manufacturing processes the original 4 x 9 matrix was reduced to a 4 x 5 matrix. For each issue, a detection strategy was developed and a recommendation, on how to handle them, were given.

While evaluating the log, some other issues became apparent. There were records which were taken while the equipment was in manual mode. Since the maintenance personnel obviously initiated the different units one by one while at the same time disobeying the intended SOO, it was determined that such records are likely to distort the mining results. Therefore it was decided to ignore data sets obtained during manual operation. Additionally, a more obscure phenomenon was found. Numerous events were recorded several times consecutively, although, within the equipment observed, each unit should be initiated only once. Closer inspection revealed that these were motions that were either abruptly interrupted during execution or which bounced back once they reached their end positions. Both are cases which will lead to excessive wear within the machine prompting the need for immediate action. It is recommended to combine these duplicate event records into a single event by using their earliest start and latest complete timestamps to eliminate any negative impact on the data-mining

efforts.

The above-described quality criteria were finally applied to equipment logs of four bottleneck stations within the Eisenach body shop. The evaluation showed that the records obtained were better than 96% complete.

The key contribution within this part of the work can, therefore, be summed up as new methodologies to discover faults within an industrial equipment log together with a description of potential root causes and recommendations on how to rectify them.

Another problem that was found within the log was that the events recorded could not be associated to a single machine cycle although it was attempted to record the current part number as an additional attribute within the event log. As previously hinted, it was determined that such an association could easier be found, if the part status within the observed station was known. It was defined that a single machine cycle could be subdivided into five sections. First, the station might need to be changed over to accept the next part style to be machined. This phase was termed setup phase and includes all motions that happen before the part being present. It is followed by the loading step, which in turn changes the stations part present status to one. Once the part is present, the process cycle is initiated, and upon its completion, the station is unloaded, causing the part present status to be reset to zero. In some instances, for example, if the processed part is mechanically ejected, there is one final step to ready the machine which is appropriately named reset step.

During this part of the research, the above heuristic rules were converted into several algorithms using Python and applied to the Eisenach body shop equipment log. An evaluation of this clustering effort yielded 4640 complete cases. The domain of Process Mining also uses the term trace which describes the sequence of events found within a case. A trace, therefore, can be used to classify cases. Since all the motions recorded within the database are labelled with a unique numeric ID, it was reasoned that a trace identifier could be generated by using a hashing algorithm over a string of the motion IDs ordered by their associated timestamps. The multiple innovative algorithms that allow for the clustering of the event log into cases without a priori knowledge along with a new algorithm to filter noise while determining mean cycle times, can be considered as the main contribution of this research step.

Within the Process Mining domain, there are different approaches to compensate for the shortcomings found within an activity log. Some researchers propose repair algorithms while others attempt to reduce the flaws impact through the design of the discovery algorithms. Careful evaluation of all the options led to the decision to remove inaccurate records within the log, which in turn causes some of the cases to be

incomplete. The missing cases prompted the need for a methodology to determine case completeness. Machine learning stipulates that it should be possible to manually tag some of the cases as being either complete or incomplete. The tagged log then could be used to learn a model with the help of different machine learning algorithms that could be used to classify the remainder of the cases. Tagging of the cases, just based on the unit IDs and the two timestamps, without consulting the underlying PLC program proved to be impossible which led to the assumption that the data available was also not sufficient for machine learning algorithms. The solution was additionally engineered features to be added to the original activity log. Because the correctness of the log has been determined to be greater than 96% it was reasoned that any case belonging to the most occurring trace must be complete. Based on subject expertise, it was defined that a case always must have a part load and unload step. Even if these are the only activities recorded the case still could be complete. That will be the case if a part bypasses a production step for any reason. Within a complete case, it also can be expected that any unit that advances also needs to be returned. If that is not the case, it could be a strong indicator that the case is incomplete. The same can be said for robots. A robot that is initiated also has to perform a process. If later is missing within the log, it can be concluded that the case is incomplete.

At first, algorithms which automatically append above described engineering features to each of the cases within the Eisenach body shop log were developed. Next, manual tagging was attempted once again for which the engineering features were helpful. In some cases, the classification still was not possible without consulting the PLC code. Finally, cross-validation was used to allow different machine learning algorithms to learn models based on the tagged data. After tuning the hyper-parameters, some of the algorithms yielded a success rate of 99%. Because of its simplicity, the decision tree was chosen. It was one of the most successful algorithms while executing the task without false positives. The formulation of five novel rules to create additional engineered features for industrial equipment logs that allow for the creation of a machine learning model to determine the completeness of the cases within the log, can be seen as the main contribution of this section. Combined all the methodologies of this research stage enable the transformation of incomplete and flawed logs into case-based records that align with the requirements of Process Discovery.

The outcome of the preprocessing step is a case-based log in which all cases have been evaluated as complete. After applying the hashing method, previously described, each one of the cases can be associated with a trace. There are three traces which are of the most interest. First, there is the most occurring/observed trace. It reflects how the equipment is behaving most of the time. Often there are two more extreme traces.

One that represents the fastest observed trace and on the opposite side, the slowest observed trace. These extremes are important because one needs to question why the most observed traces are not the fastest observed traces and what can be done to force the process to execute that way. The slowest observed traces are often the reason that throughput goals are not met although observation by experts seem to attest that the station is performing to the design intent.

In the automation domain, the sequence of operation often is visualised in the form of a Gantt chart. If this chart also includes the links between the activities, it offers an easy to read representation of the process. Although there are instances where Gantt charts can get overly complex, it still is the representation of choice for this work. The most often occurring, above determined trace is next transferred onto such a Gantt chart which allows an expert to spot potential problems right away. It needs to be mentioned that the representation at this point does not include any dependency links as they are unknown, based on a single trace. Most people, even the subject matter experts, assume that any activity that starts right after another activity has completed must be depending on that previous activity. This assumption is tolerable to discover improvement suggestions.

Some obvious observations can be made within that Gantt chart, which also have the potential to be handled by a reasoning based algorithm. The speed of a cylinder advancing and returning depends on the bore and stroke of the cylinder and the air pressure applied. If everything is kept constant, it can be assumed that the opening and closing times, reflected in the Gantt chart, should be close to equal. Deviations can be caused by wrong adjustments of the flow controls often used in such systems. Another reason for the observation of such behaviour is the recording of duplicate events. As described within the preprocessing paragraph, it is best to combine these multiple events into a single event representing the earliest start and latest complete time. The cause for this manifestation within the Gantt chart can only be discovered by evaluating the underlying data. Often multiple cylinders are connected to a single solenoid valve. In most of the cases, the bore and stroke of these cylinders are identical. Therefore, if one cylinder is faster then the others it can be reasoned that improvements are possible. Usually, the motions within automated production systems depend on each other. Completion of one activity triggers the start of the next, and so on. Any time without activity consequently is suspect. Gaps just before the unload step could also indicate a 'blocked' situation caused by the next station not being ready in time to accept the completed part.

Above described definitions were encoded into algorithms which were applied to the data of four bottleneck stations within the Eisenach body shop. The automatically

generated Gantt charts yielded improvement suggestions, ranging from 2.1 to 12.5 seconds, that were previously overlooked by experts trying to reduce the cycle time of those stations.

