CONFIDENTIAL

Identifying procedural automation opportunities in the process industry using pattern mining on logged process data

C.H.C. van der Hoorn

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

MSCCONFIDENTIAL

Identifying procedural automation opportunities in the process industry using pattern mining on logged process data

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

C.H.C. van der Hoorn

September 7, 2023

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

The work in this literature review was supported by *Shell Energy and Chemicals Park Rotterdam*. Their cooperation is hereby gratefully acknowledged.

Abstract

The process industry is increasingly oriented towards automation. Use is made of both automated systems and human operators to control processes. For this control, operators can use computer systems to execute actions and conduct procedures. Automation of procedures can improve performance and safety, while reducing workload. Identifying procedures suitable for automation currently relies heavily on manual inspection and knowledge of operational staff.

This thesis proposes a method for the analysis of stored process data to identify opportunities for procedural automation in processing plants. The goal of this analysis is the identification of action patterns in operator responses to alarms that are predictable. Action patterns which are predictable can present good opportunities for automation.

The method utilizes the physical layout of the process, combined with a statistical analysis of events occurring in the event log to identify the relevant events in response to each alarm. Sequential pattern mining is then applied to those events. This thesis introduces a novel pattern type called 'independent frequent patterns' to quantify patterns in response to an alarm. The pattern mining results are condensed into a value for the predictability of the response of an alarm.

In order to ensure validity of the patterns resulting from the method, a training and test set were constructed, consisting of alarms and the response patterns known to be the correct responses to those alarms. The method was designed using the training set and validated using the test set, by applying the method to the alarms in those sets and observing if the correct response patterns were identified. Then, the method was applied to alarms in the event log of which the correct response was not yet known, to rank those alarms on their potential for automation.

The thesis is conducted in collaboration with Shell Energy and Chemicals Park Rotterdam, and the method is applied to data supplied by them. The most common alarm messages are evaluated for the response pattern required to resolve them, the predictability of that pattern and the benefit of automating the response. From this, a ranked list of automation opportunities was constructed, which was then further discussed with operational staff.

Table of Contents

Acknowledgements

I would like to express my gratitude to Prof.dr.ir. Michel Verhaegen for his valuable support and feedback throughout this thesis. Being able to have regularly occurring meetings to discuss progress and outlook of the research has helped me reach this final result.

I'd also like to express my gratitude to ir. Rutger Jacobs, my supervisor at Shell Energy and Chemicals Park, who guided me in a helpful and friendly manner during this project. I greatly appreciate the freedom I was given to choose my own direction in the project, while also being afforded all resources necessary to bring the project to a good result.

Thanks should further go to other Shell staff who have supplied me with highly necessary resources such as data and documentation of the plant, and to all staff who have supplied me with valuable insight and advice. Specific thanks goes to Barry Cott for his advisory role in this project, Brugt Douwes for his consult on automation opportunities, Niels Mulder for his help in obtaining the event logs, and Kamil Albakri for supplying the PEFS data.

I would like to express my thanks to Annelouk for offering me distraction and much needed support during stressful times.

Finally, I'd like to thank my parents. I very much appreciate their continued support during the writing of this thesis. They provided me with a quiet workspace at their home to finalize the thesis report, and their insights while reviewing the draft version of my thesis significantly improved its quality. I am truly fortunate and deeply thankful for their belief in me.

Delft, University of Technology — C.H.C. van der Hoorn
September 7, 2023

Chapter 1

Introduction

In the process industry it is common for processes to be controlled by both an automated system and by human operators in parallel [34]. Operators can control the process manually or via a Human Machine Interface (HMI) and can perform a set of actions in order to maintain process operating conditions which are safe, reliable, efficient, and ensure profitability of the process [5,8]. In many industrial processes, actions executed by both humans and automated systems, as well as process measurements, are logged for later analysis [46].

Errors made by operators can result in dangerous conditions [4]. Performance in both safety and process efficiency can differ between operators: Although Standard Operating Procedures (SOP) are often in place, the experience of operators can affect their performance when executing procedures [59]. The industry has made efforts to further automate and reduce dependency on human operators [8, 17, 47]. Improved safety performance [8, 58], mitigating loss of knowledge due to retiring operators [55, 58], and a potential increase of efficiency [8, 17, 55, 58, 59] are most cited as reasons for pursuing further automation.

In the process industry, the focus of automation has broadened from automatically controlling the plant in steady-state conditions, to include the automation of procedures performed by operators [59]. Procedures in this context are defined as "A set of operator tasks that are conducted in a set way time-after-time to achieve a certain goal such as starting or shutting down a unit or making a product" [59]. Operators perform many procedures during their work, with varying frequency. Such procedures can have the goal to correct erroneous behaviour, e.g. making the plant return to its proper operating point after an error occurred, moving the plant towards new operating conditions, or shutting down a plant. In the process industry, the implementation of computerized systems to execute sequential tasks is known as procedural automation [58].

With many procedures in place, choosing which procedures to automate is a problem requiring the consideration of multiple aspects, including productivity, reliability, safety and cost of potential solutions [6]. Finding suitable procedures to be automated and finding methods to automate them is an ongoing topic in research, not only in industry [55, 59], but also in business [2, 28], and healthcare [63].

Research Question and Scope

The digitization of industry has led to the gathering of large amounts of process data [46,47]. While in the past operators and engineers relied on their knowledge and experience to choose which procedures to automate, recent studies propose the use of this stored process data to study the efficiency of their processes and procedures, to identify gaps in their standard operating procedures and to measure operator capability [55].

The research objective of this thesis is to find a method for leveraging stored process data for the identification of procedural automation opportunities in processing plants. The formulation of this research objective originates from Shell Energy and Chemicals Park Rotterdam, a processing plant in the Netherlands, where the research was proposed as a graduation project.

The primary focus of this research is to identify series of actions executed by operators whose execution exhibits a level of predictability, where predictability is the degree to which the operator actions in response to an alarm can be anticipated or foreseen based on the patterns observed in the event logs. Based on this proposal, the following research question was formulated:

> *"How can commonly occurring action patterns of operators in response to alarms in processing plants be extracted from stored process data for the identification and prioritization of procedural automation opportunities, considering the predictability of those patterns, their frequency, and the potential benefits of their automation?"*

This research question is addressed through the development of a method which identifies predictable action series in the responses of operators to alarms, and which assesses those responses regarding their potential for automation. This method is subsequently tested and applied to the stored data within the Shell Energy and Chemicals Pakr Rotterdam. Based on this assessment, a list of procedural automation opportunities is made.

In Figure 1-1, a schematic overview is shown illustrating the interactions between operators, the control system, and the plant. As per the research question, the goal of this research is to identify the actions performed by operator in reaction to alarms being annunciated by the control system.

Document structure

In Chapter 2, the methodology for answering the research question is further outlined. Chapter 3 provides background information necessary for the results presented in later chapters. This includes key terminology, and findings from earlier research. The method developed to identify predictable patterns in plant data is explained in Chapter 4, where it is also explained how this method was conceived. Chapter 4 provides a detailed explanation of the method developed for identifying predictable patterns in plant data and presents the process through which this method was conceived. In Chapter 5, the patterns generated by the method are presented, after which they are evaluated based on their predictability, frequency, and potential benefit of automation. This evaluation leads to a ranked list of procedural automation opportunities. In Chapter 6, the results of this thesis are summarized, the contributions are stated, and recommendations for future work are made.

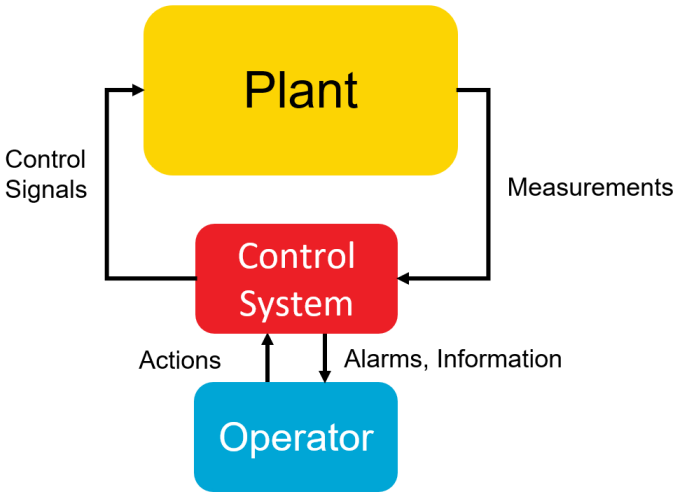

Figure 1-1: A schematic overview illustrating how operators interact with the control system, which interacts directly with the plant. The plant receives control signals from the control system, and supplies measurements to it. The Control system supplies operators with all information necessary, such as measurements and current control settings, and it annunciates alarms to the operators in case measurements are outside of their operating window. Operators can perform a number of actions to interact with the control system, and thereby change the operational mode of the plant, or resolve alarms.

Chapter 2

Methodology

The research topic of this thesis arose from a graduation project offered by Shell Energy and Chemicals Park Rotterdam. This research was conducted with their collaboration. The project description is to design a method for the identification and framing of opportunities for further procedural automation for the continuous process operations in the refinery of Shell Energy and Chemicals Park Rotterdam. Namely, the Hydrogen Converter (Hycon) and Solvent De-Asphalter (SDA) plants was investigated, overviews of which are shown in Appendices A and B respectively.

The rationalization for this project is that further automation is key in freeing operation manpower while being faced with shortage. Furthermore, automation can increase the reliability of tasks which are critical or complex. Procedural automation being the automation of tasks in processes otherwise being performed by humans offers the potential to fulfill those goals. As such, the method designed should identify automation opportunities based on the frequency, complexity and criticality of the tasks being performed.

The relative frequency and complexity of procedures will be based on both an analysis of historical data, and from conversations with operational staff in the refinery. The criticality of procedures is based on documented approaches for those procedures, and from conversations with operational staff.

2-1 Research goal

The main research goal of this thesis is designing a method to analyze historical data of the chemical plants in Shell Energy and Chemicals Park Rotterdam and to find automation opportunities. With this research goal, a literature study was conducted as a starting point for the thesis. In this, previous works were studied for methods which can be useful to the identification of automation opportunities in industry.

From this literature study, it was found that searching for frequently occurring patterns in the event log of operator actions could be a promising method to find actions which can be

automated using procedural automation. It was found that pattern recognition on alarms was attempted in previous works, but not in combination with operator actions with the goal of finding automation opportunities. Based on this, the research question stated in the introduction was introduced. This literature study was submitted as a previous deliverable.

2-2 Research plan

Based on the research question and literature study, a plan was made for the thesis research. First, the historical data of a part of the refinery will be analyzed for patterns which occur in the data. Those patterns are further analyzed for their complexity, their frequency and how predictable their occurrences are. Next, those measures are combined with the criticality of the alarms that they are in response to. This is explained in Chapter 4.

Combining those figures, a list is made of the alarm response patterns which present the most interesting automation opportunities. This is explained in Chapter 5. The most interesting automation opportunities are then discussed with operational staff to find the alarm responses that are deemed most useful to automate.

As a proof of concept, one of the alarm responses resulting from the list and discussions with operational staff is chosen and a functional description for applying procedural automation to this alarm is written. The results from the analysis of the historical data is also used in this step. Since this shows patterns of previous responses of human operators to those alarms, those patterns can be used as a starting point for writing a functional description.

2-2-1 Training and test sets

In order to design a method that can uncover relevant patterns from the dataset, it was found that a test set should be available to determine if found patterns are correct. This test set should consist of patterns which are known to be present in the dataset, and of which it is known when those patterns occur.

One clear type of pattern that is available in the data is the response of operators to alarms. A big advantage of such type of patterns is that it is easy to identify when those patterns should occur, being after the alarm is annunciated. Alarm annunciations and their time stamps are readily available in the dataset.

It was furthermore found that alarms could be interesting opportunities for procedural automation. This would be the case if we could prevent the alarm from annunciating by automating the response before the alarm threshold is reached. In such cases the workload of operators can be decreased, and the amount of upsets occurring in the plant can be reduced. A training and a test set were constructed. Both sets consist of alarm tags and sequences of the operator's responses to each of those alarms.

2-2-2 Building the method

The first step in the research project is to build a method which can uncover the operator responses to the alarms in the training set and test set. This is explained in Chapter 4, and

makes use of Sequential Pattern Mining (SPM) algorithms. It should be noted that a large advantage of such algorithms is that a measure showing how often a particular pattern is performed in response to an alarm, proportionally to how often that alarm is annunciated, is returned. This measure is called the support of a pattern, and can be used to assess how predictable the response to a particular alarm is.

The training set is used to test each individual part of the algorithm while designing the method, while the test set is used in the end to determine if the method correctly finds the response to alarms it has not seen before.

The method consist of multiple parts. Using the training set while programming each part ensures that it is known what the outcome of each particular part should be, allowing for a more effective programming process. The test set is used to prevent over-fitting the method to the training set, since the test set is used to check if the method also gives correct results on alarms that were not yet seen during programming. If the correct responses to those new alarms can be found, it is assumed that the method will also find correct responses to other alarm tags that were not seen before. This is illustrated in Figure 2-1.

The method is programmed in Python 3.10, using various additional packages, including *Numpy* [30], *Pandas* [41], and *Scipy* [53].

2-2-3 New alarms

When the method is able to return the required patterns for each alarm in the training and test set to a satisfactory degree, the method can be applied to the other alarms which the method has not yet seen before, to obtain response patterns, and their predictability and complexity. Since only the alarms with a large amount of annunciations are considered interesting as automation opportunities, the method is applied to the 500 alarms which are annunciated most often in the plant. This also reduces the processing time that the method requires.

2-2-4 Comparing automation opportunities

The responses to the different alarms are ranked as automation opportunities. This is based on the degree of predictability of the response, the amount of times this alarm is annunciated, the 'importance' of the alarm. This is further shown in Chapter 5.

Once this list is generated, the alarms showing the most promising automation opportunities are chosen. Those opportunities are discussed with operational staff of the plant in order to determine how successful the method is in finding procedural automation opportunities. As a proof of concept, one of the automation opportunities is chosen and a functional description is written for its procedural automation.

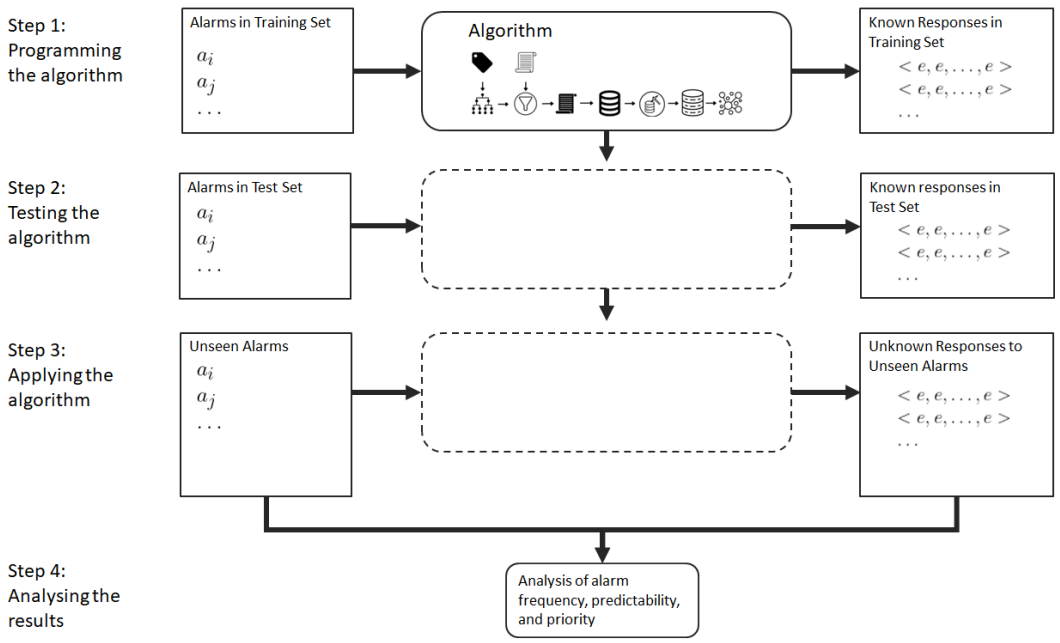

Figure 2-1: Figure showing an overview of the research plan. The research plan is broken up into four steps:
In step 1, the algorithm is designed using the alarms of the training dataset, designing each part such that it returns the required output, resulting in the known operator responses of those alarms. A more detailed overview of the algorithm is given in Figure 4-4.
In step 2, the algorithm is tested on the alarms of the test dataset, to determine if it returns the correct operator responses for alarms it has not yet seen before.
In step 3, the algorithm is applied to the set of of most often occurring alarms. Of these alarms, it is not known yet what the correct response is.
In step 4, the resulting list of alarms and the predictability of their responses is analyzed to find the most suitable automation opportunities.

Chapter 3

Literature background

This Chapter provides the background information necessary for the results presented in later chapters. First, key terminology is introduced which is used throughout this thesis. Then, the findings of earlier research which are used in this thesis are introduced and elaborated upon. These concepts are referred to in later Chapters.

3-1 Terminology

Specific terms and concepts are used in literature relating to procedures and their parts. Because the use of terms can differ between cited resources, an overview of terms used in this thesis review is given below for clarity.

Events An Event is an instantaneous occurrence which changes the process from one mode to another. In the context of the process industry, an event is associated with a tag. A tag is an identifier which references a specific item, component, piece of equipment, or process in the plant. All events have an activity associated with them, indicating how an operator interacted with the tag. The tag and activity of each event are logged, as well as the time at which an event occurred. Examples of events are incoming alarms, operator actions performed manually by operators, actions performed by operators via a Human Machine Interface (HMI), actions performed automatically by an automated system, or system messages [17].

An Event is represented by $e = (t, v, a) \in E$ and has several attributes: a time stamp $t \in \mathbb{R}^+$, a tag $v \in \mathcal{V}$, an activity $a \in \mathcal{A}_v$

$\mathcal{V}$ represent all possible tags. $\mathcal{A}_v$ represents all possible activities for tag v. For example, for an alarm with tag v_1, the set of possible activities $\mathcal{A}_{v_1}$ is equal to

$$\{\text{Alarm-Unack}, \text{Alarm-Ack}, \text{Alarm-Rtn-Ack}, \text{Alarm-Rtn-Unack}\}$$

Where an alarm can be raised, can be acknowledged, and can return to the normal state both after being acknowledged or after not being acknowledged yet.

Event logs The event log of a process $\mathcal{L} \subseteq E$ is the database in which all events that have occurred have been saved. Depending on the processes, manual actions of operators in the field or communication activities between operators may or may not be stored in the event log [31]. It's important to consider that these specific activities will not be available for analysis when reviewing the event log. This restricts the type of sequences that can be analyzed solely from the event log.

Sequences A sequence $s \in S$ is a sequential collection of a finite number of events e, and is denoted as follows:

$$s =< e_1, e_2, \ldots, e_n > \tag{3-1}$$

This follows the notation used in publications in the field of Sequential Pattern Mining (SPM), such as [25]. The sequence database S is the set of all sequences found in an event log:

$$S = \{s_i : s_i =< e_1, e_2, \ldots, e_{|s_i|} >, e_j \in \mathcal{L} \forall\, j, \} \tag{3-2}$$

Each event can only occur in one sequence of the event log, and as such $s_i \cap s_j = \emptyset, s_i, s_j \; in S$. There are several methods for constructing the sequence database from the event log, some of which are shown in Section 3-2-2. The method designed specifically for this thesis is shown in Chapter 4.

Subsequences A sequence s_a is a subsequence to sequence s_b if and only if sequence s_a is contained in sequence s_b. Sequence $s_a =< a_1, a_2, \ldots, a_n >$ is contained in sequence $s_a =< b_1, b_2, \ldots, b_m >$ if and only if integers $1 \leq i_1 \leq i_2 \leq \cdots \leq i_n \leq m$ exist, such that $a_1 = b_{i_1}, a_2 = b_{i_2}, \ldots, a_n = b_{i_n}$. This is from here on out denoted by $s_a \sqsubseteq s_b$ [25].

As an example, $s_a =< a, b, d >$ is a subsequence of $s_b =< a, b, c, d >$, while $s_c =< a, c, b >$ is not.

Patterns In the current context, a pattern is a sequence of events which occur a minimum amount of times in the sequence database S. This is the definition as used in the field of SPM [25]. While patterns are also called *frequent sequence* in literature [25], the term pattern is used throughout this document.

A sequence of events is said to be a pattern p if, and only if, $sup(p) \geq minsup$, where $sup(p)$ is the amount of sequences $s \in S$ of which p is a subsequence, and where $minsup$ is a threshold set by the user of the algorithm [25].

The set P is the set of all patterns in a sequence database S:

$$P = \{p : |\{s : s \in S, p \sqsubseteq s\}| \geq minsup\} \tag{3-3}$$

To give an example, Table 3-3 shows a sequence database S and the resulting pattern database P, where each pattern p is a subsequence to at least $minsup = 2$ sequences in S.

3-2 Alarm reduction

In the process industry, operators are alerted of potentially problematic process conditions by alarms, for example when normal operation limits are exceeded [50]. However, there are alarms which are not useful or are redundant, e.g. because they are repeated excessively in a short amount of time - called alarm chattering - or otherwise supply no new information to the operator. Such alarms are known as nuisance alarms [39]. Alarm floods occur when many alarms sound in rapid succession, often as consequence of each other [36]. A large amount of alarms can overwhelm an operator, reducing their ability to properly control the process.

