Event based diagnosis of process systems

DOI:10.18136/PE.2017.640

PhD Thesis

Author: **Attila Tóth**

Supervisor: **Katalin M. Hangos, DSc.**

University of Pannonia

Faculty of Information Technology

Doctoral School of Information Science and Technology

2016

EVENT BASED DIAGNOSIS OF PROCESS SYSTEMS

Értekezés doktori (PhD) fokozat elérése érdekében

Írta:
Tóth Attila

Készült a Pannon Egyetem Doktori Iskolájának keretében

Témavezetők: Dr. Hangos Katalin
Elfogadásra javaslom (igen / nem) ……..
(aláírás)

A jelölt a doktori szigorlaton …….%-ot ért el.

……..
(aláírás)

Az értekezést bírálóként elfogadásra javaslom:

Bíráló neve: …… igen / nem

……..
(aláírás)

Bíráló neve: …… igen / nem

……..
(aláírás)

Veszprém,……. ……..
a Bíráló Bizottság Elnöke

A doktori (PhD) oklevél minősítése:

……..
az EDT elnöke

Contents

Acknowledgments **7**

Abstract **10**

1 Introduction **15**
1.1 Motivation 15
1.2 Review of discrete methods applied in process systems 15
1.2.1 Diagnostic methods based on hazard identification information 16
1.2.2 Diagnostic methods based on clustering and statistics . 17
1.2.3 Approaches to diagnostic decomposition 18
1.3 Problem statement 19
1.3.1 Related common taxonomy 20
1.4 Thesis structure 21

2 Basic notions **23**
2.1 System model 23
2.2 The diagnostics task 24
2.3 Inputs and outputs, qualitative range space 25
2.4 Events 25
2.5 Traces 26
2.6 Faults 28
2.7 A simple composite process system 29
2.7.1 Nominal trace 30
2.7.2 Faults 30
2.8 Summary 33

3 P-HAZID diagnostics **35**
3.1 Introduction 35
3.2 Basic notions 36
3.2.1 Flattening of traces 37

3.2.2 Deviations . . . 37
3.2.3 P-HAZID table . . . 39
3.3 P-HAZID based diagnostics . . . 41
3.3.1 Additional assumptions . . . 42
3.3.2 Algorithm description . . . 42
3.3.3 Analogy with if-then rules . . . 44
3.4 Diagnostics case study . . . 44
3.4.1 Flattening of the traces . . . 45
3.4.2 Acquiring deviations from faulty traces . . . 45
3.4.3 Reasoning graphs . . . 46
3.4.4 P-HAZID table . . . 47
3.4.5 Execution of the algorithm . . . 47
3.4.6 Observations . . . 50
3.5 Summary . . . 51

4 Diagnostics using clustering **53**
4.1 Introduction . . . 53
4.2 Basic notions . . . 54
4.2.1 Mapping of qualitative values to real ones . . . 54
4.2.2 Coordinate-vectors of events and traces . . . 56
4.2.3 Event and trace distances . . . 57
4.3 The diagnostic method . . . 58
4.3.1 Additional assumptions . . . 59
4.3.2 Clustering of traces . . . 59
4.3.3 Steps of the Diagnostic Procedure . . . 61
4.3.4 Dealing with faults not considered a priori in the training set . . . 62
4.3.5 Trace coordinates with different measurement units . . . 62
4.3.6 Limitation on the lengths of traces . . . 63
4.4 A diagnostic case study . . . 63
4.4.1 Measurement errors . . . 64
4.4.2 Description of the case study . . . 65
4.4.3 Execution of the algorithm . . . 65
4.4.4 Effect of different mapping functions . . . 69
4.4.5 Observations . . . 71
4.5 Summary . . . 72

5 Decomposition in event-based diagnostics **75**
5.1 Introduction . . . 76
5.2 Components, component structure and trace decomposition . . . 76
5.2.1 Component . . . 77

5.2.2 Component graph and component path 77
5.2.3 Start condition 79
5.2.4 Component mapping function 79
5.2.5 Trace fragments 80
5.3 Component based diagnostics 81
5.3.1 Additional assumptions 81
5.3.2 General algorithm 82
5.4 A diagnostics case study 84
5.4.1 Determining the component graph 84
5.4.2 Nominal and faulty traces for components 85
5.4.3 Component based P-HAZID diagnoser 85
5.4.4 Component based Clustering diagnoser 85
5.4.5 Observations 88
5.5 Summary 90

6 Conclusion **91**
6.1 Theses 91
6.1.1 Thesis 1 - the P-HAZID diagnoser (Chapter 3) 91
6.1.2 Thesis 2 - the Clustering diagnoser (Chapter 4) 92
6.1.3 Thesis 3 - decomposition method for event-based diagnostics (Chapter 5) 93
6.2 Own publications supporting the thesis points 93
6.3 Further research directions 94
6.3.1 On-line diagnostics 94
6.3.2 Improving robustness during clustering 95
6.3.3 Validation of P-HAZID tables 95
6.3.4 Further application in real process systems 95
6.3.5 Unifying the P-HAZID and the Clustering diagnoser . 96

7 Appendix **97**
7.1 The P-HAZID reasoning method 97
7.2 Realistic use-case for the Clustering diagnoser 100
7.2.1 Tennessee-Eastman process 101
7.2.2 Preparation of the data 102
7.2.3 Results 103
7.2.4 Observations based on the results 106

Bibliography **109**

Acknowledgments

First and foremost I would like to thank my supervisor, Professor Katalin M. Hangos for her endless support and motivation during my PhD studies. I could not have been able to get this far without all the guidance she gave to me during these years.

I would also like to thank my wife for supporting me during all the long evenings and weekends when I was way too busy doing the lengthy work of the research. I will always remember all the kind words you have said when the end of this very long journey seemed too distant.

Last but not least, I would like to thank my parents for all their support and encouragement during all of my school and university years. Without them it would not have been possible for me to pursue a study of this level, which I always wanted. Specifically, I would like to thank my father for introducing me to the fascinating world of computer science.

Abstract

In order to prevent or mitigate losses caused by plant faults, early and accurate fault diagnostics during the operation of modern day process systems is a very important task. In this dissertation a few diagnostics methods for complex composite process systems controlled by operational procedures are presented.

First a methodology for capturing expertise about the execution process system operational procedures in the form of specifically constructed spreadsheets (P-HAZID tables) is described. Experts of the process system can use this methodology to create a fault model and store diagnostics related knowledge about the system. A reasoning algorithm is proposed, which can perform fault diagnosis based on this model and the actual dynamics of the plant. This method is similar to the widely known if-then rules from rule-based expert systems in artificial intelligence but extends it with the ability to reason about differences between event sequences.

A solely observation-based diagnostic approach which can work on observations of nominal and faulty modes of a process system is also proposed. The model of this diagnostic method uses a popular machine-learning technique called clustering. The algorithm suggests fault modes for the current operational procedure of the process system based on historical operational procedure logs gathered from past runs of the operational procedure under different faulty conditions.

One can see value in decomposing these diagnostic ideas to enable their use in more complex process systems and diagnostic problems. Therefore a diagnostic decomposition approach is proposed which can be used to scale up standalone diagnostic methods by bringing the diagnostics down to the level of the components from the level of the system.

Előszó

A korai és pontos hibadiagnosztikának nagy szerepe van napjaink bonyolult folyamatrendszereinek üzemeltetése során, annak érdekében hogy az esetleges meghibásodások által okozott veszteséget elkerüljük vagy csökkentsük. Ebben a disszertációban pár újszerű diagnosztikai módszer kerül ismertetésre, amit operátori eljárások által vezérelt folyamatrendszerek hibadiagnosztikájára lehet felhasználni.

Első körben egy olyan módszer kerül ismertetésre, amivel a folyamatrendszer szakértői üzemeltetési tudásukat speciális táblázatok (P-HAZID táblázatok) formájában tudják rögzíteni. Így lehetőségük nyílik arra is hogy diagnosztikai ismereteket és a hibás működés modelljeként táblázatos formában eltárolják. Egy következtetési algoritmus is ismertetésre kerül, ami képes hibadiagnosztikát végrehajtani ezen táblázat és a folyamatrendszer működése alapján. Ez a módszer a széleskörben ismert ha-akkor szabályokon alapul, kiegészítve azt az eseménysorozatokra vonatkozó következtetés képességével.

Egy pusztán megfigyelés-alapú diagnosztikai módszer is ismertetésre kerül, ami a folyamatrendszer normális és hibás megfigyelt működési módjai alapján működik. Ennek alapja egy népszerű gépi tanulási megközelítés, a csoportosítás (clustering). Ennek során a korábban felvételre került normális és hibás működési módok összehasonlításra kerülnek az aktuális működéssel, ez alapján javasol a módszer lehetséges meghibásodásokat.

Mivel nagy rendszerek és bonyolult diagnosztikai problémák esetén a felhasznált módszereket érdemes dekomponálni, ezért egy dekompozíciós megközelítés is ismertetésre kerül a disszertáció végén. Ennek segítségével egy nagyobb rendszerszintű diagnosztikai módszer több kisebb, egyszerűbb komponensszintű módszer összetételévé alakítható át.

Zusammenfassung

Heute die frühe und genaue Fehlerdiagnose spielt eine große Rolle in den Betrieb von komplexen Prozesssysteme, um die potenzielle, durch den Ausfall verursachten Verluste zu vermeiden oder zu reduzieren. In dieser Arbeit wird ein paar neue diagnostische Verfahren beschrieben, die für die Fehlerdiagnose von Operatorsverfahren gesteuerte Prozesssysteme verwendet werden können.

In der ersten Runde wird so ein Verfahren beschrieben, wobei die Expertenwissen über Betriebsbereich in Form von speziellen Tabellen (P-HAZID Tabellen) aufgezeichnet werden können. Auf dieser Weise werden wir die Gelegenheit haben die Fehlermodel zu erzeugen und die Diagnostikerfahrungen über das System zu speichern. Es wird auch eine Algorithmus dargelegt, welche fähig ist mit der Hilfe der P-HAZID Tabellen und der Prozesssysteme eine Fehlerdiagnose auszuführen. Dieses Verfahren ist weit bekannt und basiert auf den Regeln "Wenn-Dann". Dieses ergänzt den Regeln mit der Fähigkeit der Schlussfolgerung.

Eine bloße Beobachtung basierte Diagnoseverfahren ist ebenfalls vorgesehen, die auf der Grundlage der beobachteten normalen und fehlerhaften Betrieb des Prozesssystems arbeitet. Es basiert auf einem populären Maschinenlernansatz, auf der Gruppierung (Clustering). So die zuvor aufgezeichneten normalen und fehlerhaften Betriebsarten werden mit den aktuellen Betrieb vergleicht. Anhand die Ergebnisse der Vergleichungen wird Vorschlage über die Defekte gegeben.

Es wird auch eine Zersetzungsannäherung beschrieben, da es lohnt sich die verwendeten Methoden auseinander nehmen bei den großen Systemen und komplizierten Problemen. Aus mehreren, kleineren und einfacheren komponentenebene Methode kann eine größere Systemebene Methode aufgebaut werden.

Chapter 1

Introduction

1.1 Motivation

Early and accurate fault diagnostics is one of the most important challenges during the operation of modern day process systems. Primeval fault mitigation and isolation due to proper diagnostics plays a crucial role in avoiding huge losses and plant breakdowns caused by the consequences of initially smaller and isolated but propagating failures discovered too late. The application of an intelligent fault diagnostics solution for a plant might lead to safer and less complicated operation with decreased operational costs, and leaves less possibilities for human errors.

The importance of the field is indubitable, on the other hand, creating a proper diagnostic solution is not an easy task. Many aspects of the plant need to be taken into account and there is no general solution which fits all scenarios. Numerous diagnostic solutions are available in the literature, ranging from simple techniques (for instance, simple logics in the controller of the system) to very complex ones (such as rule based expert systems).

The motivation of this work is to contribute a few novel event-based qualitative diagnostics methods to this field of study.

1.2 Review of discrete methods applied in process systems

Because of its vital importance, the literature of discrete diagnostic approaches is enormously wide, with the model-based methods for fault detection and isolation are the most widespread.

1.2.1 Diagnostic methods based on hazard identification information

Depending on the a priori and measured information that is available for diagnostics, the fault diagnostic approaches for process systems are categorized into quantitative, qualitative and process history based approaches in [51]. Model-based qualitative diagnostic approaches apply approximate "qualitative" information for both the models and the measured signatures or symptoms.

Process fault diagnostics based on process and fault models had been widely described by Venkatasubramanian in review articles [49], [48] and [50]. According to [48], model based a priori knowledge can be broadly classified as quantitative and qualitative. Fault detection using these qualitative models can be performed by using expert systems with different kind of reasoning, using signed directed graphs (SDGs) for modeling cause-effect relations (for instance in [47]) or fault trees describing the relations between primary events to top level events or hazards. Fault propagation analysis [21] can be used for the identification of faults, causes and consequences in a systematic manner. In the last review article of the series by Venkatasubramanian on process systems diagnostics ([50]), process history based methods are surveyed. Instead of an apriori model, these methods require a large amount of historical process data, and they can be classified by the way they extract information from the process data (this operation is called feature extraction). Feature extraction can be qualitative (for example using rule-based expert systems or qualitative trend analysis) and quantitative (using statistical methods, such as PCA or neural networks).

Hazard identification (HAZID, see [16]) has been long taken as an independent activity from diagnostics, but the information they built on has a lot of common elements. The HAZOP (Hazard and Operability, see [5], [15], [26]) and FMEA (Failure Effect and Mode Analysis, see [6]) are two fundamentally different analysis methods for hazard identification, where HAZOP is deviation-driven and FMEA is process component-driven. Due to the complexity of real process systems, the time-consuming and error-prone manual construction of HAZOP and FMEA tables has been identified as a major bottleneck in hazard identification of process systems. Fortunately, there have been results for automated generation of them (in the previously mentioned review articles and for HAZOP, in [51] together with a concrete application in [49]). Another possible way is to use qualitative models in such an analysis (again, for HAZOP, see [10] with a batch process system application in [36]). An attempt to unite the two different diagnostic information stored in HAZOP and FMEA analysis results, called the blended HAZID methodol-

ogy was described in [42] together with its use for process system diagnostics tasks.

The blended HAZID approach (described thoroughly in [42]) combines the system-driven HAZOP and the component-driven FMEA into one methodology, attempting to minimize their weaknesses and utilize their strengths at the same time. Unlike the original HAZOP and FMEA, three main steps in the initial analysis are needed (based on [42]):

1. Decompose the system into subsystems - the analysis is done in subsystem level onwards.

2. Find deviations from intended functions, with their causes and implications.

3. Elicit the causes and effects of each fault per component in every subsystem on the function of the system.

As a result of the analysis, a cause-implication directed graph (see [34] for an example) can be drawn for each identified failure to visualize casual relationships between failures and components. The nodes in this graph are either components or functional failures and each edge represents a causal relationship between them. This graph is a powerful visualization tool for plant operators.

For describing arbitrary output signal values qualitative trend analysis (QTA) can be used, by comparing qualitative trends of nominal and actual signal values (a good example can be found in [30]). In some newer results (in [31]), these methods have been even combined to perform fault diagnostics.

1.2.2 Diagnostic methods based on clustering and statistics

As a technique used thoroughly in machine learning, clustering is used in systems for process diagnostics. For instance, as a popular method in the field, the k-means clustering algorithm (refer to [3]) is used for process system modeling in [17]. Different other approaches are using the fuzzy c-means clustering (FCM, described in [2]), a method based on the concept of fuzzy sets and logic (described originally in [28]). For example fuzzy c-means clustering for fault classification is reported in [32] and [37] while it is used for process control in [25].

A special type of historical process data are the so called alarms, the timed sequence of which has been utilized for early fault detection and diagnostics in [1]. These alarm sequences can be thought as event logs. In [46] a

process mining tool called ProM is described which is capable of discovering process models in the form of Petri Nets, using event logs collected from process systems. This tool supports conformance checking, verification, model extension and transformation as well as model discovery. A ProM extension described in [17] uses k-means clustering for categorizing event logs prior to mining them, in order to achieve faster operation. In a slightly different approach described in [40], Petri nets are used to build up models from event sequences, and the fitness and appropriateness of the model is calculated.

The most widely used quantitative feature extraction procedures use statistical methods (e.g. PCA or PLS) for process monitoring and fault detection, for which good review papers have appeared recently, see [54], [38] or [24]. A recent improvement of the PLS method capable of detecting small faults have been reported in [23]. However, these methods usually assume steady-state operation condition of the system to be diagnosed, and fail during transient operations.

1.2.3 Approaches to diagnostic decomposition

The concept of decomposing a problem into related subproblems is used widely in mathematics and computer science in order to solve bigger problems by combining the solutions for their smaller subproblems. This idea had been generalized for process system diagnostics as well in numerous cases in the past. Process systems are mostly built from similar components, therefore by decomposing them, and diagnosing the components one by one might be more effective than diagnosing the whole system at once. As described in [41] artificial intelligence based searching techniques (such as constraint satisfaction search) fall into the set of NP-hard problems (see [29]). A subset of fault diagnostics approaches uses these techniques, therefore the idea of decomposition is used at a few related works already in the field of process system diagnostics (such as the previously mentioned BL-HAZID methodology) to cope with the problem of NP-hardness.

The original HAZID method (on which the BL-HAZID approach is based on, see [5]) provides a systematic process in which it breaks down the process system into separately manageable sub-components for collecting hazards of the system. Various hazards are collected separately for the sub-components by the HAZID team performing the study, making their task less complex.

A distributed on-line diagnostics framework is proposed in a recent article [19] for fault detection, isolation and identification. This diagnostic approach scales better than the traditional centralized approaches for model-based diagnostics and it uses an extension of the Possible Conflicts (PC) decomposition technique (see [33]). The PC decomposition technique originally requires

the use of a central coordinator, but in this extended case the diagnostics is performed by independent distributed diagnosers which do not need any coordination over or communication between them.

Qualitative physics is used for process system modeling as a common sense reasoning about physical systems. This approach is based on qualitative or ordinary differential equations describing the process system to be diagnosed. These qualitative dynamic models together with many different methods use an abstract hierarchy of process knowledge which is based on decomposing the process system into subcomponents, in order to decrease computational complexity and speed up the diagnostics task.

A process system structural decomposition method for performing diagnosis based on qualitative physics is described in [21], where the the physical connections and fault propagation through them is emphasized. Apart from this, the article describes an object-oriented process system topology modeling approach with which the component-specific fault model can be shared between components of the system.

The laws of qualitative physics can be described using the concept of *qualitative reasoning*. This technique attempts to cover physical laws of the system as qualitative rules instead of ordinary differential equations (which can be considered as *quantitative*). In that way it is closer how the human mind model and reason about the behavior of a system than differential equations. Refer to [43] for a detailed description of this methodology.

1.3 Problem statement

The aim of this work is to suggest approaches for ***fault diagnostics in dynamic process systems*** using discrete time-dependent heuristics, which had not been widely investigated in the relevant literature so far. We have tried to address the following problems this dissertation:

- The traditionally static HAZID methodologies does not address the problem of diagnosing dynamic event sequences. We wanted to provide and extension to these and propose a procedure HAZID (P-HAZID) approach which might be a used in the case of diagnosing these event sequences.

- There might be cases when only the measurements are available from the diagnosable process system - without any domain-related additional model about the faults (such as a P-HAZID table). In some cases this domain-related model is time-consuming and error-prone to construct - due to the many manual steps involved. We wanted to propose a

method which is observation-based and works on external measurements without an external fault model.

- Finally, due to the fact that both of these diagnostic approaches are NP-hard, we wanted to propose a decomposition approach which can be used to effectively decrease the computational footprint of diagnostics to enable using them in more complex process systems.

1.3.1 Related common taxonomy

The diagnosis of dynamic process systems has a *distant* relationship with the taxonomy of "Dependable and Secure Computing" developed by IFIP WG 10.4 and described in [7]. The most relevant parts of this taxonomy are the following:

1. A **system** is an entity which interacts with other entities (ie. its environment) on the **system boundary**. The **service** is the behavior of a system as it is perceived by its users.

2. The part of the system boundary where service delivery takes place is the **service interface**. The part of the state which is perceivable on the service interface is the **external state** of the system, while the remaining part is its **internal state**.

3. An **error** is when a one or more external state of the system deviates from the state of the correct service. The adjudged or hypothesized cause of an error is called a **fault**. A **service failure** is an event that occurs when the delivered service deviates from the correct service. We talk about **timing failure** if the time of arrival of the service (behavior) deviates from the system function.

4. An **operational fault** is a fault which occurs during service delivery in a system. The presence of a **permanent fault** is assumed to be continuous in time.

5. **Fault diagnosis** identifies and records causes for errors.

6. The **structure** of a system is what enables it to generate the behavior. From a structural standpoint, a system is composed of a set of **components** bound together. In this regard every component is a system on its own.

For the complete taxonomy, refer to [7], in this paragraph only the relevant definitions were collected.

In this work we are dealing with *fault diagnosis* of a subset of *systems* (with a *structure*, composed of one or more *components*), more concretely chemical process systems. Our main goal is by observing the *external state* of these systems, identify *operational faults* which are also *permanent faults* (we are not dealing with other types of faults). It is theoretically possible to generalize the presented methods for other kind of systems, but that is out of scope of this work.

1.4 Thesis structure

The work is divided into these main chapters:

In **Chapter 2** the common notions used to present further diagnostic methods are explained. The general task of diagnostics is reviewed, then the basic model of the process system, the modeling of operational procedures using *events* and *traces* and the modeling of faults are explained. In the final part of the chapter a simple common case study is described which is used in further chapters to compare the various diagnostic approaches.

In **Chapter 3** the P-HAZID diagnostic methodology is discussed. First the related notions, the basic concept of the method, the different types of *deviations*, and how a *P-HAZID table* is constructed using a *reasoning graph* are discussed. The exact operation of the algorithm is described and demonstrated on the case study described in Chapter 2.

In **Chapter 4** an observation-based clustering diagnostic approach is discussed which takes the process system as a closed system and tries to reason about its faults using only the observable outputs of the system. In the first part concepts about mapping the common events and traces to a coordinate space and distance calculation in this space is reviewed, and then the algorithm is explained in detail. The execution of the method is demonstrated on the common case study from Chapter 2.

In **Chapter 5** a higher level decomposition method is described which can utilize the previously discussed diagnostic approaches in solving more complex diagnostic problems, by reducing the size of the fault model the methods. First, notions about system decomposition, *components*, *component graphs* and *component paths* are discussed. Later the higher level diagnostic approach is introduced and its operation is presented on the common case study from Chapter 2 (using the already introduced P-HAZID and Clustering diagnoser as an example).

Finally, in **Chapter 6** the theses for the presented diagnostic approaches

are summarized, along with a few possible future research directions in this field.

Chapter 2

Basic notions

In this chapter the basic notions common to all diagnostic approaches are described. First and foremost, the general system model and diagnostic problem to solve is explained in a formal manner, then the way for modeling system inputs and outputs through *events* and operational procedures using *traces* are discussed. Finally a simple common case study with single and dual faults is described, this example system will be used in further chapters for comparing the different diagnostic approaches.

2.1 System model

The diagnostic approaches described in this work are based upon the multiple input, multiple output (MIMO) causal system model of process systems. In this model, a process system processes vector-valued input and produces vector-valued output signals using an operator S. This operator models the functionality of the system and might also depend on internal unobservable system states. Among the input signals, the processed signals might contain disturbances (faults) from the environment of the system. The model is shown briefly on Fig. 2.1 and described in detail in [27].

Depending on the operator S, the system outputs at a given instant in time might depend on the state(s) of its input values (the actuator elements of the system) and the unobservable internal state of the system. The externally observable inputs and outputs with their time instant together is called an *event* while a sequence of these events (which describe an operational procedure) is referred to as *trace*. The execution of operational procedures (which are manipulating the actuators - the inputs of the system - based on the output values) are logged as traces. These concepts will be formally described further in this chapter.

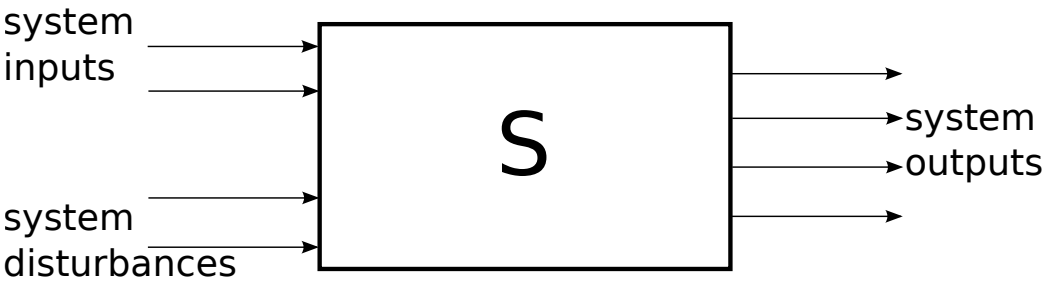

Figure 2.1: A multiple-input multiple-output system.