The contribution of this section can be summed up as a novel framework consisting of eight definitions that aid the discovery of shortcomings that typically can be found within the sequence of automated equipment accompanied by methodologies that allow for their automated annotation within sequence charts.

Industrial processes are usually controlled by programmable logic controllers that execute a fixed program. Because of that, it was expected that the log would reflect an ever-repeating sequence. It was not until the Eisenach body shop log was imported into the commercial Process Mining software DISCO that it became apparent that there was an unexpected abundance of traces. These traces prompted an in-depth analysis of the underlying causes. In the end, three main contributors were identified. In a process log, that has not been cleaned, logging issues lead to differences in the sequence being recorded. This problem can be counteracted by the preprocessing and cleaning methodologies previously described. The second reason for the different traces is the style dependency of the equipment. Different units are used for different parts being manufactured or even if the same units are used, their sequence might change. The log should be sliced by styles being processed, before any mining activity takes place, to eliminate this factor. The third category of traces is caused by partially asynchronous concurrent processes which refers to multiple, not linked activities being executed simultaneously. Depending on different factors, this will cause the sequence in the log to change. It is these traces that are needed to discover the true dependencies between activities because, in one trace, it might seem that an activity follows another where the next trace could contradict that perception. Therefore it can be reasoned that a dependency only exists if activity 'b' always follows activity 'a' and the opposite is never observed. This definition is one of the fundamental rules of Process Discovery and explains why the quality of the model discovered increases with the number of traces available. The presence of such indicators relies purely on chance which is believed to increase with the number of cases recorded.

The domain of Process Mining was established to discover business process models from business activity logs. Based on that, it seemed to be a natural fit for exploring industrial processes. In a first attempt, the cleaned and clustered log was analysed with several of the established Process Discovery algorithms, but the results did not match the expectations. This issue can be contributed to the fact that most of these algorithms are trying to balance the four quality measures fitness, precision, generalisation, and simplicity. In the eyes of an automation practitioner, disregarding the activity life

cycle during the process model discovery as most algorithms do, is counter-intuitive. Both seem to lead to the discovery of undesired dependencies even if the traces provided include all information necessary to discover a highly accurate model. This apparent gap led to several months of hypothesising and experimenting, which resulted in an algorithm that was able to discover a highly accurate model provided the needed traces where available. At that point, awareness rose that there was a second, far less publicised category of Business Process Discovery algorithms which did also consider the activity life cycle. Applying those to an industrial log lead, in two out of three cases, to the desired process model. Analysing why those algorithms performed showed that the Heuristic++ Miner is based on statistical principles which make it ignore dependencies that are only sparsely present within the log. It, therefore, could be reasoned that it discovered the desired model by chance. The β-algorithm, on the other hand, uses a clear definition to determine if a dependency exists or not. Additional complexity was added to address some other concerns, which are not relevant for automation processes. At that point, the focus shifted away from trying to define a new Process Discovery algorithm to create an algorithm that satisfies the needs of automated processes while remaining understandable.

The α_{LC}-algorithm is an activity life cycle extension to the well-known α-algorithm. Its definitions for directly following activities align with two definitions found for the β-algorithm without the added complexity. The α_{LC}-algorithm uses matrix operations that can also be understood by users outside the Process Mining community. The graphical representation of the discovered model is based on the, in the automation domain, more common flow chart instead of a Petri Net often used in Process Discovery. Experiments showed that for industrial equipment logs the α_{LC}-algorithm yielded the same results as the β-algorithm. Provided with just a few purposely chosen traces, that contain all the necessary information, the α_{LC}-algorithm outperformed the β-algorithm.

The contribution can be summed up as the development of an improved Process Discovery method to model automated manufacturing processes based on sparse industrial logs.

In the section 'Knowledge-Based Improvement Discovery' it was mentioned that one of the shortcomings of the proposed framework so far is that the dependencies are unknown. An important step towards discovering those dependencies was made in the previous section already. Up to now, there is still no certainty that the discovered model will be complete. Numerous researchers tried to address this uncertainty by providing means to gauge the quality of the discovered model. Since all these efforts determine their ratings based on the relation of the discovered model to the underlying log, the metrics can only be considered as the best guess. In the last paragraph, it was

also highlighted that one of the α_{LC}-algorithms benefits is that it can discover a highly accurate process model from a minimalistic log as long as it contains all the descriptive information. This result prompted the question if it would be possible to determine which traces must be present within the log to be sure that the discovered log will be complete.

The investigation into this problem started by plotting one of the recorded cases into a Gantt chart. Although the observer tends to assume the relations in such a chart, it became apparent, upon closer examination, that especially the activities that had potentially multiple dependencies were questionable. This suspicion became even clearer when using the α_{LC}-algorithm to create a flow chart from just that single case, and it led to the statement that all activities preceding a join require additional scrutiny. Single dependency on the other side can be considered to be correct. It also became clear that most industrial manufacturing processes start with a single activity, which often is the part being loaded and end with the part being unloaded in a single activity.

Next stood the question about what traces needed to be observed for the above-identified activities to be certain that the presumed relation holds. As previously described, Process Discovery defines a direct following relation as an activity 'b' always following an activity 'a' without ever observing the opposite. This rule allows for the definition that if activity 'b' truly depends on activity 'a' then this should still be the case if the duration of activity 'a' is extended until all remaining activities reach their complete state. Therefore if such extreme traces are recorded for all activities in question one can be certain that the process model discovered from such log with the help of the α_{LC}-algorithm will be highly accurate.

At the beginning of this conclusion, it was already explained that most industrial processes are controlled by PLCs, which often allow for external access through OPC. The concept of 'interactive trace induction' takes advantage of this capability by adding a contact into the PLC code, which allows for the delay of the activity's completion. This modification enables the recording of the extreme traces described above with the added benefit that only a few cycles need to be recorded to allow for highly accurate Process Model discovery rather than collecting months' worth of data with the hope that the appropriate cases were observed.

The idea of delaying the activity's completion through OPC was successfully tried on actual production equipment. Further experiments were contacted using the artificial sequence previously introduced, and it was shown that the framework allows for a highly accurate model discovery.

The contribution for this final part within this work can be summed up as a novel

methodology to interactively induce traces, through signal delays, that enable the discovery of a highly accurate process model by revealing the true dependencies.

In the course of this thesis, all aspects required to 'derive value from incomplete and flawed industrial equipment event logs' were investigated. Criteria and methodologies were provided that enable every step from the data acquisition over preprocessing and cleaning to dependency discovery which at the end allows for the automated annotation of improvement suggestion within a process' Gantt chart. It was shown that such a framework, applied to real-life industrial processes, can improve the throughput of a production line significantly, thus saving big sums over the lifetime of the equipment.

This research aimed to develop a framework that can automatically set up a monitoring system for an automated production system, clean the data obtained and discover a model representing the current state of the machine. The following section points out some applications for the obtained models that could be explored in the future.

6.2 Future Works

6.2.1 Process Model-Based Case Clustering

It was shown that the α_{LC}-algorithm, in conjunction with 'interactive trace induction' is capable of discovering a highly accurate process model. Essentially it is circumventing the original concept of mass data collection, preprocessing and log repair for model discovery. Knowledge-based improvement suggestions still rely on the production data to be logged. With the process model known, case clustering now could be achieved through alignment and conformance checking algorithms. At the same time also quality issues within the log could be detected and the cases checked for completeness.

