# Dynamic Fault Handling and Reconfiguration for Industrial Automation Systems

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik
der Universität Stuttgart zur Erlangung der Würde eines
Doktor-Ingenieurs (Dr.-Ing.) genehmigte Abhandlung

Vorgelegt von
Huiqiang Wang
aus Shandong, China

| | |
|---|---|
| Hauptberichter: | Prof. Dr.-Ing. Dr. h. c. Michael Weyrich |
| Mitberichter: | Prof. Dr.-Ing. Birgit Vogel-Heuser |
| Tag der Einreichung: | 13.06.2018 |
| Tag der mündlichen Prüfung: | 05.10.2018 |

Institut für Automatisierungstechnik und Softwaresysteme
der Universität Stuttgart

2018

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Automatisierungstechnik und Softwaresysteme (IAS) der Universität Stuttgart.

Mein besonderer Dank gilt meinem Doktorvater und Leiter des Instituts, Herrn Prof. Dr.-Ing. Dr. h. c. Michael Weyrich, für die fortwährende Unterstützung während der Entstehung der Arbeit, die zahlreichen wertvollen Anregungen und kritischen Diskussion sowie die Übernahme des Hauptberichts.

Frau Prof. Dr.-Ing. Brigit Vogel-Heuser danke ich für das Interesse an meiner Arbeit und die Übernahme des Mitberichts.

Ein herzlicher Dank gilt Herrn Dr.-Ing. Nasser Jazdi für sein Engagement und die wertvollen Aussprachen zum Forschungsthema und zu Fragen des beruflichen Alltags. Allen ehemaligen Kolleginnen und Kollegen am IAS gilt mein herzlicher Dank für die gute Zusammenarbeit.

Ebenso gilt mein Dank den Studierenden, die im Rahmen ihrer Master-, Studien-, Forschungs- und Bachelorarbeiten einen Beitrag zum Gelingen dieser Arbeit geleistet haben.

Schließlich danke ich meinen Eltern für die langjährige und fortwährende Unterstützung.

Stuttgart, im Oktober 2018

Huiqiang Wang

# Table of Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| ASCII | **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange |
| API | **A**pplication **P**rogramming **I**nterface |
| ATS | **A**utomation **S**ystem |
| BF | **B**asic **F**unction |
| C | **C**omponent |
| CI | **C**ommunication **I**nterface |
| DFS | **D**epth-**f**irst-**s**earch |
| EFDS | **E**xisting **F**ault **D**iagnosis **S**ystem |
| F | **F**unction |
| FDS | **F**ault **D**iagnosis **S**ystem |
| FMEA | **F**ailure Mode and **E**ffects **A**nalysis |
| JSON | **J**avaScript **O**bject **N**otation |
| MF | **M**ain **F**unction |
| NFR | **N**on-**F**unctional **R**equirement |
| R | **R**equirement |
| SR | **S**ub **R**equirement |
| S | **S**ystem |
| SADT | **S**tructured **A**nalysis and **D**esign **T**echnique |
| SF | **S**ub **F**unction |
| SOA | **S**ervice-**o**riented **A**rchitecture |
| SS | **S**ub**s**ystem |
| SysML | **S**ystems **M**odeling **L**anguage |
| TCP | **T**ransmission **C**ontrol **P**rotocol |
| UML | **U**nified **M**odeling **L**anguage |

# Glossary

**Availability:** Ability to be in a state to perform as and when required, under given conditions, assuming that the necessary external resources are provided

**Available functions:** Functions which are not affected by the fault or malfunctions in a system and still available to be performed

**Component:** A basic, relatively independent part or item of an industrial automation system, characterized by specific responsibilities, here meaning sensors, actuators, and a microcontroller in an industrial automation system

**Component model:** A conceptual model illustrates the hierarchy of the physical structure, collaboration and connections of components in a system

**Corrective maintenance:** Maintenance carried out after fault recognition and intended to put an item into a state in which it can perform a required function [DIN EN 13306:2010-12]

**Decision-making:** The process of selecting a reasonable option from possible and available choices after sufficient consideration

**Degradation:** Detrimental change in physical condition, due to time, use, or external cause

**Depth-first search:** Searching algorithm which supports the travel from the vertex to the deepest point in the vertical level before backtracking, then traveling to the other neighbor on the horizontal level [Even11]