The concept of event in this dissertation is similar to the concept of event in Discrete Event Systems (DES) theory. According to [13], a DES is a system with a discrete state space and where the transition between states are driven by events (describing a state or a signal transition on the system happening at a discrete point in time). In our case the events are also coming at discrete time intervals and describe an externally observable state of the system.

Faults of the system are modeled as externally observable permanent states or disturbances. It is assumed that the presence of faults remains constant during the fault diagnostics operation.

These concepts could be connected with the taxonomy described in Section 1.3.1(for details refer to [7]).

1. An *event* is analogous to the concept of the *external state*.
2. Consequently, a *trace* can be considered as a *service*.
3. Last but not least, *faults* of the system are *service faults* which are both *operational* and *permanent* in the taxonomy.

2.2 The diagnostics task

The general diagnostic task can be formally defined as:

Given the following:

- a (possibly faulty) process system with actuators (inputs) and observable system outputs
- an operational procedure which actuates the process system through its inputs
- a fault model (which is a representation of faults in the process system, which might occur during the execution of the operational procedure)

Determine the possible component faults (or the nominal behavior) after the execution of a possibly faulty operational procedure, based on the observed output values and the representation of faults.

In the coming chapters of the work, after the common notions are covered, we will attempt to address this problem from different viewpoints, and using different diagnostic methods.

2.3 Inputs and outputs, qualitative range space

System inputs and outputs are signals, i.e. time-dependent quantities (as described in [27]). Their range space can naturally be discrete (such as open or close for a valve) or real (a positive real value for a pressure signal). In case of uncertain values for a real valued measured signal, one can describe the actual value using a qualitative range space, which is a set of ordered mutually disjoint set of real intervals. One usually associate verbal *labels* to the intervals based on the normal operational value of the signal as follows: "N" stands for the normal range, "0", "L" and "H" denote lower and higher but acceptable intervals, while "$e-$" and "$e+$" are the unacceptably low and high values, respectively. Formally, the *qualitative range set* is described like:

$$Q = \{e-, 0, L, N, H, e+\} \tag{2.1}$$

It is possible to create a refined qualitative range set from the qualitative set Q in Eq. (2.1) by placing a new qualitative value between two already existing ones. Such refined qualitative set is given below

$$Q_{refined} = \{e-, -0, 0, 0L, L, LN, N, NH, H, H+, e+\} \tag{2.2}$$

with the newly introduced labels "-0" small negative values, "$0L$" very low, "LN" a bit low, "NH" a bit high, "$H+$" very high.

One can further refine the qualitative range set by adding new intermediate values and achieve the range space of real values in the limit.

The range space of binary discrete valued signals, such as the status of a binary valve can be described by the binary range space

$$\mathcal{B} = \{0, 1\} \tag{2.3}$$

where "0" can be associated to the closed, and "1" to the opened status.

2.4 Events

An event associated to a signal or to a set of signals is a pair of a time instance τ and the actual value(s) at this time instance for a modeled process system,

i.e. $\varepsilon_{\tau}^{(x)} = (\tau; x(\tau))$. Formally, the syntax of an input-output event (at time instant τ of an n-input m output system) is:

$$event_{\tau} = (\tau; input_1, ..., input_n; output_1, ..., output_m)$$

where the time τ is discrete, and described with its sequence number. τ is likewise called the *time* or *time instant* of the event. In the defined format, the time instant, set of input and set of outputs are separated by semicolons, while members of the input and output set are separated by commas.

For example, these events can be defined over qualitative range space in Eq. (2.1), for different process models:

- In a model with a single binary valued input and a single real valued output:
$$(1; "0"; "N")$$
- In a model with two binary inputs and a single real output:
$$(3; "1", "1"; "L")$$
or, for example:
$$(2; "1", "0"; "0")$$
- In a model with two binary inputs and two outputs:
$$(2; "0", "0"; "L", "0", "0")$$
or, for example:
$$(6; "1", "0"; "0", "N", "H")$$

2.5 Traces

Sequences formed from the events (referring to the same process system) above are called *traces* and defined formally as:

$$T(t_1, t_n) = event_{t_1}, ..., event_{t_n}$$

A trace can describe an operational procedure (a sequence of operations) on the process system. Events in the same trace always contain the same number of inputs and outputs with possibly different values. τ is strictly monotonically increasing in consecutive events in the trace.

For example, a trace for the process system in Fig. 2.2 filling up a tank with an input and an output valve using the refined qualitative set in Eq.

sequence number	system inputs		system outputs
	input valve	output valve	tank level
1	1	0	0
2	1	0	0L
3	1	0	L
4	1	0	LN
5	1	1	N

Table 2.1: "Simple tank fill" operational procedure as a trace (sequence of events). Each row represents a single event with time, input and output states (in a tabular format). System input "0" means "closed", "1" means "opened" valve states, while system output "0" means "no level", "L" means "low", "N" means "normal" levels in the tank, according to Eq. (2.1)

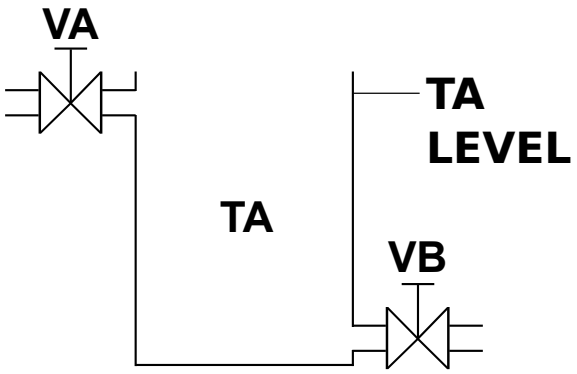

Figure 2.2: Single tank process system.

(2.2) can be defined like the one in Table 2.1. In the table each row corresponds to a single event in the trace, with its time, inputs and outputs.

An other trace on Table 2.2, which flushes the liquid from a two/tank sequential process system of Fig. 3.1 can be defined over qualitative set Eq. (2.1).

Later in the work, a trace describing normal behavior will be called as *nominal* trace, a trace describing some kind of malfunction or fault will be called *characteristic* trace, while a trace where it is unknown whether is there a malfunction or not will be called *observable* or *measured* trace.

sequence number	system inputs VA	VB	VC	system outputs TA level	TB level
1	0	1	1	N	N
2	0	1	1	N	N
3	0	1	1	L	N
4	0	1	1	0	L
5	0	0	1	0	0

Table 2.2: "Tank flush" operational procedure as a trace (sequence of events). Each row represents a single event with time, input and output states (in a tabular format). System input "0" means "closed", "1" means "opened" valve states, while system output "0" means "no level", "L" means "low", "N" means "normal" levels in the tank, according to Eq. (2.1)

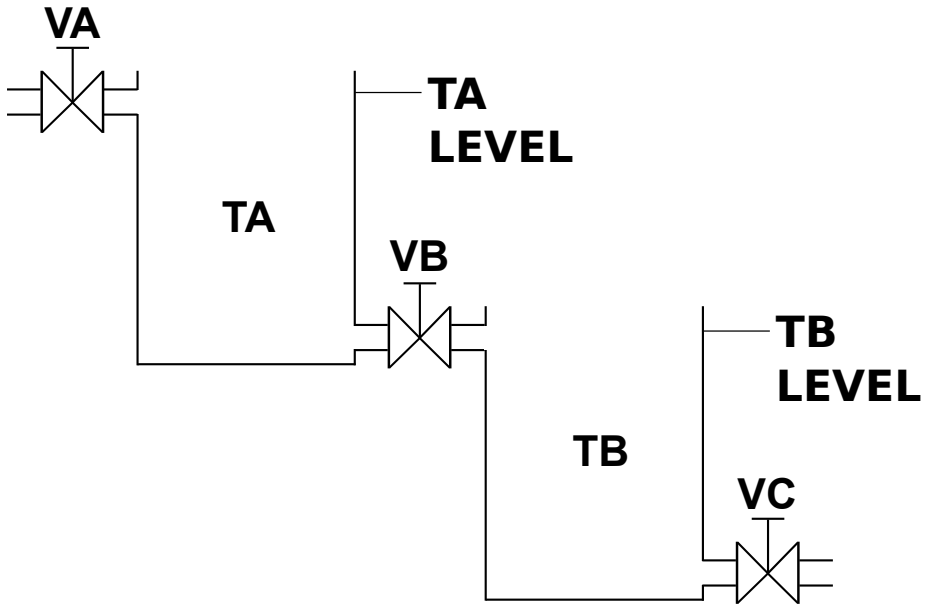

Figure 2.3: Simple sequential process system with two connected tanks.

2.6 Faults

Faults can be considered as non-observable internal system states of the process system. The general objective of the diagnostics in this work is to determine the values of these hidden internal states. The following simple assumptions are made about the nature and presence of faults:

- All inputs are considered error-free, only outputs of a system may contain abnormal values that might refer to a faulty state.

- All fault states permanent in the system during the execution (and the diagnostics) of the operational procedures, random and temporal failures are not considered.

- The structure of the system is assumed to be fixed. We are not dealing with failures which alter the structure as a result.

- Training traces are long enough to capture the transition which will be diagnosed.

- Time is always monotonically increasing and each time instance is present.

- The clock on which all event times are based on is fixed, eg. there is no time skew between any two events of the system.

From the known faults of a process system a diagnoser-specific fault model can be constructed, which can be used during the diagnostics operation in order to detect fault symptoms in measured traces coming from the process system.

2.7 A simple composite process system

A common example process system is used through the case studies to demonstrate the diagnostic approaches in further chapters. The structure of the process system can be seen in Fig. 3.4. It consists of a main tank "TA" and two identical auxiliary tanks "TB" and "TC" (half the size of the main tank each) connected with pipes to the main tank. All of the pipes have valves on them which can be either open or in closed state. The fluid flows first to the main tank through input valve "VA", then leaves it through "VB" and "VC" towards the auxiliary tanks. From the auxiliary tanks it flows out of the system via output valves "VD" and "VE".

These basic assumptions are made regarding the operation of this example process system:

1. All valves in the system are assumed to be "binary", ie. there are no intermittent states during valve operation.

2. All tanks are equipped with level sensors operating on qualitative range set of Eq. (2.1) or Eq. (2.2).

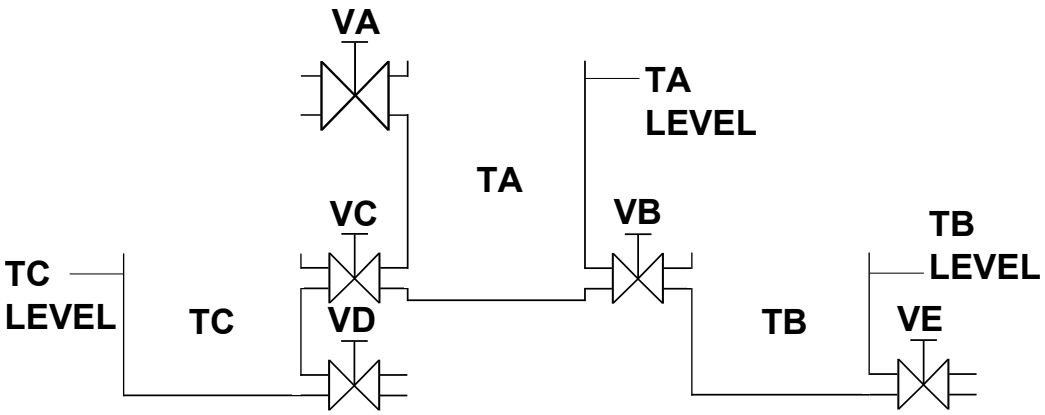

Figure 2.4: Simple process system used for the case studies.

3. Throughout the process system pipe sizes are proportional to the size of the tank they flow into. Due to this, the level in the tanks will be increased by one qualitative level per time instant, provided the input valve is opened and the output valve is closed. If the output and input valves are both in opened state, then the fluid level stays constant in the tank. If only the output valve is opened, then the fluid level will decrease by one qualitative level in the tank per time instant.

2.7.1 Nominal trace

An operational procedure which fills up all three tanks with fluid is used as a common nominal trace to compare the different diagnostic approaches. Initially all three tanks are empty and all valves are in closed state. After opening input valve "VA" tank "TA" is filled up completely with fluid, then valves "VB" and "VC" are opened and tanks "TB" and "TC" are filled up completely as well. Finally the output valves, "VD" and "VE" are opened. The operational procedure formally can be seen in Table 2.3.

2.7.2 Faults

These types of faults were taken into account for each tank in the example process system:

- The leak of the tank. The size of the leak prevents any fluid from staying inside of the tank, therefore fluid level constantly stays at qualitative value "0".

sequence	system inputs					system outputs		
number	VA	VB	VC	VD	VE	TA	TB	TC
1	1	0	0	0	0	0	0	0
2	1	0	0	0	0	L	0	0
3	1	1	1	0	0	N	0	0
4	1	1	1	0	0	N	L	L
5	1	1	1	1	1	N	N	N

Table 2.3: Nominal trace for the case study. System input "0" means "closed", "1" means "opened" valve states, while system output "0" means "no level", "L" means "low", "N" means "normal" levels in the tank, according to Eq. (2.1)

- The positive bias failure of the tank level sensor. The level sensor always detects a qualitative value one degree higher than the actual level of the tank. For instance, given the qualitative set defined in Eq. (2.1), the level sensor outputs "N" instead of "L".
- The negative bias failure of the tank level sensor. The level sensor always detects a qualitative value one degree lower than the actual level of the tank. For instance, given the qualitative set defined in Eq. (2.1), the level sensor outputs "0" instead of "L".

For reference, all faulty traces for the above mentioned *single* faults are described in Table 2.4.

Dual faults

During the presence of dual faults (which are all possible combinations of the above mentioned single faults) the combined effect of the faults on the corresponding output signals will be taken into account. The following apply during the forming of the faulty traces:

- When the faults refer to different output signals (for example, "TA" and "TB") then the effects of the faulty events are simply combined with each other, this can be done easily due to the fact that they refer to different outputs. For instance, see outputs "TA" and "TB" in fault *Leak of "TA" and negative bias of "TB"* in Table 2.5.
- When multiple faults, such as a leak and a positive or negative bias refer to the same output signal (in this case, both to the same tank component) then the effect of faults are combined using the following principles:

Leak of "TA"

sequence	system inputs					system outputs		
number	VA	VB	VC	VD	VE	TA	TB	TC
1	1	0	0	0	0	0	0	0
2	1	0	0	0	0	**0**	0	0
3	1	1	1	0	0	**0**	0	0
4	1	1	1	0	0	**0**	**0**	**0**
5	1	1	1	1	1	**0**	**0**	**0**

Leak of "TB"

sequence	system inputs					system outputs		
number	VA	VB	VC	VD	VE	TA	TB	TC
1	1	0	0	0	0	0	0	0
2	1	0	0	0	0	L	0	0
3	1	1	1	0	0	N	0	0
4	1	1	1	0	0	N	**0**	L
5	1	1	1	1	1	N	**0**	N

Leak of "TC"

sequence	system inputs					system outputs		
number	VA	VB	VC	VD	VE	TA	TB	TC
1	1	0	0	0	0	0	0	0
2	1	0	0	0	0	L	0	0
3	1	1	1	0	0	N	0	0
4	1	1	1	0	0	N	L	**0**
5	1	1	1	1	1	N	N	**0**

Positive bias fault of "TA"

sequence	system inputs					system outputs		
number	VA	VB	VC	VD	VE	TA	TB	TC
1	1	0	0	0	0	**L**	0	0
2	1	0	0	0	0	**N**	0	0
3	1	1	1	0	0	**H**	0	0
4	1	1	1	0	0	**H**	L	L
5	1	1	1	1	1	**H**	N	N

Positive bias fault of "TB"

sequence	system inputs					system outputs		
number	VA	VB	VC	VD	VE	TA	TB	TC
1	1	0	0	0	0	0	**L**	0
2	1	0	0	0	0	L	**L**	0
3	1	1	1	0	0	N	**L**	0
4	1	1	1	0	0	N	**N**	L
5	1	1	1	1	1	N	**H**	N

Positive bias fault of "TC"

sequence	system inputs					system outputs		
number	VA	VB	VC	VD	VE	TA	TB	TC
1	1	0	0	0	0	0	0	**L**
2	1	0	0	0	0	L	0	**L**
3	1	1	1	0	0	N	0	**L**
4	1	1	1	0	0	N	L	**N**
5	1	1	1	1	1	N	N	**H**

Negative bias fault of "TA"

sequence	system inputs					system outputs		
number	VA	VB	VC	VD	VE	TA	TB	TC
1	1	0	0	0	0	**e-**	0	0
2	1	0	0	0	0	**0**	0	0
3	1	1	1	0	0	**L**	0	0
4	1	1	1	0	0	**L**	L	L
5	1	1	1	1	1	**L**	N	N

Negative bias fault of "TB"

sequence	system inputs					system outputs		
number	VA	VB	VC	VD	VE	TA	TB	TC
1	1	0	0	0	0	0	**e-**	0
2	1	0	0	0	0	L	**e-**	0
3	1	1	1	0	0	N	**e-**	0
4	1	1	1	0	0	N	**0**	L
5	1	1	1	1	1	N	**L**	N

Negative bias fault of "TC"

sequence	system inputs					system outputs		
number	VA	VB	VC	VD	VE	TA	TB	TC
1	1	0	0	0	0	0	0	**e-**
2	1	0	0	0	0	L	0	**e-**
3	1	1	1	0	0	N	0	**e-**
4	1	1	1	0	0	N	L	**0**
5	1	1	1	1	1	N	N	**L**

Table 2.4: All single faults as traces in the example case study system. Output differences compared to nominal behavior are show in **bold**.

Leak of "TA" and negative bias of "TB"

sequence	system inputs					system outputs		
number	VA	VB	VC	VD	VE	TA	TB	TC
1	1	0	0	0	0	0	**e-**	0
2	1	0	0	0	0	**0**	**e-**	0
3	1	1	1	0	0	**0**	**e-**	0
4	1	1	1	0	0	**0**	**e-**	**0**
5	1	1	1	1	1	**0**	**e-**	**0**

Leak of "TB" and negative bias of "TB"

sequence	system inputs					system outputs		
number	VA	VB	VC	VD	VE	TA	TB	TC
1	1	0	0	0	0	0	**e-**	0
2	1	0	0	0	0	L	**e-**	0
3	1	1	1	0	0	N	**e-**	0
4	1	1	1	0	0	N	**e-**	L
5	1	1	1	1	1	N	**e-**	N

Leak of "TC" and positive bias of "TC"

sequence	system inputs					system outputs		
number	VA	VB	VC	VD	VE	TA	TB	TC
1	1	0	0	0	0	0	0	**L**
2	1	0	0	0	0	L	0	**L**
3	1	1	1	0	0	N	0	**L**
4	1	1	1	0	0	N	L	**L**
5	1	1	1	1	1	N	N	**L**

Table 2.5: A few examples for dual faults in the form of traces. Output differences compared to nominal behavior are shown in **bold**.

- If the bias is positive then the level sensor constantly shows a qualitative value one degree higher than the empty value. For instance, see output "TB" in fault *Leak of "TB" and negative bias of "TB"* in Table 2.5 for an example.
- If the bias is negative then the level sensor constantly shows a qualitative value one degree lower than the empty value. For instance, see output "TC" in fault *Leak of "TC" and positive bias of "TC"* in Table 2.5 for an example.
- It is assumed that the same level sensor cannot have 2 different bias faults (positive and negative) at the same time.

2.8 Summary

In this chapter the basic notions specific to the further described diagnostic approaches were presented. These notions will be used in the coming chapters as a basis, and will be extended with additional method-specific notions there. At the end of the chapter, a simple common case study was presented, which will be used in further chapters to demonstrate the capabilities of the diagnostic methods and to present the main differences between them.

Chapter 3

P-HAZID diagnostics

In this chapter, the procedure HAZID (P-HAZID) methodology, a novel off-line way of reasoning about process system faults during operational procedure execution is described in detail. The diagnostics is based upon atomic deviations of measured operational procedure events from the events in a normal procedure. These deviations are organized into a special P-HAZID table which is used by the diagnostic algorithm. When a possibly faulty observable trace need to be diagnosed, its deviations are collected first and using the P-HAZID table, the algorithm can reason about possible fault modes which might have been present in the process system during the execution of the trace.

The chapter is organized into three main parts. First, the notions specific to this methodology, such as the concept of flattening, deviations and the structure of the P-HAZID table is described. Then, the P-HAZID reasoning algorithm is discussed in detail. Finally, the execution of the method is demonstrated on a simple case study from Section 2.7 and the main characteristics of the method are described.

3.1 Introduction

The domain for the usual HAZID analysis is static in the sense, that deviations from the normal, usually steady-state operation are recorded and used. It is, however, possible to extend the BL-HAZID diagnostic idea (see Section 1.2.1 as well as [42]) to the dynamic case, when the execution of an operational procedure drives the process system from one state to another, and use it for diagnostic purposes. This extended method can be used for finding component faults based on the deviation(s) between the observable inputs and outputs of planned (nominal) and actual (characteristic) event

sequences driven by an operational procedure.

In order to achieve this, a novel dynamic procedure HAZID (P-HAZID) model (in the form of a table) and reasoning method based on the model is proposed in [52]. The use of these P-HAZID tables for diagnostic purposes had been formalized in [44], and it had been extended by taking the structural similarities of process system components into account in [45]. This approach is described in detail in this chapter.

The tabular representation form of the BL-HAZID method (with the columns "Subsystem", "Cause", "Deviation" and "Implication") is very similar to the structure of the P-HAZID table (having columns "Cause", "Deviation" and "Implication"). The difference here is while in the case of BL-HAZID, failures and failure causing events are present in the cells of the table, in the case of the P-HAZID deviations from the nominal operational procedure and their root causes are placed in the spreadsheet. The reasoning paths in the P-HAZID method, used for capturing the diagnostics is also analogous to the cause-implication graph of the BL-HAZID, but it describes causal relationships between operational procedure deviations and failure root causes. Similarly, in the case of the BL-HAZID, this graph contains failures, while in the case of the P-HAZID it contains deviations. The P-HAZID methodology can be also thought as an extension of the BL-HAZID methodology with the dimension of time.

On the other hand, the reasoning process in the P-HAZID method is analogous to the widely known concept of if-then or condition-action rules (as described in [41]). In that way it is similar to a G2 (see [14]) based fault diagnostic method described in [35] which transforms the heuristics from a HAZOP table into a set of rules as a fault model and uses if-then rule based backward chaining (see [41]) to diagnose faults in the process system.

Similar to this method, the fault reasoning phase in the P-HAZID can be performed by forward chaining (see [41]) using if-then or condition-action rules (see [41]). The details of this will be further elaborated in Section 3.3.3.

3.2 Basic notions

The most important novel concepts and tools are introduced in this section to formally define the P-HAZID table and the diagnostic method based thereon. The concepts discussed in this section are the P-HAZID specific extensions of the common notions of *event* and *trace*, as described in Section 2.4 and Section 2.5.

event sequence number	system inputs: input valve	first output valve	second output valve	system outputs: TA level	TB level
1	1	0	0	0	0
2	1	0	0	L	L
3	1	1	1	N	N

Table 3.1: "Simple tank fill" trace for the process system in Fig. . Each row represents a single event with time, input and output states (in a tabular format). In the system inputs, "1" means opened, while "0" means closed state. For the outputs, "0" means empty, "L" means low while "N" means normal level.

3.2.1 Flattening of traces

If there are multiple outputs (as described in Section 2.1) in the process system, then they need to be handled separately in a P-HAZID diagnoser, because the reasoning method works on single output values only. Therefore an initial operation need to be performed for every input trace used by the diagnoser described later. This operation is called *flattening* and it works like the following: If there are n outputs in the events of a trace then for every time instant in the trace n number of *single-output events* are generated (note that inputs are considered error-free). These partial events only contain the time instant (sequence number of the whole event), the identifier (index) and the value of the single output. During the generation of deviations, this output value determines the type of the deviation, while the identifier (index) will be the output index of the deviation (as described in Section 3.2.2).