6.2.2 Process Model-Based Log Repair

As previously stated, one of the known downfalls of any log repair effort is that it does not consider the dependencies of the activities. Instead, it assumes that a missing event is related to the prior and next activity. After discovering the process model, the dependencies are known, and there is an opportunity that the repair results might improve. Since this will not be of any benefit for this project, this possibility was not further investigated.

6.2.3 Process Model Based Improvement Suggestions

This project demonstrated that improvement suggestions could be obtained from and marked within a Gantt chart of the equipment's sequence of operations. Since a Process Model is just a different form to represent the same sequence, it should also allow for the discovery of the same performance shortcomings. This search requires annotating the process model with a duration for nodes and 'waiting times' for all the edges. Based on that information it then will be possible to detect similar and opposing motions with different cycle times, excessive fluctuations for the cycle time of a node and gaps between nodes. Since the process model shows all the dependencies for each node, it might additionally also be possible to exclude non-critical fluctuations if they happen in parallel to some other time-intensive events. Highlighting the different areas with colours would aid the clarity of the model.

6.2.4 Commissioning and Start-up

One of the main goals during start-up and commissioning is to achieve the specified cycle time for the system. Lots of time is spent in obtaining the current cycle time as well as finding ways to improve it. The proposed framework could monitor the system in the background also during this project phase and provide the programmer with annotated Gantt charts that highlight improvement opportunities.

6.2.5 Comparison Design Vs. Event Log Model

During the design phase of any manufacturing equipment, the engineers determine the sequence of the tooling. During start-up of the actual equipment often, it is found that this sequence can not perform as stated in the machine's specification. It is common practice that the start-up engineers then tweak the sequence on-site. Unfortunately, these last-minute changes often do not make it back to the documentation. Obtaining the process model from the actual equipment data will yield the real sequence, which then could be used for compliance checking.

6.2.6 Equipment Documentation

Once a manufacturing system is fully commissioned the 'as-built' status has to be documented. Often the process will no longer match the designed process because it was discovered that the system, as designed, did not meet the quality or cycle time requirements. Process Discovery can be used to discover the 'as is' process model, which then could be attached to the documentation. Ideally, the discovered model could be transformed into a Gantt chart to conform to standard documentation conventions.

6.2.7 Feedback of Real Life Data

The design process of an automated production system heavily relies on assumptions on how much time is needed for specific tasks. Timing annotations within the discovered process model could be used to correct these assumptions based on real-life data. Such a process will lead to a gradual improvement of the design process.

6.2.8 Real Time Monitoring

Diagnostic systems of automated systems are hand-coded, which leads to in-completed diagnostics due to faulty or missing logic. An automatically discovered process model can be used as the 'master'. Any deviation from that master, be it the activity life cycle or the sequence, could be considered a fault and automatically trigger a message to be displayed on the human-machine interface.

6.2.9 Automated Maintenance Task Scheduling

There are two types of maintenance that take place within manufacturing equipment. The first is the maintenance that is performed by the maintenance personnel based on faults occurring or based on a predetermined maintenance schedule. The second type includes periodic maintenance tasks related to the processes within the stations that are automatically triggered by the process controllers. It can be assumed that the maintenance personnel, for scheduled maintenance tasks, will choose a time slot that does not interfere with the production targets. Contrary, the process controllers count the number of processes and whenever a threshold is reached trigger the maintenance task without any regard to the impact for production. With the help of the process model, it would be possible to determine when there will be a gap coming up in the production flow, and the process controllers then could be forced to perform their maintenance tasks during that time.

6.2.10 Process Recommendation System

Automated manufacturing systems follow a rigidly programmed sequence of operations. If there is an interruption of that sequence, due to a mechanical fault or and human operator not following the intended process, the cell throughput will suffer. A system can be developed, that determines, based on the previously recorded cases, the best possible path for continuation of the process. Based on that information, decisions can be made to direct the human operator to the next task to be completed or to trigger a more beneficial task for the robot to perform in the meantime.

6.2.11 Predictive Maintenance

The cleaned data should also be well suited for predictive maintenance, although it does not require a Gantt chart to be created or a process model to be discovered. Instead, a long-term observation would be valuable. The proposed data collection algorithm logs the fault message associated within a production cell along with the sequential motions. Therefore it should be possible to relate faults with behaviour found in the sequential log. Literature review suggests that such tagged data could be used to predict future failures. It is also believed that the slow degradation of the units with a station will lead to more sluggish behaviour. Further research could prove if such relationships exist and if they could be used to predict impending failure. Such reasoning might require more precise timestamps and more extended term observation.

6.2.12 Maintenance Assistance

Another slightly related opportunity would be the development of maintenance assistance software. During this project, it was observed that many equipment faults could be overcome by pressing the reset button. Although the right approach would be to root cause and eliminate the faults, it might be beneficial to a strained maintenance workforce if the monitoring system would 'learn' how maintenance reacts to specific faults. The monitoring could be done through logging fault messages as well as HMI and reset button activities. This learned behaviour could eventually be executed by the monitoring PC automatically, thus relieving the maintenance workers.

6.2.13 Duration Assignment

Typically Process Discovery assigns duration to the edge between two nodes when plotting a process model. An edge represents a dependency between activities and the duration assigned is the difference between the associated timestamps. Because of the use of life cycle information, there no longer is a need to assign a duration to the edge. A benefit could only be derived if the edges were assigned a 'wait duration' which is the difference between the time this edge would have allowed for the next event to start and the actual start of the event because of other dependencies, breaks or buffers. This way, the imbalance within a system could be shown, and improvement potential could be detected. Assigning the duration to the node yields another benefit because it should be expected that this duration is nearly constant. A cylinder, for example, will typically always take the same time to extend or retract. If the value is not constant, then this would indicate a potential problem caused by an external factor. For the cylinder, such an external factor could be fluctuating air pressure.

Bibliography

Aalst, W. M. P Van Der (2018). "Relating Process Models And Event Logs-21 Conformance Propositions". In: *Ceur Workshop Proceedings* 2115, pp. 56–74.

Aalst, W. Van Der (2016a). "How To Get Started With Process Mining?" In: *How To Get Started With Process Mining?*

— (2016b). *Process Mining: Data Science In Action*. Vol. 1. Berlin, Heidelberg: Springer, pp. 3–23. DOI: 10.1007/978-3-662-49851-4_1.

Aalst, W. Van Der, A. Adriansyah, et al. (2011). "Process Mining Manifesto". In: *Business Process Management Workshops. Bpm 2011. Lecture Notes In Business Information Processing* 99, pp. 169–194. DOI: 10.1007/978-3-642-28108-2_19.

Aalst, W. Van Der, T. Weijters, and L. Maruster (2004). "Workflow Mining: Discovering Process Models From Event Logs". In: *Ieee Transactions On Knowledge And Data Engineering* 16, pp. 1128–1142. DOI: 10.1109/TKDE.2004.47.

Abbas, H. A. and A. M. Mohamed (2015). "Review On The Design Of Web Based Scada Systems Based On Opc Da Protocol". In: *Arxiv Preprint*. DOI: arXiv:1506.05069.

Accorsi, R. and T. Stocker (2011). "Discovering Workflow Changes With Time-Based Trace Clustering". In: *Simpda 2011: Data-Driven Process Discovery And Analysis*, pp. 154–168. DOI: 10.1007/978-3-642-34044-4_9.