**Failure:** Termination of the ability of an item to perform a required function

**Failure cause:** Circumstances during specification, design, manufacture, installation, use or maintenance that result in failure

**Fail-operational:** Ability of a system to continue to work in the event of a fault

**Fault:** State of an item characterized by an inability to perform a required function, excluding the inability during preventive maintenance or other planned actions, or due to a lack of external resources

**Fault Diagnosis:** Identification of the defective component, as well as malfunctions of a system, by means of analysis of the historical data

**Fault handling:** The process of responding to the fault occurring in a system and transmitting the system into a new state before the fault is removed, such as fault tolerance

**Fault location:** The defective item of a system, e.g. an element, a component, a sensor, etc.

**Fault localization:** The process of determination of the fault location in a system within a limited scope

**Formalization:** The information or knowledge formalized in specific forms and types used for the reasoning

**Function:** The special or required purpose or activity of a component or a system

**Function model:** A model represents the hierarchy of the logical structure of a system and describes combination and flow sequences of the functions, actions, or processes [Bitt12]

**Industrial automation system:** The combination of personnel, hardware, and software as a whole that can influence the secure and reliable operation of industrial processes [Siem17]

**Item:** Part, component, device, subsystem, functional unit, equipment or system that can be individually described and considered

**Knowledge:** Facts which represent the world via structured and unstructured information

**Maintenance:** Combination of all technical, administrative and managerial actions during the life cycle of an item intended to retain it in, or restore it to, a state in which it can perform the required function

**Malfunction:** An intermittent irregularity in the fulfillment of a system's desired function

**Redundancy:** A component is duplicated for another as a backup and both can perform the same required function

**Reconfiguration:** The arrangement of parts or elements in a different form, figure, or combination [Oxfo17]

**Required function:** A function, combination of functions, or a total combination of functions of an item which are considered necessary to provide a given service

**Functional requirement:** A description of a system, what the system should do or provide for users, and can be features, services, tasks, and functions supported by the solution

**Non-functional requirement:** A description of the quality attributes that must be fulfilled by the system developed, as well as functions in the system, and could be stipulated by the standard or be specifically required by the customer, such as usefulness, speed, etc.

**Requirement model:** A model represents the hierarchy of constraints of a system and indicates the relationship of the various requirements, such as reliability, safety, etc.

**Symptom:** The effect of a fault, which represents the specific changes of abnormal parameters, such as beyond the threshold

**System:** Total plant or a particular part of a plant, including various parts of components, having specific inputs and outputs

**System model:** An information model that describes a particular concrete system or some type of system description [Muth12]

**Task:** An action or piece of a work a system ought to perform or achieve

# Zusammenfassung

Die Verfügbarkeit und die korrekte Funktion von komplexen technologischen Prozessen spielen in unserer modernen Gesellschaft eine wichtige Rolle. Automatisierte Systeme werden immer komplexer, sie enthalten zahlreiche Prozesse und Teilsysteme. Dies führt zu einer immer größer werdenden Anzahl komplexer Zusammenhänge und möglicher Fehlerquellen, die in ihrer Vielzahl für den Menschen schwer beherrschbar sind. Fehler, die zur Entwicklungszeit nicht berücksichtigt wurden, können zur Laufzeit nicht abgefangen werden und führen dadurch meist zum Ausfall des gesamten Systems, obwohl die von einem Fehler nicht betroffenen Teilfunktionen weiterhin verfügbar und dadurch ein Teilbetrieb möglich wäre.