Various methods of reducing nuisance alarms in industrial applications have been researched and proposed. Simpler methods focus on individual alarms, while more complex methods target instances of multiple alarms occurring in succession. The latter do, for example, so by finding patterns (as described in the previous Section) in the incoming alarms. The application of this to alarm floods is described in this Section. Further elaboration on SPM and its implementation is elaborated on later.

3-2-1 Alarm deadbands and time-delays

Chattering or repeating alarms are alarms which repeatedly make the transition between alarm and non-alarm states. Chattering alarms are the most common nuisance alarms encountered in industrial plants [56]. Methods for reducing such alarms is a topic of active research.

In [32], a method which uses *deadbands* is shown. Once a measurement is in the alarm state, an alarm deadband requires the measurement to move an additional amount back into nominal operating conditions before the alarm will clear. This will reduce repeating of alarms for measurements which fluctuate around the alarm threshold. A different method to reduce alarms is by setting a minimum time during which the measurement must be outside of nominal operating conditions before an alarm will sound. This method is referred to as *time-delay* [7]. Both methods can be useful in reducing repeated alarms and events which need to be considered in searching for frequently occurring sequences.

3-2-2 Sequence mining

The deadband and time-delay methods described above consider each alarm individually. They do not reduce alarms during alarm floods in the case alarms are a consequence of each other. Alarms which are highly predictable are assumed to not supply new information to operators, and preventing them from annunciating can reduce the mental load on operators [36]. Industrial companies still rely mostly on expert consultation to find patterns when attempting to reduce alarm floods. Using data mining techniques to detect patterns in alarm floods is proposed to be used to find predictable alarms [14, 16, 36].

In [36] the similarity between sequences of alarms is measured using the Smith-Waterman algorithm to calculate a similarity index between different alarm flood sequences. This allows for similar patterns to be identified without them needing to be exactly equal. Such similarity measures can be used for clustering of similar sequences. Methods for measuring similarity between sequences are shown in Section 3-7-1.

In [14], sequential pattern mining is applied to alarm data in order to find sequences of alarms which are related to one another, with the goal of reducing alarm floods. Sequential Pattern Mining is further explained in Section 3-6 The authors use a dataset of alarm data of a vinyl acetate production unit, which they made available online.

The authors of [16] recognize that while finding frequently appearing patterns in industrial alarm data is possible, many spurious sequences may be generated. In [16], the sequence mining algorithm from [14] is enhanced by using hierarchical constraints which arise from the physical topology of the process, as is explained in Section 3-3.

3-3 PEFS linking

Process Engineering Flow Schemes (PEFSs) are diagrams of the physical layout of the process, showing piping and process equipment together with the instrumentation and control devices of a process. As such, a PEFS shows which measurement equipment is closely related in the physical process. This is one *rule* on which it could be assumed that two alarms have the same cause, and as such can be together in the same sequence. For example, if two pressure alarms have a physical process path connecting them, and sound a *low* alarm in a short time interval, they may be likely to have the same cause [50]. In [50] it is proposed to make use of PEFSs to infer relationships between alarms and thereby reduce the number of annunciations in an alarm flood.

Grouping alarms based on PEFSs is done in [50] by comparing alarms based on a combination of up to four properties. First, the alarm type is taken into account. Second, the time frame of two alarms is considered, taking into account that a process has dynamics such that two related alarms may not have the exact same time stamp. Next, the alarm status may be used when applicable: E.g. in the case of two related temperature alarms, it is likely that both give a *high* status. It is however unlikely that one alarm gives a *low* and the other a *high* status. Fourth, the connection type between two alarms is taken into account, which can either be physical connections via pipes and equipment, or connections through signals from the process control system. Equipment among the path between alarms may break the connection: E.g. a pump can divide the process into two zones between which pressure alarms may not be related.

In addition to analyzing alarms, this approach has the potential to incorporate operator actions of a process for preprocessing event data. Sequential pattern mining techniques - as explained in Subsection 3-6 - can then be enhanced by decreasing the amount of irrelevant frequent patterns that appeared by chance, and thereby reducing the amount of noise in the data. Since the processes in Shell Energy and Chemicals Park Rotterdam involve many different events and alarms, applying this method proved to be effective. The results are shown in Section 4-4.

3-4 Chi-squared test

In addition to the PEFS linking method outlined in the previous Section, a secondary approach to minimize the number of tags considered during the SPM step is used. This is shown

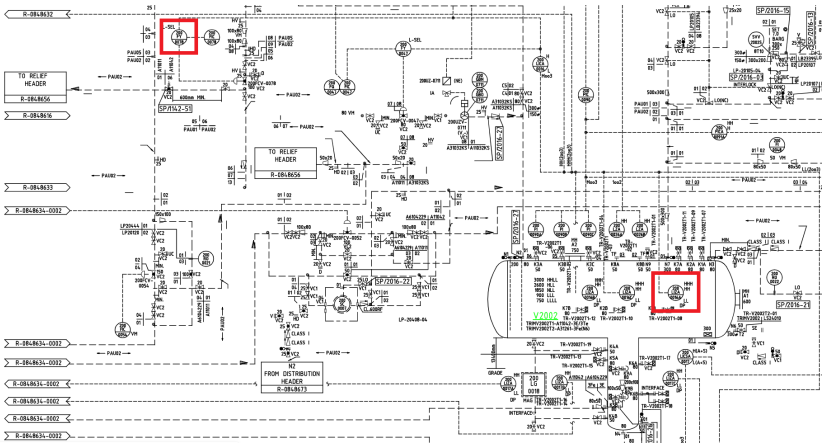

Figure 3-1: A part of an example PEFS showing two alarm tags marker in red which are physically connected via piping and equipment [20]. The method in [50] assumes that these alarm tags are more likely to be related than tags which are for example not on the same PEFSs.

in Section 4-4, and relies on the utilization of the Chi-Squared test of independence. This test is introduced here.

The Chi-Squared (χ^2) test of independence is a statistical method used to determine if the occurrence of two categorical variables is independent among a sample distribution [64]. As an illustrative example, the Chi-Squared test can determine if the the political choice of a sample population is statistically significantly correlated with their gender.

Categorical variables In the Chi-squared test, the observed frequencies of categorical variables from a number of samples are analyzed in order to determine if a statistically significant correlation between different categories of the samples is present.

For the illustrative example, a sample dataset of the political choice of 1000 people is given. Each person is either male and female, and can vote for either party A or B. The data is formatted in what is named a *contingency table*, shown in Table 3-1.

Table 3-1: Contingency table of the sample dataset, showing the political choice and gender of a population of 1000 people.

	Party A	Party B	Total
Male	295	251	546
Female	275	179	454
Total	570	430	1000

Hypotheses The Null Hypothesis H_0 in the Chi-Squared test states that the categorical variables are independent, meaning that the category of one of the variables in an observation does not provide any information about the category of the other variable. In terms of illustrative example, the Null Hypothesis would imply that gender and preference between the two political parties is not proven to be correlated.

The alternative Hypothesis H_1 states that there is an association between the two variables. In the illustrative example, it would imply that gender and preference between political parties is are related.

p-value Prior to conducting the Chi-Squared test, a significance level denoted as the p-value is chosen. The p-value represents the probability of observing an outcome at least as extreme as the outcome observed, while assuming that the null hypothesis holds true.

The Chi-Squared test results in a p-value, indicating how likely the resulting observations would have been under H_0. If the resulting p-value is small, it suggests that the observed data are highly unlikely to have occurred if the null hypothesis were true. Before conducting the Chi-squared test, a significance level must be chosen such that if the resulting p-value is above this value the Null Hypothesis is accepted, otherwise it is rejected. For the example, significance level $\alpha = 0.05$ is chosen, a commonly chosen value.

Expected frequencies The next step in the Chi-Squared test is determining what the expected frequencies of each of the categories are under the assumption that the categories are independent [64]. If $O_{i,j}$ represents the amount of observations that fall in category i for the first variable and in category j for the second variable. Then the total count for category i in the first variable is $T_{1_i} = \sum_j O_{i,j}$, the total count for category j in the second variable is $T_j = \sum_{2_i} O_{i,j}$ and the total amount of observations is $T = \sum_{i,j} O_{i,j}$.

Using this notation, the following Equation shows the expected amount of observations which fall in category i for the first variable and in category j for the second variable, under the assumption that categories are independent.

$$E_{i,j} = \frac{T_i T_j}{T} \tag{3-4}$$

Returning to the example, the following Table 3-2 shows the expected count for each combination of categories. It can be seen that the male to female proportion is equal for each political preference. Furthermore, the Party A to Party B proportion is equal for each gender. This is expected if the gender and political preference were independent [64].

Table 3-2: Expected counts for each of the categories of the example data set, under the assumption that the categories are independent.

	Party A	Party B	Total
Male	311.2	234.8	546.0
Female	258.8	195.2	454.0
Total	570.0	430.0	1000.0

Chi-squared statistic The Chi-Squared value is calculated based on the observed and expected counts of the sample. The following Equation gives the Chi-Squared value, which for the example dataset results in $\chi^2 = 4.33$.

$$\chi^2 = \sum \frac{(O_{i,j} - E_{i,j})^2}{E_{i,j}} \tag{3-5}$$

The further the observed counts lie from the expected counts, the larger the Chi-Squared value is. The Chi-Squared value is then translated to a p-value by using the Cumulative Distribution Function of the Chi-Squared Distribution, shown in Equation 3-6. In it, d is called the *degrees of freedom*, which depends on the amount of categories for each variable: $d = (n_{\text{rows}} - 1)(n_{\text{cols}} - 1)$ [64].

$$F(x, df) = \frac{\gamma\left(\frac{d}{2}, \frac{x}{2}\right)}{\Gamma(d/2)} \tag{3-6}$$

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt \tag{3-7}$$

$$\gamma(s, x) = \int_0^x t^{s-1} e^{-t} dt \tag{3-8}$$

$\Gamma(\cdot)$ is the *gamma function* shown in Equation 3-8, and $\gamma(\cdot, \cdot)$ is the *lower incomplete gamma function* shown in Equation 3-7. The p-value is calculated according to the following equation:

$$\text{p-value} = 1 - F(\chi^2, df)$$

For the example dataset, with $\chi^2 = 4.33$ and $df = (2-1)(2-1) = 1$, this results in a p-value of 0.0374. This means that, if the null hypothesis were true, the probability of obtaining these or more extreme observations is 0.0374. With the significance level $\alpha = 0.05$, the null hypothesis is rejected, and it is concluded that a correlation between gender and political preference can be found in this data.

3-5 Sequence extraction

Depending on the system, $\mathcal{L}$ contains all events of the process over a long time period, without identifiers indicating which events are associated with which abnormalities [31]. Before the process can be analyzed using Sequential Pattern Mining (SPM) (described in the next Section) it is required that separate sequences within event log $\mathcal{L}$ are generated. This separation must be performed in a manner that splits the event log into sequences of which the events occurred in relation to each other. In the context of procedural automation, such sequences are events which were all related to the same procedure. For example, an incoming alarm and all the following actions required to resolve this alarm. Separating the event log into sequences is a recognized problem in literature [1, 29, 31, 54]. Multiple methods have been proposed to separate event logs under different assumptions.

In [17], [15], and [1] a similar method is used. The method is based purely on the moment events occur. In Figure 3-2, this method is illustrated.

First, the log is segmented into operator action series using a segmentation window of predetermined length π. If two events e_1 and e_2 are operator actions, and $|t_1 - t_2| < \pi$, then e_1 and e_2 are said to be coherent. Between coherent operator action series there is a period without any operator actions longer than the segmentation window. Second, a trace window of length

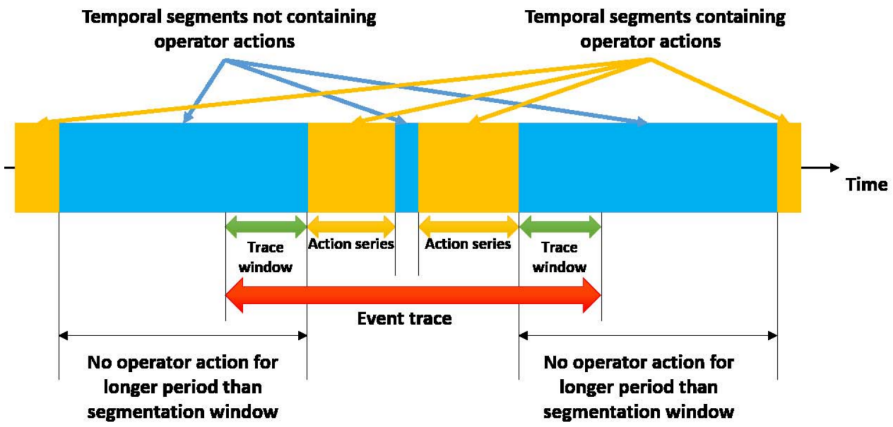

Figure 3-2: The segmentation of an event log into sequences of alarms and operator actions using a temporal method [17].

σ is used to determine what alarms and events lead up to the action series of the operator, and what the effect of the operator actions is. If $|t_1 - t_2| < \sigma$, then events e_1 and e_2 are in the same trace. As such, all actions within the operator action series, and all alarms and events outside of the action series but within the trace window are combined to form a single 'event trace': a single sequence of events assumed to belong together. This is illustrated by the red line in Figure 3-2. The length of both the segmentation and trace window depends on the process and is determined based on knowledge of process engineers [17].

In [31], sequences are constructed from an event log by simply assuming that a sequence starts with an incoming alarm and ends when this alarm is no longer active. Furthermore, a manual selection of events that will be used for the analysis is made by using process knowledge. The log is split into these different sequences under the further assumption that sequences are not overlapping in time and that no two sequences share any events.

In Section 4-5, the method sequence extraction designed for this thesis is introduced, which combines elements from the methods described above.

3-6 Sequential pattern mining

Sequential Pattern Mining (SPM) is a data mining technique which aims to discover patterns of events or items within a dataset. As was stated in the beginning of this Chapter, a pattern is a sequence of events which occur a minimum amount of times in the sequence database. Multiple types of patterns exists, the most important of which are explained in Subsection 3-6-1. By analyzing the sequence database and searching for patterns, SPM can uncover insights in underlying trends, dependencies, and behaviors present in the data.

This technique finds applications in various fields, such as market basket analysis, customer behavior analysis, process optimization, and recommendation systems [12, 25, 40, 48]. events and items can be actions performed by users over time, purchases made by customers, or interactions in a process. This can enable businesses and researchers to make decisions based on discovered patterns in the sequence database [25].

Table 3-3: List of sequences and list of corresponding patterns with $minsup = 2$. The patterns are labeled closed (C) or maximal (M) where applicable.

(a) An example sequence database S.

Sequence number	Sequence
1	<a,b,c,d>
2	<a,c,b,d>
3	<a,c,f,g,e>

(b) The resulting pattern database P, resulting from sequence database S.

Pattern	Support	Type
<a>	3	C
<a,b>	2	
<a,b,d>	2	C,M
<a,c>	3	C
<a,c,d>	2	C,M
<a,d>	2	
<b>	2	
<b,d>	2	
<c>	3	C
<c,d>	2	
<d>	2	

As stated in a Section 3-2, [14] and [16] utilize Sequential Pattern Mining in order to find sequences of alarms which are related to one another, with the goal of reducing alarm floods. By finding patterns in alarm data, the authors attempt to identify alarms of which their annunciation can be predicted based on alarms which occurred earlier. In such cases, it is possible that operators already expect those alarms to annunciate, meaning that their annunciation adds little information. Suppressing those alarms can therefore reduce nuisance alarms in alarm floods.

Objective The objective of applying this algorithm is to find patterns in the actions of operators which are highly predictable. In this context, that translates to patterns that have a high support [33]. In this thesis, Sequential Pattern Mining will be applied to both alarm events and operator action events of a processing plant. The method of doing so is explained in Chapter 4.

Various algorithms exist for the mining of patterns [22, 23, 25]. The differences in algorithms are among the types of patterns that are found, as described in Subsection 3-6-1, and their search strategy. Two different algorithms are explained in Subsection 3-6-2.

3-6-1 Pattern types

In the context of SPM multiple pattern types exist. Some popular pattern types are frequent patterns, closed frequent patterns, and maximal frequent patterns [3, 26, 57]. These pattern types are explained below. An example database and an overview of different pattern types resulting from that database are shown in Tables 3-3a and 3-3b respectively.

Frequent patterns

The most basic type of sequential patterns are frequent patterns. These are any sequence of events which occurs at least in a minimum amount of sequences in the sequence database. The amount of times a pattern occurs is called the support of the pattern, while this minimum amount is called the min supp [25].

Closed frequent patterns

Closed sequential patterns are patterns which are not strictly included in longer patterns having equal frequency. In Table 3-3b, such patterns are marked with a "C".

While the set of closed frequent patterns is smaller than the set of frequent patterns, the complete set of frequent patterns can be derived from all closed frequent patterns, and vice-versa. As such, the set of closed frequent patterns can serve as a smaller representation of the set of all frequent pattern [9].

Maximal frequent patterns

"A maximal sequential pattern is a closed pattern that is not strictly included in another closed pattern." [26]. This means that removing any closed pattern that is a subsequence of another closed pattern from the pattern database results in the set of maximal frequent patterns.

The set of closed frequent patterns cannot be derived solely from the set of maximal frequent patterns, since it is not known what the supports are from the patterns which are subsequences of the maximal frequent patterns.

3-6-2 Sequential pattern mining algorithms

Three algorithms are shown below, which respectively mine either Frequent Patterns, Closed frequent Patterns, or maximal frequent patterns.

Apriori Algorithm

The Apriori Algorithm as shown in [3] presents a basic approach to mining all frequent patterns from a sequence database. The algorithm is explained below and pseudo code is presented in Algorithm 1. The set P_k is the set of all frequent patterns of length k.

First, the set P_1, which is equal to the set of frequent events, is extracted from the sequence database S. A frequent event $e \in P_1$ is an event which occurs at least *minsup* amount of times within the sequences in the sequence database:

$$P_1 = \{e_i : |s_j : S_j \in S_j| \geq minsup, s_j \in S\}$$

Logically, only events which occur at least *minsup* times can occur in a frequent pattern. Next, all frequent events that were found are extended by combining them again with all frequent events to form a candidate patterns c:

$$C_2 = \{< e, e > \ \forall\, e \in P_1\}$$

It is then checked for each candidate pattern c_2 if it is frequent in the sequence database S, i.e. if it occurs at least *minsup* times. If not, the pattern is ignored. If a pattern is frequent, it is added to a list of frequent sequences P_2. This process of extending patterns with new events to create candidate patterns is recursively performed until no new frequent sequences can be found:

$$C_k = \{< p_{k-1}, e > \ \forall\, p_k \in P_{k-1},\ e \in P_1\}, k \neq 1$$

$$P_k = \{c \in C_k : c \geq \mathit{minsup}\}$$

Pseudo code for the Apriori Algorithm is shown in Algorithm 1, and was adapted from [3]. This type of method is referred to as a candidate generation-and-test approach, and is found to become slow for larger datasets [45]. Other algorithms were designed to improve performance on larger datasets, and are presented below. Furthermore, the Apriori Algorithm returns all frequent patterns, while the algorithms below are designed to return only closed frequent patterns or maximal frequent patterns.

Algorithm 1 Pseudo code for the Apriori Algorithm, adapted from [3]

```
P_1 = {e_i : |s_j : S_j ∈ S_j| ≥ minsup, s_j ∈ S}
for (k = 2, L_{k-1} ≠ ∅, k++) do
    for (i = 0, i < |L_{k-1}|, i++) do
        for (j = 0, j < |L_1|, j++) do
            C_k.append(< p_i, e_j >)
    L_k = ∅
    for all (c ∈ C_k) do
        for all (s ∈ S) do
            if c ∈ s then
                c.count++
        if c.count ≥ minsup then
            L_k.append(c)
```

BIDE+

While the Apriori Algorithm finds all frequent patterns, the BIDE+ algorithm discovers closed frequent patterns [57]. Mining closed frequent sequences instead of all frequent sequences has been found to result in more efficient algorithms [43, 61, 62]. Furthermore, the set of all frequent patterns can be compressed to closed frequent patterns without any information

loss, and the set of frequent patterns can again be derived from the set of closed frequent patterns [60].

The BIDE+ algorithm was introduced in [57]. The authors recognize that algorithms such as the Apriori Algorithm have poor scalability in the number of frequent patterns, since the large number of frequent patterns and candidate patterns occupy a large amount of memory, while also resulting in a large search space for the checking of candidate patterns.

The BIDE+ algorithm is similar to the Apriori Algorithm, but employs various strategies to improve efficiency. First, while the Apriori Algorithm grows each frequent pattern with all frequent events, the BIDE+ algorithm grows a pattern only with events which are locally frequent in that pattern. An event e_l is locally frequent in a pattern p with sequence database S if the event occurs in at least *minsup* amount of sequences times after the pattern p:

$$e_l = \{e : |\{s :< p, e > \sqsubseteq s, s \in S\}| \geq minsup\}$$

For each candidate pattern p_c, it must be checked if the sequence is closed. This is simply done by checking if there are locally frequent events e_l which have a larger support than the pattern p_c. If there are, the pattern is not closed. If there are not, the sequence is closed.