Agrawal, R., D. Gunopulos, and F. Leymann (1998). "Mining Process Models From Workow Logs". In: *Advances In Database Technology Edbt'98. Edbt 1998. Lecture Notes In Computer Science* 1377. DOI: 10.1007/BFb0101003.

Agrawal, R. and R. Srikant (1995). "Mining Sequential Patterns". In: *Research Report*, pp. 1–22.

Anand, S. (2009). *Design Intent Recovery From Plc Ladder Logic Programs Using Context-Free Grammar And Heuristic Algorithms*. Madras, India.

Andaloussi, A. A., A. Burattin, and B. Weber (2018). "Toward An Automated Labeling Of Event Log Attributes". In: *Bpmds 2018, Emmsad 2018: Enterprise, Business-Process And Information Systems Modeling*, pp. 82–96. DOI: 10.1007/978-3-319-91704-7_6.

Ayo, F. E., O. Folorunso, and F. T. Ibharalu (2017). "A Probabilistic Approach To Event Log Completeness". In: *Expert Systems With Applications* 80, pp. 263–272. DOI: 10.1016/j.eswa.2017.03.039.

Baier, T. et al. (2015). "Matching Of Events And Activities-An Approach Using Declarative Modeling Constraints". In: *Enterprise, Business-Process And Information Systems Modeling*, pp. 119–134. DOI: 10.1007/978-3-319-19237-6_8.

Banerjee, T. P. and S. Das (2012). "Multi-Sensor Data Fusion Using Support Vector Machine For Motor Fault Detection". In: *Information Sciences* 217, pp. 96–107. DOI: 10.1016/j.ins.2012.06.016.

Bayomie, D., A. Awad, and E. Ezat (2019). "Correlating Unlabeled Events From Cyclic Business Processes Execution". In: *Caise 2016: Advanced Information Systems Engineering*, pp. 274–289. DOI: 10.1007/978-3-319-39696-5_17.

Bayomie, D., C. Di Ciccio, and M. La Rosa (2019). "A Probabilistic Approach To Event-Case Correlation For Process Mining". In: *Er 2019: Conceptual Modeling*, pp. 136–152. DOI: 10.1007/978-3-030-33223-5_12.

Bayomie, D., I. M. A. Helal, et al. (2016). "Deducing Case Ids For Unlabeled Event Logs". In: *Bpm 2016: Business Process Management Workshops* , pp. 242–254. DOI: 10.1007/978-3-319-42887-1_20.

Bertoli, P. et al. (2013). "Reasoning-Based Techniques For Dealing With Incomplete Business Process Execution Traces". In: *Ai*Ia 2013: Advances In Artificial Intelligence Xiiith International Conference Of The Italian Association For Artificial Intelligence*, pp. 469–480. DOI: 10.1007/978-3-319-03524-6_40.

Bolt, A., W. M. P. Van Der Aalst, and M. De Leoni (2017). "Finding Process Variants In Event Logs". In: *Otm 2017: On The Move To Meaningful Internet Systems. Otm 2017 Conferences*, pp. 45–52. DOI: 10.1007/978-3-319-69462-7_4.

Boschert, S. and R. Rosen (2016). *Digital Twin The Simulation Aspect*. Cham, Switzerland: Springer, pp. 59–74. DOI: 10.1007/978-3-319-32156-1_5.

Bose, R. P. J. C., R. S. Mans, and W. M. P. Van Der Aalst (2013). "Wanna Improve Process Mining Results?" In: *2013 Ieee Symposium On Computational Intelligence And Data Mining (Cidm)*, pp. 127–134. DOI: 10.1109/CIDM.2013.6597227.

Bose, R. P. J. C., E. H. M. W. Verbeek, and W. M. P. Van Der Aalst (2011). "Discovering Hierarchical Process Models Using Prom". In: *Olympics: Information Systems In A Diverse World. Caise 2011*, pp. 33–48. DOI: 10.1007/978-3-642-29749-6_3.

Breivold, H. P. and K. Sandström (2015). "Internet Of Things For Industrial Automation–Challenges And Technical Solutions". In: *2015 Ieee International Conference On Data Science And Data Intensive Systems,* pp. 532–539. DOI: 10.1109/DSDIS.2015.11.

Browning, T. R. (2001). "Applying The Design Structure Matrix To System Decomposition And Integration Problems: A Review And New Directions". In: *Ieee Transactions On Engineering Management* 48, pp. 292–306. DOI: 10.1109/17.946528.

— (2002). "Process Integration Using The Design Structure Matrix". In: *System Engineering* 5, pp. 180–193. DOI: 10.1002/sys.10023.

Brzychczy, E. and A. Trzcionkowska (2018). "Creation Of An Event Log From A Low-Level Machinery Monitoring System For Process Mining Purposes". In: *Intelligent Data Engineering And Automated Learning Ideal 2018* 2, pp. 54–63. DOI: 10.1007/978-3-030-03496-2_7.

Burattin, A. (2015). "Heuristics Miner For Time Interval. In Process Mining Techniques In Business Environments". In: *Process Mining Techniques In Business Environments* , pp. 85–95. DOI: 10.1007/978-3-319-17482-2_11.

Burattin, A. and R. Vigo (2011). "A Framework For Semi-Automated Process Instance Discovery From Decorative Attributes". In: *2011 Ieee Symposium On Computational Intelligence And Data Mining (Cidm)*, pp. 176–183. DOI: 10.1109/CIDM.2011.5949450.

Ceravolo, P. et al. (2017). "Toward A New Generation Of Log Pre-Processing Methods For Process Mining". In: *Bpm 2017: Business Process Management Forum*, pp. 55–70. DOI: 10.1007/978-3-319-65015-9_4.

Charaniya, S. et al. (2010). "Mining manufacturing data for discovery of high productivity process characteristics". In: *Journal of biotechnology* 147, pp. 186–197. DOI: 10.1016/j.jbiotec.2010.04.005.

Che, Z. et al. (2018). "Recurrent Neural Networks For Multivariate Time Series With Missing Values". In: *Scientific Reports* 8, p. 6085. DOI: arXiv:1606.01865v1.

Chien, C. F. et al. (2005). "Cycle Time Prediction And Control Based On Production Line Status And Manufacturing Data Mining". In: *Ieee International Symposium On Semiconductor Manufacturing* 1, pp. 327–330. DOI: 10.1109/ISSM.2005.1513369.

Choudhary, A. K., J. A. Harding, and M. K. Tiwari (2009). "Data Mining In Manufacturing: A Review Based On The Kind Of Knowledge". In: *Journal Of Intelligent Manufacturing* 20, pp. 501–521. DOI: 10.1007/s10845-008-0145-x.

Cook, J. E. and A. L. Wolf (1995). "Automating Process Discovery Through Event-Data Analysis". In: *1995 17Th International Conference On Software Engineering*, pp. 73–73. DOI: 10.1145/225014.225021.

Dietterich, T. and R. S. Michalski (1983). "Discovering Patterns In Seqences Of Objects". In: *Proceedings Of The International Machine Learning Workshop*, pp. 40–57.

Dixit, P. M. et al. (2018). "Detection And Interactive Repair Of Event Ordering Imperfection In Process Logs". In: *International Conference On Advanced Information Systems Engineering* 1, pp. 274–290. DOI: 10.1007/978-3-319-91563-0_17.