Aus diesem Grund wird in dieser Arbeit ein modellgestütztes, dynamisches Fehlerbehandlungs- und Rekonfigurationssystem vorgestellt. Dieses System kooperiert dabei mit im automatisierten System integrierten oder externen Fehlerdiagnosesystemen. In diesem dynamischen Fehlerbehandlungs- und Rekonfigurationssystem werden zwei Fehlertypen unterschieden, bekannte Fehler und neue Fehler. Im Fall eines bereits bekannten Fehlers sendet das vorhandene Fehlerdiagnosesystem dem dynamischen Fehlerbehandlungs- und Rekonfigurationssystem die von ihm ermittelten Informationen zum Fehler zu. Mithilfe einer Überprüfung der Fehlerwissensbasis, welche die bekannten Fehler und die entsprechenden verfügbaren Funktionen beinhaltet, werden die verfügbaren Funktionen ermittelt und ausgegeben. Damit kann das automatisierte System rekonfiguriert und zumindest teilweise wieder in Betrieb genommen werden. In dem Fall, dass das vorhandene Fehlerdiagnosesystem einen Fehler nicht identifizieren und klassifizieren kann, sammelt das dynamische Fehlerbehandlungs- und Rekonfigurationssystem nach der Fehlermeldung des automatisierten Systems die Zustände der Komponenten und Teilsysteme des automatisierten Systems (einschließlich historischer Daten). Daraufhin wird durch Analyse der Symptome mithilfe des Anlagenmodells eine Lokalisierung der Fehler durchgeführt. Anschließend wird die Identifizierung der verfügbaren Funktionen mithilfe des Systemmodells (einschließlich Komponenten-, Funktions- und Anforderungsmodell) durchgeführt. Abschließend werden die verfügbaren Funktionen in der Fehlerwissensbasis gespeichert und das automatisierte System mithilfe der gewonnenen Informationen rekonfiguriert. Somit kann ein Totalausfall des Systems verhindert und dadurch die Zuverlässigkeit des Gesamtsystems erhöht werden. Anhand von drei Demonstratoren werden die Realisierung und die Evaluierung des Systems gezeigt. Die qualitative Evaluierung zeigt, dass das dynamische Fehlerbehandlungs- und Rekonfigurationssystem die Fehler in den automatisierten Systemen korrekt behandeln kann. Die quantitative Evaluierung zeigt weiterhin anhand von empirischer Untersuchung, dass die Verfügbarkeit der automatisierten Systeme erhöht wird.

# Abstract

The availability and proper working of complex technological processes play an important role in modern society. At the same time, industrial automation systems are becoming more and more complex. They customarily contain an extensive number of processes and subsystems which are strongly interrelated. This has led to an ever-increasing number of possible fault locations which humans find difficult to control. Faults, which are not taken into account during the development phase, cannot be intercepted or prevented at runtime and can thus lead to an overall failure of an entire industrial automation system. However, partial unaffected functions could still be available, and therefore partial operation could be possible.

For this reason, a model-based dynamic fault handling and reconfiguration system has been developed. In this thesis, faults are divided into two types: known faults and new faults. In the case of a known fault, the existing fault diagnosis system sends the fault information to the dynamic fault handling and reconfiguration system. After the investigation using the fault knowledge, which includes known faults and corresponding available functions, available functions are sent back to the industrial automation system. This allows the industrial automation system to be reconfigured and to resume a normal operating state. For a new fault, however, the existing fault diagnosis system cannot identify the fault location. The dynamic fault handling and reconfiguration system gathers the fault information, including fault diagnosis results and the historical data. To identify the fault location of the fault, the fault localization can be performed via analyzing symptoms with the help of the system model. Subsequently, the identification of available functions can be performed via the system knowledge. Finally, available functions can be stored in the fault knowledge and the industrial automation system can be reconfigured with its available functions. Hence, total failures as well as breakdown of the industrial automation system can be prevented and the system's overall system availability can be thereby increased. By means of three demonstrators (two-tank system simulator, coffee maker simulator and high-bay warehouse simulator), the realization and the evaluation of the conception of dynamic fault handling and reconfiguration are demonstrated. The qualitative evaluation shows that the dynamic fault handling and reconfiguration system can correctly deal with faults in industrial automation systems with high efficiency. The quantitative evaluation indicates that the availability can be improved with the help of the dynamic fault handling and reconfiguration system via 100 tests. With the help of the proposed dynamic fault handling and reconfiguration system, the availability of the proposed demonstrators could be enhanced.

# 1 Introduction

## 1.1 Background and Motivation

With the extensive use of automation products in industry and in daily life, automation systems have greatly improved not only the methods of production and, accordingly, productivity, but also our quality of life and lifestyle. Home automation, for example, enhances energy efficiency and smartly securing the services of homes.