For example, the three-input two-output nominal trace in Table 3.1 will be converted to this sequence of single-output events:

$$(1; TA : 0), (1; TB : 0), (2; TA : L), (2; TB : L), (3; TA : N), (3; TB : N)$$

After this initial operation the trace will be checked against deviations by comparing it to the flattened form of the nominal trace.

3.2.2 Deviations

Nominal, characteristic and observable traces can be compared by comparing their corresponding *single-output events* acquired after flattening. The difference between two corresponding *single-output events* (where the time instant is the same) is described by a single deviation. The term *deviation* is conceptually analogous to the general term of *service failure* as described in

the common taxonomy (refer to Section 1.3.1 and [7]). Deviations are formed from a deviation guide word (which describes the exact type of the deviation), the time instant to identify the event in the nominal trace and the identifier of the output to which the deviation refers to (this is the output of the single-output event). Two major deviation categories can be distinguished, a deviation might be **chronological** or **quantitative**. Chronological deviations describe that the event regarding that single output in the nominal trace did not occur at the correct time, while quantitative deviations denote the difference in the single output values between events of the same time instant. Both deviation types always refer to a single output variable in the event, if there are more outputs deviating from the nominal event in a measured event then multiple deviations are formed for that event.

The following types of chronological deviations are distinguished (along with their connection with the common taxonomy described in [7]):

- **later**: When the output value is present in the measured trace, but at a *later* time instant. According to the common taxonomy this is a **late timing fault**.
- **earlier**: When the output value is present in the measured trace, but at an *earlier* time instant. According to the common taxonomy this is an **early timing fault**.
- **never-happened**: When the particular output value *never happened* in the measured trace. According to the common taxonomy this is an **omission fault**, but not necessarily "human-made", as it is described there.

The following types of quantitative deviations are used (see Q_e in Section 2.1 for an example qualitative set):

- **slightly-greater**: When an output's qualitative value is higher than the nominal value and the difference is only one qualitative value.
- **significantly-greater**: When an output's qualitative value is higher in the measured event than the nominal one and the difference is more than one qualitative value.
- **slightly-smaller**: When an output's qualitative value is lower than the nominal value by one qualitative value.
- **significantly-smaller**: When an output's qualitative value is lower in the measured event and the difference is more than one qualitative value.

These deviations are used for capturing small fragments of the fault model (related to specific event pairs of the nominal and measured trace) in a P-HAZID table described in Section 3.2.3.

As an example, the chronological deviation **earlier**(3;TB) means that the third nominal event, taking only the output value of "TB" into account, happened at an earlier time instant in the measured trace than in the nominal trace.

As an other example, the quantitative deviation **slightly-smaller**(4;TA) denotes that at the fourth event in the measured trace the value of output "TA" was smaller than the value of "TA" in the same event in the nominal trace and for the same output by one qualitative value.

Note that for a *single* output value difference in the measured trace a chronological and a quantitative deviation can always be found, denoting the *wrong timing* of the event (as a chronological deviation) and the *incorrect value* (as a quantitative deviation) compared to the nominal event. Both of these can be included in the diagnostic reasoning to increase accuracy.

Equality of deviations

During the reasoning procedure the equality of the deviations are checked, in order to match actual deviations with already stored ones. Two deviations are said to be equal if their *type*, *time instant*, and *output identifier* are identical. For example:

- **earlier**(3;TB) and **earlier**(3;TB) are *equal*
- **earlier**(3;TB) and **earlier**(3;TC) are *not equal* because they refer to different outputs (TB and TC)
- **earlier**(3;TB) and **earlier**(2;TB) are *not equal* because they refer to different time instants (3 and 2)
- **earlier**(3;TB) and **later**(3;TB) are *not equal* because they are of different type (**earlier** and **later**)

Because inputs are considered error-free they are not present in the deviations, their values can be determined from the nominal trace and time instant if needed. Only the time instant of the event is used during the reasoning procedure, therefore inputs does not need to be present in the deviations.

3.2.3 P-HAZID table

As a combination and extension of the widely used FMEA and HAZOP analyses (for details, refer to [44] or [12] and partly [42]) the procedure HAZID

Cause	Deviation	Implication
TANK-LEAK	never-happened(2;OUT)	never-happened(3;OUT)
never-happened(2;OUT)	never-happened(3;OUT)	never-happened(4;OUT)
TANK-LEAK	slightly-smaller(2;OUT)	significantly-smaller(3;OUT)
slightly-smaller(2;OUT)	significantly-smaller(3;OUT)	significantly-smaller(4;OUT)
NEG-BIAS	slightly-smaller(1;OUT)	slightly-smaller(2;OUT)
slightly-smaller(1;OUT)	slightly-smaller(2;OUT)	slightly-smaller(3;OUT)
slightly-smaller(2;OUT)	slightly-smaller(3;OUT)	slightly-smaller(4;OUT)
POS-BIAS	slightly-greater(1;OUT)	slightly-greater(2;OUT)
slightly-greater(1;OUT)	slightly-greater(2;OUT)	slightly-greater(3;OUT)
slightly-greater(2;OUT)	slightly-greater(3;OUT)	slightly-greater(4;OUT)

Table 3.2: A fraction of a simple **P-HAZID** table for a system with two inputs and a single output "OUT".

(abbreviated as P-HAZID) analysis result can be used for fault diagnostics analyzing operational procedure execution in a given process system. The result of this P-HAZID analysis is given in the form of a table or spreadsheet, and it consists of deviations leading to possible root causes ordered in columns. Using the initial set of differences (deviations) between the measured trace and the nominal (or the characteristic) traces, the set of possible root causes can be found by a simple reasoning procedure taking the connections between rows in the table into account. Two rows are said to be connected in the table if the corresponding values in the "Deviation", "Implication" columns (last two columns) and "Cause", "Deviation" columns (first two columns) are equal according to the rules defined in Section 3.2.2. This is used during reasoning to find possible root causes (in the first column with capital letters) from a set of initial deviations by chaining rows after each other. Therefore, from an initial set of starting deviations a directed *reasoning graph* can be drawn connecting them to the possible root causes based on the P-HAZID table. Note that this is true for the other way around, eg. from a reasoning graph a P-HAZID table can be created, as it will be shown in the case study of the chapter. The reasoning graph describes causal relationships between operational procedure deviations and root causes for a particular process system. A path in this graph is called *reasoning path*. A simple example for a P-HAZID table can be found in Table 3.2.

Note that the P-HAZID table is the *fault model* (as defined in Section 2.6) of the P-HAZID diagnoser, the diagnoser uses this spreadsheet, the nominal and measured trace to detect fault modes of the system.

Construction of a P-HAZID table

Setting up the connection between deviations and determining exact root causes in the P-HAZID table is a manual task requiring domain knowledge on the diagnosable process system. However, some aspects of the analysis

might be automated, such as the flattening of traces (see Section 3.2.1) or the generation of deviations (see Section 3.2.2) between nominal and characteristic traces.

In order to create a P-HAZID table practically the following need to be done:

1. The nominal trace need to be defined, and all the different fault cases (tank leak, bias fault,...) should be identified in the system. For every fault case a corresponding characteristic trace need to be defined by domain experts or generated from the fault during a simulation of the process system.

2. All traces (nominal and characteristic) should be flattened (as described in Section 3.2.1).

3. Using the flattened traces, taking events one by one, all possible characteristic and quantitative deviations need to be generated for every fault case. Deviations shall be ordered by their time instant in decreasing order starting from the largest one, and grouped by their deviation category (chronological or quantitative).

4. Starting from the higher time instants reasoning paths can be defined leading to the specific root cause (from which the characteristic trace was created in the first place). For defining these paths expert knowledge is needed.

5. Based on the defined reasoning paths the P-HAZID table can be built by taking triplets of events from the end of the path towards the fault (which is at the beginning of the path) and placing the triplets as rows (for columns "Cause", "Deviation", "Implication") in the P-HAZID table. Except for the first triplet, every triplet shall contain the last two deviations from the preceding one. This is important because that is how the reasoning algorithm (as described in) connects rows in the P-HAZID table with each other.

3.3 P-HAZID based diagnostics

The P-HAZID algorithm requires a nominal trace, a P-HAZID table and the observable trace as an input and provides the possible set of root causes which might be present in the process system during the execution of the observable trace. The algorithm is off-line in the sense that it requires the whole observable trace as an input (it has to be already completed), although

generalizing it to an on-line case is theoretically possible (in order to work with operational procedures which are currently under execution), this is not the subject of this chapter.

3.3.1 Additional assumptions

The following assumptions are made in addition to the ones listed in Section 2.6 for this diagnostic method:

- The chosen qualitative set is fine enough to capture the transition in the nominal and faulty traces.
- A representative unique set of deviations can be generated between each faulty and nominal trace for a faulty output, based on the differences and not taking the actual nominal value of the output into account. (If two different faults generate exactly the same set of deviations the diagnostic reasoning will not be able to differentiate between the faults.)

3.3.2 Algorithm description

Using a nominal operational procedure with its corresponding P-HAZID table and a measured trace the diagnostic algorithm operates as:

1. If the measured trace contains multiple outputs, it shall be flattened as described in Section 3.2.1 in detail.
2. All deviations are collected between the flattened nominal and measured trace (which is already flattened in step 1). The deviations at the final and preceding time instant need to be collected, and assigned to the set of *final deviation pairs* by decreasing order in their time instant. The next step is executed for each possible pair of deviations in the *final deviation pair* set.
3. All the rows in the P-HAZID table are selected where exactly the first element of the pair is found in the "Deviation" column, and exactly the second element is found in the "Implication" column.
4. For all selected rows: if the selected row's "Cause" column contains a root cause, or a deviation which is not present in the original set of deviations (called *terminal deviation*) then this is added to the set of root causes and the reasoning finishes for this single path by continuing with step 6, otherwise (if the deviation is present among the initial deviations), it continues with step 5.

5. If the collected set of deviations contains the deviation in the selected row's "Cause" column then continue with step 3 by taking the deviation in the "Cause" column as the first element of the pair and the "Deviation" element as the second element of the pair (in order to find rows where the content in "Cause" equals to the "Deviation" cell and the content of "Deviation" equals to "Implication" cell).

6. Return the set of found root causes and the set of deviations which were not present at the initial set of deviations.

This diagnostic algorithm attempts to re-construct a subset of reasoning graphs from which the P-HAZID table initially created from using the method mentioned above from the set of initial deviations. Here separate cells in the P-HAZID are represented as nodes in the reasoning graph. The roots of the tree are the final deviation pairs from where the diagnostic procedure is initiated, while the leaves are the identified root causes or the deviations from which the diagnostic operation cannot proceed forward (because they were not contained in the set of initial deviations). The junctions (nodes with degree more than 2) denote rows in the P-HAZID table which are connected to multiple rows in the P-HAZID by their event values in the "Deviation" and "Implication" columns. If the reasoning cannot proceed forward from a deviation because it was missing from the set of initial deviations, this might even mean a failure, such as an incorrect deviation in the P-HAZID table (thus the algorithm might be used for validating the P-HAZID table).

The complete set of reasoning graphs, based on the simple example P-HAZID table described in Table 3.2, can be seen in Fig. 3.1.

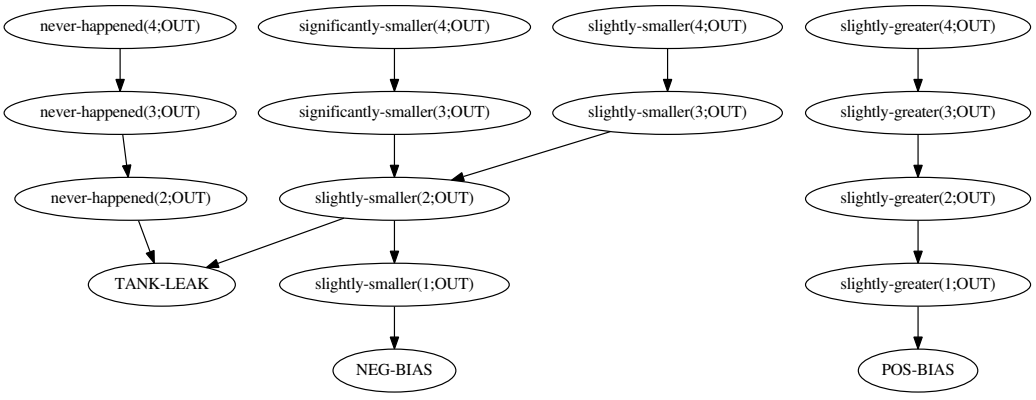

Figure 3.1: Diagnostic paths explored by the P-HAZID diagnostic algorithm for the example P-HAZID table of Table 3.2.

Cause	Deviation	Implication
TANK-LEAK	never-happened(2;OUT)	never-happened(3;OUT)
never-happened(2;OUT)	never-happened(3;OUT)	never-happened(4;OUT)

Table 3.3: A single row in the P-HAZID table is analogous to a single if-then rule.

3.3.3 Analogy with if-then rules

The reasoning paths in the P-HAZID table are analogous to an ordinary *if-then* rules used widely in artificial intelligence applications such as rule-based expert systems. For instance, if we take the simplest form of the if-then rule into account:

$$IF \ \textbf{condition} \ THEN \ \textbf{action}$$

Then, in the case of a reasoning path in the P-HAZID table, the deviations in the path can be considered as the *condition* of the rule, while the root cause at the end of the path is similar to the *action* of an if-then rule. In that way, the single reasoning path on Table 3.3 (in the form of P-HAZID rows) can be converted to the if-then rule:

$$IF \ \textbf{never} - \textbf{happened}(\textbf{4};\textbf{OUT}) \ \textbf{AND never} - \textbf{happened}(\textbf{3};\textbf{OUT})$$

$$\textbf{AND never} - \textbf{happened}(\textbf{2};\textbf{OUT})$$

$$THEN \ \textbf{TANK} - \textbf{LEAK}$$

In that way, chronological and quantitative deviations are analogous to *facts* while the set of reasoning paths can be considered as a *knowledge-base* in rule-based production systems (see [41]). In these systems the knowledge-base contains facts like the P-HAZID table is built from deviations. Due to these similarities, in a special case the P-HAZID reasoning can be also performed by an ordinary pattern matching algorithm, such as the Rete algorithm (see [20]), using the set of deviations as facts in the knowledge base.

3.4 Diagnostics case study

The simple case study from Section 2.7, shown in Fig. 3.4 is used to demonstrate the P-HAZID approach in operation. This case study, as described earlier, consisted of three fluid tanks connected in parallel, where the main tank "TA" is connected by a pipe to auxiliary tanks "TB" and "TC". Before

the actual diagnostics operation, the P-HAZID table is constructed following the process described in Section 3.2.3. Note that only single fault cases are dealt with in this case study.

3.4.1 Flattening of the traces

The nominal trace from the common case study (filling up all three tanks with liquid) described in Section 2.7.1 is flattened to this list of single-output events:

$$(1; TA : 0), (1; TB : 0), (1; TC : 0)$$
$$(2; TA : L), (2; TB : 0), (2; TC : 0)$$
$$(3; TA : N), (3; TB : 0), (3; TC : 0)$$
$$(4; TA : N), (4; TB : L), (4; TC : L)$$
$$(5; TA : N), (5; TB : N), (5; TC : N)$$

Flattening of the characteristic traces for the different faults is done in the same manner. For example the "TB-LEAK" faulty trace looks like this in flattened form (differences from the nominal trace are denoted by **bold**):

$$(1; TA : 0), (1; TB : 0), (1; TC : 0)$$
$$(2; TA : L), (2; \mathbf{TB : 0}), (2; TC : 0)$$
$$(3; TA : N), (3; \mathbf{TB : 0}), (3; TC : 0)$$
$$(4; TA : N), (4; \mathbf{TB : 0}), (4; TC : L)$$
$$(5; TA : N), (5; \mathbf{TB : 0}), (5; TC : N)$$

These flattened forms are used create the deviations for every faulty case.

3.4.2 Acquiring deviations from faulty traces

Comparison of the nominal and faulty events in flattened form resulted in many deviations in the different fault cases. For example, for the single fault cases of tank TA, the following deviations were collected:

- Leak of the tank (note that in this case all three tanks are affected):
  - **Chronological deviations:**
    never-happened(5,TA), never-happened(5,TB), never-happened(5,TC), never-happened(4,TA), never-happened(4,TB), never-happened(4,TC), never-happened(3,TA), never-happened(2,TA)

- **Quantitative deviations:**
  significantly-smaller(5,TA), significantly-smaller(5,TB), significantly-smaller(5,TC), significantly-smaller(4,TA), slightly-smaller(4,TB), slightly-smaller(4,TC), significantly-smaller(3,TA), slightly-smaller(2,TA)

- Positive bias of the tank level sensor:
  - **Chronological deviations:**
    never-happened(5,TA), never-happened(4,TA), never-happened(3,TA), earlier(2,TA), earlier(1,TA)
  - **Quantitative deviations:**
    slightly-greater(5,TA), slightly-greater(4,TA), slightly-greater(3,TA), slightly-greater(2,TA), slightly-greater(1,TA)

- Negative bias of the tank level sensor:
  - **Chronological deviations:**
    never-happened(5,TA), never-happened(4,TA), never-happened(3,TA), later(2,TA), later(1,TA)
  - **Quantitative deviations:**
    slightly-smaller(5,TA), slightly-smaller(4,TA), slightly-smaller(3,TA), slightly-smaller(2,TA), slightly-smaller(1,TA)

For the rest of the tanks the deviations for each fault can be determined in a similar fashion.

3.4.3 Reasoning graphs

Taking all the acquired deviations in decreasing order by their time instant for every fault the reasoning graphs of Fig. 3.2 for tank "TA", Fig. 3.3 for tank "TB" and Fig. 3.4 for tank "TC" are constructed. In this case a reasoning path was created for every fault case from the deviations present in that particular case in decreasing chronological order, and those paths are combined into one single directed graph at the end. The nodes are the individual deviations, and the direction of the edges between them show the direction of reasoning towards a root cause.

3.4.4 P-HAZID table

Finally, from the separate reasoning paths for different faults in the system a P-HAZID table is created, containing all the paths (with the corresponding

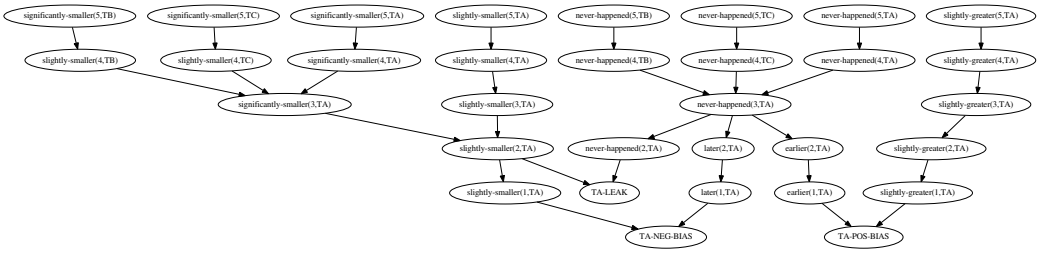

Figure 3.2: Single fault reasoning paths for tank "TA".

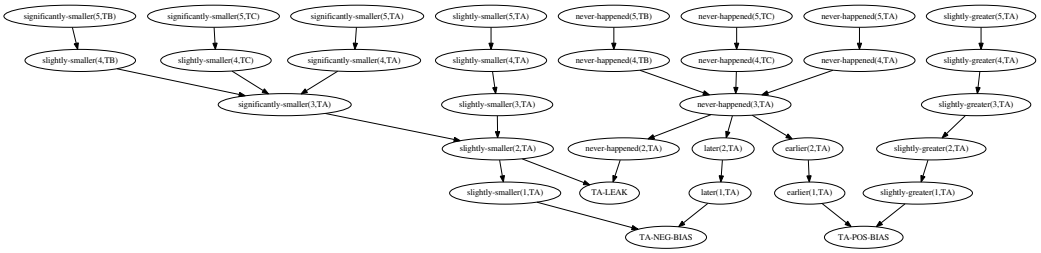

Figure 3.3: Single fault reasoning paths for tank "TB".

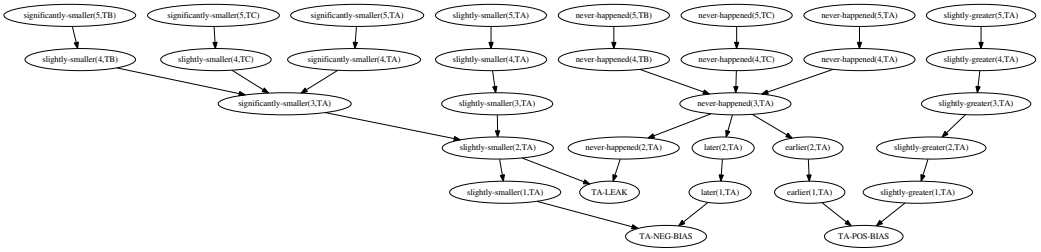

Figure 3.4: Single fault reasoning paths for tank "TC".

deviations and fault cases) in a tabular form to be used by the reasoning algorithm. This table and the observable trace is used as an input to the diagnostic algorithm. The complete table for all tanks and faults can be seen in Table 3.4.

3.4.5 Execution of the algorithm

Let us now use the above constructed P-HAZID table in Table 3.4, and a faulty trace for the *negative bias fault of the level sensor on TA* (described in Table 3.5). With these inputs a sample run of the diagnostic algorithm is demonstrated below.