Djedovic, A. et al. (2019). "A Rule Based Events Correlation Algorithm For Process Mining". In: *Iat 2019: Advanced Technologies, Systems, And Applications Iv -Proceedings Of The International Symposium On Innovative And Interdisciplinary Applications Of Advanced Technologies*, pp. 687–605. DOI: 10.1007/978-3-030-24986-1_47.

Djenouri, Y., A. Belhadi, and P. Fournier-Viger (2018). "Extracting Useful Knowledge From Event Logs: A Frequent Itemset Mining Approach". In: *Knowledge-Based Systems* 139, pp. 132–148. DOI: 10.1016/j.knosys.2017.10.016.

Eck, M. L. Van, N. Sidorova, and W. M. P Van Der Aalst (2016). "Composite State Machine Miner: Discovering And Exploring Multi-Perspective Processes". In: *Demonstration Track Of The 14Th International Conference On Business Process Management (Bpm 2016)*, pp. 73–77.

Eppinger, S. D. and T. R. Browning (2012). *Design Structure Matrix Methods And Applications*. Cambridge, UK: Mit Press.

F, C. et al. (2007). "Data Mining For Yield Enhancement In Semiconductor Manufacturing And An Empirical Study". In: *Expert Systems With Applications* 33, pp. 192–198. DOI: 10.1016/j.eswa.2006.04.014.

Fahland, D. and W. M. P Van Der Aalst (2015). "Model Repair - Aligning Process Models To Reality". In: *Information Systems* 47, pp. 220–243. DOI: 10.1016/j.is.2013.12.007.

Falcione, A. and B. H. Krogh (1993). "Design Recovery For Relay Ladder Logic". In: *Ieee Control Systems Magazine* 13, pp. 90–98. DOI: 10.1109/37.206990.

Farooqui, A. et al. (2018). "From Factory Floor To Process Models: A Data Gathering Approach To Generate, Transform, And Visualize Manufacturing Processes". In: *Cirp Journal Of Manufacturing Science And Technology* 24, pp. 6–16. DOI: 10.1016/j.cirpj.2018.12.002.

Fayyad, U., G. Piatetsky-Shapiro, and P. Smyth (1996). "The KDD process for extracting useful knowledge from volumes of data". In: *Communications of the ACM* 39, pp. 27–34. DOI: 10.1145/240455.240464.

Feldmann, K. and A. W. Colombo (1999). "Monitoring Of Flexible Production Systems Using High-Level Petri Net Specifications". In: *Control Engineering Practice* 7, pp. 1449–1466. DOI: 10.1016/S0967-0661(99)00107-0.

Fesler, M. and T. Sauter (2002). "The Fieldbus War: History Or Short Break Between Battles?" In: *4Th Ieee International Workshop On Factory Communication Systems (*, pp. 73–80. DOI: 10.1109/WFCS.2002.1159702.

Fleischmann, H. et al. (2016). "Improving Maintenance Processes With Distributed Monitoring Systems". In: *2016 Ieee 14Th International Conference On Industrial Informatics (Indin)*, pp. 377–382. DOI: 10.1109/INDIN.2016.7819189.

Fortuin, V., G. Raetsch, and S. Mandt (2019). "Multivariate Time Series Imputation With Variational Autoencoders". In: *Mathematics, Computer Science*. DOI: arXiv:1907.04155.

Fournier-Viger, P. et al. (2017). "A Survey Of Sequential Pattern Mining". In: *Data Science And Pattern Recognition* 1, pp. 54–77.

George, M. L. (2002). *Lean Six Sigma : Combining Six Sigma Quality With Lean Speed*. Vol. 1. New York, USA: The Mcgraw-Hill Companies. DOI: 10.1036/0071385215.

Gertosio, C. and A. Dussauchoy (2004). "Knowledge Discovery From Industrial Databases". In: *Journal Of Intelligent Manufacturing* 15, pp. 29–37. DOI: 10.1023/B:JIMS.0000010073.54241.e7.

Gonzalez, I. et al. (2017). "Integration Of Sensors, Controllers And Instruments Using A Novel Opc Architecture". In: *Sensors* 17, p. 1512. DOI: 10.3390/s17071512.

Grieves, M. and J. Vickers (2017). *Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior In Complex Systems*. Cham, Switzerland: Springer, pp. 85–113. DOI: 10.1007/978-3-319-38756-7_4.

Groeger, C. et al. (2012). "Data mining-driven manufacturing process optimization". In: *Proceedings of the world congress on engineering* 3, pp. 4–6.

Gschwandtner, H. et al. (2012). "A Taxonomy Of Dirty Time-Oriented Data". In: *Multidisciplinary Research And Practice For Information Systems*, pp. 58–72. DOI: 10.1007/978-3-642-32498-7_5.

Günther, C. W. and A. Rozinat (2012). "Disco: Discover Your Processes". In: *Demonstration Track Of The 10Th International Conference On Business Process Management (Bpm 2012)* 940, pp. 40–44.

Harding, J. A., M. Shahbaz, and A. Kusiak (2006). "Data Mining In Manufacturing: A Review". In: *Journal Of Manufacturing Science And Engineering* 128, pp. 969–976. DOI: 10.1115/1.2194554.

Haubeck, C. et al. (2014). "Interaction Of Model-Driven Engineering And Signal- Based Online Monitoring Of Production Systems". In: *Iecon 2014 - 40Th Annual Conference Of The Ieee Industrial Electronics Society*, 2571 –2577. DOI: 10.1109/IECON.2014.7048868.

Hee, K. M. Van, Z. Liu, and N. Sidorova (2011). "Is My Event Log Complete? - A Probabilistic Approach." In: *2011 Fifth International Conference On Research Challenges In Information Science*, pp.1–12. DOI: 10.1109/RCIS.2011.6006848.

Helal, I. M. A., A. Awad, and A. El Bastawissi (2015). "Runtime Deduction Of Case Id For Unlabeled Business Process Execution Events". In: *2015 Ieee/Acs 12Th International Conference Of Computer Systems And Applications (Aiccsa)*, pp.1–8. DOI: 10.1109/AICCSA.2015.7507132.

Heynicke, R. et al. (2017). "Io - Link Wireless Enhanced Sensors And Actuators For Industry 4 . 0 Networks". In: *Ama Conferences 2017 Sensor 2017 And Irs2 2017*, pp. 134–138. DOI: 10.5162/sensor2017/A8.1.

Ho, G. T. S. et al. (2006). "An Intelligent Production Workflow Mining System For Continual Quality Enhancement". In: *International Journal Of Advanced Manufacturing Technology* 28, pp. 792–802. DOI: 10.1007/s00170-004-2416-9.

Hoffmann, M. et al. (2016). "Continuous Integration Of Field Level Production Data Into Top-Level Information Systems Using The Opc Interface Standard". In: *Procedia Cirp* 41, pp. 496–501. DOI: 10.1016/j.procir.2015.12.059.

Holzinger, A. (2013). "Human-Computer Interaction and Knowledge Discovery (HCI-KDD): What is the benefit of bringing those two fields to work together?" In: *International Conference on Availability, Reliability, and Security* 1, pp. 319–328. DOI: 10.1007/978-3-642-40511-2_22.