In order to reduce the amount of times that the above operations need to be executed, patterns which cannot be extended to form closed sequences are identified by further analysis of the locally frequent events, and they are not further investigated, resulting in an algorithm which finds closed frequent patterns efficiently for lower support values. Furthermore, BIDE+ demonstrates linear scalability in both runtime and memory usage as the number of sequences in the sequence database increases [57].

VMSP

In [26], an algorithm called VMSP (Vertical mining of Maximal Sequential Patterns) is proposed to mine only the set of maximal frequent patterns, whose meaning was explained in the previous Subsection. It is similar to the Apriori Algorithm, but instead of keeping a list of all patterns that can be found in the database, a structure Z is kept. This structure contains all maximal patterns found up until now. When a new pattern s_a is generated and found frequent, it is compared with each maximal pattern $s_b \in Z$ to determine if a pattern s_b exists such that s_a is contained in s_b. If this is the case, then s_a is not maximal and is not inserted in Z. If this is not the case, then s_a is maximal up until this point, and is inserted in Z. When a new s_a is inserted in Z, then the pattern s_a is again compared with each pattern $s_b \in Z$, each s_b which is contained by s_a is removed from Z. [26] proposes several optimizations to reduce the amount of comparisons that need to be made.

3-7 Sequence clustering

As shown in the previous Section, applying Sequential Pattern Mining (SPM) to a sequence database results in a list of patterns that are present in the database. As stated before, the support of a pattern is in this context a measure for its predictability.

The response of an alarm which consists of a small set of very similar patterns can still be suitable for automation. This is because automating a response consisting of very similar patterns is generally less complex than automating responses that involve significantly different patterns [55].

As such, it was found that if an alarm has several responses which are very similar, their predictability should be based on the combined support of those patterns. For instance, two patterns which differ only with the order of two events being switched should not diminish the overall predictability of the resulting responses, since some operators may prefer to do perform one action first, and others may prefer to perform another first. The supports of these closely related patterns should be combined to form a single measure of predictability. This ensures that the overall predictability metric encompasses all relevant patterns. this is further explained in Section 4-7.

In the subsections below, several methods for comparing sequences are shown. These methods will be considered for the combination of the support of similar patterns in Section 4-7.

3-7-1 Sequence similarity

The Needleman-Wunsch and Smith-Waterman algorithms are a dynamic programming algorithms for the alignment of sequences [38,52]. An inherent feature of both methods is that they result in the optimal alignment of various sequences, meaning that sequences are arranged such that they are as similar as possible, as is shown in Table 3-4b. The Needleman-Wunsch algorithm is explained here. The algorithm guarantees an optimal alignment by considering possible alignments and efficiently calculating the scores using dynamic programming. It is commonly used in bio-informatics to align pairs of sequences, such as DNA or protein sequences. It aims to find the optimal alignment by maximizing a similarity score based on specific match, mismatch, and gap penalties.

Table 3-4: An example of a Needleman-Wunsch [38] scoring table in which two sequences are aligned, and their alignment is scored. The alignment score is equal to the value in the bottom right of the table, resulting in an alignment score of 11. .

(a) The scoring table for the Needleman-Wunsch sequence allignment method.

		sequence 2						
		-	a	b	d	e	f	g
sequence 1	-	**0**	-1	-2	-3	-4	-5	-6
	a	-1	**2**	1	0	-1	-2	-3
	b	-2	1	**4**	3	2	1	0
	c	-3	0	3	**2**	1	0	-1
	d	-4	-1	2	**5**	4	3	2
	e	-5	-2	1	4	**7**	6	5
	f	-6	-3	0	3	6	**9**	8
	g	-7	-4	-1	2	5	8	**11**

(b) The two original sequences and their resulting alignment resulting from the scoring table.

	Original Seqs.	Alligned Seq.
Seq. A	a,b,d,e,f,g	a,b,-,d,e,f,g
Seq. B	a,b,c,d,e,f,g	a,b,c,d,e,f,g

The algorithm makes finding the optimal alignment a dynamic programming problem in the form of a scoring matrix, such a matrix is shown in Table 3-4a. The matrix is initialized

with scores for aligning gaps and the first sequence against the second sequence. Then, for each cell in the matrix, the algorithm calculates the score based on three possibilities: a match/mismatch, a gap in the first sequence, or a gap in the second sequence.

To compute the score for each cell, the algorithm considers the scores from adjacent cells and determines if a match/mismatch or gap penalty results in the largest score. This process continues until the entire scoring matrix is filled, after which the path resulting in the highest alignment score is chosen. This is shown in Table 3-4.

3-7-2 Damerau-Levenshtein

The Damerau-Levenshtein [37] method compares two sequences by finding the shortest path to change the first sequence into the second. The algorithm allows for removal, addition, substitution and swapping of characters (or events) in the sequence. The amount of operations required to change one sequence into the other is called the edit distance d_e. The lower the edit distance, the more similar the two sequences are.

In Table 3-5, the steps to change sequence A into sequence B are shown. The edit distance is in this case $d_e = 3$.

Table 3-5: An example of the steps for the Damerau-Levenshtein algorithm, the steps for changing a sequence A into a sequence B are shown, resulting in an *edit distance* d_e of 3.

	Description	**Sequence**
	Sequence A	$< a, b, c, d, e, f >$
Step 1	Remove f	$< a, b, c, d, e >$
Step 2	add g	$< a, b, c, d, e, g >$
Step 3	swap c and d	$< a, b, d, c, e, g >$
	Sequence B	$< a, b, d, c, e, g >$

3-7-3 PCA

Principal Component Analysis (PCA) is a technique to reduce the dimensionality of large data sets. The method transforms data represented by a large set of parameters into a a dataset represented by fewer parameters, in a way that the maximum amount of information is preserved [27].

PCA attempts to find directions (called principal components) that maximize the variance of the data in a limited dimensional space, and to project the original data on that space. The first principal component is the direction that accounts for the most variance, the second principal component is the direction that accounts for the second most variance, and so on. By doing this, PCA can capture the main patterns and structures of the data, and project those on a lower dimensional space, while discarding potential noise and redundancy [27].

In order to apply PCA, each observation in the data must be in the form of a vector of equal length containing numerical data. In the bio-informatics field, use is made of PCA [44] to cluster similar DNA sequences together [35]. DNA sequences consist of letters A, C, G, T, and gaps $-$, and have the following form:

$$s_{dna} < C, G, \ldots, T >$$

Such sequences are not vectors containing numerical data, but rather sequences of categorical data. In order to use PCA, they must first be transformed. In [35], the letters are first one-hot encoded [27], where each letter or gap is turned into a vector: $A = [1, 0, 0, 0, 0]$, $C = [0, 1, 0, 0, 0]$. Each sequence is then turned into a vector by combing the one-hot encoded letters, each of those vectors represents a single observation for the PCA algorithm:

$$v_{dna}^{T} = \begin{bmatrix} 0 & C & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \ldots & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Chapter 4

Response pattern searching algorithm

In this Chapter, the results to the research question *"How can patterns in actions of operators in a processing plant be extracted from stored process data for the identification and prioritization of procedural automation opportunities, considering the predictability of those patterns, their frequency, and the potential benefits of their automation?"* are presented. A method was constructed which analyzes alarm responses of operators one by one, giving the most common operator response to each alarm as an output. There are multiple steps to this method, First, the operator actions which are likely to be relevant to the resolving of the alarm are determined, then sequences of operator actions after alarms occur are extracted from the event log. After this, patterns in those sequences are found. Those patterns are analyzed for the most relevant responses per alarm. Finally, the different alarm responses are compared in order to find those responses which are most suitable to be automated. The following section will first present the event logs named in the research question. In Section 4-2, the training and test datasets are explained. In the Sections thereafter, the method is explained. For each of the steps in the method, the results of an example alarm are shown.

4-1 Data set

An event log $\mathcal{L}$ is assumed to be the collection of events recorded by the control system of the process plant, containing all alarm annunciations, and all actions performed by operators. An event is represented by $e = (t, v, a) \in E$ and has several attributes: a time stamp $t \in \mathbb{R}^+$, a tag $v \in \mathcal{V}$, and an activity $a \in \mathcal{A}_v$, where the set of possible activities $\mathcal{A}_v$ depends on tag v. For example, alarm events are all events directly associated with alarm tags. The activities for those alarms are shown in Table 4-1.

The log available for this project consists of the events of the Hydrogen Converter (Hycon) and Solvent De-Asphalter (SDA) in the Energy and Chemicals Park Rotterdam of Shell. Overviews of both plants are shown in Appendices A and B respectively. The data was recorded over the period from 2019-12-08 to 2023-03-23 and supplied by Shell.

Table 4-1: All possible activities $a \in \mathcal{A}_v$ when v is an alarm tag.

Activity	Description
Unack_Alm	Alarm is sounded and is not yet acknowledged
Unack_Rtn	Returned to normal, without having been acknowledged by operator
Ack_Alm	Alarm was acknowledged by the operator
Ack_Rtn	Returned to normal, after having been acknowledged by operator

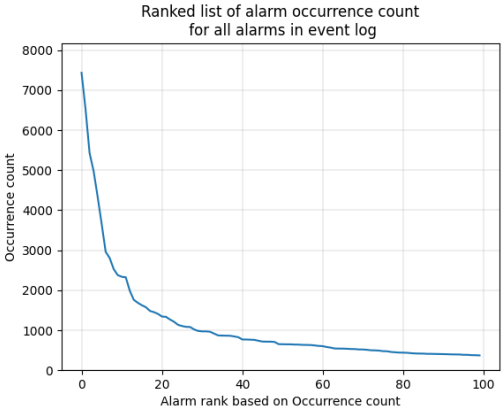

Figure 4-1: Total occurrence count per alarm for the 100 most occurring alarms.

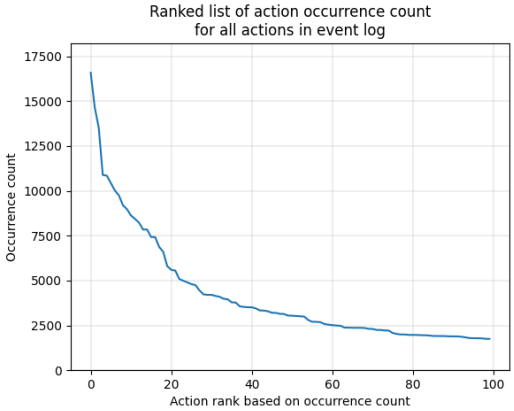

Figure 4-2: Total occurrence count per action for the 100 most occurring actions.

There are 2266 different alarms in the supplied event log. In Figure 4-1 the total occurrence count of the 100 most frequently occurring alarms is shown. Action events are actions performed by operators via a Human Machine Interface (HMI). Those actions can change parameters in the control system of the plant. The set of all actions in the event log is much larger than the set of all alarms: there are 28646 different actions in the event log. In Figure 4-2, the occurrence counts for 100 most frequently occurring actions are shown.

4-2 Training and Test Data Set

As stated in Chapter 2, a training and test data set were constructed to be used for designing and testing the algorithm. Each of these sets consists of the alarm tag which have a reference operator response which is known to be the required response to resolve those alarms. The sets were constructed by using plant documentation and by consulting with operational staff.

The algorithm is developed such that by using a particular alarm tag as an input, an operator response is given as the output. Using the training set, it can be tested if the algorithm correctly performs each step such that for an alarm in the training set, the correct output is given, eventually leading to the reference operator response. This allows for a more targeted approach while designing the algorithm, and allows the quality of the results to be determined.

The test set is then used in a similar fashion to test if the method works sufficiently well after being finished. From this it is assumed that the method also works on alarms which were not yet seen during the design of the algorithm. This allows the algorithm to be applied to the entire alarm database, resulting in operator responses and their predictability.

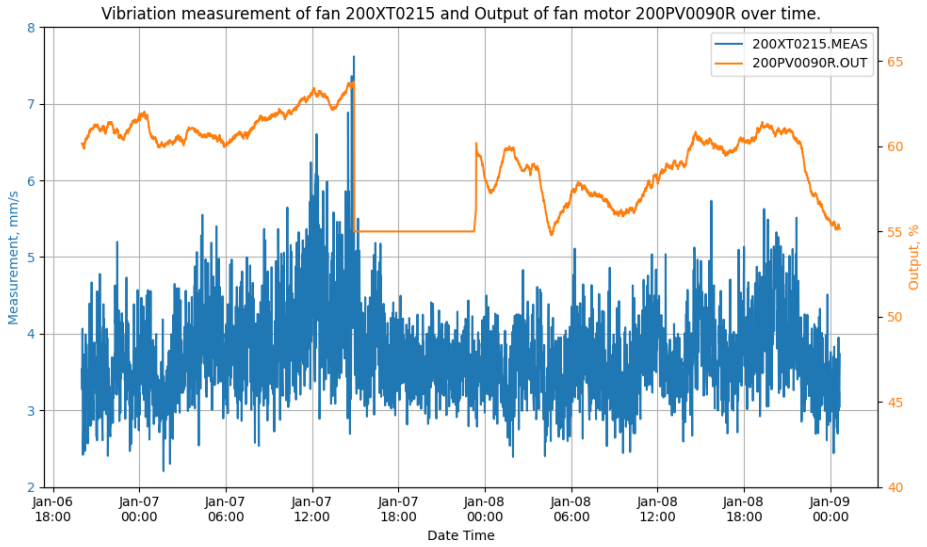

Figure 4-3: The vibration measurements 200XT0215 of the K2011H fan, and the output of fan motor 200PV0090R over time. In the time range, an occurrence of the 200XT0215 measurement exceeding its alarm limits occurs, and subsequent operator actions can be identified.

Reference Operator Responses

For 20 alarms, the responses of operators to resolve those alarms were documented. In based on documentation and on consultation with several operators, these alarms were chosen and their respective operator responses were formulated. In Appendix C, the alarm tags and responses are shown.

In Table 4-2, an example is shown of alarm *"200XT0215.HIABS"*. This is an alarm that annunciates when large vibrations are registered in one of the fans in the plant. In the documentation of the plant, a description on possible methods for resolving this alarm is presented to operators. The described response for this particular alarm is *"Reduce the speed of the fan if possible and see if this reduces vibrations."* [19]. In Figure 4-3, a occurrence of this procedure is shown. At around January 7th 15:00, the vibration measurements exceed their limits, and an alarm is annunciated to the operator. The operator subsequently takes the 200PV0090R fan in manual control, and gives it a lower output. Some time later, around Jan 7th 23:00, the fan is placed back into automatic control.

4-3 Method Overview

In this section, an overview of the method for extracting patterns in the alarm response of operators is presented. The found patterns are then further analyzed for the alarms whose responses yield the best automation opportunities. The different stages of the method are shown, it will be explained how they were designed, and the analysis results are shown for example single alarms.

The method of analyzing a particular alarm consists of multiple different stages. In Figure 4-4, an outline is shown of these different stages. Starting with the alarm and event logs, first

Table 4-2: Table showing the known alarm response to alarm *"200XT0215.HIABS"*. a) the alarm is annunciated by the system, b) the alarm is acknowledged by the operator, c) the fan controller (*"200PV0090R"*) is taken into manual operating mode, allowing the operator to manually change its workload. d) The operator reduces the fan workload. e) The alarm is resolved. f&g) the fan is taken back into automatic operating mode, where the Distributed Control System (DCS) determines its workload.

ID	Event
a	200XT0215.HIABS.UNACK_ALM
b	200XT0215.toACK
c	200PV0090R.MA
d	200PV0090R.OUT.DEC
e	200XT0215.HIABS.ACK_RTN
f	200PY0090R_I.BI07.toSet
g	200PY0090R_I.BI07.toReset

the event tags relevant for the response of a particular alarm are determined, as is explained in Section 4-4. Next, sequences of the relevant events are extracted from the event log by following a time frame after annunciations of the alarm , as is explained in Section 4-5. A pattern recognition algorithm is then applied to the found sequences, to identify common patterns in the responses of operators to the alarm, the application of this algorithm is shown in Section 4-6. The results from applying this algorithm are then further processed to obtain the most relevant patterns for each alarm in Section 4-7 The different alarms are ranked based on their patterns and other parameters to find the most suitable responses to be automated, which is shown in Chapter 5.

4-4 Event Reduction

In the pattern recognition stage, the amount of different types of events being considered is important to both the quality of the results and the time it takes the algorithm to produce those results [11]. As stated in Section 4-1, the complete set of events possible in a chemical plant is large with over 30000 different event types. Instead of analyzing the entire event log of a chemical plant, it is needed to make a smaller selection by only using the events which are likely to have been in response to an alarm are considered. The search space for the algorithm is first reduced by constructing a set of events which a are likely to be important to the response of the alarm.

This reduction is performed in two stages. First, only the event tags associated with equipment that is physically related to the original alarm tag is selected, by searching the PEFSs of the process. Then, only the events which are likely to be related to the alarm based on the time at which they are occurring are kept. The stages are further explained in the following subsections.

4-4-1 PEFS Linking

The PEFSs of a process show how substances in a process flow between different pieces of equipment, and how the equipment is connected. A PEFS also indicates which PEFSs are

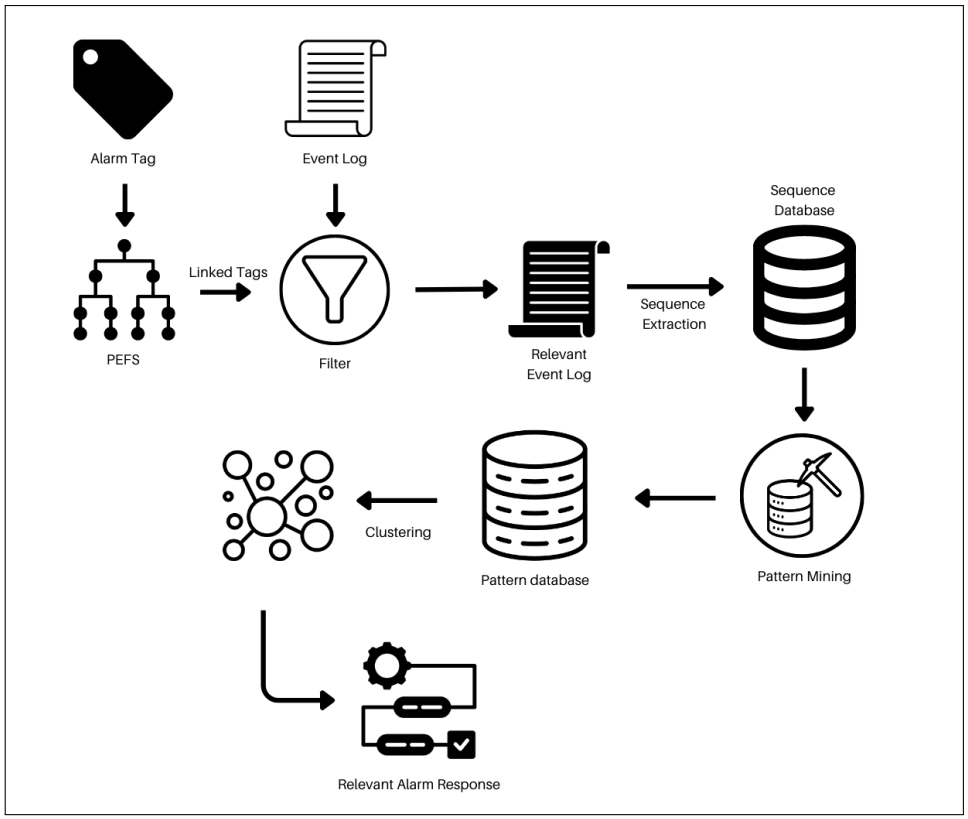

Figure 4-4: An overview of the method for the analysis of a single alarm tag: The event log is filtered by retaining only the events e whose tags v are linked to the alarm tag via the Process Engineering Flow Schemes (PEFSs). Next, sequences of events after the occurrence of an alarm are extracted from the event log and stored in the sequence database. Sequential pattern mining is applied to the sequence database in order to find all patterns which occur within the sequence database. The found patterns are then clustered based on similarity, and only the most relevant alarm responses are kept and returned.

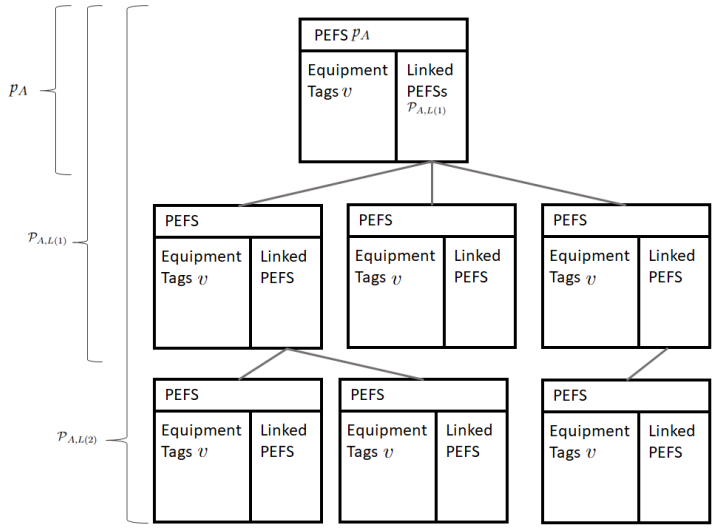

Figure 4-5: An Illustration of searching for equipment tags v that are linked via PEFS to alarm tag A. The set $\mathcal{V}_{A,n}$ consists of all event tags v which are present on the PEFS linked to p_A in at most n steps, denoted by $\mathcal{P}_{A,L(n)}$

.

connected to it. In [50], use is made of PEFSs to infer relationships between alarms, for the analysis of alarm floods. Under the assumption that alarms can be often be resolved by interacting with equipment which exists on the same or connected PEFS, this is extended to include operator actions in this thesis. This assumption was made after discussing with operational staff within Shell Energy and Chemicals Park Rotterdam. This assumption was furthermore confirmed by the test dataset in the next Chapter, where all event tags in the known responses of the test alarms are indeed found on either the PEFS of the alarm tag, or a PEFS directly linked to it.