The faulty trace first need to be converted to a flattened form, to a list of *single-output events*. This form in this case is the following (differences from

Cause	*Deviation*	*Implication*
never-happened(3,TA)	never-happened(4,TA)	never-happened(5,TA)
never-happened(2,TA)	never-happened(3,TA)	never-happened(4,TA)
TA leakage	never-happened(2,TA)	never-happened(3,TA)
significantly-smaller(3,TA)	significantly-smaller(4,TA)	significantly-smaller(5,TA)
slightly-smaller(2,TA)	significantly-smaller(3,TA)	significantly-smaller(4,TA)
TA leakage	slightly-smaller(2,TA)	significantly-smaller(3,TA)
TB leakage	never-happened(4,TB)	never-happened(5,TB)
TB leakage	slightly-smaller(4,TB)	significantly-smaller(5,TB)
TC leakage	never-happened(4,TC)	never-happened(5,TC)
TC leakage	slightly-smaller(4,TC)	significantly-smaller(5,TC)
slightly-smaller(3,TA)	slightly-smaller(4,TA)	slightly-smaller(5,TA)
slightly-smaller(2,TA)	slightly-smaller(3,TA)	slightly-smaller(4,TA)
TA negative bias	slightly-smaller(2,TA)	slightly-smaller(3,TA)
never-happened(3,TA)	never-happened(4,TA)	never-happened(5,TA)
later(2,TA)	never-happened(3,TA)	never-happened(4,TA)
later(1,TA)	later(2,TA)	never-happened(3,TA)
TA negative bias	later(1,TA)	later(2,TA)
slightly-greater(3,TA)	slightly-greater(4,TA)	slightly-greater(5,TA)
slightly-greater(2,TA)	slightly-greater(3,TA)	slightly-greater(4,TA)
TA positive bias	slightly-greater(2,TA)	slightly-greater(3,TA)
never-happened(3,TA)	never-happened(4,TA)	never-happened(5,TA)
earlier(2,TA)	never-happened(3,TA)	never-happened(4,TA)
earlier(1,TA)	earlier(2,TA)	never-happened(3,TA)
TA positive bias	later(1,TA)	later(2,TA)
slightly-smaller(3,TB)	slightly-smaller(4,TB)	slightly-smaller(5,TB)
slightly-smaller(2,TB)	slightly-smaller(3,TB)	slightly-smaller(4,TB)
TB negative bias	slightly-smaller(2,TB)	slightly-smaller(3,TB)
never-happened(3,TB)	never-happened(4,TB)	never-happened(5,TB)
later(2,TB)	never-happened(3,TB)	never-happened(4,TB)
later(1,TB)	later(2,TB)	never-happened(3,TB)
TB negative bias	later(1,TB)	later(2,TB)
slightly-greater(3,TB)	slightly-greater(4,TB)	slightly-greater(5,TB)
slightly-greater(2,TB)	slightly-greater(3,TB)	slightly-greater(4,TB)
TB positive bias	slightly-greater(2,TB)	slightly-greater(3,TB)
never-happened(3,TB)	never-happened(4,TB)	never-happened(5,TB)
earlier(2,TB)	never-happened(3,TB)	never-happened(4,TB)
earlier(1,TB)	earlier(2,TB)	never-happened(3,TB)
TB positive bias	later(1,TB)	later(2,TB)
slightly-smaller(3,TC)	slightly-smaller(4,TC)	slightly-smaller(5,TC)
slightly-smaller(2,TC)	slightly-smaller(3,TC)	slightly-smaller(4,TC)
TC negative bias	slightly-smaller(2,TC)	slightly-smaller(3,TC)
never-happened(3,TC)	never-happened(4,TC)	never-happened(5,TC)
later(2,TC)	never-happened(3,TC)	never-happened(4,TC)
later(1,TC)	later(2,TC)	never-happened(3,TC)
TC negative bias	later(1,TC)	later(2,TC)
slightly-greater(3,TC)	slightly-greater(4,TC)	slightly-greater(5,TC)
slightly-greater(2,TC)	slightly-greater(3,TC)	slightly-greater(4,TC)
TC positive bias	slightly-greater(2,TC)	slightly-greater(3,TC)
never-happened(3,TC)	never-happened(4,TC)	never-happened(5,TC)
earlier(2,TC)	never-happened(3,TC)	never-happened(4,TC)
earlier(1,TC)	earlier(2,TC)	never-happened(3,TC)
TC positive bias	later(1,TC)	later(2,TC)

Table 3.4: A complete P-HAZID table for the three tank composite process system (see Fig.) for single faults. Relevant rows for the case study are colored with gray.

sequence	system inputs					system outputs		
number	VA	VB	VC	VD	VE	TA	TB	TC
1	1	0	0	0	0	e-	0	0
2	1	0	0	0	0	0	0	0
3	1	1	1	0	0	L	0	0
4	1	1	1	0	0	L	L	L
5	1	1	1	1	1	L	N	N

Table 3.5: Operational procedure for the negative bias fault of the level sensor on Tank TA.

the nominal trace are denoted by **bold** text):

$$(1; \mathbf{TA} : \mathbf{e}-), (1; TB : 0), (1; TC : 0),$$

$$(2; \mathbf{TA} : \mathbf{0}), (2; TB : 0), (2; TC : 0),$$
$$(3; \mathbf{TA} : \mathbf{L}), (3; TB : 0), (3; TC : 0),$$
$$(4; \mathbf{TA} : \mathbf{L}), (4; TB : L), (4; TC : L),$$
$$(5; \mathbf{TA} : \mathbf{L}), (5; TB : N), (5; TC : N)$$

Comparing the form with the nominal flattened form (described in Section 3.4.1) these deviations can be determined (as the trace shows, the sensor level for "TA" was always *one qualitative value* below the nominal value):

$$never - happened(5, TA), never - happened(4, TA)$$

$$never - happened(3, TA), later(2, TA), later(1, TA),$$
$$slightly - smaller(5, TA), slightly - smaller(4, TA),$$
$$slightly - smaller(3, TA), slightly - smaller(2, TA),$$
$$slightly - smaller(1, TA)$$

From these sets, the set of final deviations contains two pairs (by taking the deviations from the last and preceding time instants, "4" and "5"):

$$never - happened(5, TA), never - happened(4, TA)$$

$$slightly - smaller(5, TA), slightly - smaller(4, TA)$$

Based on these acquired final deviation pairs and the P-HAZID table of Table 5.4 the diagnostic algorithm (as described in Section 3.3.2) can be started. Relevant rows participating in the diagnostics are marked with a darker color in the P-HAZID table shown in Table 5.4. Taking the first final deviation pair, "never-happened(4,TA)" is found in the "Deviation", and "never-happened(5,TA)" is found in the "Implication" column in the P-HAZID table for a row (denoted by a darker gray color in Table 5.4), so the content of the corresponding "Cause" column is examined. In this case this column contains deviation "never-happened(3,TA)", which is, in turn, present in the set of the initial deviations. In this case, diagnostics moves on, now looking for the pair "never-happened(3,TA)" and "never-happened(4,TA)" in columns "Implication" and "Deviation". This pair can be found just a row below the previously investigated one, with "later(2,TA)" in the "Cause" column which is *also* present in the list of initial deviations.

So the above mentioned procedure continues until the root cause, in this case, "TA-NEG-BIAS" (the root cause for the negative bias fault of tank "TA") is found four rows below the initial pair in the table. Similar reasoning is done for the "slightly-smaller" deviations starting with "slightly-smaller(4,TA)" and "slightly-smaller(5,TA)" (the corresponding section is part of the dark gray area in the table).

The reasoning paths explored for these deviations are depicted in Fig. 3.5 with solid edges.

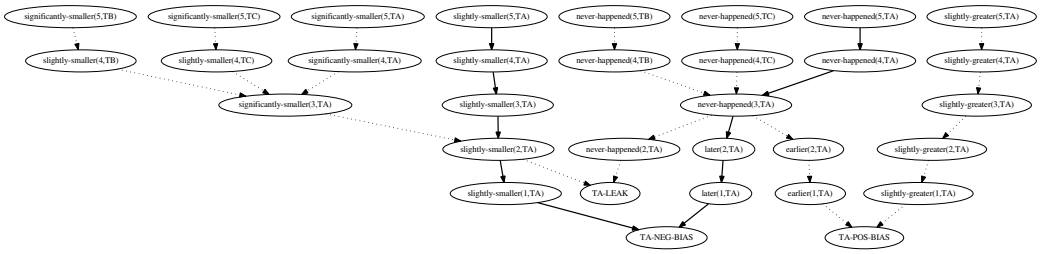

Figure 3.5: All reasoning paths for Tank "TA". Paths relevant to the negative bias fault of the case study are drawn with solid, with other not-relevant paths denoted with dotted edges.

At then end of the diagnostics operation, the procedure emits the root cause, namely that there seems to be a *negative bias failure* with the level sensor on TA ("TA-NEG-BIAS" on the graph). This concludes how the P-HAZID reasoning works in a simple example case.

3.4.6 Observations

These observations can be made regarding the algorithm described in this chapter.

Diagnostic accuracy. If two fault cases result in the same set of deviations then they cannot be distinguished by the algorithm due to the fact that exactly the same diagnostics path is explored by it.

Complexity. Even in simple cases the P-HAZID table can grow very large and complex due to the number of deviations. (The case study already contained 52 rows despite its simplicity.) The complexity brings in the increased possibility for errors and the need for validation upon changes.

P-HAZID table creation. The flattening of traces and collecting the set of deviations between nominal and characteristic traces can be done automatically. On the other hand, expert and/or domain knowledge is needed to set up the reasoning paths between different deviations for a fault. If the reasoning paths are set up, creating the P-HAZID table from it can be done algorithmically.

P-HAZID table validation. Validation can be performed using the algorithm itself, by running it with the complete set of faulty traces (from which the P-HAZID table was created from). Using this, every single reasoning path in the table can be explored and the possible faults can be corrected. Also, faults which lead to the same root cause (due to the structure of the P-HAZID table) can be discovered. The exact details of this validation is not in the scope of this work.

Dual faults. Detection of dual faults, which refer to different output variables is theoretically possible. Due to the fact that deviations refer to single outputs, in this case multiple, independent reasoning paths are explored by the algorithm, and the independent fault modes are determined as the output of the algorithm. However, if the variables refer to the same output variable (like in the case of dual fault "TA leak and TA negative bias") are not always separable from the single fault "TA leak" or "TA negative bias". This happens because they generate the same set of chronological deviations.

Rule-based reasoning. After the deviations had been collected, the reasoning procedure can also be performed using an ordinary pattern matching algorithm (such as the Rete algorithm) or rule engine. In order to achieve this, from every reasoning path an appropriate rule need to be created (as described in 3.3.3) and the deviations between the nominal and faulty traces need to be put as facts into the knowledge-base of the system.

3.5 Summary

In this chapter a novel methodology for diagnosing process systems based on operational procedure execution was described. This methodology can be thought as an extension of the BL-HAZID methodology (described in [34]) with the dimension of time. It needs a nominal trace (where no faults were present), a P-HAZID table as a distilled form of fault model about the process system, and a measured trace. When executed with these inputs it

returns the set of possible root causes which might have been present in the process system during the execution of the observable trace.

The method can deal with single and dual faults (in case they refer to different outputs). Low-level diagnostic information about chronological and quantitative differences between corresponding events of the nominal and observable trace can be captured in the form of deviations and by combining them together in the form of a P-HAZID table, reasoning about possible system level faults in the process system can be performed by the diagnostic algorithm.

As a disadvantage, some parts (such as setting up the proper reasoning paths) in the creation of the P-HAZID table cannot be automated, requiring domain expert knowledge (which might be a difficult task in some cases). Also, the P-HAZID table can grow large and complex, making changes in it more difficult (even though theoretically the algorithm can be used for validating the P-HAZID table after changes were done).

Chapter 4

Diagnostics using clustering

In this chapter the Clustering diagnoser, a cluster-based diagnostic approach is described which can detect faults in process systems during operational procedure execution. This diagnoser is based upon transforming the traces (the representations of an operational procedure execution) to a coordinate space with distance metrics, and then the calculate the distance of this vector representation from known faulty representations in order to find the closest one. In order to create the diagnostic model (which is practically a set of labeled cluster centroids in this case) the algorithm need to be trained with traces coming from known nominal and faulty cases. Prior to that an application-specific mapping function need to be defined as well (which will be used to transform the traces, so that the algorithm can work on them).

In the first part of the chapter the notions specific to clustering, such as mapping function, event and trace coordinates and a way for validating the model is discussed. The second part describes the clustering algorithm in detail. In the third part, using the case study from Section 2.7, the algorithm is demonstrated and its main properties are collected.

4.1 Introduction

One of the most important aspect of a fault diagnostics algorithm for process systems is the fault model which requires significant amount of human expertise and work to set up and maintain. The main aim of the Clustering diagnoser is to suggest a data-driven diagnostic procedure which requires less amount of human assistance during the setup of the fault model (it can create its own fault model from observations) and still remains feasible as a fault diagnostic method. The diagnoser is based on the k-means clustering algorithm (see [3]).

The most significant difference between this approach and the one described in Chapter 3 is that instead of a P-HAZID diagnostics spreadsheet, in this case an application-specific mapping function and a distance metric need to be defined which maps traces with qualitative-valued outputs to a coordinate space (with a distance metric defined), where by clustering these coordinates, a fault model is created (in a form of multiple cluster centroids). The algorithm need to be trained with (possibly many) traces, coming from nominal and faulty scenarios, to form the centroids referring to each scenario. After the training step, when a measured trace is observed, its nearest centroid (fault mode) can be determined, based on the previously defined distance metrics.

Based on other forms of clustering similar diagnostic methods are available in the relevant literature, refer to the literature review in Section 1.2.2 for more details.

4.2 Basic notions

A few diagnostic notions are specific to this diagnostic approach, because in this case traces are not compared on an event-level but as a whole in an application-specific coordinate space. These notions are reviewed in this section. Like in the case of the P-HAZID diagnoser they are based upon the basic notion of *event* and *trace* from Section 2.4 and Section 2.5.

4.2.1 Mapping of qualitative values to real ones

In order to be able to define distances between events and traces, one can convert the qualitative values of the outputs present in events and traces back to real numbers using a mapping function $M : Q \mapsto \mathbb{R}$. In the case of qualitative range space defined in Eq. (2.1) the linear mapping function defined in Eq. (4.1) can be used.

$$M_{linear}(q) = \begin{cases} -1.0 \;\; if \;\; q = e- \\ 0.0 \;\; if \;\; q = 0 \\ 1.0 \;\; if \;\; q = L \\ 2.0 \;\; if \;\; q = N \\ 3.0 \;\; if \;\; q = H \\ 4.0 \;\; if \;\; q = e+ \end{cases} \tag{4.1}$$

The mapping function is application and signal (output) specific at the same time. Separate mappings can be used for different outputs (due to

different output ranges, for example) and it is possible to define a mapping function which weights more the possibly faulty output values (compared to the nominal values) for a single output. Such $non-linear$ mapping can be defined with Eq. (4.2) for the qualitative range set Q in Eq. (2.1). The nominal values ("0", "L"" and "N") are placed next to each other, while the possibly faulty output values ("$e-$", "H" and "$e+$") are placed farther away in both directions.

$$M_{non-linear}(q) = \begin{cases} -10.0 \;\; if \;\; q = e- \\ -2.0 \;\; if \;\; q = 0 \\ -1.0 \;\; if \;\; q = L \\ 0.0 \;\; if \;\; q = N \\ 10.0 \;\; if \;\; q = H \\ 20.0 \;\; if \;\; q = e+ \end{cases} \tag{4.2}$$

In this case nominal outputs are "0", "L", "N", while "H", "$e+$" and "$e-$" denote a fault, therefore they are weighted accordingly.

A linear mapping function can be defined for the refined qualitative range set of Eq. (2.2), as well with Eq. (4.3) below.

$$M_{finer}(q) = \begin{cases} -1.0 \;\; if \;\; q = e- \\ -0.5 \;\; if \;\; q = 0- \\ 0.0 \;\; if \;\; q = 0 \\ 0.5 \;\; if \;\; q = 0L \\ 1.0 \;\; if \;\; q = L \\ 1.5 \;\; if \;\; q = LN \\ 2.0 \;\; if \;\; q = N \\ 2.5 \;\; if \;\; q = NH \\ 3.0 \;\; if \;\; q = H \\ 3.5 \;\; if \;\; q = H+ \\ 4.0 \;\; if \;\; q = e+ \end{cases} \tag{4.3}$$

For the sake of completeness, it is worth mentioning that event *input* values can be also converted to real numbers. For instance, in the case of the two-valued qualitative set of Eq. (2.3) the mapping in Eq. (4.4) can be used. (This function can be considered as an identifying function for the set.)

$$M_{boolean}(q) = \begin{cases} 1.0 \;\; if \;\; q = 1 \\ 0.0 \;\; if \;\; q = 0 \end{cases} \tag{4.4}$$

The effect of using different mapping functions for the output values is described in the first case study in Section 4.4.

4.2.2 Coordinate-vectors of events and traces

Coordinate-vector of events Events with quantitative inputs and outputs can be converted to real-valued vectors (coordinates) using an *event mapping function* $G_{IO} : E \mapsto \mathbb{R}^r$, where E is the space of events and r can be defined as:

$$r = (number\ of\ inputs + number\ of\ outputs)$$

The individual qualitative mapping functions described in Section 4.2.1 can be inverted, it is possible to define mappings which transform from Q back to $\mathbb{R}^r$. Therefore for the transformation of the individual input and output values to $\mathbb{R}^r$, inverse functions of these mappings can be used. For example, the event $(1;"1","0";"N")$ having two inputs ("1" and "0") and one output ("N") is mapped to vector $[1.0, 0.0, 2.0]$ using the inverse of qualitative mapping function M_{linear} described in Eq. (4.1) for the single output and the inverse of the input mapping $M_{boolean}$ from Eq. (4.4) for the single input. The sequence number of the event is not converted in this case, because it is assumed to be always monotonically increasing.

If inputs are also assumed to be failure-free they can be removed from the event representation for diagnostics. In that way a different mapping function $G_O : E \mapsto \mathbb{R}^p$ can be used, where

$$p = (number\ of\ outputs)\ .$$

The output of G_O is called the *event coordinate form*. This form contains only the transformed values of the outputs.

For example, the same event $(1;"1","0";"N")$ in event output coordinate form is just $[2.0]$, using the inverse of mapping function M_{linear} from Eq. (4.1). (Note that because inputs are considered error-free the mapping function $M_{boolean}$ is not used anymore.)

Coordinate-vectors of traces A trace can be also converted to an m length list of r dimensional real-valued vectors, where the sequence number of an event is omitted, r is the dimension of the event as before, and m is the length of the trace. This form can be considered as a piece-wise linear trajectory in an r dimensional space, like the centroids for the first case study described in Section 4.4 on Fig. 4.4.4.

For example, a trace T consists of 4 consecutive events

$$T = (1;"1","0";"0"), (2;"1","0";"L"), (3;"1","0";"N"), (4;"1","1";"N")$$

can be converted to the following 4 long list of 3 dimensional real vectors using function G_{IO}:

$$G_{IO}(T) = [[1.0, 0.0, 0.0], [1.0, 0.0, 1.0], [1.0, 0.0, 2.0], [1.0, 1.0, 2.0]]$$

The inverse of qualitative mapping function M_{linear} from Eq. (4.1) is used for converting individual outputs and the inverse of the input mapping $M_{boolean}$ from Eq. (4.4) for individual inputs. In every element of the vector the first two real numbers correspond to the inputs (two in this case) followed by the real value of the single output (we have a single output only in this example case).

In a similar fashion to events, if inputs are considered error-free they can be removed from the trace representation, by using the same mapping function G_O for every event. This vectorial form is called the *trace coordinate form*. As before, this form only uses the output mapping function M_{linear} from Eq. (4.1).

For example, the trace from the previous example in trace coordinate form is the following:

$$G_O(T) = [[0.0], [1.0], [2.0], [2.0]] \ .$$

4.2.3 Event and trace distances

Event-to-event distance Distance between events are calculated by using a distance function D between the corresponding coordinates of the events (already in *event coordinate form*). For example, the distance between the two-output event

$$G_O(event_1) = [2.0, 2.0]$$

and two-output event

$$G_O(event_2) = [4.0, 2.0]$$

is calculated as follows (using the Euclidean distance as D):

$$D(G_O(event_1), G_O(event_2)) = \sqrt{(2.0 - 4.0)^2 + (2.0 - 2.0)^2} = 2.0 \ .$$

Trace-to-trace distance Distance between traces in trace coordinate form are calculated by summing the distance values between corresponding events

in the traces. Trace to trace distance is interpreted only between traces of equal length.

Let $\varphi(i)$ denote the ith event in trace φ. Based on this, the distance between trace φ_1 and trace φ_2 can be calculated as described in Eq. (4.5) formally (where m is the number of events):

$$E(\varphi_1, \varphi_2) = \sum_{i=1}^{m} D(\varphi_1(i), \varphi_2(i)) \tag{4.5}$$

For instance, the distance between the two-output trace in *trace coordinate form*

$$[[0.0, 0.0], [1.0, 2.0], [2.0, 3.0]]$$

and the two-output trace in *trace coordinate form*

$$[[0.0, 0.0], [1.0, 1.0], [1.0, 1.0]]$$

can be calculated as (using the Euclidean distance as D like in the case of events):

$$E(\varphi_1, \varphi_2) = \sqrt{(0.0 - 0.0)^2 + (0.0 - 0.0)^2} +$$
$$\sqrt{(1.0 - 1.0)^2 + (1.0 - 2.0)^2} +$$
$$\sqrt{(1.0 - 2.0)^2 + (1.0 - 3.0)^2} =$$
$$\sqrt{0 + 0} + \sqrt{0 + 1} + \sqrt{1 + 4} = 0 + 1 + 2.236 = 3.236$$

It is theoretically possible to use other distance functions. However in this chapter only the Euclidean distance is used for calculating distances between events and traces. Note that this simple distance function in its current form can only compare traces of equal length.

4.3 The diagnostic method

The proposed diagnostic method uses training traces which belong to the identified normal or faulty modes of the system in order to recognize the faulty mode of a not known trace (further referred as *measured trace*).

The traces in the training set are annotated and labeled by the operating personnel with the faulty mode they recognized. This label may refer to a variety of faults, disturbances or malfunctions or even to a combination of those. This opens up possibilities to diagnose both internal faults, such as a broken pipe or leaking tank in the system, and external disturbances such as changes in the process feed using the proposed method.

The normal operation is considered as a special faulty mode with no fault, therefore we also need observed traces in the training set characterizing this situation. When only traces of normal operation are available with some threshold distance characterizing its accuracy, then only fault detection is possible, i.e. one can decide if a measured trace belongs to the normal operation mode, or some fault occurred the nature of which is not known.

Objective of diagnosis Given a set of training traces from different faulty and fault-free operational scenarios, and a possibly faulty measured trace (both is given in trace coordinate form) identify the operational scenario to which the measured trace is - most likely - belongs to, based on its distance from the centroids calculated from the training data.

4.3.1 Additional assumptions

The following assumptions are made in addition to the ones listed in Section 2.6 for this diagnostic method:

- The length of training traces and diagnosable traces are the same. This is required because the simple Eucledian distance function (see Section 4.2.3) works on traces of equal length.

- Enough number of training traces are available so that the diagnoser can be trained appropriately. This number is application-specific, but the accuracy of the centroids can be checked by validating the fault model (we will address this in Section 4.3.2).

4.3.2 Clustering of traces

As described in [4] in detail, clustering in general is a form of unsupervised learning where the objective is to find regularities (certain patterns occur more often than others) in a vector space. In our case the vector space is formed by the traces in trace coordinate form in accordance with the assumptions listed in Section 4.3.1. In this regard, a cluster can be defined as a set of training traces with similar patterns (having the same fault) while a centroid (center of a cluster) can be defined as a mean of these traces in trace coordinate form.

A popular method of clustering a vector space with distance metrics is the K-Means Clustering algorithm where the number of clusters, K is given as an input. For more details, see [4]. After conversion of the traces, the K-Means clustering algorithm is executed with $K = 1$ for every diagnostic

scenario (faulty and fault-free) to find a single centroid for the set of training traces having the same pattern. Because of this, the diagnostic approach described here - not like clustering in general - can be considered as a form of *supervised learning*, the centroids representing the different scenarios are trained separately.

In order to describe the clustering algorithm formally, a few *basic definitions* are needed.

1. Given a distance metric D (such as the Euclidean distance described in Section 4.2.3).

2. Given a set X, let us denote the number of elements in X by $|X|$.

3. Given n diagnostic scenarios let i be the scenario index going from 1 to n.

4. Given a set of traces Y in trace coordinate form, and centroids Z and W. Let the relation Y *belongs to* Z denote the set of traces from Y which are closer to Z than W using distance metric D. Similarly, Y *belongs to* W denotes the set of traces from Y which are closer to W than Z using the same distance metric D. Consequently, $|Y| \geq |Y \textit{ belongs to } C|$ for every centroid C.

Acquiring and validating the cluster centres For every faulty scenario i a set of traces in trace coordinate form are provided for creating and validating the centroids. This given set is split into a training set T_i (for creating the centroids) and a validation set V_i (for performing validation of the centroids). The split is homogeneous and the ratio $\frac{|T_i|}{|V_i|}$ is application specific.

Executing the K-Means clustering with $K = 1$ on every training set T_i, the centroid C_i is formed for scenario i. These centroids are created in single trace coordinate form, and they might not be equal to any specific input trace of the training set. A centroid, like a trace is a piece-wise linear trajectory in an m dimensional space where m is the number of outputs, and the length of the piece-wise linear trajectory is the length of the trace (number of events in the trace). For example, for a training set which contains 100 event long traces, and 20 output values for every event, the representation of the trace will be a 100 long line in 20 dimensional space.

After every centroid C_i is formed from the training sets, the validation sets are used to calculate the *fault detection rate* (FDR_i) for every faulty scenario i using the formula defined in Eq. (4.6). The sequence $\{FDR_i | i = 1...n\}$

also gives an overall fitness of the model, where

$$FDR_i = \frac{|V_i \; belongs \; to \; C_i|}{|V_i|} \tag{4.6}$$

Note that this FDR value is conceptually the same as the value which was the base for the comparison for the different diagnostic approaches in review article [54].

4.3.3 Steps of the Diagnostic Procedure

The steps are executed in two phases: (i) an off-line **training phase** which creates the trace clusters identified with a fault label and its centroid, and (ii) an on-line **diagnosis phase** which can be executed with the known clusters for an measured trace we want to diagnose.

1. **Training phase.** Every input trace is converted to trace coordinate form using the method in Section 4.2.2 for every training scenario. Because inputs are considered as fixed and error-free (both in their number and value) and sequence numbers are increasing strictly monotonically (due to the basic assumptions laid down in Section 4.3.1), only outputs are participating further in clustering (the inputs and the sequence numbers are not present in this form).

2. As defined in Section 4.3.2, sets T_i, V_i and C_i are created for each training scenario $i = 1, ..., n$.

3. The diagnostic model is validated using C_i and sets V_i after all centroids are determined. FDR_i values are calculated for every training scenario as described in Section 4.3.2 Eq. (4.6) for each $i = 1, ..., n$.

4. Each cluster centre C_i is labeled with the inputs of training scenario i and the particular fault (those are fixed).

5. **Diagnosis phase.** Given a measured trace which is converted into *trace coordinate form*, the nearest centroid can be determined by computing its distances from centroids C_i for training scenarios $i = 1, ..., n$, using a distance such as the one described in Section 4.2.3, and finding the closest C_i. The fault index i which corresponds to the nearest centroid is regarded as the most probable fault mode of the system during the execution of the measured trace.