Hou, T. S., W. L. Liu, and L. Lin (2003). "Intelligent Remote Monitoring And Diagnosis Of Manufacturing Process Using An Integrate Approach Of Neural Networks And Rough Sets". In: *Journal Of Intelligent Manufacturing* 14, pp. 239–253. DOI: 10.1023/A:1022911715996.

Houyou, A. M. et al. (2012). "Agile Manufacturing: General Challenges And An Iot@ Work Perspective". In: *2012 Ieee 17Th International Conference On Emerging Technologies & Factory Automation (Etfa 2012)*, pp. 1–7. DOI: 10.1109/ETFA.2012.6489653.

Hu, H., Z. Li, and A. Wang (2006). "Mining Of Flexible Manufacturing System Using Work Event Logs And Petri Nets". In: *International Conference On Advanced Data Mining And Applications Adma 2006*, pp. 360–387. DOI: 10.1007/11811305_42.

Huang, W. et al. (2017). "Real-Time Data Acquisition Support For Iec 61499 Based Industrial Cyber-Physical Systems". In: *Iecon 2017 - 43Rd Annual Conference Of The Ieee Industrial Electronics Society*, pp. 6689–6694. DOI: 10.1109/IECON.2017.8217168.

Jaber, A. A. and R. Bicker (2014). "The State Of The Art In Research Into The Condition Monitoring Of Industrial Machinery". In: *International Journal Of Current Engineering And Technology* 4, pp. 1986–2001.

Kagermann, H., W. D. Lukas, and W. Wahlster (2011). "Industrie 4.0: Mit Dem Internet Der Dinge Auf Dem Weg Zur 4. Industriellen Revolution". In: *Vdi Nachrichten* 13, p. 2.

Kherbouche, M. O., N. Laga, and P. A. Masse (2017). "Towards A Better Assessment Of Event Logs Quality". In: *2016 Ieee Symposium Series On Computational Intelligence (Ssci)*, pp. 1–8. DOI: 10.1109/SSCI.2016.7849946.

Kim, T. T. T. and H. Werthner (2011). "An Ontology-Based Framework For Enriching Event-Log Data". In: *Semapro 2011 : The Fifth International Conference On Advances In Semantic Processing*, pp. 110–115.

Kim, W. et al. (2003). "A Taxonomy Of Dirty Data". In: *Data Mining And Knowledge Discovery* 7, pp. 81–99. DOI: 10.1023/A:1021564703268.

Koehler, W. and Y. Jing (2018). "A Novel Block-Based Programming Framework For Non-Programmers To Validate Plc Based Machine Tools For Automotive Manufacturing Facilities". In: *2018 11Th International Conference On Developments In Esystems Engineering (Dese)*, pp. 202–207. DOI: 10.1109/DeSE.2018.00046.

Lee, J. (2003). "E-Manufacturing Fundamental, Tools, And Transformation. Robotics And Computer-Integrated Manufacturing". In: *Robotics And Computer-Integrated Manufacturing* 19, pp. 501–507. DOI: 10.1016/S0736-5845(03)00060-7.

Lee, J., B. Bagheri, and C. Jin (2016). "Introduction To Cyber Manufacturing". In: *Manufacturing Letters* 8, pp.11–15. DOI: 10.1016/j.mfglet.2016.05.002.

Lee, J., B. Bagheri, and H. Kao (2015). "A Cyber-Physical Systems Architecture For Industry 4.0-Based Manufacturing Systems". In: *Manufacturing Letters* 3, pp. 18–23. DOI: 10.1016/j.mfglet.2014.12.001.

Lee, J., E. Lapira, et al. (2013). "Recent advances and trends in predictive manufacturing systems in big data environment". In: *Manufacturing letters* 1, pp. 38–41. DOI: 10.1016/j.mfglet.2013.09.005.

Leemans, S. J. J., D. Fahland, and W. M. P Van Der Aalst (2013). "Discovering Block-Structured Process Models From Event Logs Containing Infrequent Behaviour". In: *Bpm 2013: Business Process Management Workshops*, pp. 66–78. DOI: 10.1007/978-3-319-06257-0_6.

— (2018). "Scalable Process Discovery And Conformance Checking". In: *Software & Systems Modeling* 17, 599 631. DOI: 10.1007/s10270-016-0545-x.

— (2016). "Using Life Cycle Information In Process Discovery". In: *Bpm 2016: Business Process Management Workshops*, pp. 204–217. DOI: 10.1007/978-3-319-42887-1_17.

Levine, J. R. (2009). *Flex & Bison: Text Processing Tools*. Vol. 1. Sebastopol, USA: O'Reilly Media, Inc.

Li, C., M. Reichert, and A. Wombacher (2008). "Mining Process Variants: Goals And Issues". In: *2008 Ieee International Conference On Services Computing* 2, pp. 573–576. DOI: 10.1109/SCC.2008.103.

Li, J., D. Liu, and B. Yang (2007). "Process Mining: Extending ?-Algorithm To Mine Duplicate Tasks In Process Logs". In: *Apweb 2007, Waim 2007: Advances In Web And Network Technologies, And Information Management*, pp. 396–407. DOI: 10.1007/978-3-540-72909-9_43.

Luengo, J., S. Garcia, and F. Herrera (2012). "On The Choice Of The Best Imputation Methods For Missing Values Considering Three Groups Of Classification Methods". In: *Knowledge And Information Systems* 32, pp. 77–108. DOI: 10.1007/s10115-011-0424-2.

L et al. (2012). "Data Transformation And Semantic Log Purging For Process Mining". In: *Caise 2012: Advanced Information Systems Engineering*, pp. 238–253. DOI: 10.1007/978-3-642-31095-9_16.

Martin, N. et al. (2019). "Interactive Data Cleaning For Process Mining: A Case Study Of An Outpatient Clinic'S Appointment System". In: *Caise 2019:International Conference On Advanced Information Systems Engineering*, pp. 1–12.

McLean, R. and J. Antony (2014). "Why continuous improvement initiatives fail in manufacturing environments? A systematic review of the evidence". In: *International Journal of Productivity and Performance Management* 63, pp. 370–376. DOI: 10.1108/IJPPM-07-2013-0124.

Mizuya, T., M. Okuda, and T. Nagao (2017). "A case study of data acquisition from field devices using opc ua and mqtt". In: *56th Annual Conference of the Society of Instrument and Control Engineers of Japan*, pp. 611–614. DOI: 10.23919/SICE.2017.8105594.

Monostori, L. et al. (2016). "Cyber-Physical Systems In Manufacturing". In: *Cirp Annals* 65, pp. 621–641. DOI: 10.1016/j.cirp.2016.06.005.

Moritz, S. and T. Bartz-Beielstein (2017). "Imputets: Time Series Missing Value Imputation In R". In: *The R Journal* 9, pp. 207–218. DOI: 10.32614/RJ-2017-009.

Müller, A. C. and S. Guido (2016). *Introduction To Machine Learning With Python*. Vol. 1. Sebastopol, USA: O'Reilly Media, Inc.

Nakajima, S. (1989). *Tpm Development Program : Implementing Total Productive Maintenance*. Vol. 1. Portland, USA: Productivity Press.

Nemeth, M. and A. Peterkova (2018). "Proposal Of Data Acquisition Method For Industrial Processes In Automotive Industry For Data Analysis According To Industry 4.0". In: *2018 Ieee 22Nd International Conference On Intelligent Engineering Systems (Ines)*, pp. 157–162. DOI: 10.1109/INES.2018.8523853.