To pursue higher customer convenience, however, simple and single-function automation systems can no longer satisfy people's modern demands in the daily life. Therefore, complex automation systems that integrate more functions into one automation system, e.g. the CNC machine, have inevitably become the developmental trend of science and technology. This has unavoidably led to an increase in the complexity of industrial automation systems. In addition, to afford higher global benefits, manufacturers tend to sell their developed automation products not only in a limited area, but also to all possible corners of the world.

As a basic requirement, customers need to be supported by having a stable industrial automation system, which enables them to provide their services continuously. Therefore, the importance of system availability has increased significantly. Availability is an important characteristic that plays an increasingly important role in industrial automation systems. Not only must these systems exhibit various functionalities, they must furthermore attain these functionalities under given operating conditions in a specific time interval. Despite very high availability, faults in an industrial automation system can never be completely avoided during the entire lifecycle of a system. For this reason, a special approach in handling faults during the runtime of an industrial automation system is required. The term fault, which describes an undesirable and unwanted situation in which one or more (active or potential) disturbances occur, addresses the cause of an incident [Ebel08]. The most frequent causes for faults arising at runtime, and thus being responsible for the decreasing availability of the entire industrial automation system, are listed as follows:

- Short development time and budget constraints: limitations of software development time and budget always affect the quality and functionality of the automated system [BaPr04].
- No systematic reuse: Reuse concepts are essential for cost reduction. However, a systematic approach to the increasing complexity of automated systems is not always possible [Lugu03].
- Too few resources during the test phase: To avoid faults in the software of automated systems, the test must contain as many test cases as possible. However, this is often very expensive and even impossible because of short deadlines and restrictions on resources [KiPo02].

- Lack of experience among developers: The knowledge of developers contributes significantly to software quality. Inexperienced developers tend to have misconceptions due to a lack of knowledge about the context of a system [MoWe00].

- Insufficient knowledge of the environment in the development phase: The environment in which industrial automation systems operate is never fully known in the developmental phase. The increasing mobility and various applications of industrial automation systems reinforce this effect. However, information concerning the environment is essential for the development of industrial automation systems.

Considering the five reasons mentioned, one or more faults remain in industrial automation systems. When such a fault occurs suddenly during the service lifecycle, an industrial automation system cannot avoid terminating its service. In such cases, the expectation of a customer is that the downtime used for repairing is reduced to a minimum and that services of the industrial automation system will be resumed as soon as possible. Correspondingly, the availability of these complex automated products' functions plays an increasing role in modern society.

## 1.2 Challenges of Maintenance for Industrial Automation Systems

An internal event or a change in the environment, or a wrong action by a human operator, can result in a failure of a component. A fault in a single component can have a significant impact on the availability and performance of the system, even halt an entire system. Faults of industrial automation systems are, in fact, impossible to be avoided in their entirety during the development phase. A certain number of faults will occur during customer use. Therefore, manufacturers face the serious choice of either solving the faults as soon as possible, or decreasing the fault effect of an automation system before the fault is removed. Unfortunately, several serious gaps impede manufacturers from performing the usual maintenance:

- **Increasing complexity of industrial automation systems:** Complexity here is addressed from two perspectives. For one single industrial automation system, more functions are integrated into one industrial automation system to meet growing customer demands, for instance, in an all-in-one washer-dryer instead of a washing machine and a dryer [BDW14]. For various industrial automation systems, the producer often manufactures different goods, e.g. different generations of smartphones, different coffee machines, etc. This results in a challenge for the reasonable maintenance of various products. In addition, frequent upgrades of industrial automation systems are required to adapt to the quick changes of customer demands and market changes. Hence, the complexity of industrial automation systems hinders effective support [FSV13].

- **Effective maintenance for worldwide distribution:** The market is no longer confined to a certain region or country, but focused on the global scope, i.e. the global village. This means that automation products produced by manufacturers will be sold to different countries on a global scale [PSU13]. The traditional maintenance concept is to cover a large number of sales areas with fixed maintenance centers. The maintenance staff can provide customers with timely fault maintenance. However, whether from the cost control perspective or the timely service perspective, this traditional human-service based approach has been unable to adapt to the changes of worldwide distribution. In addition, because of the fast upgrade of industrial automation systems, this approach requires a long waiting time and high costs for training and consulting ordinary maintenance staff. For this reason, overcoming the limitation of long distances and the dependence on maintenance staff are another challenge of modern maintenance [ImSa13].