4.3.4 Dealing with faults not considered a priori in the training set

When *unknown faults* (faults missing from the training set for the diagnoser) are present during the *Diagnosis phase* of the diagnosis (see Section 4.3.3), we can define a modified distance function having a *distance threshold* T in order to separate these cases from the known faults. With this value, a modified distance function in Eq. (4.7) (based on the simple Eucledian distance E described in Eq. (4.5)) can be used

$$E_T(\varphi_1, \varphi_2, T) = \begin{cases} E(\varphi_1, \varphi_2) & E(\varphi_1, \varphi_2) <= T \\ \infty & otherwise \end{cases} \tag{4.7}$$

in a way that if the distance is ∞ then the result of diagnosis is "Unknown". The distance threshold T can be chosen based on the diameter (the maximum distance between elements in the training set belonging to the same centroid). In general, it can be said that a diagnostics method shall prepare for the presence of unknown faults. By using a distance function as the one described in Eq. (4.7) this can be achieved. Application of a distance function like this is out of the scope of this chapter.

4.3.5 Trace coordinates with different measurement units

If the different trace coordinates have diverse measurement units, distance calculation using the Eucledian distance might return incorrect results. This happens because this simple distance assumes uncorrelated inputs with equal variances. If this poses a problem, then a different distance metric need to be used. For instance, the Mahalanobis distance (see Eq. 4.8) can be used instead. This distance function can be considered as a generalization of the Eucledian distance - when C is the identity matrix it behaves identically. Choosing an appropriate value for C, differences in measurement units can be handled. The Mahalanobis distance can also take dependencies or correlations between coordinates into account.

$$D_M(\varphi_1, \varphi_2) = \sqrt{(\varphi_1 - \varphi_2)C^{-1}(\varphi_1 - \varphi_2)} \tag{4.8}$$

For more details on this distance function, refer to [3]. In this chapter, for the sake of simplicity we are only dealing with the Eucledian distance, because all of the measurement units on the outputs are the same, and it is assumed that they are not in correlation with each other.

4.3.6 Limitation on the lengths of traces

One of the assumptions listed in Section 4.3.1 was that the length of the traces need to be the same for the diagnoser to function properly (this was needed because of the simple Eucledian distance function). In reality this limitation makes the practical applications of the method difficult in some cases, due to the fact that traces might vary in length. Multiple approaches can be tried to deal with this problem:

1. Traces can be simply trimmed to equal length, by taking the shortest trace from the training and validation set and trimming all other traces to this length. The disadvantage of this method is that some data inevitably will be lost. In the example case study described in Section 7.2.1 of the Appendix this approach was chosen.

2. A technique called Dynamic Time Warping (DTW) can be used to calculate "distance-like" metrics for sequences (traces in our case). This method can find the minimal distance between two sequences which vary in time and speed - this allows traces of different length to be compared with each other. The algorithm is used eg. in speech recognition to compare different speech signals with each other (in speech differences in time and speed are very common). A good overview article on DTW can be found at [39], while an application of DTW for time series analysis can be read at [8]. Unlike in the previous proposal, no data will be lost in this case.

Dealing with this constraint on the length of traces is out of the scope of this chapter. Different applications of the Clustering diagnoser might require different solutions for this limitation problem.

4.4 A diagnostic case study

To demonstrate the Clustering diagnostic approach a simple three-tank case study from Section 2.7, shown in Fig. 3.4 is used, like in the case of the P-HAZID diagnostics in Chapter 3, with the nominal procedure filling up all the tanks. As described earlier, this process system consists of three tanks, where the main tank "TA" is connected to auxiliary tanks "TB" and "TC". The nominal operational procedure fills up the whole system with fluid then opens both output valves on the auxiliary tanks.

In this case study (in order to further illustrate the capabilities of the diagnostic algorithm) the following differences are present compared to the P-HAZID case study described in Chapter 3:

1. When training the Clustering diagnoser with many traces, effect of measurement errors are simulated. Some of the training traces are having incorrect qualitative output values described in Section 4.4.1. The rationale behind this is twofold: first, to show how the diagnostic approach handles a more realistic scenario, and secondly, to show the effect of using different output mapping functions during diagnostics.

2. Single and dual faults were considered in this case, in order to show that the diagnostic algorithm can deal with dual fault scenarios as well. In the case of the P-HAZID table based diagnoser only single failures were dealt with, although theoretically it would have been possible to use it for diagnosing dual faults with some limitations. These limitations do not apply for the Clustering diagnoser.

The aim of the case study is to demonstrate the execution of the algorithm, and to survey the effects of three different mappings on the diagnostic accuracy.

4.4.1 Measurement errors

Given a qualitative set Q, for every qualitative value $q \in Q$ the

$$neighbourhood(q, \ell)$$

is defined as a set containing all elements from Q which, respecting the ordering of Q, are not farther than a given $\ell > 0$ natural number from q. The neighborhood does not contain the element itself, $q \notin neighborhood(q, n)$. The parameter ℓ is called the level of neighborhood.

For example, given the qualitative set defined in Eq. (2.2) then

$$neighborhood(N, 2) = \{L, LN, NH, H\},$$

as these are the qualitative values not farther than "2" from "N".

Qualitative output values usually come from measurements which might be prone to measurement errors. The errors might be large enough (or the qualitative set can be fine enough) so the observed value does not match the actual value even in the considered qualitative range space.

Given a set of qualitative values Q, such as the ones described in Eq. (2.1) or Eq. (2.2), a function

$$ERRSIM(q, \ell, p) \ : \ Q \mapsto Q$$

can be defined to simulate the effect of measurement errors, which transforms a qualitative value q to a qualitative value w from the neighborhood

(considering neighborhood level ℓ) of the value q with a given probability p. This function is useful to simulate the effect of sporadic measurement errors in large number of training-input traces.

4.4.2 Description of the case study

The case study had three different runs. The output mapping function (as described in 4.2.1) was different between the runs, and the effects of the different mappings on the accuracy of the diagnostics are surveyed and compared. The used distance metric was the Euclidean distance as described in Section 4.2.3.

First the reference traces for the nominal case and for every faulty case were created. Training trace sets were formed from each reference trace copying them 5000 times and applying a simulated measurement error function (using $ERRSIM$ as described in Section 2.1) on each set with 6% error probability ($p = 0.06$) with neighbor level = 2 ($\ell = 2$) in the refined case of Section and neighbor level = 1 ($\ell = 1$) in the other two case studies.

These training trace sets were given as input to the algorithm as described in Section 4.3.2 so that the diagnostic model (practically the distribution of the centroids for every scenario in the coordinate space) could have been created. This model is created for every mapping function, and then they were compared with each other. The distribution of FDR values as it was described in Section 4.3.2 was used as a basis for comparison.

4.4.3 Execution of the algorithm

After the training traces are provided to the algorithm, the traces are converted to trace-coordinate form after the input and time is trimmed from the events of the trace.

So for example the nominal trace of Table 4.1 having a single measurement problem for output "TB" is mapped to this trace coordinate using the refined mapping function from Eq. (4.3), with the effect of the measurement error denoted in **bold**:

$$(0.0, 0.0, 0.0), (1.0, 0.0, 0.0), (2.0, \mathbf{0.5}, 0.0), (2.0, 1.0, 1.0), (2.0, 2.0, 2.0)$$

Note that due to Step 1 of the diagnostic procedure described in Section 4.3.3 *only* output values are converted, the inputs and the time instant were trimmed from the event prior to the conversion operation (due to the fact that the inputs are assumed to be free of faults and the time instant is always increasing due to the interpolation step).

sequence	system inputs					system outputs		
number	VA	VB	VC	VD	VE	TA	TB	TC
1	1	0	0	0	0	0	0	0
2	1	0	0	0	0	L	0	0
3	1	1	1	0	0	N	**0L**	0
4	1	1	1	0	0	N	L	L
5	1	1	1	1	1	N	N	N

Table 4.1: Nominal operational procedure having a problem due to measurement error in output "TB" for the third time instant (in **bold**). System input "0" means "closed", "1" means "opened" valve states, while system output "0" means "no level", "L" means "low", "$0L$" means a value between "normal" and "low", "N" means "normal" levels in the tank, according to Eq. (2.2).

Every other input trace is converted in the similar manner. After conversion, as described in Section 4.3.2, k-means clustering with $K = 1$ is executed to find the centroids in every case. Fig. 4.1 shows nine of these centroids in the form of three dimensional piece-wise linear trajectories. On this figure on every subgraph a single centroid can be seen which is a 5 point long trajectory in most cases - due to the fact that the trace is also 5 event long - except for the bottom three graphs where all points overlap each other, and the middle graph in the left column where some points overlap. For all cases the three axes represent the qualitative fluid level values in the three tanks (TA, TB and TC).

After the centroids are determined, the diagnoser is trained and it is ready to accept observable traces and assign them to possible faults. This is done by converting the trace first to trace coordinate form, and then finding the nearest centroid. For example, for the trace in Table 4.2, these coordinates are assigned (using mapping function in a similar fashion from Eq. (4.3)):

$$(0.0, 0.5, 0.0), (1.0, 0.5, 0.0), (1.5, 0.5, 0.0), (1.5, 1.5, 1.0), (1.5, 2.0, 1.5)$$

Based on the distribution of the centroids and a previously trained diagnoser, the results of the diagnostics (the distances from the centroids) can be seen in Table 4.3. Based on this result the trace is diagnosed as *nominal* by the algorithm (the nominal centroid was the closest to the coordinate vector of the trace). In fact, this trace was created artificially from the nominal trace, with applying more simulated errors on it that the measurement error simulation function from Section 4.4.2 would have added. Note the nature of the severe measurement error in the output value "TB":

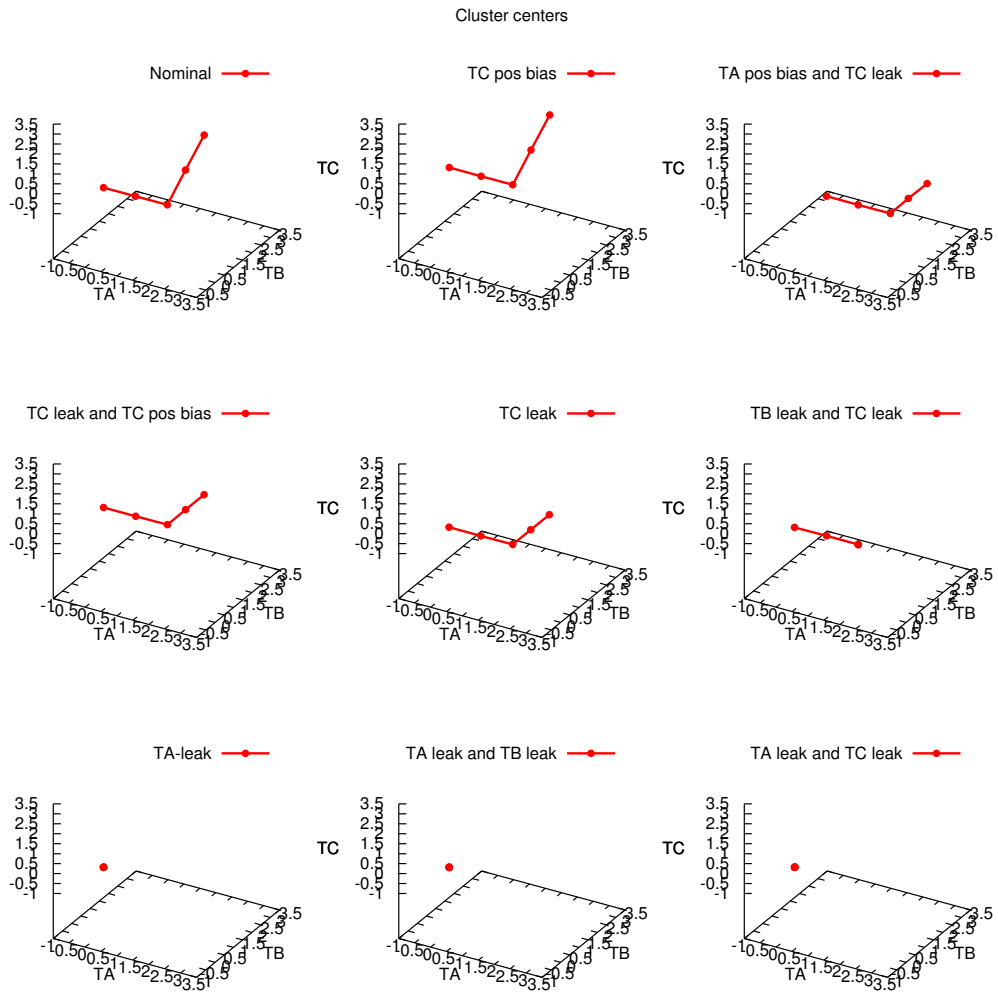

Figure 4.1: A few interesting centroids for some fault cases in piece-wise linear trajectory form. The three axes represent values in the tanks "TA", "TB" and "TC".

- in the first three instants it is "$0L$" instead of "0"
- while in the fourth instant it is "LN" instead of "L".

This causes the second most probable fault mode to be *TB leak and TB positive bias* while the third most probable to be *TB positive bias* (even though the trace was created originally from the nominal trace, in its value for "TB" it is closer to these faults).

sequence number	system inputs					system outputs		
	VA	VB	VC	VD	VE	TA	TB	TC
1	1	0	0	0	0	0	**0L**	0
2	1	0	0	0	0	L	**0L**	0
3	1	1	1	0	0	**LN**	**0L**	0
4	1	1	1	0	0	**LN**	**LN**	L
5	1	1	1	1	1	**LN**	N	**LN**

Table 4.2: Measured trace with many measurement-related problems (denoted by **bold**).

centroid (scenario) label	distance from centroid
NOMINAL	1.4127
TB-LEAK and TB-POS-BIAS	1.7316
TB-POS-BIAS	1.7320
TA-NEG-BIAS	1.9861
TA-NEG-BIAS and TB-POS-BIAS	2.2189
TC-LEAK and TC-POS-BIAS	2.2325
TC-LEAK	2.2358
TC-NEG-BIAS	2.4121
...	...

Table 4.3: The closest centroids for the coordinates of trace in Table 4.2

4.4.4 Effect of different mapping functions

The case study was executed multiple times to survey the effects of different mapping functions.

Coarse linear mapping

First a very simple linear mapping function (see Eq. (4.1)) was used, and the FDR values for every scenario were calculated. The ordered FDR values for this case can be seen in Fig. 4.2.

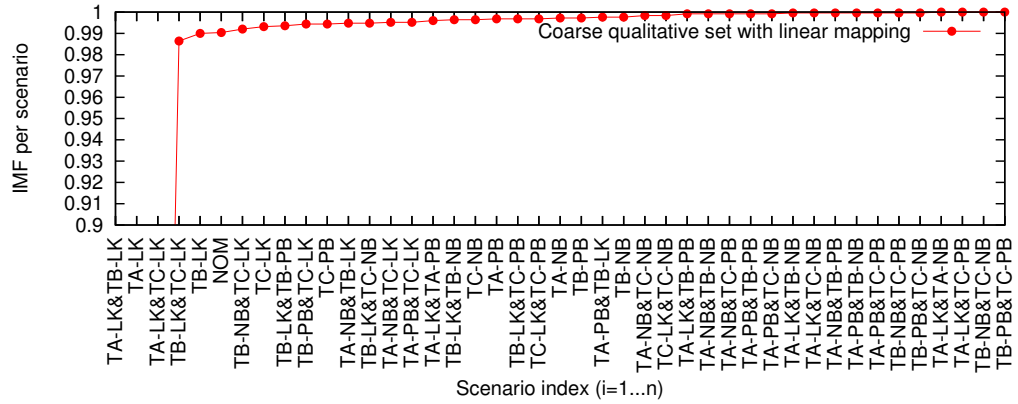

Figure 4.2: FDR values in increasing order for the coarse linear mapping function. Values for *TA leak*, *TA and TB leak* and *TA and TC leak* are smaller than 0.9 hence not shown.

Coarse nonlinear mapping

In the second case a nonlinear mapping function (see Eq. (4.2)) was used instead of the linear one. The expectation in this case was to have increased diagnostic accuracy because the function would place obviously erroneous outputs (unmeasurably low "$e-$" or the dangerously high "H") farther away from each other in the coordinate space. However, due to the presence of the measurement error, these erroneous values were used instead of otherwise nominal outputs. Because of this, the centroids became less accurate than even in the coarse linear case. This effect can be seen in Fig. 4.3 via the ordered FDR values of the model.

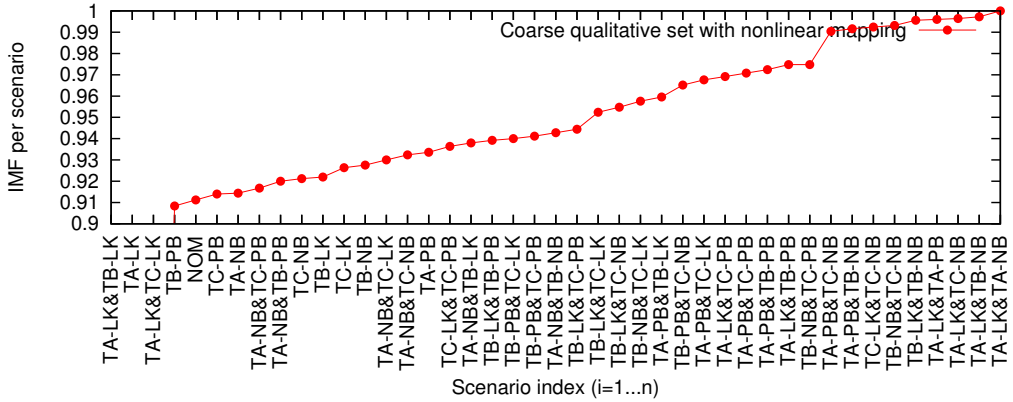

Figure 4.3: FDR values in increasing order for the coarse nonlinear mapping function. Values for *TA leak*, *TA and TB leak* and *TA and TC leak* are smaller than 0.9 hence not shown.

Refined linear mapping

In the third case, using a linear mapping function with higher accuracy (see Section 4.3) was used. This function, despite the presence of simulated measurement errors, could improve the overall accuracy of the diagnostics operation. The ordered FDR values can be seen in Fig. 4.4 for this mapping function.

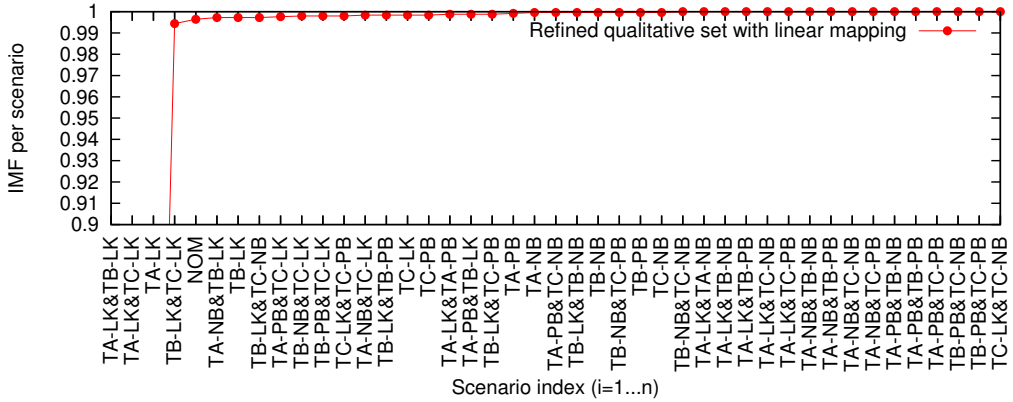

Figure 4.4: FDR values in increasing order for the refined linear mapping function. Values for *TA leak*, *TA and TB leak* and *TA and TC leak* are smaller than 0.9 hence not shown.

4.4.5 Observations

The following observations can be made on the diagnostic algorithm:

Significance of the mapping function and distance metric. Diagnostic accuracy highly depends on the correct trace mapping function and distance metric. The case studies shown significant differences in the result of diagnostics on practically the same data set using just different mapping functions (even though in this case the same distance metric is applied):

- The diagnostic accuracy improved when a linear mapping function with refined qualitative set was used (see the FDR distribution on Fig. 4.4).

- The diagnostic accuracy worsened when a non-linear mapping function was used (see the FDR distribution on Fig. 4.3). In this case the combination of the simulated measurement error and the non-linear mapping had been responsible for the loose position of the centroids and the slightly less accurate diagnostics.

A proper combination of a mapping function and distance metric places traces belonging to different scenarios in a greater distance from each other, in that way cluster centers can be formed more appropriately and diagnostic accuracy can be improved.

Overlapping cluster centers. If, due to the mapping or the nature of the input data, traces map very closely to each other then their centroids will be also very close in the coordinate space, which makes them indistinguishable from each other. This phenomenon can be observed in Fig. 4.1 for faults *TA leak*, *TA leak and TB leak* and *TA leak and TC leak*. In this case, due to the leak of "TA" no fluid can get into the system, so "TB" and "TC" will be empty as well (the leak on these tanks cannot be separately detected). This causes the mappings for the three traces on top of each other. This can be seen on the FDR distribution graphs in Fig. 4.2, 4.3 and 4.4 where all the FDR values for these scenarios were far below than 0.9 (90%), at around 0.3 (30%). The relevant centroids can be seen in Fig. 4.1 in the bottom row.

Simulated measurement errors. If simulated measurement errors are eliminated, then - expect for the case when cluster centers are overlapping - the diagnostics has 100% accuracy in every scenario.

Faults affecting different outputs. If the faulty scenarios affect different output variable(s) on the process system, and their effects are independent of each other, they can be distinguished even though they appear simultaneously like in the case of the P-HAZID diagnoser. Taking a simple example a bias failure in "TA" and a leak on "TC" at the same time affect different outputs (level sensor of "TA" and "TC"), they are not related to each other, hence they can be detected and distinguished from the rest of the faults. See the scenario *TA positive bias and TC leak* in Fig. 4.1.

Faults affecting the same outputs. If the faulty scenarios affect the same output variable on the process system, but their effect is independent of each other, they can still be distinguished. For example, a tank leak causes the level to be constant "0" in the tank, but a positive bias failure changes the sensor value to constant "L" level. See the centroids from faults *TC leak and TC leak and TC positive bias* in Fig. 4.1.

Similarity to the K-NN classifier. The diagnostic method in its current form is analogous to the *K-nearest-neighbor classifier* (*K-NN classifier*) described in [3], which is a supervised classification method. This is due to that the most probable fault is determined by calculating the closest centroid to the coordinate form of the diagnosable trace (which can be considered as a border case of the K-NN classification problem with K=1).

Application specific distance metric. The distance function is independent of the method and need to be chosen appropriately based on the process system. In many cases a simple Eucledian distance might work, however as described in Section 4.3.4 and 4.3.5, it has several issues that need to be addressed (such as how to detect unknown faults, and what to do when coordinates have different measurement units) that can be solved by choosing an appropriate distance function, which suits the application more.

4.5 Summary

In this chapter the Clustering diagnoser was described which can suggest fault mode(s) for operational procedures in process systems like the previously described P-HAZID diagnoser. It uses a transformed representation of traces to a coordinate space to perform diagnostics. Distance of the representation from known nominal and faulty trace representations are calculated, the label of the closest representation is given as the result of the diagnostics as the most probable fault mode (or the nominal behavior).

This method can deal with occurrences of single and dual faults, even when the faults has an effect on the same output variable (this is a significant difference compared to the P-HAZID diagnostic approach). As the case study in this chapter had shown it, the diagnostic algorithm could deal with measurement errors - which might be inevitable when a large number of traces are collected from process systems for training purposes.

On the other hand, an application-specific mapping function and distance metric need to be defined which can place different faults far away from each other in the coordinate space where clustering occurs so that the diagnostics operation has acceptable accuracy. However, apart from definition of this transformation the diagnostic process can be fully automated.

In Chapter 7, the operation of the Clustering diagnoser is also demonstrated on a commonly known control and benchmark process system, the Tennessee-Eastman Challenge problem.

Chapter 5

Decomposition in event-based diagnostics

In this chapter a diagnostic decomposition method is presented, which is based on a component model of a process system and can be applied with the already discussed P-HAZID (Chapter 3) and Clustering (Chapter 4) event-based diagnostic approaches in order to decrease their complexity and to share common parts of the fault model needed by these methods during diagnosis. Using this higher-level approach, traces can be distributed among separate components of the system and fault diagnostics can be performed on a component-local level.