Nicola, M. et al. (2017). *Scada Systems Architecture Based On Opc Servers And Applications For Industrial Process Control*. Baile Govora, Romania, pp. 222–232.

Nowaczyk, S. et al. (2018). "Monitoring Equipment Operation Through Model And Event Discovery". In: *Intelligent Data Engineering And Automated Learning Ideal 2019* 2, pp. 41–53. DOI: 10.1007/978-3-030-03496-2_6.

Oksanen, T., P. Piirainen, and I. Seilonen (2015). "Remote Access Of Iso 11783 Process Data By Using Opc Unified Architecture Technology". In: *Computers And Electronics In Agriculture* 117, pp. 141–148. DOI: 10.1016/j.compag.2015.08.002.

Ouelhadj, D., C. Hanachi, and B. Bouzouia (2000). "Multi-Agent Architecture For Distributed Monitoring In Flexible Manufacturing Systems (Fms)". In: *Proceedings 2000 Icra. Millennium Conference. Ieee International Conference On Robotics And Automation. Symposia Proceedings (Cat. No.00Ch37065)* 3, pp. 2416–2421. DOI: 10.1109/ROBOT.2000.846389.

Palluat, N., D. Racoceanu, and N. Zerhouni (2006). "A Neuro-Fuzzy Monitoring System. Application To Flexible Production Systems". In: *Computers In Industry* 57, pp. 528–538. DOI: 10.1016/j.compind.2006.02.013.

Park, J., D. Lee, and H. Bae (2014). "Event-Log-Data-Based Method For Efficiency Evaluation Of Block Assembly Processes In Shipbuilding Industry". In: *Icic Express Letters Part B: Applications* 5, pp. 157–162.

Pethig, F. et al. (2012). "A generic synchronized data acquisition solution for distributed automation systems". In: *IEEE 17th International Conference on Emerging Technologies & Factory Automation*, pp. 1–8. DOI: 10.1109/ETFA.2012.6489655.

Phaithoonbuathong, P. et al. (2010). "Web Services-Based Automation For The Control And Monitoring Of Production Systems". In: *International Journal Of Computer Integrated Manufacturing* 23, pp. 126–145. DOI: 10.1080/09511920903440313.

Polczynski, M. and A. Kochanski (2010). "Knowledge Discovery and Analysis in Manufacturing". In: *Quality Engineering* 22, pp. 169–181. DOI: 10.1080/08982111003742855.

Polyvyanyy, A. et al. (2017). "Impact-Driven Process Model Repair". In: *Acm Transactions On Software Engineering And Methodology (Tosem)* 25, p. 28. DOI: 10.1145/2980764.

Pourmirza, S., R. Dijkman, and P. Grefen (2015). "Correlation Mining: Mining Process Orchestrations Without Case Identifiers". In: *Service-Oriented Computing. Icsoc 2015. Lecture Notes In Computer Science* 9435. DOI: 10.1007/978-3-662-48616-0_15.

Qi, Q. and F. Tao (2018). "Digital Twin And Big Data Towards Smart Manufacturing And Industry 4.0: 360 Degree Comparison". In: *Ieee Access* 6, pp. 3585–3593. DOI: 10.1109/ACCESS.2018.2793265.

Rahm, E. and H. H. Do (2000). "Data Cleaning: Problems And Current Approaches". In: *Data Engineering* 23, pp. 3–13.

Raman, V. and J. M. Hellerstein (2001). "Potter'S Wheel: An Interactive Data Cleaning System". In: *Proceedings Of The 27Th Vldb Conference* 1, pp. 381–390.

Reboredo, P. and M. Keinert (2013). "Integration Of Discrete Manufacturing Field Devices Data And Services Based On Opc Ua". In: *Iecon 2013 - 39Th Annual Conference Of The Ieee Industrial Electronics Society*, 4476 –4481. DOI: 10.1109/IECON.2013.6699856.

Rogge-Solti, A. et al. (2013). "Repairing Event Logs Using Stochastic Process Models". In: *Technische Berichte Des Hasso-Plattner-Instituts Für Softwaresystemtechnik An Der Universität Potsdam* 78.

Rosen, R. et al. (2015). "About The Importance Of Autonomy And Digital Twins For The Future Of Manufacturing". In: *Ifac-Papersonline* 48, pp. 567–572. DOI: 10.1016/j.ifacol.2015.06.141.

Sauter, T. (2010). "The Three Generations Of Field-Level Networks?Evolution And Compatibility Issues". In: *Ieee Transactions On Industrial Electronics* 57, 3585 – 3595. DOI: 10.1109/TIE.2010.2062473.

Schlechtendahl, J. et al. (2015). "Making Existing Production Systems Industry 4.0-Ready". In: *Production Engineering* 9, pp. 143–148. DOI: 10.1007/s11740-014-0586-3.

Schleipen, M. (2008). "Opc Ua Supporting The Automated Engineering Of Production Monitoring And Control Systems For Information And Data Processing". In: *2008 Ieee International Conference On Emerging Technologies And Factory Automation*, pp. 640–647. DOI: 10.1109/ETFA.2008.4638464.

Schleipen, M. et al. (2016). "Opc Ua & Industrie 4.0 - Enabling Technology With High Diversity And Variability". In: *Procedia Cirp* 57, pp. 315–320. DOI: 10.1016/j.procir.2016.11.055.

Schwarz, M. H. and J. Boercsoek (2007). "A Survey On Ole For Process Control (Opc)". In: *Acs'07 Proceedings Of The 7Th Conference On 7Th Wseas International Conference On Applied Computer Science* 7, pp. 186–191.

Shimm, G. (2004). “Mining Exact Models Of Concurrent Workflows”. In: *Computers In Industry* 53, pp. 265–281. DOI: 10.1016/j.compind.2003.10.003.

Sommer, M. (2012). “Zeitliche Darstellung Und Modellierung Von Prozessen Mit Hilfe Von Gantt-Diagrammen”. PhD thesis. Ulm, Germany.

Son, S. et al. (2014). “Process Mining For Manufacturing Process Analysis: A Case Study”. In: *Proceeding Of 2Nd Asia Pacific Conference On Business Process Management*.

Song, M., C. W. Günther, and W. M. P. Van Der Aalst (2008). “Trace Clustering In Process Mining”. In: *Bpm 2008: Business Process Management Workshops* 1, pp. 109–120. DOI: 10.1007/978-3-642-00328-8_11.

Srikant, R. and R. Agrawal (1996). “Mining Sequential Patterns: Generalization And Performance Improvements”. In: *Advances In Database Technology Edbt ’96* 1057. DOI: 10.1007/BFb0014140.

Steward, D. S. (1981). “The Design Structure System: A Method For Managing The Design Of Complex Systems”. In: *Ieee Transactions On Engineering Management* 28, pp. 71–74. DOI: 10.1109/TEM.1981.6448589.

Suriadi, S. et al. (2017). “Event Log Imperfection Patterns For Process Mining: Towards A Systematic Approach To Cleaning Event Logs”. In: *Information Systems* 64, pp. 132–150. DOI: 10.1016/j.is.2016.07.011.