In Shell Energy and Chemicals Park Rotterdam, PEFSs are available for the processes and are digitized such that a database exists containing which event tags are present on each PEFS. Furthermore, each PEFS shows what other PEFSs are linked to it. This database was utilized to make a selection of event tags to be considered in later stages of the alarm response analysis. This is illustrated in Figure 4-5.

First, an alarm tag A is chosen, after which the PEFS database is searched for the PEFS p_A on which the alarm tag is present. Next, a search is performed for all PEFSs $p \in \mathcal{P}_{A,L(1)}$ which are linked to PEFS p_A. Here, $\mathcal{P}_{A,L(n)}$ are all PEFSs linked to p_A in at most n steps. $\mathcal{P}_{A,L(2)}$ would thus be all PEFSs linked to PEFS p_A by at most two steps. Finally, all event tags v which are present on PEFSs $\mathcal{P}_{A,L(n)}$ are said to be linked to alarm tag A in n steps, and are denoted by set $\mathcal{V}_{A,n}$

Results for Example Alarm

For the example alarm *"200XT0215.HIABS"* introduced in Subsection 4-2, the resulting PEFSs for $n = 1$ are shown in Table 4-3a. In total, 383 event tags are found, a selection of which are shown in Table 4-3b, where the event tags in the known response are shown in bold. The original database contained 13139 tags, using the *PEFS linking* method, a 97%

Table 4-3: a) Overview of all PEFSs linked to alarm tag *"200XT0215*, b) Selection of event tags on the different linked PEFSs.

(a)

Set	PEFS
$p_{200XT0215}$	R-0848633
$\mathcal{P}_{200XT0215,L(1)}$	R-0848630
	R-0848632
	R-0848634
	R-0848656
	R-0848657
	R-0848659
	R-0848664
	R-0861396

(b)

PEFS	Event tag
R-0848630	200FB0037
R-0848633	200PV0090Q
R-0848633	**200PV0090R**
R-0848633	200PY0090Q
R-0848633	**200PY0090R**
R-0848633	200XT0214
R-0848633	**200XT0215**

reduction in the amount of event tags was thus made. Although for this particular example all relevant tags can already be found on the PEFS on which the alarm tag can be found, this is not the case for all alarm tags.

4-4-2 Tags Correlated with Alarm

To further decrease the size of the set of events considered, a metric is introduced to determine if the occurrence of an event tag is correlated to the annunciation of an alarm. This is determined by comparing the amount of times the event tag occurs in a window of time after an alarm occurred, and the amount of times that event tag occurs in the entire dataset. The method was constructed under the assumption that actions which are performed in response to a particular alarm will occur relatively often in a time window after the alarm is annunciated than events which are unrelated to that alarm.

Time Window

The windows in which an event $e = (t, v, a)$ needs to occur to be counted as occurring in response to an alarm A are denoted as $T_{A,i}$. They are constructed by taking the time $t_{0,i}$ at which an alarm is sounded for the $i-th$ time, ($a = Unack\_Alm$) until time $t_{1,i}+T_t$, where $t_{1,i}$ is the time at which the alarm returns to its normal state after having been acknowledged ($a = Ack\_Rtn$) and T_t is a *trailing window*. As such, event e should occur at time $t_0 \leq t \leq (t_1+T_t)$ in order to be in the window for alarm A.

This is illustrated in Figure 4-6. The implementation is similar to the implementation shown in [14]. However, a trailing window of fixed size is used, instead of extending the trailing window for every new event that occurs in it. The latter method was found to result in alarm windows that were too long in this dataset, because of the high frequency at which events are executed by operators.

The amount of events with tag v occurring within the windows of alarm A is from here on out denoted as $|e_{v,I_A}|$, while the amount of events with tag v outside the windows of alarm A is denoted as $|e_{v,O_A}|$. The total amount of occurrence of events with tag v is denoted as $|e_v|$.

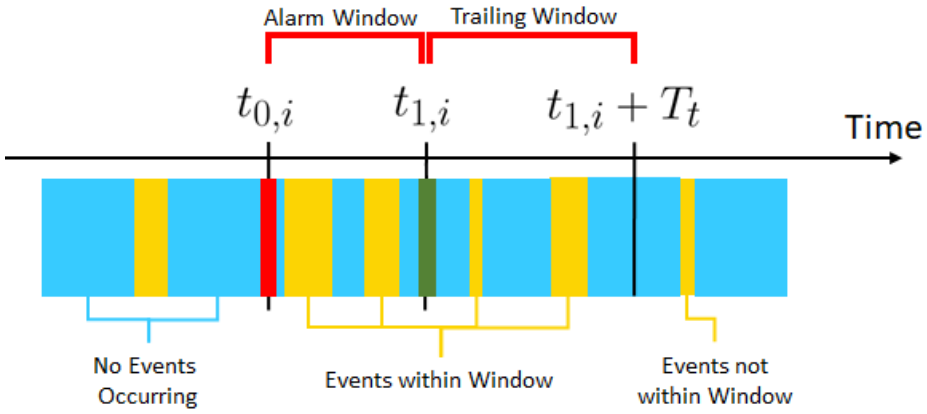

Figure 4-6: Time window used for extracting sequences of events from the event log. The red part denote the annunciation of the alarm in question, while the green part denotes the alarm being resolved. The yellow parts denote other events occurring, and the blue parts denote no events occurring.
In the context of finding relevant events for an alarm (Section 4-4), the tags of the events within the window are simply counted. In the context of sequence extraction (Section 4-5), the events within the window are combined into a single sequence and placed into the sequence database.

To determine if the occurrence of an event tag is correlated to the recent annunciation of an alarm, a number of different methods can be used. Two different methods were considered and tested. The results are compared in Table 4-4.

Occurrence Ratio

The simplest method is to consider the occurrence ratio r between the amount of events with tag v occurring within the alarm windows of alarm A and the total amount of occurrences of events with tag v:

$$r = \frac{|e_{v,I_A}|}{|e_v|} \tag{4-1}$$

An event tag v which is correlated to the occurrence of alarm A would be expected to occur more often in the time windows after the alarm annunciations, and as such have a higher occurrence ratio r. A selection of the results for alarm *200XT0215* can be seen in Table 4-4. The three event tags that are required for the reference operator response to alarm *200XT0215* have an occurrence ratio higher than most other event tags: They are all in the top 7 out of 383 event tags.

Chi-Squared

A second method based in statistics is to assess the probability of observing the distribution of $|e_{v,I_A}|$ and $|e_{v,O_A}|$ via the Chi-Squared test. This gives a value for the probability of the observed distribution occurring, under the assumption that the event tag is not related to the alarm. If this value is small, it is unlikely that the alarm and the event tag are not related.

The frequency of stochastic events within a time interval conforms to a Poisson Distribution, under the assumption that the time interval is independent of the occurrence of events. The Poisson Distribution is characterized by its Probability Mass Function (PMF) shown in the following equation, where $P(x)$ is the probability of x occurring, k is the amount of occurrences within a window, and λ is the average rate at which event occur per window length [13].

$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!} \tag{4-2}$$

If an event tag v is not correlated to an alarm A, the assumption can be made that the occurrences of event tag v follow a Poisson Distribution with respect to the alarm windows of alarm A. In order to test this assumption, the Chi-Squared test can be used to find the probability of the observed distribution occurring under the hypothesis that the event tag v is not correlated to alarm A. If this distribution is found the be very unlikely, the hypothesis can be rejected, and event tag v and alarm A are assumed to be related. The Chi-Squared test was introduced in Section 3-4.

The Chi-Squared test compares the observed amount with the expected amount of occurrences in and out of the alarm window of alarm A. For a Poisson process, the expected amount of occurrences $E[N_T]$ of event e within a window of length T is given as:

$$E[N_T] = \lambda T \tag{4-3}$$

where *lambda* is the rate in the PMF given in equation 4-2 [13]. For an event tag v, the rate of occurrence λ is equal to the total amount of events with that tag $|e_v|$ occurring in the database, divided by the total length of time T_T that the database covers [13]:

$$\lambda_v = \frac{|e_v|}{T_T} \tag{4-4}$$

If e_{v,A_I} are all occurrences of event tag v, inside alarm windows of alarm A, the expected amount $E[|e_{v,A_I}|]$ is equal to the sum of expected occurrences of all separate alarm windows $T_{A,i}$:

$$E[|e_{v,A_I}|] = \sum_{i=0}^{n} E[N_{T_{A,i}}] = \sum_{i=0}^{n} \lambda T_{A,i} = \lambda \sum_{i=0}^{n} T_{A,i} = \frac{|e_v|}{T_T} T_A = |e_v| \frac{T_A}{T_T} \tag{4-5}$$

In which n is the total amount of alarm windows for alarm A, and T_A is the total amount of time covered by the alarm windows for alarm A. Similarly, the expected amount of events with event tag v outside of the alarm windows is proportional to the amount of time not covered by the alarm windows. This results in the following equations for the expected amount of event with tag v occurring within and outside of the alarm windows of alarm A, assuming that the event tag v and alarm A are not correlated:

$$E[|e_{v,A_I}|] = |e_v| \frac{T_A}{T_T} \tag{4-6}$$

$$E[|e_{v,A_O}|] = |e_v| \frac{T_T - T_A}{T_T} \tag{4-7}$$

By formulating a null-hypothesis stating that the occurrence of events with tag v are uncorrelated to alarm A being (recently) annunciated, a Chi-Squared test can be used to test if

Table 4-4: A Comparison between metrics for different event tags for alarm *200XT0215.HIABS*. The operator actions in the known alarm response are shown in **bold**. The event tags are sorted on their Chi-Square score. For this alarm, the relevant event tags are in the tags with the largest Chi-Square value.

Alarm	Ratio	Chi-Squared
200PV0090R	0.39	3065
200XT0215	0.62	5500
200PY0090R_I	0.34	670.6
200PV0090Q	0.22	628.9
200PV0090A	0.14	345.1
200PV0090C	0.15	287.7
200PC0090	0.07	219.1
200PV0090K	0.14	206.2
200PY0090Q_I	0.21	205.5
200XT0214	0.62	185.0
200PY0090A_I	0.16	150.7

this hypothesis holds for a certain p-value [64]. If the hypothesis does not hold, the event is assumed to be correlated to the alarm, and is used in further analysis.

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i} = \frac{(|e_{v,A_I}| - E[|e_{v,A_I}|])^2}{E[|e_{v,A_I}|]} + \frac{(|e_{v,A_O}| - E[|e_{v,A_O}|])^2}{E[|e_{v,A_O}|]} \tag{4-8}$$

In Table 4-4 the results for different event tags in response to alarm *200XT0215.HIABS* are shown. The Table shows the 10 event tags with the largest Chi-Square. Based on the Chi-Square test, those event tags are the most likely to be correlated to alarm *200XT0215.HIABS*. In the Table, it can be seen that the tags present in the reference operator response to example alarm *200XT0215.HIABS* have the 3 largest Chi-squared values, out of a total of 383 tags found by linking the PEFS method. This indicates that the observed distribution of those tags has the smallest probability of occurring under the assumption that they are unrelated with the alarm.

Based on the results shown in Table 4-4, and from results of other alarms considered, the Chi-Squared method is determined to be the most effective. In order to implement this method, the n_v event tags with the largest Chi-Square value are chosen as the relevant event tags for an alarm, where n_v should be chosen by the user of the method. Those event tags are kept in the resulting event log for further analysis.

This approach was designed for this thesis, and is novel in the context of analyzing actions performed by human operators in the process industry. In the results of the training and test dataset of different alarms (shown in the following Chapter) the Chi-Squared test is able to determine the events which were indeed performed in response to the different alarms.

4-4-3 Resulting Event Log

For example alarm *200XT0215.HIABS*, initially starting with 13139 different event tags in the event log, selecting the event tags linked by PEFSs reduced this to 383 different event

tags. The Chi-Squared test reduces this to a number of event tags which can be decided by the user of the method. From the results from the test-database in Chapter 5, it was determined that 30 is a suitable amount of event tags to be used for this step.

Comparison of Computational Performance

It is possible to directly use the Chi-Square in order to find the n most likely alarms, without first selecting events which are linked via PEFSs. From a comparison between the two methods, it was found that while the resulting tags were equal, preselecting the event tags with the PEFSs, and then analyzing only their Chi-square values is computationally faster than applying the Chi-squared test to all event tags. For alarm *200XT0215.HIABS*, both methods were tested on the system described in the methodology for 10 runs. The former method took on average 5.71 seconds for finding the event tags on the PEFS and then finding their Chi-Square values combined. The latter takes on average 85.6 seconds for finding the Chi-Square values of all event tags. As such, the faster combination of both methods is chosen for the alarm analysis method.

4-5 Sequence Extraction

In Section 4-6, patterns will be mined from the responses of operators to alarms. As explained in Chapter 3, a pattern is a sequence of events which occur a minimum number of times in a sequence database. As such, a sequence database first needs to be extracted from the event log for use in the pattern mining algorithm. Each sequence consists of a number of events occurring within a specific time window. The extraction of that database is explained in this Section.

In order for the database to be used for pattern mining, several considerations need to be made. Extracting sequences involves defining a time frame following an alarm to capture subsequent events that contribute to the operator's response to the alarm. It must be ensured that each event occurrence only appears one sequence, to avoid the occurrence being over represented in the pattern mining results.

Alarm messages that are resolved without any operator intervention are considered chatter and are perceived as extraneous noise, as is explained in Subsection 4-5-3. Operator actions that are directly repeated within a sequence are reduced to a single occurrence of that operator action. This was done in order to shorten the event sequences while preserving all information. This is explained in Subsection 4-5-4

4-5-1 Sequence Window

A sequence consists of all relevant events (resulting from the methods presented in the previous Section) which occur in a time window. In order to construct the sequence database, the same time windows are used as described in Subsection 4-4-2. Each window $T_{A,i}$ in the analysis of alarm A spans from $t_{0,i}$ to $t_{1,i} + T_t$, where $t_{0,i}$ is the moment of annunciation of the alarm, $t_{1,i}$ is the moment it is resolved, and T_t is a trailing window for which the length needs can be chosen by the user of the method. This is illustrated in Figure 4-6: As indicated in the

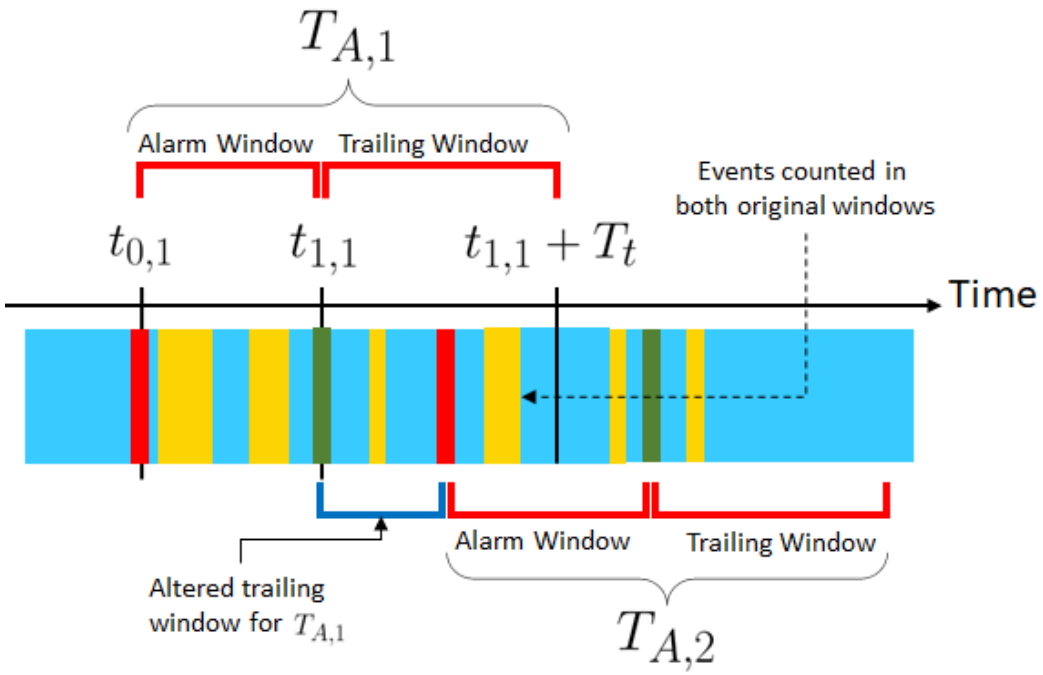

Figure 4-7: Figure illustrating the possible over representation of events when extracting sequences from the event log, and the solution to this problem.

image, certain events are counted in both alarm windows when the trailing windows are used as explained above.

4-5-2 Preventing Over Representation of Events

To ensure the accuracy of the extracted sequences and the resulting patterns, it is essential that each relevant event appears in no more than one sequence of the sequence database. If a single event or a number of sequential events are counted in multiple sequences, the results of Sequential Pattern Mining (SPM) will be distorted and those events will be over represented. If a number of sequential events are counted in multiple sequences, those can erroneously be seen as a pattern. This is illustrated in Figure 4-7. As indicated in the image, certain events are counted in both alarm windows when the trailing windows are used as explained above.

To prevent events from being over represented, each event should only be assigned to a single sequence. Therefore, the alarm windows $T_{A,i}$ should not be overlapping. In order to achieve this, each instance i of alarm A is given its own trailing window $T_{t,i}$, as shown in equation 4-9, in which $\min(\cdot,\cdot)$ is the smaller of the two arguments. The individual trailing windows are thus shortened such that none overlap with the next instance of the alarm annunciating. In Figure 4-7, this is illustrated with the *Altered Trailing window for* $T_{A,1}$.

$$T_{t,i} = \min(T_t,\ t_{0,i+1} - t_{1,i}) \tag{4-9}$$

In Table 4-5, a selected portion of the event log of events relevant for the example alarm *200XT0215.HIABS* is shown, together with the resulting assignments with and without the

Table 4-5: Comparison of results for extracting sequences from a selection of the event log of alarm *200XT0215.HIABS* with and without the alteration of the trailing window T_t. Without alteration, some events are represented in both sequence 1 and sequence 2, possibly resulting in over representation of those events in the SPM step.

		Without Alteration		With Alteration	
Time stamp	Event	Seq. 1	Seq. 2	Seq. 1	Seq. 2
0:00	200XT0215.HIABS.UNACK_ALM	X		X	
0:00	200XT0215.toACK	X		X	
0:00	200PV0090R.MA	X		X	
0:01	200PV0090R.OUT.DEC	X		X	
0:02	200PV0090R.OUT.DEC	X		X	
0:02	200XT0215.HIABS.ACK_RTN	X		X	
1:02	200PV0090R.OUT.INC	X		X	
1:06	200PY0090R_I.BI07.toSet	X		X	
1:06	200PY0090R_I.BI07.toReset	X		X	
1:08	200XT0215.HIABS.UNACK_ALM		X		X
1:08	200XT0215.toACK	X	X		X
1:08	200PV0090R.OUT.DEC	X	X		X
1:10	200XT0215.HIABS.ACK_RTN		X		X
3:02	200PV0090R.OUT.INC	X	X		X
3:02	200PY0090R_I.BI07.toSet	X	X		X
3:02	200PY0090R_I.BI07.toReset	X	X		X

trailing window alteration. From the table, it can be seen that the method without alteration of the trailing window T_t splits the event log into two sequences, where some of the events are counted in both sequence 1 and 2. The method with alteration splits the event log into two sequences, both containing the pattern of interest, and no events overlapping.

4-5-3 Removing Trivial sequences

When extracting sequences in the manner described above, a number of trivial sequences resulting from alarm chattering may be included. A sequence is considered to be trivial when no operator action was necessary to resolve the alarm. This is the case when the only operator action occurring during the annunciation of the alarm was the acknowledgement of the alarm. There can however be events included in the trailing window after the alarm was resolved. An example of such a sequence is shown in Table 4-6.

Leaving such sequences in the sequence database would result in the events occurring after the alarm was resolved to be included in the pattern mining stage. Since those actions were not performed by the operator to resolve the alarm, they are regarded to be noise and should not be included in the sequence database. Furthermore, such sequences can distort the support of patterns in the database. The support is the percentage of sequence in which a pattern shows up. By leaving trivial sequences in the sequence database, the total amount of sequences in the database will be higher, meaning the support value will be distorted.