This method itself is independent from the specifics of the underlying diagnostic algorithms, while at the same time it can be used to decrease the overall complexity of the methods. This is achieved by bringing the diagnostics down to a more localized level, closer to the actual components of the process system.

This chapter is organized into three main parts. First, the notions specific to decomposition (component based system model, component graphs, trace fragments and how they are distributed between separate components) are discussed. In the second part the decomposition algorithm is presented. Finally, as a case study, a decomposed P-HAZID and Clustering diagnoser is demonstrated on the process system described in Section 2.7, and the main properties of this new approach are surveyed. The decomposition approach is compared with the standalone diagnostic approaches (the same case study is used in all three chapters), and its main advantages over them are collected at the end of the chapter.

5.1 Introduction

Using the previously discussed P-HAZID and Clustering diagnostic methods (in Chapter 3 and in Chapter 4) for fault diagnostics, root causes in faulty process systems can be found in principle, but in the case of more complex systems creating the fault model and performing the diagnostics might be a very complex task. This due to the fact that the fault diagnostics problem they attempt to solve is an NP-hard problem. This practically means that in the worst case the complexity grows with the number of components in an exponential fashion, which might cause a very large number of rows in the P-HAZID table (making it very difficult to create and maintain), or a very large number of dimensions in the case of the Clustering diagnoser (which can seriously limit the performance of the clustering operation).

The structural decomposition method in this chapter tries to address this problem, by decomposing the diagnostic task to smaller sub-tasks with lower complexity, better runtime and a reduced fault model. This higher-level method will be demonstrated on the already described P-HAZID and Clustering fault diagnostic methods from Chapter 3 and Chapter 4. In [45] the idea of structural decomposition had been applied for the P-HAZID diagnoser, and demonstrated on a case study briefly.

As a main foundation of the P-HAZID approach, the BL-HAZID metholodology (see Section 1.2.1 as well as [42]) also uses the idea of structural decomposition. For instance, during the BL-HAZID analysis this helps the analysis team focusing on a single subsystem at one time, making their task significantly less complex.

Refer to the literature review in Section 1.2.3 for further applications of the idea of decomposition on the field of process system diagnostics.

5.2 Components, component structure and trace decomposition

Prior to going into details of the method, the notions specific for the decomposition approach are discussed, such as *components*, *component graphs* and how traces are distributed among separate components during diagnosing the system using *trace fragments*. These notions depend on the common definitions of *event* and *trace* already discussed in Section 2.4 and Section 2.5.

Using the physical characteristic of the system and the system-level nominal operational procedure (trace), *components* can be identified in the system and the connection between them can be modeled by a directed graph, called

a *component graph*. Using this component graph and a defined *starting component* (this denotes the component where the operational procedure starts at) all possible *component paths* (different execution paths in the component graph) of the system can be determined. Diagnosis progresses until the last component in every component path is reached which has a true *starting condition* (denoting whether a component can be reasonably diagnosed). Component paths can be diagnosed in parallel, independently of each other. Further in this section, these notions are demonstrated and discussed.

5.2.1 Component

A *component* is a part of the process system which can have faults on its own, and it is elementary, i.e. it contains no other components. It has related inputs and outputs in the global trace, based on these, its fault can be diagnosed by a diagnostic algorithm. Process systems are made of independent components, for example, a tank or a pipe in a process system can be considered as a component. In order to diagnose it, the component has diagnoser-specific fault model assigned, such as a P-HAZID table or a list of nominal and faulty cluster centers. This model is local to the component, and depends on its type. Therefore, it can be shared among components which belong to the same type during diagnostics.

5.2.2 Component graph and component path

Based on a given trace, a process system can be considered as a set of *connected components* where the connections depend on the order the trace actuates components in the system. A *component graph* of a process system is a directed graph of components and their connections with each other. The *nodes* in this graph are the components of the process system, while the *edges* describe the order these components are used in the trace. For instance, if A and B are nodes in the component graph, then an edge exists between A and B if the operational procedure actuates component B after component A in the trace. Therefore, the nodes in a component graph are structure specific, while the edges are operation (trace) specific. For instance, for a different trace (operating on the same process system) a graph with different edges and the same set of nodes can be created.

A component with no incoming edges is called a *starting component* in the component graph, while a component with no outgoing edges is called a *final component*. From the nominal trace the *starting component* and the possible *final component*(s) can always be determined - the *starting component* is always the first, while the *final component*(s) are always the last

component(s) which are actuated in the trace.

A *component path* is a path in the component graph from a *starting component* to a *final component*. Component paths can be diagnosed separately in parallel with each other.

A very simple example component graph along with the corresponding process system can be seen in Fig. 5.2 for the trace in Table 5.1 (this is in fact the common process system and trace described in Section 2.7). This process system has three tank components which can be diagnosed separately. The nominal trace for the graph fills up all three tanks with fluid. Tank "TA" is a bigger tank with one input ("VA") and two output pipes ("VB" and "VC"), while tanks "TB" and "TC" are smaller tanks with a single input ("VC" and "VB") and output pipes ("VD" and "VE"). "TB" and "TC" belong to the same component type, so the same fault model can be used to diagnose them. The nominal trace is the same which had been already described in Section 2.7.1. In this example component graph, based on the nominal operational procedure in Table 5.1, two component paths can be defined (both of them are two element long): ("TA","TB") and ("TA","TC").

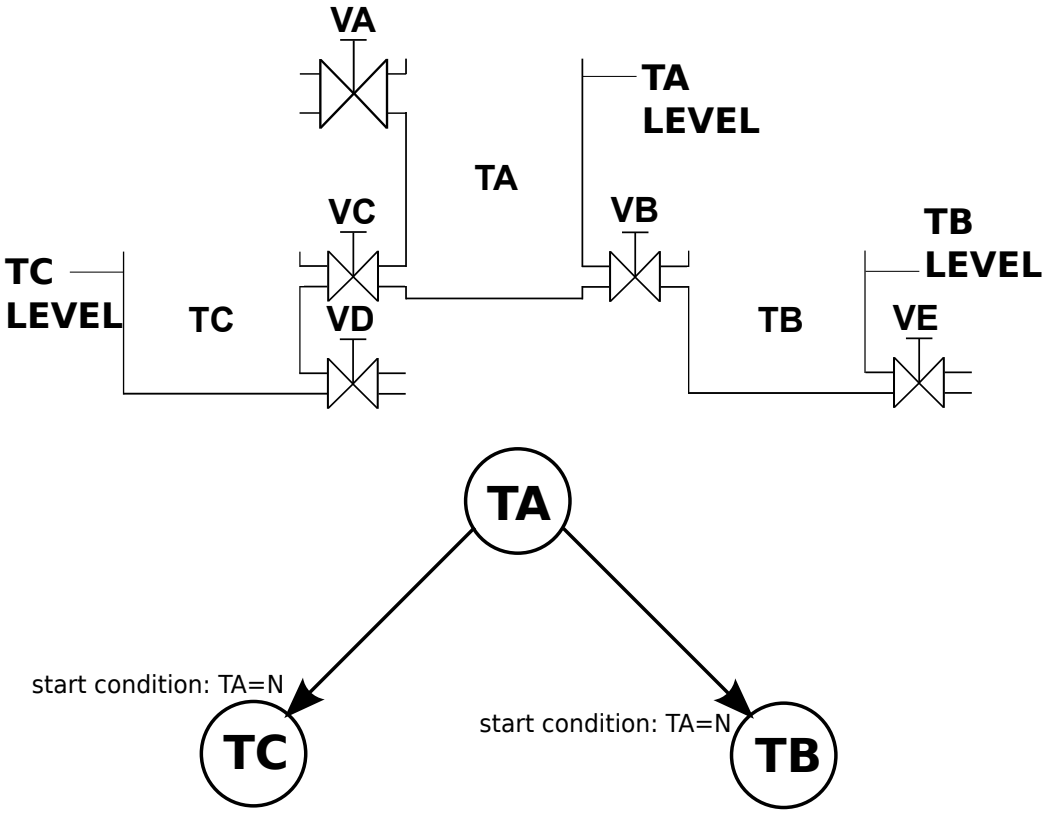

Figure 5.1: Example component graph for a simple serial system.

sequence number	system inputs					system outputs		
	VA	VB	VC	VD	VE	TA	TB	TC
1	1	0	0	0	0	0	0	0
2	1	0	0	0	0	L	0	0
3	1	1	1	0	0	N	0	0
4	1	1	1	0	0	N	L	L
5	1	1	1	1	1	N	N	N

Table 5.1: Complete nominal trace for the process system in Fig. 5.2 with all components.

5.2.3 Start condition

Optionally, a component can have a *start condition* which describes a global system-level condition on input and output value(s) of the diagnosable system under which the particular component can be reasonably diagnosed. This condition usually used to skip further diagnostics on the component path if a severe fault is present at a component in it.

Using a start conditions for a components during diagnostics in simple cases can make the result cleaner and easier to interpret. Note that this is only possible when a simple expression over the input and output values of the component can be defined to determine if it is functioning properly or not.

For instance, in the case of the component graph in Fig. 5.2, for components "TB" and "TC" such conditions can be that the fluid has reached nominal level ("*N*") in the preceding component "TA". If "TA" is already having a severe fault, there is no point in trying to diagnose the following components in the component graph. For instance, this severe fault can be a leakage which prevents "TA" from keeping any fluid inside.

5.2.4 Component mapping function

In a trace the inputs and outputs for different components have different identifiers, even though the components belong to the same type (and exactly the same fault model, such as their P-HAZID table could be used to diagnose them). This mapping is specific to the system and unique to every component (it is not the same between even components belonging to the same type). In this case a *component mapping function* can be used to transform the global input and output identifiers to have a common component-local identifier. These common component-local identifiers can be referred from the fault model for the component (such as the P-HAZID table or centroids for the

Clustering diagnoser) in order to make it shareable for different components of the same type.

For example, taking the trace in Table 5.1 into account, where "TB" and "TC" are components of the same type:

- If "TB" need to be diagnosed as a component, two input valve states ("VB" and "VE") and a single tank level ("TB") need to be considered.
- If "TC" need to be diagnosed as a component, two input value states ("VC" and "VD") and a single tank level ("TC") need to be considered.

In order to make the fault model shareable, two different component mapping functions can be used for the tanks. For "TB" the function can be seen in Eq. (5.1), while for "TC" it can be seen in Eq. (5.2). After the mapping is complete the inputs "IN","OUT" and the output "LEVEL" can be referred from the P-HAZID table (provided we are using a P-HAZID diagnoser) instead of "VB","VE","TB" in case of "TB" and "VC","VD","TC" in case of "TC". For example, in the case of the a P-HAZID diagnoser, without the mapping the diagnostics would require two different P-HAZID tables, with exactly the same content but different identifiers.

$$M_{TB}(id) = \begin{cases} "IN" & id = "VB" \\ "OUT" & id = "VE" \\ "LEVEL" & id = "TB" \end{cases} \tag{5.1}$$

$$M_{TC}(id) = \begin{cases} "IN" & id = "VC" \\ "OUT" & id = "VD" \\ "LEVEL" & id = "TC" \end{cases} \tag{5.2}$$

5.2.5 Trace fragments

Based on the decomposition idea, every trace that operates on the whole process system can be split into smaller *trace fragments* (sub-traces) referring to different components of the process system. A *trace fragment* for a component is the part of the overall trace which refers to that single component only. It has different component-local inputs and outputs compared to the overall trace (other component inputs and outputs are not considered and identifiers are mapped according to Section 5.2.4) and limited length (other parts of the trace are not taken into account) compared to the original trace. A component-related *trace fragment* is sufficient for diagnosing the component (along with the diagnoser specific fault model, such as the P-HAZID table of the component).

If there are multiple components in the system, then there are inputs which directly relate to neighboring components (like a valve on a pipe which is between two tanks, feeding one from the fluid coming from the other belongs to both tanks). These inputs are included in the trace fragments of both components. Moreover, there should be a specific time instant in the trace in which this boundary input is manipulated (for instance, in case of the valve switched to open from closed) then this time instant need to be present in both trace fragments for the neighboring components (at the end of the preceding, and at the beginning of the next one). These events are not the same (due to the different component-level input/output mapping) but they are created from the same global event in which the process system input was actuated. Such events which relate to neighboring components are called *boundary events* and inputs manipulated in them (like the valve in the previous example) are the *boundary inputs*. This effect causes trace fragments to overlap each other in the set of inputs and in time as well.

Using the component graph in Fig. 5.2 the trace in Table 5.1 can be split up to the three trace fragments in Table 5.2. In this case the boundary inputs are "VB" and "VC" (valves between "TA";"TB" and "TA";"TC") and the boundary event is the event at time instant 3 (when boundary inputs "VB" and "VC" are opened).

5.3 Component based diagnostics

Given a system-level trace and a system broken up into separate components, the component-level diagnostics can utilize the original single component diagnostic idea to search for potential failures or root causes on a single component. This is the basis of the general decomposition method. It takes a component path of the system, localized diagnosers for every component and a observed trace as an *input* and provides the list of root causes per component as an *output*.

5.3.1 Additional assumptions

The following assumptions are made in addition to the ones listed in Section 2.6 for this diagnostic method:

- It is possible to structurally decompose the process system into subcomponents. If this cannot be done, then the method is working on a single component only, therefore it has no advantage over applying just the underlying component-level diagnostics on that component.

Trace fragment for tank component "TA"				
sequence	inputs			output
number	IN	OUT-1	OUT-2	LEVEL
1	1	0	0	0
2	1	0	0	L
3	1	1	1	N
Trace fragment for tank component "TB"				
sequence	inputs			output
number	IN	OUT		LEVEL
1	1	0		0
2	1	0		L
3	1	1		N
Trace fragment for tank component "TC"				
sequence	inputs			output
number	IN	OUT		LEVEL
1	1	0		0
2	1	0		L
3	1	1		N

Table 5.2: Trace fragments for the process system in Fig. 5.2.

- The faults happening at the border of two adjacent components in the component tree are not taken into account (like a fault in a pipe which connects two tank components). If this poses a problem, then the structural decomposition can be made finer (in this example case the pipe need to be a component as well).

5.3.2 General algorithm

As it is discussed previously, the decomposition algorithm can be applied to multiple lower level diagnostic approaches, such as the P-HAZID diagnoser (discussed in Chapter 3) and Clustering diagnoser (Chapter 4). The steps of this algorithm are the following:

1. From the component graph, all possible component paths (paths from a starting component to a final component) are formed.

2. Trace fragments (as described in Section 5.2.5) are formed based on different component paths, and distributed among separate components of the system. In this step, all inputs and outputs in the original trace are transformed to inputs and outputs local to the component by using a component mapping function as described in Section 5.2.4.

3. For all the component paths at the last component where the start condition is not fulfilled the path is truncated. As described in Section 5.2.3, the rationale behind this is that components will not get diagnosed when it does not make sense to diagnose them due to a preceding and severe fault in an other component.

4. Looping over the truncated component paths a *component local diagnoser* is executed for every component. A *component local diagnoser* can be any of the following:

   P-HAZID local diagnoser. A local P-HAZID diagnoser, which handles the component as a whole system (as described in Chapter 3). In this case the diagnoser uses a P-HAZID table relevant for that component and trace fragment only, which might be considerably smaller and easier to set up than a P-HAZID table for the overall system.

   Clustering local diagnoser. A local Clustering diagnoser, which, like in the case of the P-HAZID local diagnoser, handles the component as a single whole process system and uses the same diagnostic idea which was already described in Chapter 4. In this case this diagnoser utilizes a list of centroids which is valid for that type of the component only. The number of dimensions for the centroids in this case are equal to the number of output(s) of the component plus the length of the trace fragment. Like in the case of the P-HAZID, this might be considerably smaller than taking the outputs of the whole process system and the length of the whole trace into account. Moreover, if the list of centroids are distributed among local diagnosers diagnosing components of the same type, only a single training and validation phase is needed for them.

   Any other event sequence-based algorithm for fault diagnostics. An arbitrary event-based diagnostics algorithm can be used as a component-local diagnoser. Like the other two approaches, this diagnoser can share the same component-local fault model between same component type(s). We are not dealing with this case in this work.

5. After the diagnosers are finished with the diagnostics, root causes are collected and returned.

Note that this diagnostic algorithm does not have a separate method-specific *fault model* (as described in Section 2.6). The main advantage of the

method is the ability to decompose the fault model, therefore the union of the component fault models for the underlying methods (such as the P-HAZID table or the list of centroids) can be thought as an overall fault model for the algorithm. If there are multiple components of the same type then this union of local fault models might contain less redundancy than the fault model for the overall system.

This concludes the description of the higher-level decomposition procedure. In the next section the decomposition and the diagnostics based thereon is demonstrated on a common case study focusing on the differences between component-local and standalone versions of the P-HAZID and Clustering diagnoser.

5.4 A diagnostics case study

In the last section of the chapter the general decomposition algorithm is applied to the common case study described in Section 2.7. At first, the component graph for the case study is discussed, then the two different component local diagnosers (a P-HAZID and a Clustering diagnoser) are described in detail. The differences between the standalone and decomposed diagnosers are listed (the case study is the same as in Chapters 3 and 4). Finally in the section, general observations regarding the decomposition approach are collected.

5.4.1 Determining the component graph

The component graph for the case study can be seen in Fig. 5.2. There are three components in the graph, namely the main tank "TA", and the two auxiliary tanks "TB" and "TC". Based on the nominal trace a local diagnoser with exactly the same fault model can be used for "TB" and "TC". Theoretically, because it has multiple output valves, a slightly different version should be used for "TA", but in this special case - due to reasons listed later at the component-level diagnosers - the same fault model which is in use for "TB" and "TC" can be used for "TA" as well.

A starting condition is used for tanks "TB" and "TC" that is based on the fluid level in "TA". If the fluid level in "TA" does not reach nominal level, "TB" and "TC" will not be diagnosed.

5.4.2 Nominal and faulty traces for components

The nominal trace, which fills up the system with fluid can be seen in Table 5.1. During the identification of the faulty traces the single and dual faults described in Section 2.7.2 were considered. For these faults, the corresponding faulty trace fragments are shown in Table 5.3 along with the nominal trace (referring to a single component only). The local diagnostic algorithms for the auxiliary tanks ("TB" and "TC") are based on these traces, while the diagnoser for the main tank ("TA") has an additional second output valve which behaves exactly the same as the first output (according to the nominal trace both output valves are opened at the same time).

5.4.3 Component based P-HAZID diagnoser

As a local diagnoser, a P-HAZID diagnoser (see Chapter 3) can be used for diagnostics. For a single component, the P-HAZID diagnostic graph in Fig. 5.2 is utilized for performing the diagnostics. This graph is created from the deviations generated using the traces in Table 5.3 using the general method for P-HAZID table construction in Section 3.2.3. The reasoning paths on the graph are simpler than the reasoning paths in Figures 3.2, 3.3 and 3.4 (created for the original overall P-HAZID case). Moreover, there is no need to use three different diagnostic graphs for the three tanks anymore, this simplified one is sufficient for performing the diagnostics. Therefore the component-local P-HAZID table is significantly smaller than Table 5.4. It has only 10 rows compared to the 52 row table in Table 3.4.

Otherwise, the P-HAZID based diagnostics is identical the one described in Chapter 3, despite the fact that it handles all three components as three isolated systems, and the results from this local diagnostics is collected by the decomposition algorithm.

Note that even though the trace fragments are not exactly the same for all the tanks ("TA" has an additional input compared to the other two tanks), the same P-HAZID table can be used for all the tanks. This is because the nominal trace fragments are of equal length and describe the same phenomenon (which fills up the tank).

5.4.4 Component based Clustering diagnoser

Besides the P-HAZID, a Clustering diagnoser is feasible as a local diagnoser. After training from the trace fragments, the centroids in Fig. 5.3 can used in a local diagnoser (based on the training data their position might slightly differ). In contrast with the non-decomposed clustering case study described

Nominal trace			
sequence	inputs		output
number	IN	OUT	LEVEL
1	1	0	0
2	1	0	L
3	1	1	N
Tank Leak trace			
sequence	inputs		output
number	IN	OUT	LEVEL
1	1	0	0
2	1	0	0
3	1	1	0
Positive Bias trace			
sequence	inputs		output
number	IN	OUT	LEVEL
1	1	0	L
2	1	0	N
3	1	1	H
Negative Bias trace			
sequence	inputs		output
number	IN	OUT	LEVEL
1	1	0	e-
2	1	0	0
3	1	1	L
Positive Bias with Tank Leak trace			
sequence	inputs		output
number	IN	OUT	LEVEL
1	1	0	L
2	1	0	L
3	1	1	L
Negative Bias with Tank Leak trace			
sequence	inputs		output
number	IN	OUT	LEVEL
1	1	0	e-
2	1	0	e-
3	1	1	e-

Table 5.3: Nominal and faulty traces (with single and dual faults) used in the case study.

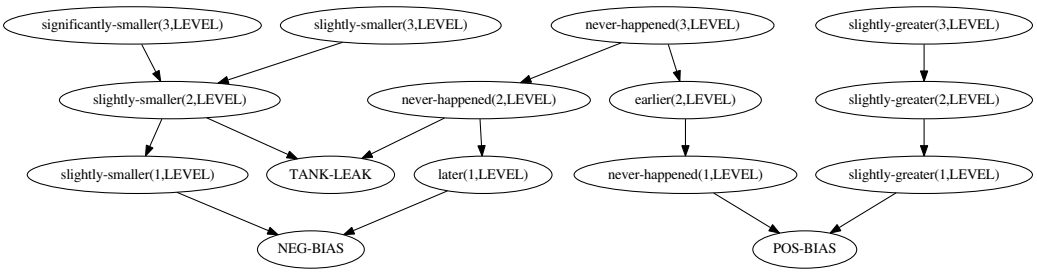

Figure 5.2: Diagnostic graph for the component specific P-HAZID table in Table 5.4.

Cause	*Deviation*	*Implication*
slightly-greater(1) POS-BIAS	slightly-greater (2) slightly-greater(1)	slightly-greater(3) slightly-greater (2)
never-happened(1) POS-BIAS	earlier(2) never-happened(1)	never-happened(3) earlier(2)
slightly-smaller(1) NEG-BIAS	slightly-smaller(2) slightly-smaller(1)	slightly-smaller(3) slightly-smaller(2)
never-happened(1) NEG-BIAS	never-happened(2) never-happened(1)	never-happened(3) never-happened(2)
TANK-LEAK TANK-LEAK	slightly-smaller(2) never-happened(2)	significantly-smaller(3) never-happened(3)

Table 5.4: Component specific P-HAZID table. Relevant for a single tank (and trace fragment) only.

in Section 4.4.2, there are only 6 centroids in this case (covering the nominal, single and dual fault modes of a *single* component) instead of the 42 centroids in the non-decomposed case (covering nominal and faulty modes for all three tanks at the same time). The centroids in this component-level case have 3 dimensions only (because only one tank is considered, and the nominal trace length is three), in contrast with the 9 dimensions in the non-decomposed case (because all tank levels are considered there with the trace length of three).

Apart from this, as in the case of the decomposed P-HAZID diagnoser, the Clustering diagnostic algorithm operates in an identical manner like the one described in Chapter 4, the component is diagnosed independently of the other components.

In this case, inputs are not used, so the same centroids can be used for all three tanks (all of them has a single output).