- **Lack of reasonable measures for new faults as well as faults that will always exist and cannot be prevented:** In an ideal situation, the industrial automation system is able to overcome all faults, or holds the right measures in reserve for all possible faults to prevent them. And although some faults can be detected or are already known in the development phase, there are still no corresponding measures to prevent them: for instance, redundancy of components is not possible because of limitations of cost [IPW10]. In addition, as mentioned in the last section, an industrial automation system cannot avoid the occurrence of new faults [PDK15]. In this case, it is very hard to perform the process fault diagnosis matching with known fault cases, and to take effective solutions to remove the fault, or even reduce the fault effect. Therefore, an effective concept is required for handling new faults or reducing the fault effect, which is unpreventable within the operational phase.

- **Weakness of integrated fault diagnosis systems:** As a necessary maintenance solution, current industrial automation systems usually have their own their specific professional fault diagnosis systems in order to monitor system behaviors, provide basic fault diagnosis functions, and afford certain essential instructions for customers [Roth10]. Due to the limitation of hardware capacity and the development cycle, integrated fault diagnosis systems are unable to comprise far too modern and complex fault diagnosis algorithms and approaches. In addition, the upgrade of integrated fault diagnosis systems affords plenty of room for further development costs. If the industrial automation system is updated, the fault diagnosis system also has to also be updated; however, this update is not always available. The upgrading of the new operational system of android-based smartphones is, for instance, incompatible with the old smartphones due to the limitation of hardware capabilities [FrGö15]. Hence, to overcome the weakness of the integrated fault diagnosis system, a new approach is required.

- **Lack of skilled knowledge of domestic consumers:** Users usually do not have the necessary electrical training or an education in electronics [FrGö15]. Hence, users lack specific technical knowledge and practical experience in knowing how the industrial automation system works,

in comprehending the fault messages provided by the fault diagnosis system, and in finding out the reasons for poor performance. In such a situation, the user has to rely on technical maintenance staff instead to remove or replace the faulty component [Jela12].

## 1.3 Objective of the Research Work

In line with the previously mentioned challenges, the aim of this research is to present a novel fault handling and reconfiguration approach which enables the system to work with partially available functions in order to prevent system breakdown of an industrial automation system should there be a component fault. Referring to the proposed concept, the following objectives will be strived for:

- **Providing higher availability** for industrial automation systems: For this objective, this research attempts to prevent a longtime breakdown of an entire system. In the event of a fault, partial functions of an industrial automation system can be assured with the intention of providing further services to users, i.e. reducing downtime. With reduced downtime, the availability of an industrial automation system can be improved, compared with the original case.

- **Automatic support** of handling faults and reconfigurations for industrial automation systems: Referring to this objective, the proposed concept suggests that the intervention of human beings, i.e. the normal user and the maintenance service provider, can be avoided if possible by increasing handling efficiency as well as further reducing the downtime. This objective can be achieved through establishing communication, confirming available functions, and performing the reconfiguration in industrial automation systems automatically. It is worth nothing that the identification of available functions requires the knowledge of the system's internal structure. Hence, it behooves this research to consider the definition, establishment, formalization, and utilization of a system model.

- **Handling faults dynamically**: Concerning this objective, the proposed concept enables known faults as well as new faults to be dealt with. According to the challenges in the last section, known faults can be easily detected by the existing fault diagnosis system (EFDS) via process data and known symptoms, like limit and trend checking. The concept should cooperate with the EFDS to identify if the fault is known or new. Additionally, the proposed concept supposes a method to identify the fault location, establish a reasonable system model and confirm the fault effect as well as available functions with the help of historical data instead of depending on real-time data.

## 1.4 Overview regarding the Thesis

An industrial automation system usually possesses a specific fault diagnosis system to monitor the operating state. However, such a fault diagnosis system can only handle known faults which are determined in the development phase. For new faults, there is no solution for a consistent fault handling and removal of faults. This circumstance can lead to a breakdown of the entire industrial automation system. In fact, faults can only impact parts of the system, but do not necessarily impact the entire system. In another words, some functions of the industrial automation system can still be available. Hence, a novel approach is required to handle new faults and to guarantee the performance of the available functions which are not impacted by the fault.