Finally, sequential pattern mining can be computationally expensive, and its computational cost increases as the amount of sequences increases. Removing trivial sequences of which

Table 4-6: A sequence of events resulting from an alarm chatter, where no intervention of the operator was needed to resolve the alarm.

Time Stamp	Event
2023/3/13 14:52	200XT0215.HIABS.UNACK_ALM
2023/3/13 14:53	200XT0215.toACK
2023/3/13 14:54	200XT0215.HIABS.ACK_RTN
2023/3/13 15:04	200XT0215.HIABS.UNACK_ALM
2023/3/13 15:04	200XT0215.toACK
2023/3/13 15:06	200XT0215.HIABS.ACK_RTN
2023/3/13 15:07	200XT0215.HIABS.UNACK_ALM
2023/3/13 15:07	200XT0215.toACK
2023/3/13 15:11	200XT0215.HIABS.ACK_RTN
2023/3/13 15:12	200XT0215.HIABS.UNACK_ALM
2023/3/13 15:13	200XT0215.toACK
2023/3/13 15:14	200XT0215.HIABS.ACK_RTN

Table 4-7: Table presenting a sequence containing repeated events. The operator action *200PV00090R.OUT.DEC*, which decreases a fan speed is repeated four times.

Time Stamp	Event
2022/12/23 18:05	200XT0215.HIABS.UNACK_ALM
2022/12/23 18:06	200XT0215.toACK
2022/12/23 18:06	200PV0090R.MA
2022/12/23 18:06	200PV0090R.OUT.DEC
2022/12/23 18:06	200PV0090R.OUT.DEC
2022/12/23 18:06	200PV0090R.OUT.DEC
2022/12/23 18:06	200PV0090R.OUT.DEC
2022/12/23 18:07	200XT0215.HIABS.ACK_RTN
2022/12/24 03:22	200PY0090A_I.BI07.toSet
2022/12/24 03:22	200PY0090A_I.BI07.toReset
2022/12/24 03:22	200PY0090R_I.BI07.toSet
2022/12/24 03:22	200PY0090R_I.BI07.toReset

it is known that they contain no relevant patterns decreases the search space makes the sequential pattern mining stage less computationally expensive. For those reasons, all such trivial sequences are removed from the sequence database before continuing with the SPM step.

4-5-4 Removing Repeating Events

During operation, operators may repeatedly perform certain actions. For example, they may increase a set point of a controller value in a number of steps. In order to decrease the amount of different events, an actions which changes a set point is appended with either the *".INC"* or *".DEC"* suffix, and the exact value to which the set point is changed is ignored. Repeated changes in set point will result in a sequence as shown in Table 4-7.

Repeated actions increase the length of sequences. As the number of repetitions increases, the

length of the sequences grows, making the search space larger and more complex, resulting in more computational cost in the sequential pattern mining algorithm.

Repetition of an action also introduces redundancy and noise in the patterns. The presence of repeated actions can overshadow other important patterns or distort the overall sequence. It becomes challenging to distinguish between meaningful repetitions and noise, making it harder to identify significant patterns. Furthermore, repeated actions will be present more frequently in the sequence database, meaning that the support for such actions is distorted. For those reasons, repeated actions are regarded to be noise, and are removed from the sequences.

4-5-5 Results for Example Alarm

For the example alarm *200XT0215*, 84 sequences were found over a period of 3 years. The average sequence length is 15.6 events. The largest sequence has a length of 64 events. By manual inspection, 17 instances of the reference operator response are found in the data.

4-6 Sequential Pattern Mining

Because of the complexity of plants in the process industry, multiple alarms can be annunciated simultaneously or in quick succession. As such, any sequence in the sequence database of a particular alarm may contain events which were not in response to that alarm. However, patterns which occur in a large amount of sequences in the sequence database are more likely to have been the response of operators to the alarm. In order to find those operator responses to an alarm, patterns are mined from the sequence database. As explained in the literature review in Chapter 3, a pattern p is any sequence of events e which occurs at least a minimum amount of times as a subpattern of the sequences in sequence database S_A. This minimum threshold is called the *minimum support*. An example was presented in Chapter 3, in Table 3-3.

Pattern mining is specifically designed to discover meaningful and previously unknown patterns in large datasets [14]. As such, pattern mining is suitable to automatically find patterns which are not easily found through a simpler analysis method or manual inspection. This capability is valuable in understanding how operators respond to alarms over time, and for identifying common response sequences.

4-6-1 Patterns Types

As was explained in the previous Chapter, research in pattern mining has resulted in a range of pattern mining algorithms with different characteristics and capabilities [25]. Some of these algorithms represent enhanced versions of their predecessors, aiming to improve efficiency and scalability, optimizing the algorithms' execution time or memory usage, making them more suitable for mining patterns from larger datasets [57]. Different pattern mining algorithms target the discovery of different pattern types. The proper pattern type should be chosen and depends on the goal of the analysis [25].

Table 4-8: Table with a comparison of the support values for a selection of patterns resulting from the sequence database of example alarm *200XT0215.HIABS*. The table shows that for the longest pattern in the pattern database, the support values are all the same. When counting all patterns, patterns will have support larger than or equal than the frequent patterns they are a subsequence of. The column with closed frequent pattern support show that a pattern is not counted if its support is equal to a that of pattern of which it is a subsequence. Since patterns 2 to 6 are subsequences of sequence 1, their support is 0 for when counting maximal sequences.
In order to maintain legibility, the letters map to the following events: a) *200XT0215.HIABS.UNACK_ALM*, b) *200XT0215.toACK*, c) *200PV0090R.MA*, d) *200PV0090R.OUT.DEC*, e) *200XT0215.HIABS.ACK_RTN*, f) *200PY0090R_I.BI07.toSet*, g) *200PY0090R_I.BI07.toReset*

			Support			
#	Pattern	length	All	Closed	Max	Independent
1	a,b,c,d,e,f,g	7	17	17	17	17
2	a,b,c,d,e	5	32	32	-	15
3	a,b,c,d	4	32	-	-	-
4	a,b,c	3	43	43	-	11
5	a,c	2	46	46	-	3
6	c	1	46	-	-	-

In Sequential Pattern Mining (SPM), the support of a pattern is a primary indicator for its relevance. Therefore, patterns which are more meaningful should have a larger support than those which are not meaningful. Below, the pattern types introduced in the previous Chapter are again named, and their suitability for the goal in this thesis is discussed. Furthermore, a novel pattern type, designed specifically for this thesis is introduced. In Figure 4-8, the different pattern counts of different patterns in the sequence database of alarm *200HY0037.LOABS* are shown. The reference operator response for this alarm is 4 events long.

Frequent Patterns

The most basic approach is to find all frequent patterns in the sequence database which occur at least the *minsup* amount of times. The result from this approach is illustrated in Table 4-8. This method results in a very large amount of patterns, where every subsequence of larger patterns is also counted as a distinct pattern. If those smaller patterns have the same support as larger patterns, they do not supply any new information: Their support could already be derived from the larger pattern's support [9]. This makes finding the most relevant response to an alarm difficult, since shorter subsequences will inherently have at least the same support as the sequences they are contained in, and possibly a larger support. As such this approach would be biased towards short sequences, which is not desirable in the context of finding response patterns to alarms. This is also shown in Figure 4-8, where the shortest patterns have the highest support, this is the case for any sequence database.

Closed Patterns

A different approach is to search for closed frequent patterns. A closed frequent patterns is a pattern that is not strictly included in another pattern with the same support [26]. This

approach greatly reduces the resulting amount of patterns by removing any pattern which is a subsequence of another pattern with the same support. However, this does not simplify the process of finding the most relevant response to an alarm, since shorter patterns will inherently have either a support of zero, or a larger support than sequences that they are a subsequence of. This means that this method is still biased towards shorter sequences, as is shown in Figure 4-8.

Maximal Patterns

"A maximal pattern is a closed pattern that is not strictly included in another closed pattern." [26]. Thereby, any closed frequent pattern that is a subsequence of another closed frequent pattern in the pattern database is removed. This greatly reduces the amount of resulting patterns. However, this method will remove any pattern which is a subsequence of a larger pattern, even if the larger pattern has a much smaller support than the smaller pattern. As such, this method favours larger patterns. The largest patterns are not necessarily the response that is searched for as reference response. As is shown in the example in Figure 4-8, the pattern with the highest support in the maximal patterns is of length 5, with a support of 25. The true reference pattern is of length 4, of which there are 37 instances in the sequence database.

Independent Patterns

In order to remedy the problems of the pattern types named above, a novel method of counting pattern support is proposed in this thesis. The goal of this method is to find the patterns which are the most relevant to the solving of the alarm, without being biased towards shorter or longer sequences. Because shorter closed frequent patterns can act as building blocks of longer patterns, the sequences in which the longer closed pattern occurs are counted towards the support of both the longer and shorter closed pattern. As such, a pattern which is a subsequence of a larger pattern is likely to appear more frequently in the closed pattern database than the larger pattern. It is therefore hard to identify the most relevant pattern. In order to solve this problem, a new pattern type is proposed, where the support of each pattern is calculated differently than for previously proposed patterns types.

By analyzing the pattern databases resulting from the test alarms, the insights were gained that a sequence containing the reference operator response will likely also contain other events in it. Furthermore, a pattern which is a subsequence of a larger pattern that appears less frequently when considered independently (in sequences where it is not a subsequence of the larger pattern) than the larger pattern itself is likely to be less significant than the larger pattern.

In order to find the most relevant patterns under these insights, only *independent* occurrences of patterns are kept. Here, an independent occurrence is any occurrence of a pattern which was not already counted as an occurrence for a larger pattern of which the smaller pattern is a subsequence.

In order to achieve this, the support of sequences in the closed pattern database is altered by systematically reducing the support count of smaller subsequences that were already accounted for in longer patterns: If a pattern p_1 is supported by sequences s_1, s_2, and s_3 and

if a smaller pattern $p_2 \sqsubseteq p_1$ is supported by sequences s_1, s_2, s_3, and s_4, then the pattern p_2 is said to now be *independently* supported only by sequence s_4. As such, the new support for the sequence is 1.

A pseudo algorithm for this systematic approach is shown in Algorithm 2. A Python implementation of this algorithm was written for this Thesis, and is shown in Appendix D.1.

In Table 4-9, an example closed pattern database is shown, with the resulting independent pattern database. Furthermore, in Figure 4-8, the maximum independent pattern support for the length of each pattern length is shown for alarm *200HY0037.LOABS* from the test database. The known correct response for this alarm is a pattern of length 4. Only when independent patterns are used, does the correct pattern have the highest support.

Since closed frequent patterns are needed to find the independent frequent pattern, an algorithm should be chosen to find the closed frequent pattern. For this, the BIDE+ algorithm introduced in Section 3-6 was chosen, since this algorithm was faster than other options when tested on the different alarm sequence databases. The algorithm is furthermore suited for mining large databases with low support [57]. A Python implementation of this algorithm was written to use in this thesis, but a readily available implementation written in Java was tested to be faster, and therefore used instead [24].

Algorithm 2 Pseudo code for the algorithm for finding the set of independent frequent patterns P_I and their support, starting with a database P_C containing all closed frequent patterns. A Python implementation is shown in Appendix D.

```
P_C = set of closed patterns
S = Sequence Database
P_I = ∅                                   ▷ Initialize set of independent patterns
for (p ∈ P_C) do                          ▷ For each pattern in the closed pattern database
    S_p = {s ∈ S : p ⊑ s}                 ▷ Initialize independent support as closed sequence support
    P_s = {p_s : p ⊑ p_s, p_s ∈ P_C}      ▷ Identify patterns containing the current pattern
    for (p_s ∈ P_s) do                    ▷ For each supersequence
        S_ps = {s ∈ S : p_s ⊑ s}          ▷ Find all sequences that support the super sequence
        S_p = S_p \ S_ps                  ▷ Remove those sequences from the sequences supporting p
    if |S_pI| ≥ minsup then               ▷ If the independent support ≥ minsup,
        P_I = P_I ⋃ (p, |S_pI|)           ▷ add the pattern and its support to P_I
```

4-6-2 Results for Example Alarm

Analyzing the sequence database of example alarm *200XT0215.HIABS* described in Section 4-5 using the independent pattern type results in 35 different patterns, ranging from 4 to 7 events in length. In Figure 4-9, the distribution of the resulting pattern lengths and their support values are shown. The independent pattern with the largest support is the reference operator response shown in Table 4-2 with 17 occurrences.

Further inspection shows that a different pattern with a high support is similar to this pattern, with the *200PV0090R.OUT.DEC* event replaced for *200PV0090R.OUT.INC*. In this pattern, the operator responded by increasing the fan speed instead of decreasing it. After consulting with operational staff, this was found to be indeed be a valid response: The fan has a workload

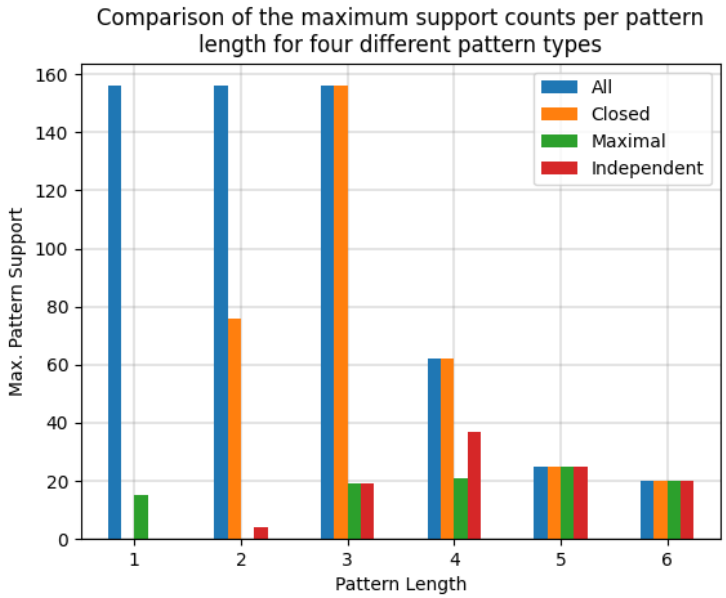

Figure 4-8: Comparison of the results of applying different pattern mining algorithms to the sequence database of alarm *200HY0037.LOABS* with a minimum support of 10%. Four pattern types of patterns were mined, resulting in all patterns, closed patterns, maximal patterns and independent patterns. The reference operator response is of length four. The pattern of length four with the highest support is the reference pattern for each pattern type, except in case of the maximal sequences, where the reference pattern was absorbed by a larger pattern. Only in case of the newly proposed indepent patterns is the reference pattern the pattern with the largest support.

Table 4-9: Comparison of a selection of closed and independent patterns and their counts arising from an example sequence database for minsup = 2. The example was adapted from [23]. Note that the sequence $< a, b >$ is only independently supported by sequence 5, and as such does not meet the $minsup$ threshold.

(a) Sequence Database

ID	Sequences
1	<a,c,b,g,f>
2	<a,c,b,f>
3	<a,b,e,f>
4	<g,a,b,f>
5	<a,f,b>
6	<e,a,f>

(b) Closed Patterns

ID	Pattern	Support	Sequences
1	<a,c,b,f>	2	1,2
2	<a,b,f>	3	1,2,3,4
3	<a,f>	3	1,2,3,4,5,6
4	<a,b>	3	1,2,3,4,5
...	...	...	...

(c) Independent Patterns

ID	Pattern	Support	Sequences
1	<a,c,b,f>	2	1,2
2	<a,b,f>	2	3,4
3	<a,f>	2	5,6
...	...	...	...

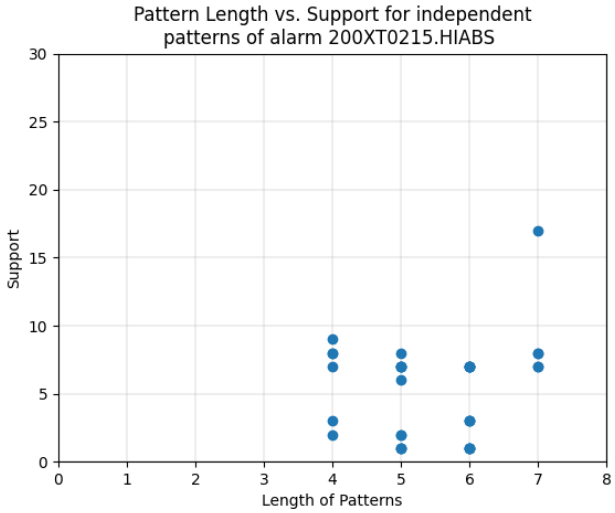

Figure 4-9: Scatter plot showing the different support values for the independent patterns mined from the sequence database of alarm *200XT0215.HIABS*

at which resonance occurs. In order to steer the fan away from this workload, an operator can either increase or decrease the fans speed.

Apart from the main response, deviations from expected patterns can thus be detected. In the context of operator response to alarms, this allows for the identification of both common response patterns and exceptional response scenarios. By analyzing both frequent and rarer patterns, insights can be gained into the typical behaviors of operators as well as unusual or abnormal response patterns that may be useful to inspect further.

4-7 Combining similar patterns

The SPM step results in a list of response patterns found in the sequence database S. For each pattern, its independent support is given in the pattern database P. In order to determine how suitable the response of an alarm is to automate, it should be determined how predictable the response of that alarm is. As stated before, the predictability is the degree to which the operator actions in response to an alarm can be anticipated or foreseen based on the patterns observed in the event logs.

For example, in Table 4-10, two different patterns are shown, where the first is the most frequently occurring pattern in response to alarm *200XT0215.HIABS*, and the second is a very similar pattern, with only the action of setting the controller in manual mode is missing. Such a pattern could arise if the controller was already set in manual mode. Instead of using only the support value for the first pattern, it was found that combining the two patterns yielded results better representing the predictability of the response pattern.

In order to extract a single predictability value from the pattern database, the support of very similar patterns should thus be combined. If a single alarm can be resolved by a small number of response patterns which are very similar, the alarm response can still be suitable for automation, since automating a procedure consisting of very similar responses is generally

Table 4-10: Comparison of two patterns in response to the alarm *200XT0215.HIABS*

ID	Pattern
1	200XT0215.HIABS.UNACK_ALM, 200XT0215.toACK, 200PV0090R.MA, 200PV0090R.OUT.DEC, 200XT0215.HIABS.ACK_RTN, 200PY0090R_I.BI07.toSet, 200PY0090R_I.BI07.toReset
2	200XT0215.HIABS.UNACK_ALM, 200XT0215.toACK, 200PV0090R.OUT.DEC, 200XT0215.HIABS.ACK_RTN, 200PY0090B_I.BI07.toSet, 200PY0090B_I.BI07.toReset

less complex than automating procedures that involve significantly different patterns [55], as was also explained in Chapter 3. Various methods of combining similar patterns were investigated, and are explained below.

4-7-1 PCA

First, it was attempted to project the different patterns on low dimensional plane using Principal Component Analysis (PCA) (see Chapter 3), and then clustering the resulting groups of patterns in that sub space. Clustering sequences using PCA is done in the bio-informatics field for the clustering of DNA sequences [35]. It was however found that this method requires longer sequences than the patterns found in the responses of alarms. For example, in [35] DNA sequences of lion species are clustered, which consist of DNA sequences with lengths of around 1700 letters. The average pattern found in the responses of alarms is only around 6 events long. Using PCA on alarm response sequences did not result in meaningful clusters for the different patterns.

4-7-2 Similarity scores

As was shown in Section 3-7, various methods exist to directly compare two sequences, and thus two patterns. The Needleman-Wunsch [38] method and the Smith-Waterman [52] method use dynamic programming to first find an optimal alignment of for two sequences, and then compare how similar the sequences are.

The Smith-Waterman method was used in the context of mining alarm sequences in both [12] and [36]. Both works used this method to compare alarm flood sequences. In [42] the Needleman-Wunsch algorithm is also used in the context of alarm floods.

It was however found that both methods were less suited for finding the similarity between two response patterns to alarms. This is because both methods do not account for the order of elements in sequences to be switched. It was found that some operators may prefer to perform one action first, while others may prefer to perform another first, in otherwise the same alarm response. Accounting for this is thus needed.

Both algorithms realign the events by creating gaps, but maintaining the original orders. To illustrate this, the example in Table 4-11 was constructed. In pair 1, the sequences are equal, except for 2 events being switched. This results in an alignment score of 2. In pair 2, the sequences differ more, but the alignment score is also 2.

Table 4-11: Results of alignment of pairs of sequences via the Smith-Waterman method with match score 1, mismatch score -1, and gap score -1.

Pair	Original sequences	Aligned sequences	Alignment score
1	$< A, B, D, C, E >$	$< A, B, -, D, C, E >$	2
	$< A, B, C, D, E >$	$< A, B, C, D, -, E >$	
2	$< A, B, C, D, E >$	$< A, B, -, C, D, E >$	2
	$< A, B, F, C, E >$	$< A, B, F, C, -, E >$	

4-7-3 Edit Distance

The Damerau-Levenshtein [37] method finds the shortest path to change one sequence into the other, allowing for removal, addition, substitution and swapping of characters. The amount of operations required to form one sequence into the other is called the edit distance. (See Subsection 3-7-1). By allowing characters (or events) to be swapped, the disadvantage of the similarity methods is remedied. In Table 3-5, in Chapter 3, an example is shown of the edit distance of two sequences.