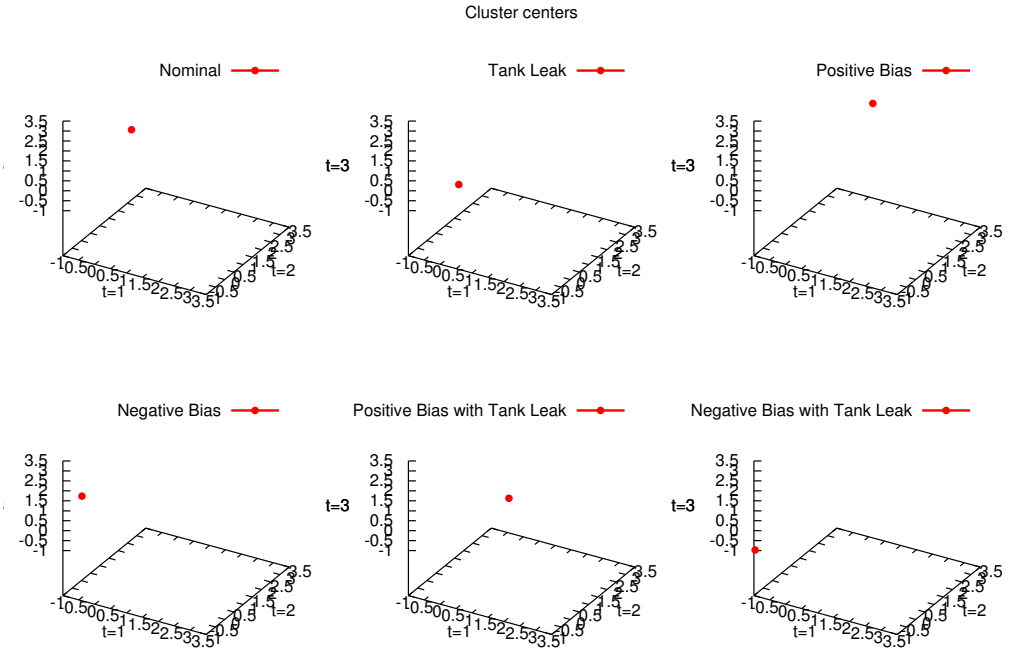

Figure 5.3: Example centroids for the decomposed Clustering diagnoser. The three axes represent the three time instants (1,2,3) in the nominal and faulty component traces from Table 5.3.

5.4.5 Observations

Based on the case study the diagnosers using the decomposition algorithm have the following characteristics:

Decreased complexity. If a process system with many components (therefore with many inputs and outputs) need to be diagnosed, and decomposition is not applied, the complexity of the diagnostics will be higher compared to the case when decomposition is applied. For example, as it had been shown

in this case study, the size of the P-HAZID table in case of the P-HAZID diagnoser and the number of dimensions of the centroids in the case of the Clustering diagnoser were smaller. Using this decomposition strategy, the complexity of a diagnostic solution can be decreased, provided that isolated components can be identified in the process system.

Sharing the fault model. If there are components in the process system, a localized fault model can be created for each component, and then this can be shared among the component-local diagnosers for the similar components. This makes the model smaller, easier to create and more efficient to use. For example, in the case study described in Section 2.7, the fault model about the tanks can be shared, the same P-HAZID table or set of centroids can be used during diagnostics.

Dealing with propagating faults. A properly created component graph and start conditions for the components can ensure that the root cause for a propagating failure (such as a TA *leak* fault for the process system described in Fig. 5.2, which leads to no fluid present in TB and TC) can be found. On the other hand, this causes that some types of multiple faults (such as TA *leak and* TB *positive bias*, see Fig. 4.1) in the case of the Clustering diagnoser cannot be diagnosed. This side-effect can be considered as a trade-off for the reduced fault model required for performing the diagnosis.

Mixing of different diagnostic approaches. The decomposed diagnostics approach described in this chapter can be considered as a higher-level diagnostic extension of the already described P-HAZID and Clustering algorithms. The two methods can be even mixed, in that way always an appropriate lower-level diagnostic method can be used (for example, if a P-HAZID table cannot be easily constructed, then it can be substituted with a Clustering diagnoser). The decomposition approach is independent of the lower-level algorithms used, so theoretically it is possible to use a third diagnoser algorithm for the components.

Lack of central coordination. Central coordination is only needed to decide which diagnosers shall be started (this depends on the *start condition* of the individual components), and to collect the results form the different diagnosers. Otherwise the diagnosers do not depend on and communicate with each other.

Limits of applicability. If the process system cannot be split up into separate components (because it contains only one component or the relations between its components prevent isolated diagnostics on them), then a standalone approach shall be used instead.

5.5 Summary

In this chapter a higher-level decomposition approach for process system fault diagnostics was described. Using this method, the diagnosable process system can be decomposed into separate components and these components can be diagnosed in isolation using suitable event-based diagnostic approaches, such as the P-HAZID approach described in Chapter 3 or the Clustering approach described in Chapter 4.

Using a simple component graph model behind the process system and the diagnosable trace, this method can deal with finding the root cause of propagating failures between separate components, and perform a more efficient diagnostics by not diagnosing components which cannot be reasonably diagnosed (due to severe faults in other components). However this requires that this severe condition can be described using a simple condition over the observable output(s) of the component(s).

The approach described in this chapter can be considered as an extension for other diagnostic approaches, it allows these lower-level diagnostic methods to scale better for more complex process systems. Applying the decomposition idea to larger process systems it can make them diagnosable with reduced fault model (such as smaller P-HAZID tables in case of the P-HAZID diagnoser or clusters in fewer dimensions in the case of the Clustering diagnoser). The method itself is independent of the underlying component-level diagnostic approach, in theory other event-based local diagnosers can be used or they can be of mixed types.

Chapter 6

Conclusion

In this work a few novel ways for performing process system diagnostics were proposed. These approaches act as valid options for fault diagnostics besides the many already available approaches in the field, by attempting to perform discrete time-dependent heuristics during operation. This had not yet been widely investigated in the relevant literature.

In this final chapter the novel scientific results of the work are summarized as theses, and the relevant scientific publications are collected. Finally, some possible further research directions are identified, which might serve as a guideline for continuing research in this direction.

6.1 Theses

During the presentation of the diagnostic approaches in this work, the Multiple-Input Multiple-Output (MIMO) model for process systems were used, where a process system processes (inputs) and produces (outputs) vector-valued signals (see [27] for further details). A single externally observable state, which is a sequence of observable input (processed) and output (produced) values of the system was described by an event. The dynamism of a system was captured in the form of *traces*, which describe a sequence of these events of the system during the execution of an operational procedure.

6.1.1 Thesis 1 - the P-HAZID diagnoser (Chapter 3)

Relevant publications: [THWS12],[THWS13a],[TWSH14]

I have developed a novel diagnostic reasoning approach which is based on differences between observed and fault-annotated traces of a complex process

system.

- A method-specific P-HAZID table as an extension to the BL-HAZID table in non-transient fault scenarios was defined that is constructed by the domain experts and serves as the fault model of the method.
- A method was proposed to form reasoning paths leading to specific root causes of faults in the system. A reasoning graph is built from these paths in order to perform the diagnostic reasoning.
- The proposed reasoning procedure uses the deviations between the observed and the nominal fault-free traces together with the reasoning graph during the execution of a trace.
- I have shown that this method can be tracked back to reasoning with if-then rules.

6.1.2 Thesis 2 - the Clustering diagnoser (Chapter 4)

Relevant publication: [TH16]

I have defined an observation-based diagnostic approach which can detect nominal and non-transient faulty states based on externally monitored traces in a complex process system.

- Historical traces from nominal and faulty scenarios are converted to a vectorial representation in a multi-dimensional coordinate space.
- The fault model is trained and validated from this representation and is a list of centroids (cluster centers). Training is done using a machine learning technique called k-means clustering.
- Fault diagnosis is performed by transforming a measured trace to the same vectorial representation and determining the nearest centroid using a properly selected distance function.
- The diagnostic method can tolerate the presence of measurement errors in the training set.

6.1.3 Thesis 3 - decomposition method for event-based diagnostics (Chapter 5)

Relevant publication: [TWSH14]

I have proposed a higher-level diagnostic approach which can decompose the task of fault diagnostic to possibly overcome the NP-hardness of the general fault diagnostics problem using a structural decomposition of the process system.

- I have developed a structural decomposition method, which works based on the process system and operational procedure.
- The method can connect individually operating lower-level component-specific diagnosers with each other and utilize their findings to solve the overall diagnostics task. It is possible to use component diagnosers of different type for diagnosing separate components.
- In order to reduce its complexity and size the diagnosers can share common parts of the fault model between each other.

6.2 Own publications supporting the thesis points

The thesis points were presented in my following relevant publications:

[TH16] A. Tóth and K. M. Hangos. A diagnostic method based on clustering qualitative event sequences. *Computers and Chemical Engineering.* ***IF=2.581***, 95:58–70, 2016.

[THWS12] A. Tóth, K. M. Hangos, and Á. Werner-Stark. Hazid information based operational procedure diagnosis method. In *12th International PhD Workshop on Systems and Control*, pages 1–6, Veszprém, Hungary, August 2012.

[THWS13a] A. Tóth, K. M. Hangos, and Á. Werner-Stark. A model based diagnosis method for discrete dynamic processes using event sequences. In *Factory Automation 2013 Conference*, pages 114–119, Veszprém, Hungary, May 2013.

[TWSH14] A. Tóth, Á. Werner-Stark, and K. M. Hangos. A structural decomposition-based diagnosis method for dynamic process systems using hazid information. *Journal of Loss Prevention in the Process Industries.* ***IF=1.409***, 31:97–104, 2014.

6.3 Further research directions

The diagnostic approaches described in this work can be extended with the following additional capabilities in the future:

6.3.1 On-line diagnostics

So far in this work every approach was dealing with historical off-line events and traces for which the execution was already completed. However, it is possible to extend these methods by performing real-time diagnostics based on partial information from the measured traces, which are still under execution on the system.

In the case of the P-HAZID diagnoser. The P-HAZID reasoning algorithm can be extended to work with a limited (already available) set of deviations and explore all possible root causes based on the P-HAZID table. This would result on a greater set of root causes, but over time, as new events (and possibly new deviations) would come in, the result would converge to the result of the off-line diagnostics. The rate of the convergence depends on the current faults of the system and the faults defined in the P-HAZID table.

In the case of the Clustering diagnoser. The Clustering algorithm can be extended to work with a limited number of event coordinates, and calculate the distance from the centroids based only on these coordinates (trimming the remaining coordinates, which are not yet available in the measured trace from the centroids). Over time, this approach would converge to the result of the off-line diagnostics.

Note that the algorithm initially would be less accurate in the on-line cases compared to the off-line case (due to the lack of data), but the results (diagnosed faults) would appear much more earlier compared to the off-line case. During the diagnosis of long operational procedures this approach would be very useful, because that it can highlight problems even before the run of the operational procedure on the process system is finished.

6.3.2 Improving robustness during clustering

Due to the fact that currently it is based on the K-means clustering method, the Clustering diagnoser might not be robust enough to handle possible outlier elements. These elements might come from the properties of the used mapping or the actual characteristics of the underlying process system.

Therefore the robustness of the Clustering approach can be improved by:

- Change the clustering method to the *K-medoid* clustering (see [53], which tolerates outlier elements in a better fashion. In this case the centroid (the prototype of the cluster) is also a real trace from the training set, which can be useful for manual interpretation of the centroids.

- Instead of performing *K-means clustering* with $K = 1$ to determine the centroids and a distance calculation, the *K-nearest-neighbors* (*K-NN*) classification technique can be used with an appropriately selected $K \in \mathbb{Z}$ for diagnosing the trace. Note that due to the size of the data diagnosis might take longer compared to the original approach (which performed the clustering on the failures one-by-one and calculating the distance from a single point). On the other hand, this clustering approach not only tolerates outlier elements, but can tolerate clusters with special shapes in a more robust fashion than the approaches listed before. Based on its faults this might be desired by the concrete application.

6.3.3 Validation of P-HAZID tables

A modified version of the P-HAZID algorithm can be used not only for actual diagnostics, but for the validation of P-HAZID tables during construction time. The algorithm would not work on real process system data in this case, but it would explore all possible reasoning paths of the given (possibly incomplete) P-HAZID table. This would provide valuable information about the reasoning graphs in the P-HAZID table, and would be helpful in discovering human mistakes during the construction of these tables. This can be very important during the creation of bigger P-HAZID tables.

6.3.4 Further application in real process systems

There have been already a realistic application described for the Clustering diagnoser in the Appendix (based on the Tennessee-Eastman problem). More realistic or real case studies would be needed to discover the real strengths

and the possible limitations of the approaches described in this work. Taking these findings into account, the methods can be further improved and optimized.

6.3.5 Unifying the P-HAZID and the Clustering diagnoser

Predicate abstraction is a formal logical analysis technique. Originally described in [22], this method can partition modes of operation of a system into abstract states, using the boolean valued predicates describing these modes as an input. The technique is used widely, eg. in the field of software verification (see [9]).

In that way this approach has similarities with the P-HAZID diagnoser (which can be also described as a set of rules containing conditions or predicates as described in Section 3.3.3). On the other hand, it is like the Clustering diagnoser (which also partitions the nominal and faulty modes of operation into different centroids or abstract states during training). Therefore it can be said that this technique can unify the theories behind the two different diagnosers. This connection can be further investigated in order to establish a consistent relationship between the two methods.

Chapter 7

Appendix

7.1 The P-HAZID reasoning method

In this part of the Appendix, a more practical description of the P-HAZID reasoning described in Chapter 3 is presented. This might help readers who want to implement the reasoning procedure on their own.

The method attempts to find the root causes, if there are any, based on the following input:

- **P-HAZID** table (as a spreadsheet) The table is assumed not to contain duplicate rows containing the same **Cause**, **Deviation** and **Implication** triplet.
- Nominal trace (**nominalTrace**) as a timed event sequence as described in Section 2.5. It is assumed that the number of events in the nominal trace is not less than 2.
- Characteristic trace (**chrTrace**) as a timed event sequence as described in Section 2.5.

The output is a set of identified root causes (**IRC**) and identified non-root causes (**INC**) for the analysis trace. The procedure attempts to identify the root causes using a search of Algorithm 1. If such root causes are found, they are appended to the set of **IRC**. If the method cannot perform the search onwards from a specific non-root cause, then this cause is added to the set of **INC**. This can happen either because of the **P-HAZID** table is incomplete or the cause to continue the search with is not present in the set of deviations. Both output sets might be empty, when there are no deviations (a nominal trace is provided as an input to the algorithm).

The notations which are used in the description of the reasoning procedure are defined as follows:

Precedence. Let the notation $(event_x) \prec (event_y)$ mean that $event_x$ precedes $event_y$ in a trace in time.

All root causes. Let the set **RC** contain all possible root causes for the provided **P-HAZID** table. The root causes can be acquired from the provided **P-HAZID** table by collecting all elements from the **Cause** column which are not deviations.

Cell reference in the HAZID table. Let the value $cause(R)$, $dev(R)$ and $imp(R)$ refer to the deviation or root cause of the corresponding column in the **P-HAZID** table at row number R.

Deviation containment. Let **DEV** be a set of $(time) \times (deviation)$. In this sense let the notation $cause(R) \in \mathbf{DEV}$ mean that *cause* is an element of **DEV** for an arbitrary *time*.

Accessing events. Let $eventSequence(time)$ refer to the *event* happened at time *time* in *eventSequence*.

Sequence length. Let $length(eventSequence)$ refer to the number of events in the sequence.

Projection. If $pair = (p, q)$ is an ordered pair then let $proj_1(pair) = p$ and $proj_2(pair) = q$.

Set containment. If $\mathbf{a} \in \mathbf{SET}$ then the operation $\mathbf{SET} \leftarrow \mathbf{SET} \bigcup \mathbf{a}$ has no effect, every element may present only once in a set.

The method is described in Algorithm 1 in detail. First, the deviations need to be collected. Then, if the system under diagnosis is a multiple output system, deviations need to be flattened, ie. from all multiple output deviations single-output deviations need to be generated (where the type of the deviation is identical, but there is exactly one output present per deviation). If the system is a single output system, then this step is skipped. After this step all flattened deviations along with their time instances are collected and stored in **DEV**. Devations are formed when an event in the observed trace is deviating from its nominal correspondent, according to the ordering relations listed in Section 3.2.2. Sets **INC** and **IRC** are given empty initial values. The set of final deviation pairs (**FDP**) is also calculated, this set contains all deviations of **chrTrace** from **nominalTrace** at the last and it's preceding time instant where deviations are present as ordered pairs. The

Algorithm 1 Pseudo-code for the P-HAZID reasoning method

1: **DEV** $\leftarrow \{\emptyset\}$
2: $\mathbf{t}^* = length(nominalTrace)$
3: **for** $T := 1$ to $\mathbf{t}^*$ **do**
4: **for all** deviation D of $chrTrace$ from $nominalTrace$ at time T **do**
5: **for all** single-output-deviation SOD from FLATTEN((T,D)) **do**
6: **DEV** $\leftarrow$ **DEV** $\bigcup SOD$
7: **end for**
8: **end for**
9: **end for**
10: **INC** $\leftarrow \{\emptyset\}$
11: **IRC** $\leftarrow \{\emptyset\}$
12: $\mathbf{t_{FDP}}=\{\exists \mathbf{t}, \exists \mathbf{d_1}, \exists \mathbf{d_2}, (\mathbf{t}, \mathbf{d_1}) \in \mathbf{DEV}, (\mathbf{t}-\mathbf{1}, \mathbf{d_2}) \in \mathbf{DEV}\}$
13: $\mathbf{FDP} \leftarrow \{(\forall \mathbf{d_1}, \forall \mathbf{d_2} | (\mathbf{t_{FDP}}-\mathbf{1}, \mathbf{d1}) \in \mathbf{DEV}, (\mathbf{t_{FDP}}, \mathbf{d2}) \in \mathbf{DEV}, (\mathbf{t_{FDP}}-\mathbf{1}, \mathbf{d_1}) \times (\mathbf{t_{FDP}}, \mathbf{d_2})\}$
14: **for all** $pair \in$ **FDP do**
15: $startDeviation \leftarrow proj_1(pair)$
16: $startImplication \leftarrow proj_2(pair)$
17: STEP(startDeviation, startImplication)
18: **end for**
19: **procedure** STEP(deviation,implication)
20: **if** $\exists R, deviation = dev(R), implication = imp(R)$ **then**
21: **for all** $\{R, dev(R) = deviation$ **and** $imp(R) = implication\}$ **do**
22: **if** $cause(R) \in$ **RC then**
23: **IRC** $\leftarrow$ **IRC** $\bigcup cause(R)$
24: **return**
25: **else**
26: **if** $cause(R) \in$ **DEV and** $cause(R) \prec dev(R)$ *in* **DEV then**
27: STEP(cause(R), dev(R))
28: **else**
29: **INC** $\leftarrow$ **INC** $\bigcup cause(R)$
30: **return**
31: **end if**
32: **end if**
33: **end for**
34: **else**
35: **INC** $\leftarrow$ **INC** $\bigcup cause(R)$
36: **return**
37: **end if**
38: **end procedure**

end of the characteristic trace is not necessarily deviating from the nominal trace, therefore the deviation pair from which the reasoning can be initiated might not be situated at the final and preceding time instant of the trace. After the initialization, a recursive reasoning procedure is initiated on the **P-HAZID** table using the **FDP** set as a starting point.

The core of the recursion checks the *deviation* and *implication* rows in the **P-HAZID** table and according to the corresponding *cause* value it

1. either, if *cause* is a root cause then adds it to the list of root causes **IRC** and returns

2. or, if *cause* is a non-root cause but it cannot continue the search it adds *cause* to the list of non-root causes **INC** and returns

3. or, if *cause* is a non-root cause and it can continue the search then it moves to a corresponding next row in the **P-HAZID** table by calling itself recursively.

The reasoning stops with the reasoning because of either the corresponding cause (with which the search need to be continued):

1. is not present in the set of deviations prior to the actual deviation,

2. or, with its corresponding implication does not exist in the **P-HAZID** table for any row as implication and deviation. It is possible that the **P-HAZID** table is not complete in this case.

In either cases the algorithm adds the cause to the set of **INC** and moves on with the next possible final deviation pair, if available.

After the recursion finished, set **IRC** contains the set of identified root causes, while the set **INC** contain the set of deviations which might lead to a failure but either do not occur in the list of deviations in advance of the actual deviation or the search could not find any row in the **P-HAZID** table with which it could move forward.

7.2 Realistic use-case for the Clustering diagnoser

In this part of the Appendix, a more realistic process system example, the Tennessee-Eastman Problem is used to demonstrate the diagnostic capabilities of the Clustering diagnoser presented in Chapter 4. Like in the case of the previous three-tank case study, for the various disturbances (faults) of

the problem the fault detection ratio (FDR) is calculated (as described in Section 4.3.2). A similar survey for many different statistical methods had been performed in [54] on the same process system and disturbances for other statistical diagnostic methods (such as PCA or PLS).

7.2.1 Tennessee-Eastman process

The Tennessee Eastman process (later mentioned as TEP) is widely used and accepted for developing, studying and comparing process control and diagnostics algorithms. It consists of a reactor/separator/recycle arrangement involving two simultaneous gas-liquid exothermic reactions and two additional byproduct reactions. The process has 12 available valves for manipulation and 41 available output measurements for monitoring or control. In the case study the first 15 of the original 20 simulated disturbances are considered (see Table 7.1 for details), due to the fact that the last 5 disturbances are of type "Unknown", and we wanted to emphasize diagnosing the known faults in the case study. Note that these "not known" disturbances were part of the actual diagnosis for the rest of the disturbances, but their FDR values have not been calculated, and observations have not been made for them.

The TEP produces two products, an inert and a byproduct from four reactants (there are eight components altogether, A, B, C, D, E, F, G and H). The following reactions take place based on the components in this example process system (based on [18]):

$$A(g) + C(g) + D(g) \rightarrow G(liq) \quad (Product\ 1)$$

$$A(g) + C(g) + E(g) \rightarrow H(liq) \quad (Product\ 2)$$

$$A(g) + E(g) \rightarrow F(liq) \quad (Byproduct)$$

$$3D(g) \rightarrow 2F(liq) \quad (Byproduct)$$

These components are also shown on the flow-sheet of the process system in Fig. 7.1.

For more details, refer to [18]. In order to be more consistent with the original article, the term *disturbance* will be used for faults in the description of this case study.

The original model was written in FORTRAN, but in this case study the revised MATLAB version of the TEP (described in [11]) was used for generating the training traces for the algorithm.

The MATLAB model has already contained simulated measurement errors, so the measurement error generation approach described in Section 4.4.1

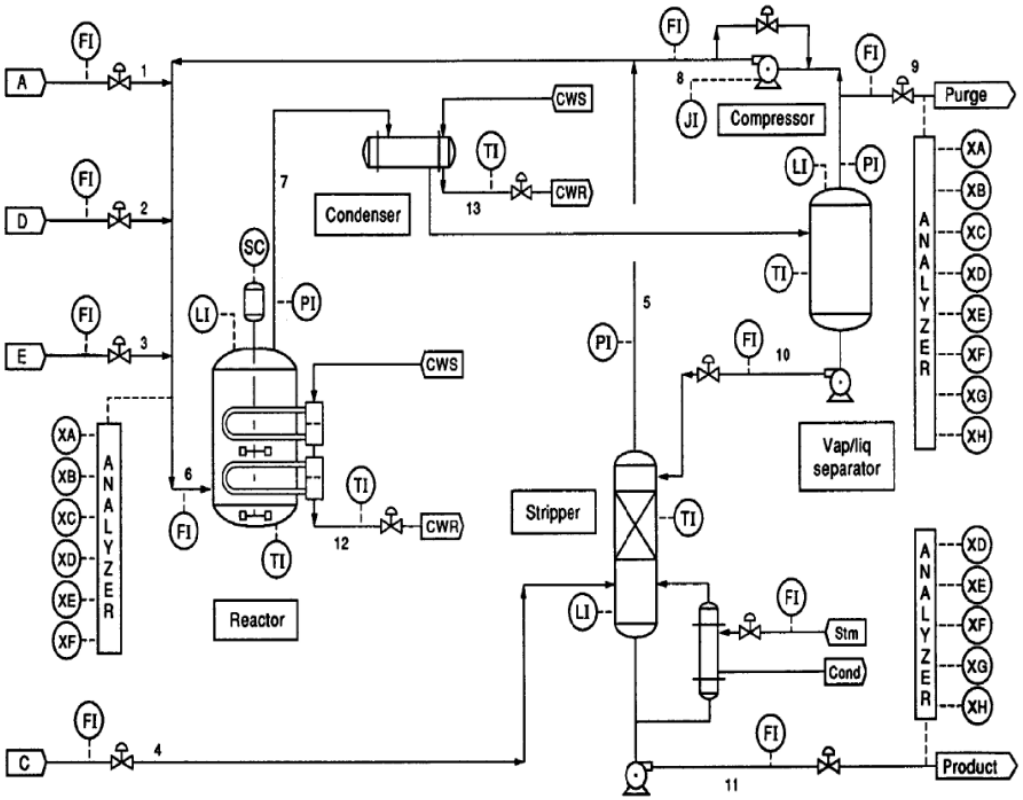

Figure 7.1: Tennessee Eastman Challenge Problem from [18].

was not used in this case, the raw values were just taken from the simulated model without change. Due to the available functionality of the model, only single disturbances were considered in this case.