Theorin, A. et al. (2017). “An Event-Driven Manufacturing Information System Architecture For Industry 4.0”. In: *International Journal Of Production Research* 55, pp. 1297–1311. DOI: 10.1080/00207543.2016.1201604.

Uhlemann, T. H. J. et al. (2017). “The Digital Twin: Demonstrating The Potential Of Real Time Data Acquisition In Production Systems”. In: *Procedia Manufacturing* 9, pp. 113–120. DOI: 10.1016/j.promfg.2017.04.043.

Ungurean, I., N. C. Gaitan, and V. G. Gaitan (2014a). “Transparent Interaction Of Scada Systems Developed Over Different Technologies”. In: *2014 18Th International Conference On System Theory, Control And Computing (Icstcc)*, pp. 476–481. DOI: 10.1109/ICSTCC.2014.6982462.

— (2014b). “Transparent Interaction Of Scada Systems Developed Over Different Technologies”. In: *In 2014 18Th International Conference On System Theory, Control And Computing (Icstcc)*, pp. 476–481. DOI: 10.1109/ICSTCC.2014.6982462.

Verbeek, H. M. W, J. Buijs, and B. F. Van Dongen (2010). “Prom 6: The Process Mining Toolkit”. In: *Bpm 2010 Demonstration Track 8Th International Conference On Business Process Management Bpm’10*, pp. 34–39.

Verhulst, R. (2016). *Evaluating Quality Of Event Data Within Event Logs: An Extensible Framework*. Eindhoven, Netherlands.

Veryha, Y. (2005). “Going Beyond Performance Limitations Of Opc Da Implementation”. In: *2005 Ieee Conference On Emerging Technologies And Factory Automation* 1, p. 50. DOI: 10.1109/ETFA.2005.1612501.

Vrigant, P. et al. (2018). “Opc Ua: Examples Of Digital Reporting Applications For Current Industrial Processes”. In: *2018 International Conference On Electrical, Control, Automation And Robotics (Ecar 2018)*, pp. 305–313.

Walicki, M. and D. R. Ferreira (2011). “Sequence Partitioning For Process Mining With Unlabeled Event Logs”. In: *Data&Knowlage Engineering* 70, pp. 821–841. DOI: 10.1016/j.datak.2011.05.003.

Waljee, A. K. et al. (2013). "Comparison Of Imputation Methods For Missing Laboratory Data In Medicine". In: *Bmj Open* 3, e002847. DOI: 10.1136/bmjopen-2013-002847.

Wan, J. et al. (2017). "A Manufacturing Big Data Solution For Active Preventive Maintenance". In: *Ieee Transactions On Industrial Informatics* 13, pp. 2039–2047. DOI: 10.1109/TII.2017.2670505.

Wang, J. et al. (2015). "Cleaning Structured Event Logs: A Graph Repair Approach". In: *2015 Ieee 31St International Conference On Data Engineering*, pp. 30–41. DOI: 10.1109/ICDE.2015.7113270.

Wang, K. et al. (2007). "Review on Application Data Mining in Product Design and Manufacturing". In: *Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)* 4, pp. 613–618. DOI: 10.1109/FSKD.2007.482.

Waschull, S., J. C. Wortmann, and J. A. C. Bokhorst (2018). "Manufacturing Execution Systems: The Next Level Of Automated Control Or Of Shop-Floor Support?" In: *Ifip International Conference On Advances In Production Management Systems* 1, pp. 386–393. DOI: 10.1007/978-3-319-99707-0_48.

Weerdt, J. De et al. (2011). "A Robust F-Measure For Evaluating Discovered Process Models". In: *2011 Ieee Symposium On Computational Intelligence And Data Mining (Cidm)*, pp. 148–155. DOI: 10.1109/CIDM.2011.5949428.

Weijters, A., W. M. P. Van Der Aalst, and A. K Alves De Medeiros (2006). "Process Mining With The Heuristics Miner-Algorithm". PhD thesis. Eindhoven, Netherlands, pp. 1–34.

Weiss, S. M. et al. (2010). "Rule-based data mining for yield improvement in semiconductor manufacturing". In: *Applied Intelligence* 33, pp. 318–329. DOI: 10.1007/s10489-009-0168-9.

Wen, L. et al. (2009). "A Novel Approach For Process Mining Based On Event Types". In: *Journal Of Intelligent Information Systems* 32, pp. 163–190. DOI: 10.1007/s10844-007-0052-1.

Wilamovski, B. M. and J. D. Irwin (2016). *Industrial Communication Systems*. Vol. 2. Boca Raton, USA: Crc Press.

Windmann, S. et al. (2015). "Big Data Analysis Of Manufacturing Processes". In: *12Th European Workshop On Advanced Control And Diagnosis* 659. DOI: 10.1088/1742-6596/659/1/012055.

Wolper, P. and D. Leroy (1993). "Reliable Hashing Without Collision Detection". In: *Courcoubetis C. (Eds) Computer Aided Verification. Cav 1993. Lecture Notes In Computer Science* 697. DOI: 10.1007/3-540-56922-7_6.

Wuest, T. et al. (2016). "Machine learning in manufacturing: advantages, challenges, and applications". In: *Production & Manufacturing Research* 4, pp. 23–45. DOI: 10.1080/21693277.2016.1192517.

Yahya, B. N. (2014). "The Development Of Manufacturing Process Analysis: Lesson Learned From Process Mining". In: *Jurnal Teknik Industri* 16, pp. 97–107. DOI: 10.9744/jti.16.2.95-106.

Yang, H., B. Van Dongen, et al. (2012). "Estimating Completeness Of Event Logs". In: *Bpm Center Reports* 1204.

Yang, H., A. Hm Ter Hofstede, et al. (2010). "On Global Completeness Of Event Logs". In: *Bpm Center Report*, pp. 1–25.

Yang, H., M. Park, et al. (2014). "A System Architecture For Manufacturing Process Analysis Based On Big Data And Process Mining Techniques". In: *2014 Ieee International Conference On Big Data (Big Data)*, pp. 1024–1029. DOI: 10.1109/BigData.2014.7004336.

Yang, H., L. Wen, and J. Wang (2012). "An Approach To Evaluate The Local Completeness Of An Event Log". In: *2012 Ieee 12Th International Conference On Data Mining*, pp. 1164–1169. DOI: 10.1109/ICDM.2012.66.

Yurin, A. Y. (2012). "An Approach For Definition Of Recommendations For Prevention Of Repeated Failures With The Aid Of Case-Based Reasoning And Group Decision-Making Methods". In: *Expert Systems With Applications* 39, pp. 9282–9287. DOI: 10.1016/j.eswa.2012.02.076.

Zaki, J. (2001). "Spade: An Efficient Algorithm For Mining Frequent Sequences". In: *Machine Learning* 42, pp. 31–60. DOI: 10.1023/A:1007652502315.

Zhang, Y., Y. Lu, and B. Yang (2017). "Parsing Statement List Program Using Flex And Bison". In: *2017 First International Conference On Electronics Instrumentation & Information Systems (Eiis)*, pp. 1–4. DOI: 10.1109/EIIS.2017.8298547.

Zhang, Y., G. Zhang, et al. (2015). "Real-Time Information Capturing And Integration Framework Of The Internet Of Manufacturing Things". In: *International Journal Of Computer Integrated Manufacturing* 28, pp. 811–822. DOI: 10.1080/0951192X.2014.900874.