For the predictability value, the supports of patterns with an edit distance d_e smaller than some user defined maximum edit distance m_{d_e} are combined into a single predictability figure. For this, the total support is divided by the amount of non-trivial sequences of alarm, as shown in the following equation, where $\sum supp_{I,(d_e \leq m_{d_e})}$ is the sum of the independent support of all patterns with edit distance d_e smaller than m_{d_e} and N_{nt} is the amount of non trivial sequences.

$$pred = \frac{\sum supp_{I,(d_e \leq m_{d_e})}}{N_{nt}} \tag{4-10}$$

Results for example Alarm

In Table 4-12, an example is shown for alarm *200PC0081.LOOUT*, the total support becomes the sum of all independent support values. For this alarm, that results in

$$pred = \frac{19 + 12 + 10 + 10}{103} = 0.51 \tag{4-11}$$

Table 4-12: Example of the most frequent response pattern to alarm *200PC0081.LOOUT*, and the other found patterns which have an edit distance of 1 from the most frequent pattern.

	Pattern	Ind. Supp	Edit dist.
Most freq.	200PC0081.LOOUT.UNACK_ALM, 200PC0081.toACK, 200FC0040.SPT.DEC, 200PC0081.LOOUT.ACK_RTN	19	0
Similar	200PC0081.LOOUT.UNACK_ALM, 200PC0081.toACK, 200FC0040.SPT.DEC, 200PC0081.LOOUT.ACK_RTN, 200FC0003.SPRAMP.toSet	12	1
	200PC0081.LOOUT.UNACK_ALM, 200PC0081.toACK, 200FC0040.SPT.DEC, 200PC0081.LOOUT.ACK_RTN, 200FC0040.SPT.INC	10	1
	200PC0081.LOOUT.UNACK_ALM, 200PC0081.toACK, 200FC0040.SPT.DEC, 200PC0081.LOOUT.ACK_RTN, 200FC0040.SPT.DEC	10	1

Chapter 5

Testing and Application of Algorithm

The results of the testing and application of the algorithm from Chapter 4 are shown. First, the results from the training set, and the algorithm parameters resulting in those results are presented. Next, the results on the test-set for verification of the algorithm are shown. Finally, the results from applying the algorithm to the set of unseen alarms are presented.

5-1 Training Alarm Set

In Table C-1 of Appendix C, the set of training alarms is shown. As explained in the Methodology, this set results from an analysis of the documentation of the plant of interest, and discussions with operational staff. Some of the alarms in the training set were shown in the previous Chapter.

Table 5-1: Results of applying the algorithm on the training set, with the support value for the most frequently found pattern. For each alarm, the reference pattern shown in Table C-1 was found.

Alarm	**Non trivial Sequences**	**Support**
15TC404.LOABS	256	0.17
15TC410.HIOUT	154	0.18
15TT406.LOABS	198	0.15
17PC055.HIABS	48	0.15
17PC149.LOABS	54	0.2
200HY0037.LOABS	157	0.24
200PC0081.LOOUT	105	0.18
200XT0200.HIABS	91	0.16
200XT0215.HIABS	85	0.2
210PT0010.HIABS	78	0.18

Table 5-2: The user-definable parameters used in the algorithm presented in Chapter 4 and the values chosen. T_t is the maximal length of the trailing window in the alarm windows. *minsup* is the minimum support a pattern should have, here shown in a percentage of the amount of sequences in the sequence database. n_v is the maximum amount of relevant tags resulting from the Chi-Squared test. m_{d_e} is the maximum edit distance allowed for a pattern to be considered similar to the most frequent pattern.

Parameter	Value
T_t	12 hours
minsup	8%
n_v	30
m_{d_e}	1

The algorithm is able to identify all required operator responses, as indicated by the results in Table 5-1. The algorithm makes use of a number of parameters that can be chosen by the user of the algorithm. Those parameters are: the number of relevant tags n_v (Subsec. 4-4-2), the max. length of trailing window T_t (Sec. 4-5), the minimal support of patterns *minsup* (Sec. 4-6), the maximum edit distance m_{d_e} (Sec. 4-7 In order to uncover the operator responses in the training set, the parameters need to be properly tuned. The resulting values for the parameters are shown in Table 5-2.

Finding the parameters was carried out manually. For each step, the results were inspected for various parameter values. As will be noted in the recommendations for future work, finding those parameters could be performed using an optimization scheme [49], where the cost function is based on the amount of correct response patterns found for each alarm.

5-2 Test Alarms

In Table C-2 in Appendix C, the set of test alarms is shown. The set of test alarms is used to evaluate the performance of the method on alarms which were not seen before by the method, and were not used for its design or tuning. It was tested if the algorithm can find the reference patterns as the most frequent patterns, without relying on the reference pattern being similar to the most frequent pattern. If that was not the case, it was then inspected if the pattern was present in one of the similar patterns.

In Table 5-3, the results are shown. In the table, it can be seen that for most test alarms, the found operator response is equal to the reference response. There are two alarms for which this is not the case. For the alarm *15YB020.LLABS*, the most found pattern is a response in which the operator simply mutes the annunciation of the alarm, this pattern is too different from the reference pattern to find the reference pattern in the similar patterns. The reference pattern was however present in the patterns database, with the support shown in the Table. For the alarm *15TT407.LLABS*, the most frequent pattern is very similar to the reference pattern, but contains an extra operator action.

Since most of the reference patterns are correctly identified, it was found that the results were sufficient to continue with other unseen alarms in order to find the most predictable alarm responses.

Table 5-3: Table presenting the results of the method applied to the test set. From the table, it can be seen that most of the reference responses are found directly as the most occurring independent frequent pattern. Two alarms are not directly found as the most frequently occurring pattern.

Alarm	Non trivial Sequences	Most frequent pattern?	In similar Pattern	Support of Ref. Pattern
15FC046.LOABS	85	Yes		0.11
15TC321.LOABS	90	Yes		0.19
15YB020.LLABS	240		No	0.10
16FC073.LOABS	187	Yes		0.13
200HY0029.LOABS	87	Yes		0.16
15TT407.LLABS	657		Yes	0.16
200XT0203.HIABS	15	Yes		0.20
200XT0206.HIABS	263	Yes		0.30
200XT0210.HIABS	87	Yes		0.14
230TC0001.HIABS	121	Yes		0.11

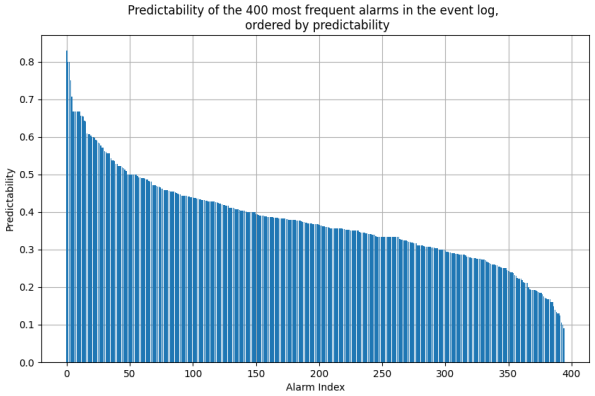

Figure 5-1: Graph presenting the predictability value for the 400 most frequent alarms.

5-3 Analyzing unknown alarms

The tested algorithm for obtaining predictability values for the response patterns to alarms can now be applied to alarms for which the correct response pattern is not known. The algorithm was applied to the 400 most frequent alarms in the dataset. The amount of annunciations for the first 100 of those was shown in Figure 4-1.

The resulting predictability values for the 400 most frequent alarms are shown in Figure 5-1. From the plot, it can be seen that a small number of alarms are significantly more predictable than other alarms.

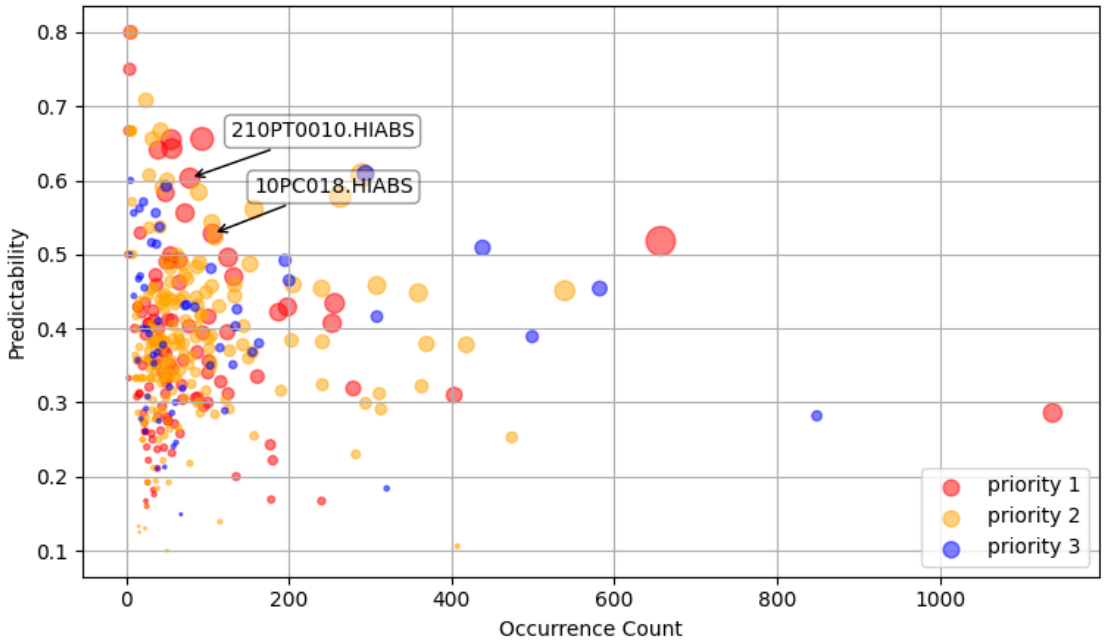

Figure 5-2: Scatter plot of the analysis of the different alarms, horizontal axis indicating the amount of non trivial alarm occurrences, vertical axis indicating the predictability of the alarm response. The location of alarm *210PT0010.HIABS* and of alarm *10PC018.HIABS* are shown. The size of each alarm marker indicates the score it received when comparing automation opportunities.

5-4 Identifying automation opportunities

Once the algorithm was applied to the 400 most frequently occurring alarms, a ranked list was made based on the results. This list is constructed based on the amount of times the alarm occurs, the predictability of the response pattern, and the priority level of the alarm.

The priority of the alarm is an indication for how long operators have before they must act to resolve an alarm from the start of annunciation. The priority is split into three levels: 1, 2, and 3, where level 1 is the highest priority, with the shortest allowable response time. The priority level of each alarm is directly based on documentation of the plant being analyzed.

The amount of non trivial occurrences was chosen instead of the total amount of alarm annunciations, because counting all occurrences of an alarm will also count the times it chatters. Using the non trivial sequences gives a relevant figure for the amount of times an alarm necessitates a response.

In Figure 5-2, the results for the 400 alarms are shown, where the horizontal axis indicates the amount of non trivial occurrences (see subsection 4-5-3) and the vertical axis indicates the predictability value (see subsection 4-7).

5-4-1 Ranking the alarms

The rank of each alarm is based on a score, which combines the occurrence count, the predictability and the priority of the alarm. This is done by the following equation, which will be further explained below:

$$score = pred^2 * \sqrt{N_{nt}} * (4 - prio) \tag{5-1}$$

In which *pred* represents the predictability value for the alarm response, N_{nt} represents the amount of non-trivial occurrences of the alarm, and *prio* represents the priority level of the alarm from 1 to 3.

This scoring equation was devised through an iterative process, attempting to combine the three terms in a manner which results in a fair evaluation for automation opportunities. Alarms which present good automation opportunities are, as stated in previous Chapters, alarms which have a predictable response so that their automation is feasible, and occur often enough and have a large enough priority to justify their automation.

After starting with the multiplication of all terms: $score = pred * N_{nt} * (4 - prio)$ the quadratic term for the predictability was added after observing that the predictability term needed to be more significant. The square root was added after observing that the outcomes were too heavily skewed towards alarms with high occurrences, causing the predictability term to still be underrepresented. It's should be noted that individuals have the flexibility to formulate their own scoring equations, tailoring them to accentuate the importance of any of the terms.

The equation shown above was applied to all 400 of the most frequent alarms. In Figure 5-2, the size of each alarm marking indicates its score relative to the other alarms. A larger marker indicates a higher score.

The top 10 alarms with the highest score are shown in Table 5-4. In the table, the values of the different metrics are shown, as well as the score that results from equation 5-1. A brief description is shown for each tag. Tags starting with a 1 are present in the Hydrogen Converter (Hycon) shown in Appendix A. Tags starting with a 2 are present in the Solvent De-Asphalter (SDA) shown Appendix B. The impact is divided into three categories: *potential trip*, *sub optimal performance* and *equipment damage*. Note that the priority and the impact are not always proportional in severity. The priority indicates how much time an operator has before he must act, while the impact indicates what the potential consequence is if the operator does not successfully resolve the situation.

5-4-2 Inspection of the ranked alarms

The different alarms in Table 5-4 were discussed with operational staff in the refinery. In those discussions, it was discussed which alarms would be good opportunities and which alarms would not be.

Certain alarms, namely the *10PY224.HHABS*, *10PT224.HHABS*, and the *13LC040.HIABS* alarms are already associated with automated procedures. The first two are alarms indicating that certain filters are ready to be cleansed. The cleansing of those filters is performed via an automated procedure. The third is an alarm which indicates that the level in a vessel of

Table 5-4: Table presenting the 10 alarms with the highest scores resulting from equation 5-1. A brief description of each tag and the potential impacts associated with the alarms are shown in the left most columns [21].

Alarm	Occ.	Pred.	Prio.	Score	Tag Description	Potential Impact
15TT407.LLABS	657	0.518	1	20.63	Measure bottom temperature of C1505 Condensor	Potential trip
10PY224.HHABS	290	0.607	2	12.55	Cleanse frequency filters	Sub optimal performance
13LC040.HIABS	93	0.656	1	12.45	Level measurement transport oil supply	Sub optimal performance
200XT0206.HIABS	263	0.578	2	10.84	Measures vibration levels in fan	Equipment damage
210PT0010.HIABS	78	0.603	1	9.63	Measures pressure in transfer pipeline	Potential trip
17PC115.HIABS	55	0.655	1	9.55	Controls flue gas to furnace burners	Potential trip
10PT224.HHABS	539	0.451	2	9.44	Measures pressure over filters	Sub optimal performance
10FC047.LOABS	56	0.643	1	9.28	Controlling washing oil to heat exchanger	Equipment damage
15TC404.LOABS	256	0.434	1	9.04	Controls temperature in condensor by flow to heat exchanger	Sub optimal performance
10PC018.HIABS	106	0.528	1	8.61	Controls suction pressure in compressor stage	Potential trip

transport oil (oil used for the transport medium for catalyst) exceeds the designated range. The operators intentionally allow the vessel to be filled beyond this range as they require a larger quantity of transport oil for a specific automated procedure. As such, none of the three alarms was deemed interesting as a further automation opportunity.

The *15TT407.LLABS*, the *17PC115.HIABS*, and the *15TC404.LOABS* alarms are all associated with equipment which is directly connected to distillation columns. Distillation columns are complex pieces of equipment in the process industry with complex dynamical properties [51]. Therefore, the operational staff advised against correcting deviating behaviour using automated procedures. Other control methods, such as Model Predictive Control (MPC) were deemed more appropriate.

For the *200XT0206.HIABS* and the *10FC047.LOABS* alarms, the conclusion is that their impact is not severe enough to justify their automation of the response. The first is a vibration alarm on a fan (the alarm is part of the test set). The second is a flow alarm to a heat exchanger.

The *210PT0010.HIABS* and the *10PC018.HIABS* alarms were indeed deemed as interesting automation opportunities. The operational staff agreed that the required response to resolve the situation causing such alarms is relatively clearly defined, and they are relevant enough to be automated. In the next Section, these two alarms are further discussed, as well as the situations leading to these alarms and the possible automated resolving of those situations.

5-5 Proposal for automation opportunities

As explained in the previous Section, alarms *210PT0010.HIABS* and *10PC018.HIABS* are two alarms which scored high on the ranked list for automation opportunities. Operational staff agreed that these alarms can be good automation opportunities, as they are relatively straightforward to resolve, and influence few other installations.

5-5-1 210PT0010.HIABS

The *210PT0010* is a Pressure Transmitter in a pipeline in the solvent storage system of the SDA, shown in Figure 5-3. The solvent storage is schematically shown in Figure B-1 in Appendix B, where "solvent surge" is the same vessel as the V2002 in Figure 5-3. Alarm *210PT0010.HIABS* annunciates when the pressure in the pipeline between vessels V2101/2 and V2002 becomes too high. If the pressure in this pipeline increases beyond a certain threshold, the pressure relieve valve will open, and excess pressure will be relieved by disposing solvent via the flare. This situation should be prevented for environmental and efficiency reasons.

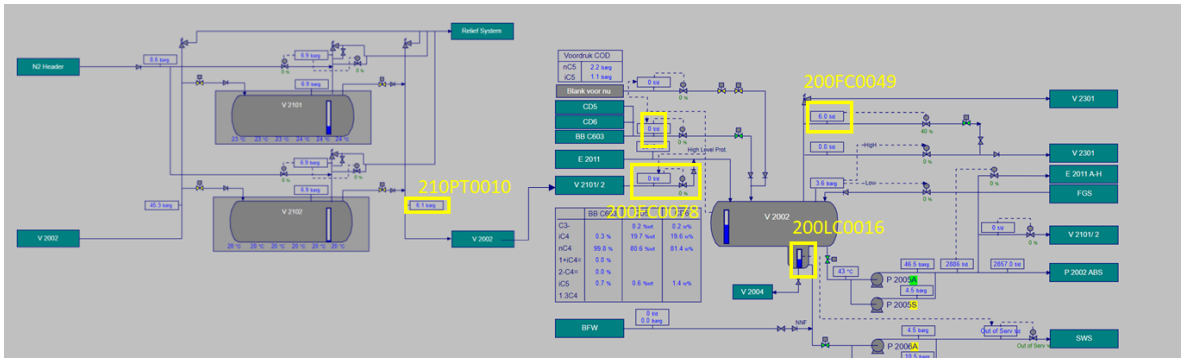

Figure 5-3: Overview of the solvent storage system in the SDA, of which *210PT0010* is a part. Various relevant tag numbers are highlighted.

In Table 5-5, the most frequent response pattern in response to alarm 210PT0010.HIABS is shown, along with the response patterns which have an edit distance of $d_e = 1$. All of the patterns increase the output of *200FC0078*, thereby increasing the flow to vessel V2002, which reliefs pressure and resolves the alarm. In the documentation of Shell, this is also the suggested solution [21]. In Figure 5-4, an occurrence of the pressure alarm *210PT0010.HIABS* annunciating, and the alarm being resolved by increasing the flow in the *200FC0078* is shown.

Other tags are also used in the patterns after the alarm has been resolved, namely the *200LC0017*, and the *200FC0049*. Both tags are shown in Figure 5-3.

The *200LC0017* is the level controller in the V2002. If more flow is supplied to the vessel, the level in the vessel will change. *200FC0049* controls the flow of purge gas out of the V2002. Purge gas is an inert gas present in a vessel to prevent the formation of an ignitable atmosphere. If a flow of gas is directed from V2101/2 towards V2002, the purge gas flow may need to be increased to prevent pressure build up in the V2002. As such, the patterns found for this alarm were found to be valid responses to the alarm, confirming the solution in

Table 5-5: Table showing the most frequent response pattern to alarm *210PT0010.HIABS*, and the patterns which have an edit distance d_e of 1 from this pattern.

	Response Pattern	**Supp.**	**Edit Dist.**
highest supp.	210PT0010.HIABS.UNACK_ALM, 210PT0010.toACK, 200FC0078.OUT.INC, 210PT0010.HIABS.ACK_RTN, 200FC0078.OUT.DEC	14	0
$d_n \leq 1$	210PT0010.HIABS.UNACK_ALM, 210PT0010.toACK, 200FC0078.OUT.INC, 210PT0010.HIABS.ACK_RTN, 200FC0078.OUT.INC, 200FC0078.OUT.DEC	12	1
	210PT0010.HIABS.UNACK_ALM, 210PT0010.toACK, 200FC0078.OUT.INC, 210PT0010.HIABS.ACK_RTN, 200FC0078.OUT.DEC, 200LC0017.SPT.DEC	10	1
	210PT0010.HIABS.UNACK_ALM, 210PT0010.toACK, 200FC0078.OUT.INC, 210PT0010.HIABS.ACK_RTN, 200FC0078.OUT.DEC, 200FC0049.SPT.INC	9	1
	210PT0010.HIABS.UNACK_ALM, 210PT0010.toACK, 200FC0078.OUT.INC, 200FC0078.OUT.DEC, 210PT0010.HIABS.ACK_RTN, 200FC0078.OUT.DEC	9	1

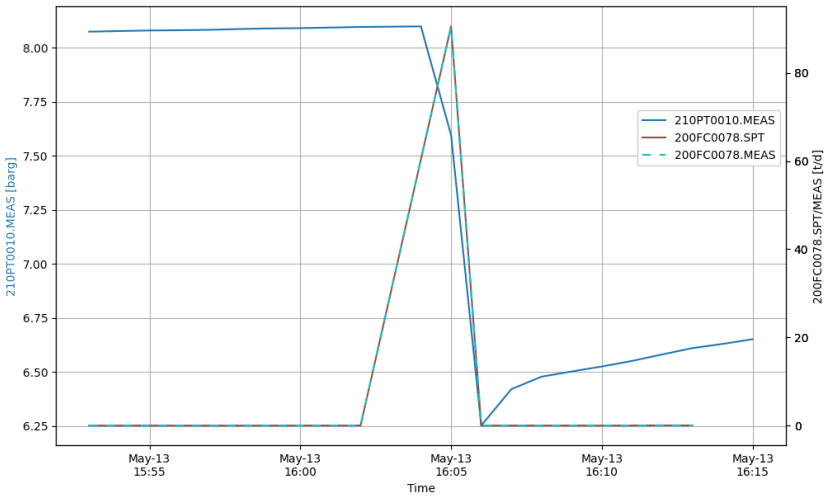

Figure 5-4: Graph showing the trend of the *210PT0010* measurement. On the left y-axis, the range for this measurement is are shown. On the right y-axis, the range for the flow through the *200FC0078* is shown. Both the set point and the measurements for this flow are shown in the graph. An occurrence of of the high pressure alarm is shown, as well as the resolving of that alarm using the flow controller.

the documentation, and identifying extra steps that need to be taken into account to ensure proper operation.