7.2.2 Preparation of the data

In this case study the diagnostic algorithm's ability to identify the various disturbances (considered as fault modes from the algorithm's perspective) are surveyed for the TEP. Two operational modes, an "open-loop" mode and a controlled steady-state mode (refer to "Mode 1" in [18]) were considered. Inputs were modified by the simulated controller in steady state mode but were not taken into account. Also, only a subset of the original 41 outputs (22 "Continuous process measurements", see Table 4 in [18]) were taken into account in the case study.

The reason for this is that we wanted to focus on the continuous measurements only during diagnosis (the rest of the outputs were relatively infrequently sampled process measurements measuring concentrations in the

Identifier	Disturbance	Type
IDV(1)	A/C feed ratio, B composition constant (stream 4)	Step
IDV(2)	B composition. A/C ratio constant (stream 4)	Step
IDV(3)	D feed temperature (stream i)	Step
IDV(4)	Reactor cooling water inlet temperature	Step
IDV(5)	Condenser cooling water inlet temperature	Step
IDV(6)	A feed loss (stream I)	Step
IDV(7)	C header pressure loss - reduced availability (stream 4)	Step
IDV(8)	A, B, C feed composition (stream 4)	Random variation
IDV(9)	D feed temperature (stream 2)	Random variation
IDV(10)	C feed temperature (stream 4)	Random variation
IDV(11)	Reactor cooling water inlet temperature	Random variation
IDV(12)	Condenser cooling water inlet temperature	Random variation
IDV(13)	Reaction kinetics	Slow drift
IDV(14)	Reactor cooling water valve	Sticking
IDV(15)	Condenser cooling water valve	Sticking

Table 7.1: A subset of the disturbances, along with their types of the Tennessee Eastman Challenge problem, as described in the original article [18].

output of the process system, which would not change the findings of the case study). Therefore the output of the simulation was a set of events (each containing 22 output values) ordered by time. A single event was describing a state of the system at a different sequence number for a single execution. This is converted to the trace format required by the algorithm where inputs were not considered (there was no input changes during the execution of traces), while the list of outputs contained every output from the simulation.

After the results had been collected from the MATLAB simulator, the traces were trimmed to equal length (this was one of the assumptions from Section 4.3.1). Moreover, the raw data was sampled at different intervals to determine the effect of sampling on the diagnostic accuracy. The reactor in the "open loop" case always shut down after approximately 1 hour of operation (due to the high pressure threshold built into the MATLAB model), while in the steady-state case ("Mode 1") always a 5-hour model MATLAB simulation was performed (the model could have been executed longer in this case).

7.2.3 Results

The corresponding output values were normalized and converted to trace coordinate form using a qualitative mapping function.

Normalization was performed so that the same qualitative function could be used for all outputs, this made the execution of the diagnosis simpler. In both cases the complete traces were kept and compared with each other.

Finally, the diagnostic algorithm was executed for the traces, centroids were formed from the training traces and the fault (disturbance) detection rates ($FDRs$) were calculated from the validation set for every disturbance case.

Table 7.2 shows a summary about the most important properties of the executed cases. These are the following:

1. **Number of training and validation traces.** The number of times the simulation was executed for every disturbance scenario described in Table 7.1 to get the traces required by the algorithm. The first half of the traces was used for training while the second half is for validation of the trained centroids During the training phase the centroids were calculated one by one for each disturbance. On the other hand, during validation the distance from all the centroids was calculated for each trace in every validation set. Based on this the formula for the FDR value for disturbance scenario d can be seen on Eq. 7.1. For more information, refer to 4.3.2 for details).

$$FDR_d = \frac{Number\ of\ correctly\ identified\ traces}{Number\ of\ all\ traces} \tag{7.1}$$

   We have experimented with other training/validation ratios, such as 4:1 but we have not seen significant differences in the results of the case study.

   The exact number of training and validation traces (simulator runs) was chosen in a way that we have enough traces for each disturbance to compare the diagnostic accuracy between them. A couple of hundred traces per disturbance proved to be more than enough for this purpose.

2. **Trace trim.** Traces are trimmed at this length after conversion. Trimming is needed so that every trace participating in the diagnosis will have the same length (to comply to the assumptions in Section 4.3.1).

   This was required because for some of the disturbances the MATLAB simulation (see [11]) produced fewer number of events due to the fact that internal error thresholds (eg. "Low stripper liquid level" in the case of IDV(6) in Mode 1) were met, which correctly caused the simulation to halt immediately. This resulted in shorter traces for these disturbances. In order to comply with the diagnostic assumptions in

Section 4.3.1 (all traces shall have the same length), longer traces for other disturbances were trimmed accordingly, so that every trace would have the same length.

3. **Sampling rate.** This item describes how the simulation output was sampled. For example, "7" means that every seventh event was taken from the simulation output (the rest was thrown away), "1" means that every event was kept.

   In the "Mode 1" case due to the chosen sampling rate, the traces needed to be trimmed at an earlier event, so that traces for every disturbance have the same length. This effect can also be seen in Table 7.2.

4. **Qualitative sets.** The resolution of qualitative sets used to represent the previously normalized output values. After normalization, every output had the same range, so the same qualitative output mapping function could be used for them. In every case (except for ∞) a linear qualitative mapping function (like the one described in Eq. (4.1)) was used, which divided the interval $[0, 1]$ into n qualitative sets evenly.

   (a) ∞ means that qualitative sets are not used for the output.

   (b) *Refined* means a 8-element refined qualitative set for the output:

   $$Q = \{0, 0L, L, LN, N, NH, H, H+\} \tag{7.2}$$

   (c) *Coarse* means a 3-element qualitative set for the output:

   $$Q = \{0, L, N\} \tag{7.3}$$

The parameters in Table 7.2 were chosen based on preparatory simulation experiments, so they are valid for this case study only, and might be different for other process systems or operational procedures.

After the traces were converted, they were given as input to the diagnostic algorithm. The first half of them were used for training (calculating the centroids using K-Means clustering with $K = 1$), while the remainder was used for validating these clusters and determine the FDR values for every disturbance in every case study. These values are collected in Table 7.3. The original article also contained disturbances from type "Unknown". Due to the not known nature of these disturbances they are not displayed in the Table 7.3 for the case studies - they were only taken into account during calculating of the FDR values for the other, known disturbances.

Case study	Qualitative sets	Number of training and validation traces	Trace trim	Sampling rate
(18) Mode 1 #1	∞	366	100	1
Mode 1 #2	Refined	366	100	1
Mode 1 #3	Coarse	366	100	1
Mode 1 #4	∞	366	59	7
Mode 1 #5	Refined	366	59	7
Mode 1 #6	Coarse	366	59	7
(18) Open Loop #1	∞	300	100	1
Open Loop #2	Refined	300	100	1
Open Loop #3	Coarse	300	100	1

Table 7.2: Basic properties of the executed case studies.

Dist. ID	Mode 1 #1	Mode 1 #2	Mode 1 #3	Mode 1 #4	Mode 1 #5	Mode 1 #6	Open Loop#1	Open Loop#2	Open Loop#3
IDV(1)	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	83.333%	84.0%	**100.0%**
IDV(2)	98.37%	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	9.333%	8.0%	19.333%
IDV(3)	27.174%	34.783%	34.239%	21.196%	15.761%	26.630%	7.333%	12.0%	40.0%
IDV(4)	**100.0%**	**100.0%**	**100.0%**	38.043%	73.370%	86.957%	**100.0%**	99.333%	**100.0%**
IDV(5)	**100.0%**	**100.0%**	**100.0%**	58.696%	**100.0%**	95.652%	**100.0%**	**100.0%**	**100.0%**
IDV(6)	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**
IDV(7)	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**
IDV(8)	21.196%	32.065%	14.674%	47.283%	71.739%	55.435%	1.333%	5.333%	5.333%
IDV(9)	14.674%	21.196%	16.848%	16.848%	23.370%	23.370%	6.0%	3.333%	8.0%
IDV(10)	8.696%	28.804%	24.457%	34.783%	30.435%	53.804%	8.0%	4.667%	10.0%
IDV(11)	5.978%	46.196%	29.891%	72.283%	48.370%	88.587%	6.667%	8.667%	21.333%
IDV(12)	9.783%	42.935%	35.87%	22.283%	38.587%	65.217%	14.667%	9.333%	37.333%
IDV(13)	2.717%	5.435%	45.109%	18.478%	20.652%	19.022%	4.667%	8.0%	10.0%
IDV(14)	92.391%	94.565%	97.826%	88.043%	67.935%	97.826%	5.333%	6.667%	6.0%
IDV(15)	13.587%	23.913%	18.478%	17.935%	16.848%	17.391%	6.0%	4.667%	8.0%

Table 7.3: FDR values in % for the different case studies. The "Dist. ID" is the identifier from Table 7.1.

7.2.4 Observations based on the results

Based on these results the following observations can be made:

1. It can be seen in Table 7.3, that many of the *Step* type disturbances, such as IDV(1), IDV(6) or IDV(7), of the TEP were detected with 100% FDR. This means that the diagnostic algorithm could isolate a separate centroid – farther from the rest – for these disturbance types successfully. A reason for this in the case of IDV(7) can be seen in Fig. 7.2 which shows a TEP output ("A and C feed (stream 4)") for all the considered disturbance scenarios and cases, with IDV(7) as bold black line while the rest of the disturbances with ordinary red lines.

   This figure shows a significant contrast between IDV(7) and all the rest of the disturbances in the value of the TEP output for all the executed case studies described in Table 7.2. The dissimilarity in the value is so outstanding in this case that not even:

   - a significant change in the Sampling Rate (from "1" to "7", where $\frac{6}{7} \approx 85\%$ of the traces were dropped, see diagrams in the first row in Fig. 7.2),
   - the use of a very coarse 3-element qualitative set

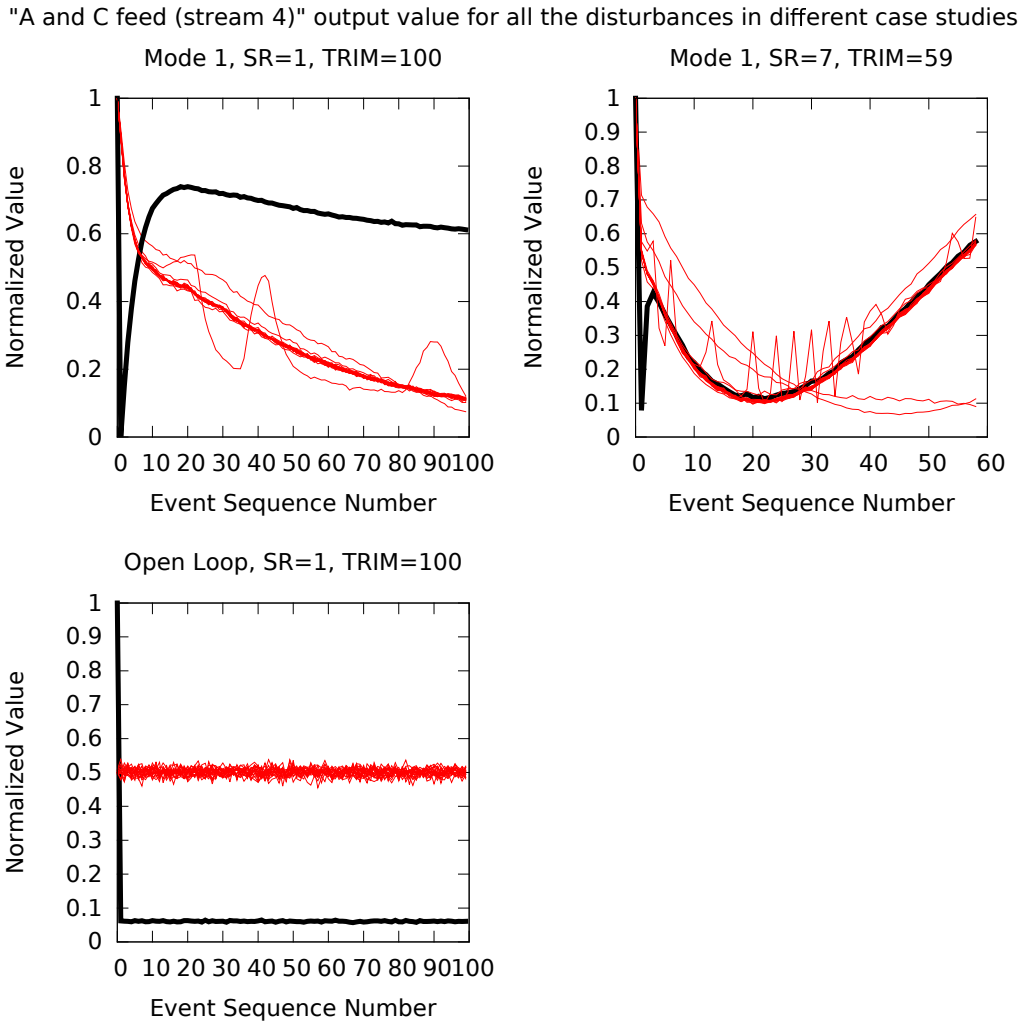

Figure 7.2: Distribution of a single output over all scenarios and case studies. TEP disturbance IDV(7) with bold black line, while the rest of the disturbances are shown with ordinary red lines. SR= Sampling Rate, TRIM=Trace trim (according to Table 7.2). Normalization was performed by transforming the values of "A and C feed" to the interval of [0,1] for every disturbance per scenario right after the simulation. This operation was required to be able to use the same qualitative mapping function for all the outputs during diagnosis.

could not affect the diagnostic capability of the algorithm for this disturbance significantly.

Differences like this placed the centroid for IDV(7) farther than the

centroids for the other disturbance scenarios, and this is responsible for the 100% accuracy in this case. Similar differences to this in output values are responsible for the 100% accuracy in some of the other cases when *Step* type disturbances were diagnosed. *Step* type disturbances were assumed among the diagnostic assumptions in Section 2.6. (Faults (disturbances) are permanent and their number is fixed a priori.) Note that in most other cases increasing the roughness of the qualitative sets might result in highly inaccurate diagnostics. However in this case the output change caused by the fault was so outstanding that not even a very coarse, 3-element qualitative set could not affect the diagnostic accuracy drastically.

2. *Step* type disturbances could be well diagnosed in case of both the "Open Loop" operational mode and the steady-state ("Mode 1") operational mode of the simulation.

3. In case of the "Mode 1" case studies the use of different Sampling rates ("1" and "7") had no significant effect on the diagnostic result for Step type disturbances IDV(1), IDV(2), IDV(6) and IDV(7). These disturbances could still be diagnosed for both Sampling rates, despite the fact that the diagnosis was based on a fraction of the available events only (every seventh event). Due to the loss of data during sampling, the FDR for some other *Step* type disturbances (IDV(4) and IDV(5)) reduced drastically.

4. The use of the three different qualitative sets (as described in Section 7.2.3) had also no effect on the diagnostic result.

5. Disturbances from other types (such as *Random Variations*) could be detected with a very low FDR. In this case individual centroids are created by the clustering overlapped each other, therefore the disturbance could not have been identified by the algorithm properly using distance calculation.

Bibliography

[1] C. Agudelo, F. M. Anglada, E. Q. Cucarella, and E. G. Moreno. Integration of techniques for early fault detection and diagnosis for improving process safety: Application to a fluid catalytic cracking refinery process. *Journal of Loss Prevention in the Process Industries*, 26:660–665, 2013.

[2] E. Alpaydin. Soft vector quantization and the em algorithm. *Neural Networks*, 11:467–477, 1998.

[3] E. Alpaydin. *Introduction to Machine Learning, Second Edition.* The MIT Press, Cambridge, Massachusets, London, England, 2010.

[4] E. Alpaydin. *Introduction to Machine Learning, Second Edition.* The MIT Press, Cambridge, Massachusets, London, England, 2010.

[5] Standards Australia. *Hazard and operability studies (HAZOP studies) Application Guide.* 2003. AS IEC 61882-2003.

[6] Standards Australia. *Analysis techniques for system reliability - Procedure for failure mode and effects analysis (FMEA).* 2008. AS IEC 60812-2008.

[7] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1:11–33, 2004.

[8] B. Balasko, Z. Banko, and J. Abonyi. Analyzing trends by symbolic episode representation and sequence alignment. In *2007 Mediterranean Conference on Control and Automation (MED 2007)*, pages 1–6, Athens, Greece, June 2007.

[9] T. Ball, R. Majumdar, T. Millstein, and S. K. Rajamani. Automatic predicate abstraction of c programs. In *Proceedings of the ACM SIGPLAN 2001 Conference on Programming Language Design and Implementation*, pages 203–213, New York, NY, USA, 2001.

[10] V. Bartolozzi, L. Castiglione, A. Piciotto, et al. Qualitative models of equipment units and their use in automatic hazop analysis. *Reliability Engineering and Systems Safety*, 70:49–57, 2000.

[11] A. Bathelt, N. L. Ricker, and M. Jelali. Revision of the tennessee eastman process model. In *9th IFAC Symposium on Advanced Control of Chemical Processes ADCHEM 2015*, volume 48, pages 309–314, Whistler, Canada, 2015.

[12] I. T. Cameron and R. Raman. *Process Systems Risk Management*, volume 6 of *Process Systems Engineering*. Elsevier Academic Press, San Diego, CA, 2005.

[13] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, Boston, Dordrecht, London, 1999.

[14] Gensym Corporation. G2 developer's guide, http://www.gensym.com/, 2005.

[15] F. Crawly and B. Tyler. *HAZOP: Guide to best practice.* The Institution of Chemical Engineers, Rugby, United Kingdom, 2000.

[16] F. Crawly and B. Tyler. *Hazard Identification Methods.* The Institution of Chemical Engineers, Rugby, United Kingdom, 2003.

[17] A.K. Alves de Medeiros, A. Guzzo, and G.Greco et al. *Process Mining Based on Clustering: A Quest for Precision*, volume 4928. Springer Berlin Heidelberg, Berlin, 2008.

[18] J. J. Downs and E. F. Vogel. A plant-wide industrial process control problem. *Computers and Chemical Engineering*, 17:245–255, 1993.

[19] A. Bregon et al. An event-based distributed diagnosis framework using structural model decomposition. *Artificial Intelligence*, 210:1–35, 2014.

[20] Charles L. Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19(1):17–37, 1982.

[21] H. A. Gabbar. Improved qualitative fault propagation analysis. *Journal of Loss Prevention in the Process Industries*, 20:260–270, 2007.

[22] S. Graf and H. Saidi. Construction of abstract state graphs with pvs. *Lecture Notes in Computer Science*, 1254:72–83, 1997.

[23] F. Harrou, M. N. Nounou, H. N. Nounou, and M. Madakyaru. Pls-based ewma fault detection strategy for process monitoring. *Journal of Loss Prevention in the Process Industries*, 36:108–119, 2015.

[24] A. Cinar J. MacGregor. Monitoring, fault diagnosis, fault-tolerant control and optimization: Data driven methods. *Computers and Chemical Engineering*, 47:111–120, 2012.

[25] H. G. Kim and C. Kim. Interval clustering algorithm for fast event detection in stream monitoring applications. *Pattern Recognition Letters*, 36:171–176, 2014.

[26] T. A. Kletz. *Hazop and Hazan - Identifying and assessing process industry hazard. 4th edition.* The Institution of Chemical Engineers, Rugby, United Kingdom, 2003.

[27] J. Bokor K.M. Hangos and G. Szederkényi. *Analysis and Control of Nonlinear Process Systems.* London, 2004.

[28] L.A.Zadeh. Fuzzy logic and approximate reasoning. *Synthese*, 30(3-4):407–428, 1975.

[29] H. R. Lewis and C. H. Papadimitriou. *Elements of the Theory of Computation.* Upper Saddle River, New Jersey 07458, 1998.

[30] M. R. Maurya, R. Rengaswamy, and V. Venkatasubramanian. Fault diagnosis by qualitative trend analysis of the principal components. *Chemical Engineering Research and Design*, 83:1122–1132, 2005.

[31] M.R. Maurya, R. Rengaswamy, and V. Venkatasubramanian. A signed directed graph and qualitative trend analysis-based framework for incipient fault diagnosis. *Chemical Engineering Research and Design*, 85:1407–1422, 2007.

[32] D. Mercurio, L. Podofillini, E. Zio, and V.N. Dang. Identification and classification of dynamic event tree scenarios via possibilistic clustering: Application to a steam generator tube rupture event. *Accident Analysis and Prevention*, 41:1180–1191, 2009.

[33] Noemi Moya, Gautam Biswas, Carlos J. Alonso-Gonzalez, and Xenofon Koutsoukos. Structural observability. application to decompose a system with possible conflicts. In *21st International Workshop on the Principles of Diagnosis*, Portland, Oregon, October 2010.

[34] E. Németh and I. T. Cameron. Cause-implication diagrams for process systems: Their generation, utility and importance. *Chemical Engineering Transactions*, 31:193–198, 2013.

[35] E. Németh, R. Lakner, K.M. Hangos, and I.T Cameron. Prediction-based diagnosis and loss prevention using qualitative multi-scale models. *Information Sciences*, 177:1916–1930, 2007.

[36] C. Palmer and P. W. H. Chung. An automated system for batch hazard and operability studies. *Reliability Engineering and System Safety*, 94:1095—-1106, 2009.

[37] M. Petković, M. R. Rapaić, Z. D. Jeličić, and A. Pisano. On-line adaptive clustering for process monitoring and fault detection. *Expert Systems with Applications*, 39:10226–10235, 2012.

[38] S. J. Qin. Survey on data-driven industrial process monitoring and diagnosis. *Annual Reviews on Control*, 36:220–234, 2012.

[39] C. A. Ratanamahatana and Keogh. E. Everything you know about dynamic time warping is wrong. In *Third Workshop on Mining Temporal and Sequential Data (KDD-2004)*, Seattle, WA., August 2004.

[40] A. Rozinat and W.M.P. van der Aalst. Conformance checking of processes based on monitoring real behavior. *Information Systems*, 33:64–95, 2008.

[41] Stuart J. Russel and Peter Norvig. *Artificial Intelligence. A modern approach.* Prentice Hall, Englewood Cliffs, New Jersey 07632, 1995.

[42] B. J. Seligmann, E. Németh, K. Hangos, and Ian T. Cameron. A blended hazard identification methodology to support process diagnosis. *Journal of Loss Prevention in the Process Industries*, 25:746–759, 2012.

[43] J.L. Top and J.M. Akkermans. Qualitative reasoning about physical systems: an artificial intelligence perspective. *Journal of the Franklin Institute*, 328:1047–1065, 1991.

[44] A. Tóth, K. M. Hangos, and Á. Werner-Stark. Hazid information based operational procedure diagnosis method. In *12th International PhD Workshop on Systems and Control*, pages 1–6, Veszprém, Hungary, August 2012.

[45] A. Tóth, K. M. Hangos, and Á. Werner-Stark. A model based diagnosis method for discrete dynamic processes using event sequences. In *Factory Automation 2013 Conference*, pages 114–119, Veszprém, Hungary, May 2013.

[46] W.M.P. van der Aalst, B.F. van Dongen, and C.W. Gunther et al. Prom 4.0: Comprehensive support for real process analysis. 4546:484–494, 2007.

[47] H. Vedam and V. Venkatasubramanian. Signed digraph based multiple fault diagnosis. *Computers and Chemical Engineering*, 21:655–660, 1997.

[48] V. Venkatasubramanian, R. Rengaswamy, and S. N. Kavuri. A review of process fault detection and diagnosis part ii: Qualitative models and search strategies. *Computers and Chemical Engineering*, 27:313–326, 2003.

[49] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri. A review of process fault detection and diagnosis part i: Quantitative model-based methods. *Computers and Chemical Engineering*, 27:293–311, 2003.

[50] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri. A review of process fault detection and diagnosis part iii: Process history based methods. *Computers and Chemical Engineering*, 27:327–346, 2003.

[51] V. Venkatasubramanian, J. S. Zhao, and S. Viswanathan. Intelligent systems for hazop analysis of complex process plants. *Computers and Chemical Engineering*, 24:2291–2302, 2000.

[52] Á. Werner-Stark, Erzsébet Németh, and K. M. Hangos. Knowledge-based diagnosis of process systems using procedure hazid information. In *15th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 385—-394, 2011.

[53] J. Xin and H. Jiawei. *K-Medoids Clustering*. Springer US, Boston, MA, 2010.

[54] S. Yin, S. X. Ding, A. Haghani, H. Hao, and P. Zhang. A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark tennessee eastman process. *Journal of Process Control*, 22:1567–1581, 2012.