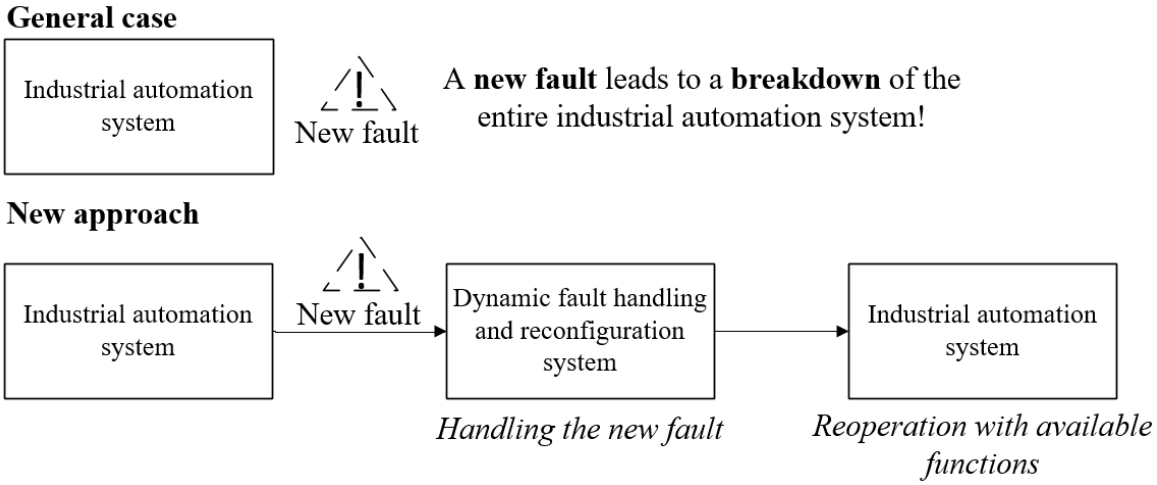



Figure 1.1: Handling new faults in industrial automation systems with new approach

To introduce the developed approach, the thesis is divided into three major parts: fundamentals and existing methods for handling faults in Chapter 2 and Chapter 3, the approved conception of handling either known or new faults in Chapter 4 and Chapter 5, and the realization and evaluation of the concept in Chapter 6.

To handle faults in industrial automation systems, fundamentals concerning industrial automation systems, fault handling and reconfiguration will be researched in **Chapter 2** (see Figure 1.2).The aim of this thesis is the improvement of the availability of industrial automation systems. Thus, the availability will be clearly defined with MTTF and MTTR. Furthermore, the fundamentals of the system model will be introduced for describing the structure of industrial automation systems. For the purpose of handling faults, it is necessary to present the basics of the faults. Moreover, different fault handling strategies and principles will be considered. To activate still available functions and to combine the fault handling strategies, reconfiguration approaches have to be researched in order to adapt the approved concept. To perform reconfiguration properly, the reconfiguration principle will be outlined. At the end of this section, based on the objective of the thesis and the fundamentals, five requirements concerning the objective, the system model, fault handling and reconfiguration are indicated for establishing a novel fault handling conception.