This can be a straightforward procedure to automate. It would prevent operators from having to increase the flow in the *200FC0078* themselves. Additional investigation is however needed to understand in what situations two other equipment tags need to be operated. For this, the data resulting from the analysis in this thesis can be used.

5-5-2 10PC018.HIABS

The *10PC018* is a Pressure Controller in the Fresh Gas Compressor K1001A, shown in Figure 5-5. This compressor is part of the Hycon plant, and is shown in Appendix A on the top of Figure A-1. The *10PC018* controls the pressure in the pipeline before compressor stage 2 of this compressor bank.

The most frequent pattern in the sequence database for alarm *10PC018.HIABS* is shown in Table 5-6. The similar patterns with an edit distance $d_e = 1$ from the most frequent pattern are also shown in the Table.

Table 5-6: Table showing the most frequent response pattern to alarm *10PC018.HIABS*, and the patterns which have an edit distance d_e of 1 from this pattern.

	Response Pattern	**Supp.**	**Edit Dist.**
Highest supp.	10PC018.HIABS.UNACK_ALM, 10PC018.toACK, 10HC055.OUT.INC, 10PC018.HIABS.ACK_RTN, 10HC055.OUT.DEC	18	0
$d_e \leq 1$	10PC018.HIABS.UNACK_ALM, 10PC018.toACK, 10HC055.OUT.INC, 10PC018.HIABS.ACK_RTN	10	1
	10PC018.HIABS.UNACK_ALM, 10PC018.toACK, 10HC055.OUT.INC, 10PC018.HIABS.ACK_RTN, 10HC055.OUT.DEC, 10HC055.OUT.INC	9	1

All of the patterns make use of the *10HC055* tag in order to resolve the alarm. The location of this tag is shown in Figure 5-5, along with other relevant tags. The tag represents a bias which is applied to the *10FC055* output. In Figure 5-6, the effect of increasing the bias can be seen. If the output of Flow Controller *10FC055* is increased, its corresponding valve is opened further, resulting in more gas flowing through the compressors. This increase in flow results in a smaller pressure difference in each compressor stage. The bias thus allows an operator to decrease the pressure *10PC018* by increasing the bias *10HC055*.

The Shell documentation gives a different solution for the resolving of the alarm however, by using the *10PC018*, *10PC200* or *10PC203* tag [21]. After closer inspection of the found patterns, it was found that those tags are in fact rarely used to resolve this alarm, instead the *10HC055* tag is used as stated above.

In order to automate the response to a high value in *10PC018*, a closer examination is required of the situations in which specific actions are chosen. While additional investigation is required, the analysis presented in this thesis offers valuable insights and data that can help in this analysis.

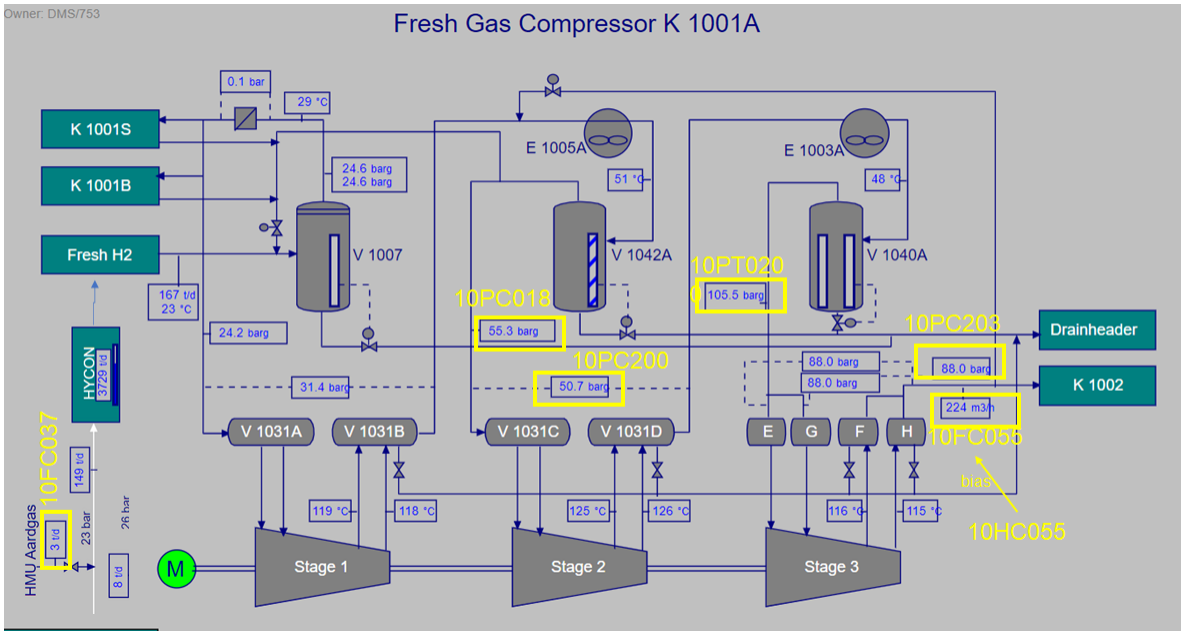

Figure 5-5: Overview of compression stages in the Hycon, of which *10PC018* is a part. Various relevant tag numbers are highlighted.

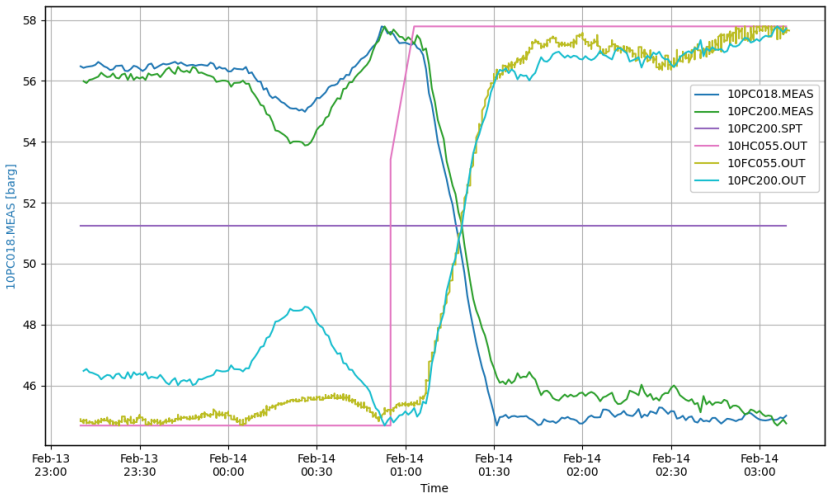

Figure 5-6: Graph showing the trends of various values relevant for the *10PC018* alarm. On the y-axis, the range for the value of *10PC018.MEAS* is shown, this concerns the pressure measurement in bar gauge. The ranges for the other values are omitted, because plotting them would make the graph cluttered. The relative trends of those values can however be discerned. The time range on the y-axis was chosen such that an alarm occurrence of alarm 10PC018 was present in the data.

Chapter 6

Conclusions and Recommendations

6-1 Project Summary

This thesis proposes a method for the analysis of stored process data to identify opportunities for procedural automation in processing plants. This method was designed to answer the following research question:

> *"How can commonly occurring actions patterns of operators in response to alarms in processing plants be extracted from stored process data for the identification and prioritization of procedural automation opportunities, considering the predictability of those patterns, their frequency, and the potential benefits of their automation?"*

The most effort was directed to designing the method for determining the predictability of an alarm response. The resulting method utilizes the physical layout of the process, combined with a statistical analysis of events occurring in the event log to identify the relevant events in response to each alarm. Sequential Pattern Mining is then applied to those events. The pattern mining results are condensed into a value for the predictability of the response to an alarm.

In order to ensure validity of the patterns resulting from the method, a training and test set were constructed, consisting of alarms and the response patterns known to be the correct responses to those alarms. The method was designed using the training set and validated using the test set, by applying the method to the alarms in those sets and observing if the correct response patterns were identified. Then, the method was applied to alarms in the event log of which the correct response was not yet known, to rank those alarms on their potential for automation.

The thesis was conducted in collaboration with Shell Energy and Chemicals Park Rotterdam, and the method is applied to data supplied by them. The most common alarm messages are evaluated for the response patterns required to resolve it, the predictability of those patterns

and the potential benefit of automating the response. From this, a ranked list of automation opportunities was constructed, which was then further discussed with operational staff.

Two alarms were identified that presented interesting opportunities for automation. The analysis also allows for a comparison between the found responses to alarms and the responses that are described in plant documentation.

The method was written such that it is easily applicable to datasets of other plants, provided that all required data is available: an event log containing both alarms and operator actions, and Process Engineering Flow Schemes (PEFSs) containing the physical layout of the process. It is however recommended that the parameters used in the method (Table 5-2) are tuned to each new plant, using a training and test set consisting of alarms of that plant.

6-2 Contributions

To the best of the authors knowledge, this thesis is the first work in literature that designs a method specifically to find automation opportunities based on the analysis of plant data. The method for identifying automation opportunities designed in this thesis consists of multiple parts. The following contributions were furthermore made during the design of this method:

- First, a statistical method for determining which event tags are relevant to the resolving of a specific alarm was developed, based on the Chi-Square test. This allows the method to only include a selected set of events in the pattern mining step, reducing noise of unrelated events and reducing computational time in the pattern mining step.

- Next, a novel pattern type called 'independent frequent patterns' was introduced to quantify patterns in response to an alarm: A pattern that occurs a *minsup* amount of times as an independent occurrence. An independent occurrence is any occurrence of a pattern which was not already counted as an occurrence for a larger pattern of which the smaller pattern is a subsequence.

  This pattern type gives a better indication of how often a particular response pattern resolves an alarm and reduces the disadvantages of other pattern types, which are either biased towards very short or very long sequences.

- A single value for the predictability of an alarm response was condensed from the resulting pattern database. By finding the edit distance between the most frequent pattern and all other patterns, similar response patterns were combined.

- Finally, a method was designed to rank automation opportunities based on the results of the analysis of the event log. This method ranks the alarms based on their occurrence frequency, predictability, and their priority. While this method gives a good initial ranking to determine which alarms are interesting to analyse further, it must be noted that conversations with operational staff remain important to understand if those alarms are indeed good automation opportunities.

6-3 Discussion and recommendations for future work

The results of this research show that an analysis of the data of a plant is a promising approach for the identification of automation opportunities in the response of operators to incoming alarms. It was however found that discussions with operational staff of the plant remain necessary for the final choice in what alarm responses are suitable for automation.

The current method focuses solely on alarms. This choice was made because incoming alarms supply a clearly defined moment at which operators are expected to start their response, and alarms often have a clearly documented expected response. Applying a pattern mining algorithm to an event log with the goal of finding general procedures which are interesting may however yield interesting results. It is in that case recommended that such a method is applied to a plant of which it is known that certain action patterns regularly arise, such as plants which produce in batches.

The chi-square test for finding relevant event-tags for each alarm shows good results. All tags in the training and test sets being among the event tags rated most likely to be relevant to the response of a particular alarm. The p-values resulting from the Chi-Square test are however extremely low. As such, there may be more appropriate statistical test methods for finding the event tags which are more likely to be relevant to an alarm.

Using the edit distance to compare the most frequent pattern to the other patterns showed promising results. It is however possible that a group of patterns which among themselves are very similar, but are all very different from the most frequent pattern, have a combined support that is higher than that of the frequent pattern. In such cases, that group of patterns will not be counted towards the predictability, since they are not similar enough to the most frequent pattern. A brief investigation was done in other clustering methods that may be able to better combine supports of similar patterns. From this, the use of Laplacian Eigenmaps [10] seems like a promising option to device such a scheme, and as such may be worth further exploring.

As stated in Chapter 5, the choice of parameters was performed by manual inspection of the results for different choices for the parameters. Using the training set, it is possible to design an optimization scheme that searches for the optimal value for each parameter. Here, the optimal value should be such that the support for the reference pattern for each alarm is as large as possible, while being the largest support out of all patterns. For such methods, it is however important that the set which is used for the optimization contains enough example patterns [27].

The amount of reference responses to alarms in the training and test set is currently limited to 10 in each set. This choice was made, because finding such reference responses was a time consuming task, and this research project was on tight schedule. In future research, such sets can be expanded to contain more examples. This is especially recommended if the optimization based approach to finding parameters explained above is used.

Appendix A

Hycon Overview

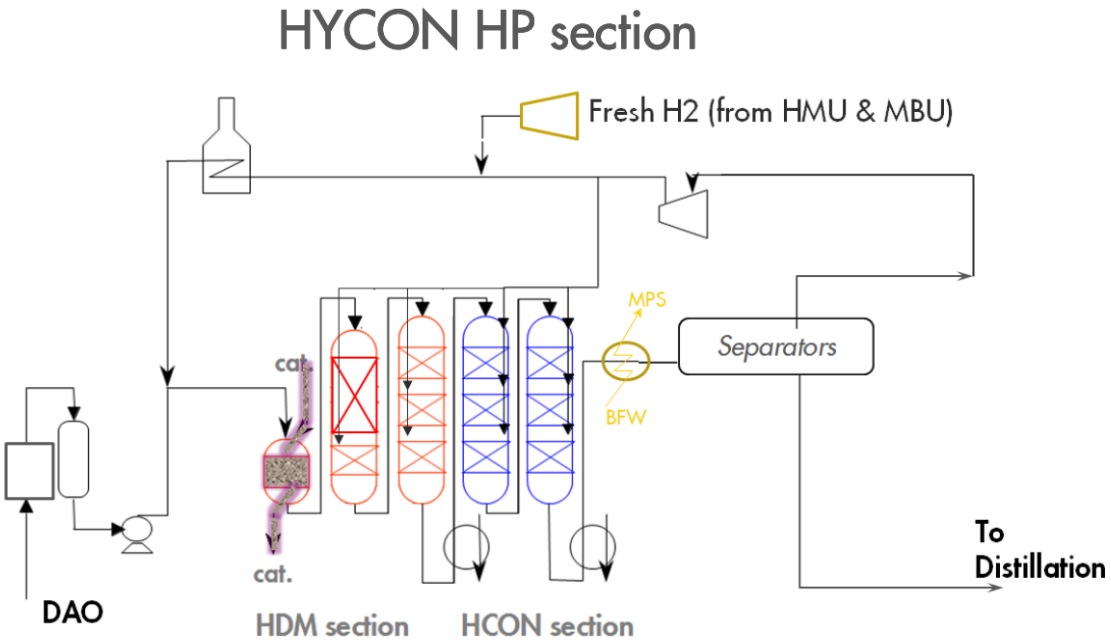

Figure A-1: Overview one of the Hydrogen Converter (Hydrogen Converter (Hycon)) plant [18], showing part where hydrocarbons are cracked into smaller molecules using hydrogen, takes feed from Solvent De-Asphalter (SDA).

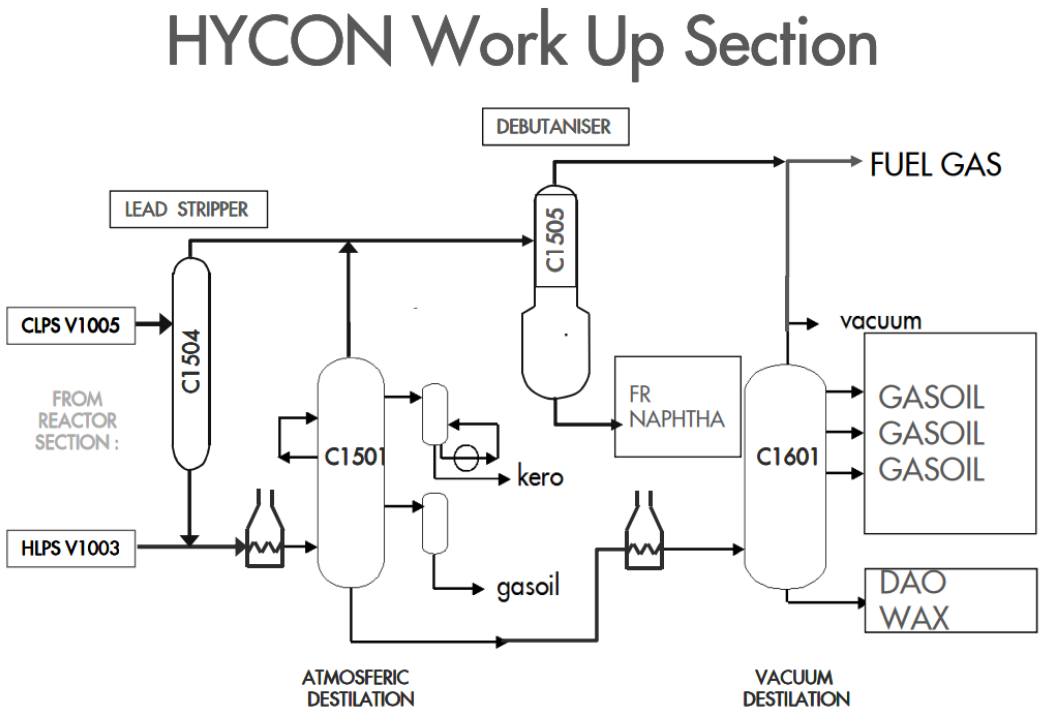

Figure A-2: Overview two of Hydrogen Converter (Hycon) plant [18], showing part where feed from the cracker is being separated into various different products.

Appendix B

SDA Overview

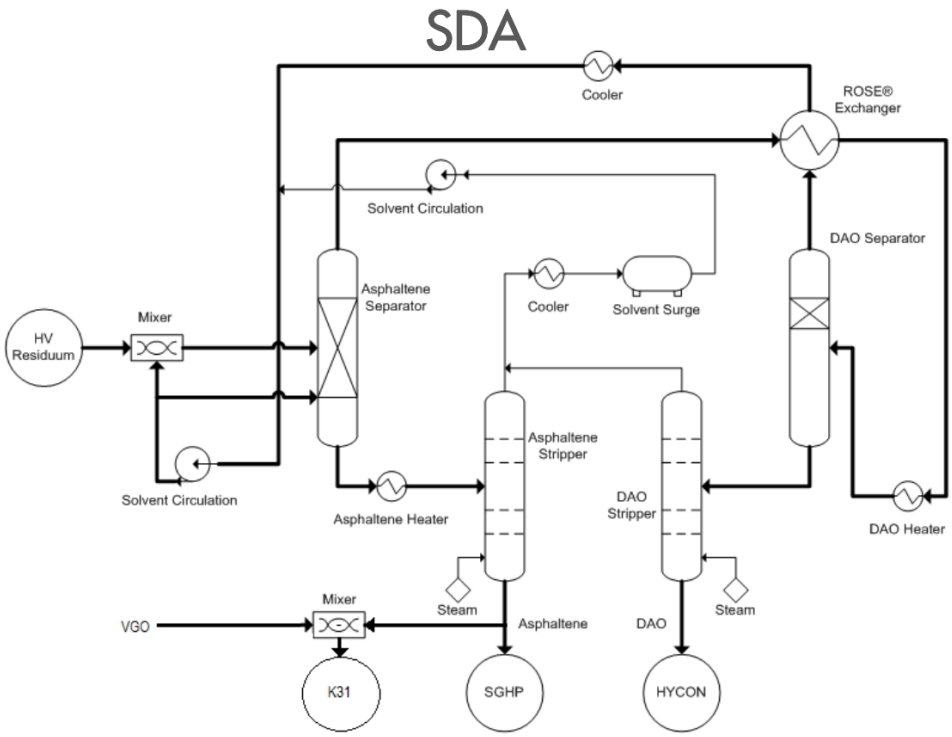

Figure B-1: Overview of Solvent De-Asphalter (SDA) plant, which creates De-Asphalted Oil (DAO) as feed for the Hydrogen Converter (Hycon) [18].

Appendix C

Operator Response Reference Sets

Table C-1: Training set of reference operator responses for a number of alarms.

Alarm	Reference Response Pattern
15TC404.LOABS	15TC404.LOABS.UNACK_ALM, 15TC404.toACK, 15TC404.SPT.INC, 15TC404.LOABS.ACK_RTN
15TC410.HIOUT	15TC410.HIOUT.UNACK_ALM, 15TC410.toACK, 15TC410.SPT.DEC, 15TC410.HIOUT.ACK_RTN
15TT406.LOABS	15TT406.LOABS.UNACK_ALM, 15TT406.toACK, 15TC404.SPT.INC, 15TT406.LOABS.ACK_RTN, 15TC404.SPT.DEC
17PC055.HIABS	17PC055.HIABS.UNACK_ALM, 17PC055.toACK, 17PC055.OUT.INC, 17PC055.HIABS.ACK_RTN, 17PC055.OUT.DEC
17PC149.LOABS	17PC149.LOABS.UNACK_ALM, 17PC149.toACK, 17PC149.SPT.INC, 17PC149.LOABS.ACK_RTN, 17PC149.SPT.DEC
200HY0037.LOABS	200HY0037.LOABS.UNACK_ALM, 200HY0037.toACK, 200FC0037.SPT.INC, 200HY0037.LOABS.ACK_RTN
200PC0081.LOOUT	200PC0081.LOOUT.UNACK_ALM, 200PC0081.toACK, 200FC0040.SPT.DEC, 200PC0081.LOOUT.ACK_RTN
200XT0200.HIABS	200XT0200.HIABS.UNACK_ALM, 200XT0200.toACK, 200PV0090A.MA, 200PV0090A.OUT.DEC, 200XT0200.HIABS.ACK_RTN, 200PY0090A_I.BI07.toSet, 200PY0090A_I.BI07.toReset
200XT0215.HIABS	200XT0215.HIABS.UNACK_ALM, 200XT0215.toACK, 200PV0090R.MA, 200PV0090R.OUT.DEC, 200XT0215.HIABS.ACK_RTN, 200PY0090R_I.BI07.toSet, 200PY0090R_I.BI07.toReset
210PT0010.HIABS	210PT0010.HIABS.UNACK_ALM, 210PT0010.toACK, 200FC0078.OUT.INC, 210PT0010.HIABS.ACK_RTN, 200FC0078.OUT.DEC

Table C-2: Test set of reference operator responses for a number of alarms.

Alarm	Reference Response Pattern
15FC046.LOABS	15FC046.LOABS.UNACK_ALM, 15FC046.toACK, 15FC046.OUT.INC, 15FC046.LOABS.ACK_RTN
15TC321.LOABS	15TC321.LOABS.UNACK_ALM, 15TC321.toACK, 15TC321.SPT.INC, 15TC321.LOABS.ACK_RTN
15YB020.LLABS	15YB020.LLABS.UNACK_ALM, 15YB020.toACK, 15FC016.SPT.INC, 15YB020.LLABS.ACK_RTN
16FC073.LOABS	16FC073.LOABS.UNACK_ALM, 16FC073.toACK, 16FC073.SPT.INC, 16FC073.LOABS.ACK_RTN, 16FC073.SPT.DEC
200HY0029.LOABS	200HY0029.LOABS.UNACK_ALM, 200HY0029.toACK, 200FC0029.SPT.INC, 200HY0029.LOABS.ACK_RTN
15TT407.LLABS	15TT407.LLABS.UNACK_ALM, 15TT407.toACK, 15TC404.SPT.INC, 15TT407.LLABS.ACK_RTN
200XT0203.HIABS	200XT0203.HIABS.UNACK_ALM, 200XT0203.toACK, 200PV0090D.MA, 200PV0090D.OUT.DEC, 200XT0203.HIABS.ACK_RTN, 200PY0090D_I.BI07.toSet, 200PY0090D_I.BI07.toReset
200XT0206.HIABS	200XT0206.HIABS.UNACK_ALM, 200XT0206.toACK, 200PV0090G.MA, 200PV0090G.OUT.DEC, 200XT0206.HIABS.ACK_RTN, 200PY0090G_I.BI07.toSet, 200PY0090G_I.BI07.toReset
200XT0210.HIABS	200XT0210.HIABS.UNACK_ALM, 200XT0210.toACK, 200PV0090L.MA, 200PV0090L.OUT.DEC, 200XT0210.HIABS.ACK_RTN, 200PY0090L_I.BI07.toSet, 200PY0090L_I.BI07.toReset
230TC0001.HIABS	230TC0001.HIABS.UNACK_ALM, 230TC0001.toACK, 230TC0001.SPT.DEC, 230TC0001.HIABS.ACK_RTN

Appendix D

Independent Pattern Mining - Python

Code Listing D.1: Python Code for the function which mines independent patterns from a closed pattern database. The function starts with the closed pattern database shown in Table 4-9b, and returns the independent patterns shown in Table 4-9c for $minsup = 2$. In Algorithm 2, the pseudo code of this algorithm is shown including comments.

```
# subpat function: returns True if [small] is a subpattern of [large]
def is_subpat(small, large):
    it = iter(large)
    return all(any(c == ch for c in it) for ch in small)

# mine the independent patterns and their supports
def mineIndependent(P_C,minsup):
    P_I = list()

    for pat_id, (pattern,supporting_seqs) in enumerate(P_C):
        S_p = supporting_seqs.copy()
        P_s = [p_c_id for p_c_id in range(0,pat_id) if is_subpat(pattern, P_C[
                                                    p_c_id][0])]

        for p_s in P_s:
            S_Ps = P_C[p_s][1]
            S_p = S_p.difference(S_Ps)

        if (len(S_p) >= minsup):
            P_I.append((pattern,S_p))
    return P_I

# start with closed pattern databsae:
P_C = [
(["a","c","b","f"], {1,2}),
(["a","b","f"], {1,2,3,4}),
(["a","f"],     {1,2,3,4,5,6}),
(["a","b"],     {1,2,3,4,5})]

P_I = mineIndependent(P_C,2)
print(P_I)
```

Bibliography

[1] Janos Abonyi and Gyula Dorgo. Process mining in production systems. In *2019 IEEE 23rd International Conference on Intelligent Engineering Systems (INES)*, pages 000267–000270. IEEE, 2019.

[2] Julie S Ågnes. Gaining and training a digital colleague: Employee responses to robotization. *The Journal of Applied Behavioral Science*, 58(1):29–64, 2022.

[3] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499. Santiago, Chile, 1994.

[4] Junyong Bae, Geunhee Kim, and Seung Jun Lee. Real-time prediction of nuclear power plant parameter trends following operator actions. *Expert Systems with Applications*, 186:115848, 2021.

[5] Junyong Bae and Seung Jun Lee. Framework for operator manipulation validation system using plant parameter prediction. In *Transactions of the Korean Nuclear Society Autumn Meeting*, pages 24–25, 2019.

[6] J.F. Bard. A Multiobjective Methodology for Selecting Subsystem Automation Options. *Management Science*, 32(12):1628–1641, 12 1986.

[7] Tord Bergquist, Jonas Ahnlund, and Jan Eric Larsson. Alarm reduction in industrial process control. In *EFTA 2003. 2003 IEEE Conference on Emerging Technologies and Factory Automation. Proceedings (Cat. No. 03TH8696)*, volume 2, pages 58–65. IEEE, 2003.

[8] Peter T Bullemer and John R Hajdukiewicz. A study of effective procedural practices in refining and chemical operations. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 48, pages 2401–2405. SAGE Publications Sage CA: Los Angeles, CA, 2004.

[9] Artur Bykowski and Christophe Rigotti. A condensed representation to find frequent patterns. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 267–273, 2001.

[10] Guangliang Chen. Laplacian eigenmaps. https://www.sjsu.edu/faculty/guangliang.chen/Math253S20/lec12laplacian.pdf. Accessed: 2023–8-10.

[11] Ming-Syan Chen, Jiawei Han, and Philip S. Yu. Data mining: an overview from a database perspective. *IEEE Transactions on Knowledge and data Engineering*, 8(6):866–883, 1996.

[12] Yue Cheng, Iman Izadi, and Tongwen Chen. Pattern matching of alarm flood sequences by a modified smith–waterman algorithm. *Chemical engineering research and design*, 91(6):1085–1094, 2013.

[13] Frederik Michel Dekking, Cornelis Kraaikamp, Hendrik Paul Lopuhaä, and Ludolf Erwin Meester. *A Modern Introduction to Probability and Statistics: Understanding why and how*, volume 488. Springer, 2005.

[14] Gyula Dorgo and Janos Abonyi. Sequence mining based alarm suppression. *IEEE Access*, 6:15365–15379, 2018.

[15] Gyula Dorgo and Janos Abonyi. Learning and predicting operation strategies by sequence mining and deep learning. *Computers & Chemical Engineering*, 128:174–187, 2019.

[16] Gyula Dorgo, Kristof Varga, and Janos Abonyi. Hierarchical frequent sequence mining algorithm for the analysis of alarm cascades in chemical processes. *IEEE Access*, 6:50197–50216, 2018.

[17] Gyula Dorgo, Kristof Varga, Mate Haragovics, Tibor Szabo, and Janos Abonyi. Towards operator 4.0, increasing production efficiency and reducing operator workload by process mining of alarm data. *Chemical Engineering Transactions*, 70:829–834, 2018.

[18] Shell Energy and Chemicals Park Rotterdam. "An introduction to all Pernis plants". Internal Company Document, 2017. Not publicly accesible, original source in Dutch, translated by author.

[19] Shell Energy and Chemicals Park Rotterdam. "ACM TAG INFORMATION - TAG#: U200_C41:200XT0215". Internal Company Document, 2023. Not publicly accesible, original source in Dutch, translated by author.

[20] Shell Energy and Chemicals Park Rotterdam. "SDA, 2000: PEFS - solvent surge drum". Internal Company Document, 2023. Not publicly accesible.

[21] Shell Energy and Chemicals Park Rotterdam. "Universal Tag Locator". Internal Company Document, 2023. Not publicly accesible, original source in Dutch, translated by author.

[22] Philippe Fournier-Viger, Yangming Chen, Farid Nouioua, and Jerry Chun-Wei Lin. Mining partially-ordered episode rules in an event sequence. In *Intelligent Information and Database Systems: 13th Asian Conference, ACIIDS 2021, Phuket, Thailand, April 7–10, 2021, Proceedings 13*, pages 3–15. Springer, 2021.

[23] Philippe Fournier-Viger, Antonio Gomariz, Manuel Campos, and Rincy Thomas. Fast vertical mining of sequential patterns using co-occurrence information. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 40–52. Springer, 2014.

[24] Philippe Fournier-Viger, Jerry Chun-Wei Lin, Antonio Gomariz, Ted Gueniche, Azadeh Soltani, Zhihong Deng, and Hoang Thanh Lam. The spmf open-source data mining library version 2. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part III 16*, pages 36–40. Springer, 2016.

[25] Philippe Fournier-Viger, Jerry Chun-Wei Lin, Rage Uday Kiran, Yun Sing Koh, and Rincy Thomas. A survey of sequential pattern mining. *Data Science and Pattern Recognition*, 1(1):54–77, 2017.

[26] Philippe Fournier-Viger, Cheng-Wei Wu, Antonio Gomariz, and Vincent S Tseng. Vmsp: Efficient vertical mining of maximal sequential patterns. In *Canadian conference on artificial intelligence*, pages 83–94. Springer, 2014.

[27] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow.* " O'Reilly Media, Inc.", 2022.

[28] Jerome Geyer-Klingeberg, Janina Nakladal, Fabian Baldauf, and Fabian Veit. Process mining and robotic process automation: A perfect match. In *BPM (Dissertation/Demos/Industry)*, pages 124–131, 2018.

[29] Berno Theo Greyling and Wyhan Jooste. The application of business process mining to improving a physical asset management process: a case study. *South African Journal of Industrial Engineering*, 28(2):120–132, 2017.

[30] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, sep 2020.

[31] Wenkai Hu, Ahmad W Al-Dabbagh, Tongwen Chen, and Sirish L Shah. Process discovery of operator actions in response to univariate alarms. *IFAC-PapersOnLine*, 49(7):1026–1031, 2016.

[32] Alan J Hugo. Estimation of alarm deadbands. *IFAC Proceedings Volumes*, 42(8):663–667, 2009.

[33] Richard Karoly and Janos Abonyi. Multi-temporal sequential pattern mining based improvement of alarm management systems. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 003870–003875. IEEE, 2016.

[34] Eric Knapp. Chapter 5 - how industrial networks operate. In Eric Knapp, editor, *Industrial Network Security*, pages 89–110. Syngress, Boston, 2011.

[35] Tomokazu Konishi, Shiori Matsukuma, Hayami Fuji, Daiki Nakamura, Nozomi Satou, and Kunihiro Okano. Principal component analysis applied directly to sequence matrix. *Scientific Reports*, 9(1):19297, 2019.

[36] Shiqi Lai and Tongwen Chen. A method for pattern mining in multiple alarm flood sequences. *Chemical Engineering Research and Design*, 117:831–839, 2017.

[37] Vladimir I Levenshtein et al. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union, 1966.

[38] Vladimir Likic. The needleman-wunsch algorithm for sequence alignment. *Lecture given at the 7th Melbourne Bioinformatics Course, Bi021 Molecular Science and Biotechnology Institute, University of Melbourne*, pages 1–46, 2008.

[39] Elham Naghoosi, Iman Izadi, and Tongwen Chen. Estimation of alarm chattering. *Journal of Process Control*, 21(9):1243–1249, 2011.

[40] Takanobu Nakahara and Katsutoshi Yada. Analyzing consumers' shopping behavior using rfid data and pattern mining. *Advances in Data Analysis and Classification*, 6:355–365, 2012.

[41] The pandas development team. pandas-dev/pandas: Pandas, feb 2020.

[42] Md Rezwan Parvez, Wenkai Hu, and Tongwen Chen. Comparison of the smith-waterman and needleman-wunsch algorithms for online similarity analysis of industrial alarm floods. In *2020 IEEE Electric Power and Energy Conference (EPEC)*, pages 1–6. IEEE, 2020.

[43] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering frequent closed itemsets for association rules. In *Database Theory—ICDT'99: 7th International Conference Jerusalem, Israel, January 10–12, 1999 Proceedings 7*, pages 398–416. Springer, 1999.

[44] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572, 1901.

[45] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Transactions on knowledge and data engineering*, 16(11):1424–1440, 2004.

[46] Janne Pietilä and Olli Haavisto. Comparison of operator performance in a mineral processing plant. In *2010 IEEE International Conference on Systems, Man and Cybernetics*, pages 2773–2779. IEEE, 2010.

[47] Marcos Quiñones-Grueiro, Alberto Prieto-Moreno, Cristina Verde, and Orestes Llanes-Santiago. Data-driven monitoring of multimode continuous processes: A review. *Chemometrics and Intelligent Laboratory Systems*, 189:56–71, 2019.

[48] AA Raorane, RV Kulkarni, and BD Jitkar. Association rule–extracting knowledge using market basket analysis. *Research Journal of Recent Sciences ISSN*, 2277:2502, 2012.

[49] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[50] Markus Schleburg, Lars Christiansen, Nina F Thornhill, and Alexander Fay. A combined analysis of plant connectivity and alarm logs to reduce the number of alerts in an automation system. *Journal of process control*, 23(6):839–851, 2013.

[51] Sigurd Skogestad et al. Dynamics and control of distillation columns: A tutorial introduction. *Chemical Engineering Research and Design*, 75(6):539–562, 1997.

[52] Temple F Smith, Michael S Waterman, et al. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

[53] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[54] Michał Walicki and Diogo R Ferreira. Sequence partitioning for process mining with unlabeled event logs. *Data & Knowledge Engineering*, 70(10):821–841, 2011.

[55] David Walker. Using machine learning to enhance operator performance. *The APPEA Journal*, 60(2):681–684, 2020.

[56] Jiandong Wang and Tongwen Chen. An online method to remove chattering and repeating alarms based on alarm durations and intervals. *Computers & Chemical Engineering*, 67:43–52, 2014.

[57] Jianyong Wang and Jiawei Han. Bide: Efficient mining of frequent closed sequences. In *Proceedings. 20th international conference on data engineering*, pages 79–90. IEEE, 2004.

[58] Maurice Wilkins and Marcus Tennant. Advances in procedural automation in the chemical industry. In *Computer Aided Chemical Engineering*, volume 31, pages 1115–1119. Elsevier, 2012.

[59] Maurice J Wilkins and Steve Lazok. Modular procedural automation innovation for upstream processes. In *SPE Intelligent Energy International*. OnePetro, 2012.

[60] Youxi Wu, Changrui Zhu, Yan Li, Lei Guo, and Xindong Wu. Netncsp: Nonoverlapping closed sequential pattern mining. *Knowledge-based systems*, 196:105812, 2020.

[61] Xifeng Yan, Jiawei Han, and Ramin Afshar. Clospan: Mining: Closed sequential patterns in large datasets. In *Proceedings of the 2003 SIAM international conference on data mining*, pages 166–177. SIAM, 2003.

[62] Mohammed J Zaki and Ching-Jui Hsiao. Charm: An efficient algorithm for closed itemset mining. In *Proceedings of the 2002 SIAM international conference on data mining*, pages 457–473. SIAM, 2002.

[63] Teresa Zayas-Cabán, Saira Naim Haque, and Nicole Kemper. Identifying opportunities for workflow automation in health care: lessons learned from other industries. *Applied Clinical Informatics*, 12(03):686–697, 2021.

[64] Minhaz Fahim Zibran. Chi-squared test of independence. *Department of Computer Science, University of Calgary, Alberta, Canada*, 1(1):1–7, 2007.

Glossary

Notation	Description	Page List
C	A set of candidate patterns	19
E	The set of all possible events	9, 10, 25
P	The set of all patterns found in the sequence database	10, 17–19, 42, 44
S	The set of sequences of events	10, 17–19, 44
Z	A database with al maximal patterns found	20
$\mathbb{R}^+$	Set of all positive real numbers	9, 25
$\mathcal{A}_v$	The set of actions for event tag v	9, 25, 26
$\mathcal{L}$	The database with all occurred events	10, 15, 25
$\mathcal{P}$	A set of PEFSs p	30, 31
$\mathcal{V}$	The set of all possible tags	9, 25
$minsup$	Minimum support of a sequence to be considered a pattern	10, 17–19
p	The mathematical representation of a PEFS	30, 31
π	A segmentation window	15
σ	A trace window	16
a	The action of an event	9, 25, 31, 76
c	A candidate pattern	19
d_e	The edit distance allowed for a pattern to be named a similar pattern.	22, 53–55

Notation	Description	Page List
e	A single event, consisting of a time stamp t, a tag v, and an activity a: $e = (t, v, a)$	9, 10, 15, 16, 18, 19, 25, 29, 31, 39
p	A pattern, or frequent sequence, in the pattern database	10, 19
s	A sequence of events	10, 18–20
t	The time stamp of an event	9, 15, 16, 25, 31, 76
v	The tag of an event	9, 25, 29, 31–33, 76
closed frequent pattern	A pattern that is not strictly included in another pattern with the same support	17–19, 40–42
event	A discrete occurrence representing either an alarm or an operator action.	10, 16, 17, 41
event log	The plant log containing all events occurring over a time period	10, 34
event tag	A label given to an action or alarm for the identification of the equipment it is associated with.	28, 30–32, 34, 35
frequent pattern	A pattern that occurs a minimum number of times in the sequence database.	17–20, 40
independent frequent pattern	A pattern that occurs a *minsup* amount of times as an independent occurrence.	42, 49
maximal frequent pattern	A closed pattern that is not strictly included in any other closed pattern	17–20
min supp	Minimum support of a sequence to be considered a pattern	18
non trivial sequence	A sequence resulting from an alarm window that contains operator actions between the alarm starting and stopping to annunciate.	50

Notation	Description	Page List
pattern	A sequence occurring a set minimum amount of times in a database.	10, 11, 16, 17, 35, 36, 40, 41
pattern database	A database containing all pattern found in the sequence database.	41
predictability	The degree to which the operator actions in response to an alarm can be anticipated or foreseen based on the patterns observed in the event logs	2, 21, 44, 49
procedural automation	The automation of procedures within a process industry plant	1
reference response	The pattern that is, based on documentation and consult with operational staff, the expected response to an alarm	49
sequence	A number of events occurring sequentially.	10, 16, 36, 40
sequence database	A database containing all sequences extracted from the event log	10, 16, 20, 36, 40, 41
sequential pattern mining	The searching of patterns in a sequence database	37
subsequence	A sequence contained by another sequence	10, 40, 41
support	The amount of sequences in the sequence database containing a specific pattern	18, 20, 37, 39–44

Acronyms

Notation	Description	Page List
DAO	De-Asphalted Oil	63
DCS	Distributed Control System	28
HMI	Human Machine Interface	1, 9, 26
Hycon	Hydrogen Converter	5, 25, 51, 55, 56, 61–63
MPC	Model Predictive Control	52
PCA	Principal Component Analysis	22, 23, 45
PEFS	Process Engineering Flow Scheme	12, 13, 28–31, 34, 35, 58
PMF	Probability Mass Function	33
SDA	Solvent De-Asphalter	5, 25, 51, 53, 61, 63
SOP	Standard Operating Procedure	1
SPM	Sequential Pattern Mining	7, 10–12, 15–17, 20, 36–38, 40, 44