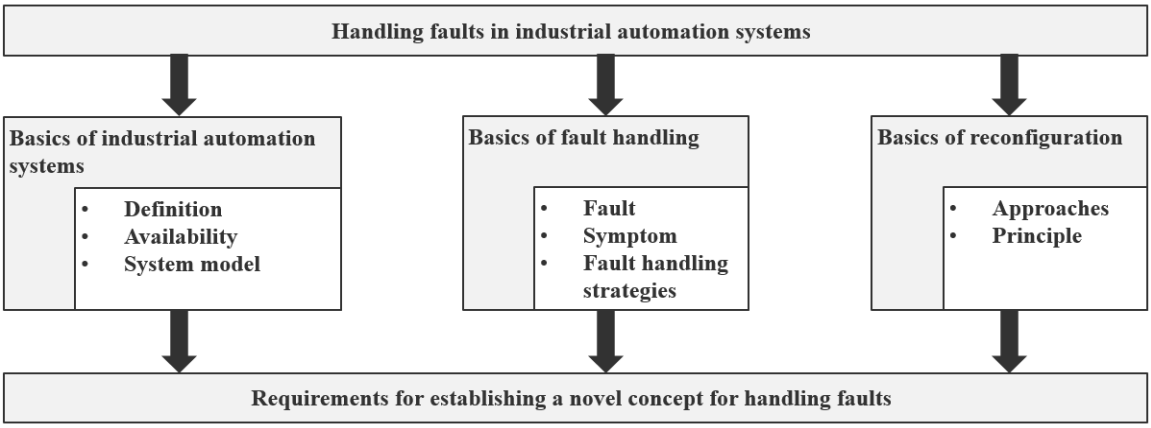



Figure 1.2: Schema of basics concerning handling faults in industrial automation systems

The aim of the thesis is to enhance availability of industrial automation systems by performing still available functions in case of a fault. To realize this aim, two major functionalities are required: handling faults with available functions, and analyzing new faults and their impacts properly. To handle faults, a survey of existing methods of handling faults is conducted, covering aspects like fault prevention, fault tolerance, fault removal and fault forecasting. With regard to new faults, the approved concept has to be able to determine the fault location and the fault impact in an industrial automation system. This requires a very comprehensive knowledge of the system structure and its behaviors. Establishing a proper system model is required to attain such system knowledge. Hence, four major system modeling methods are presented and compared, namely: process-oriented system modeling, data-oriented system modeling, state-oriented system modeling and object-oriented system modeling. Thus, methods concerning handling faults and system modeling are surveyed in **Chapter 3** (see Figure 1.3).

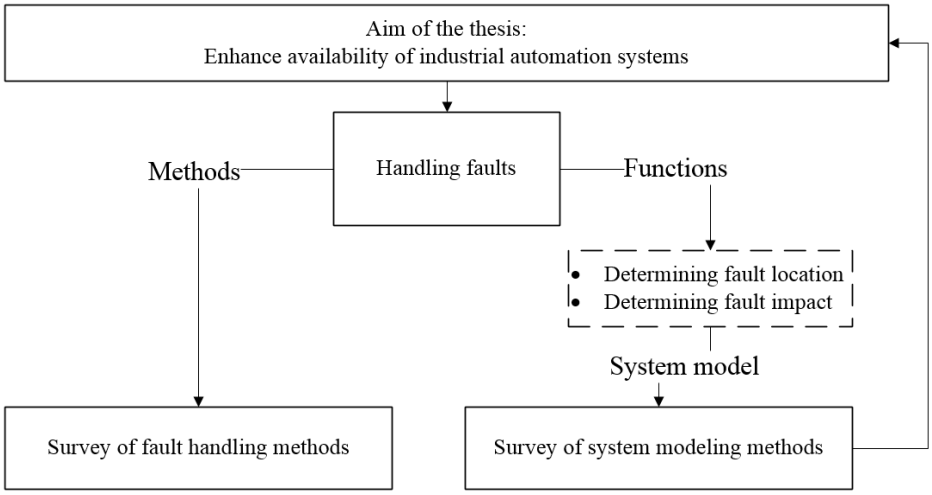



Figure 1.3: Survey methods of handling faults and system modeling

The challenge of handling faults is to handle new faults that did not occur in the development phase. Thus, comprehensive system knowledge is required to conduct a reasonable procedure for determining the impact of a new fault. Following the general development approach of an industrial automation system according to a V-model, a development sequence is carried out

through the following steps: requirements analysis, system design, implementation with components and test. **Chapter 4** proposes a system model to describe an industrial automation system, which includes a component model, a function model and a requirement model, shown in Figure 1.4. However, in order to propose computer-supported automatic assessment, the system model has to be formulated appropriately. Two functionalities, namely the fault localization and the identification of available functions, are supposed to characterize the impact of a fault. For the fault localization, specific attributes concerning faults, symptoms and system parameters extend the system model. Through combining the fault information with the existing fault diagnosis system, the fault location within the component model can be confirmed. To formulate the system model, this thesis utilizes metrics to indicate the relations among components, functions and requirements, and specific rules to indicate the condition whether functions and requirements are available. Thus, relations are able to provide a route for determining the fault propagation with the determined fault location within the component model, as well as the defective components. Rules are used to evaluate the availability of functions.

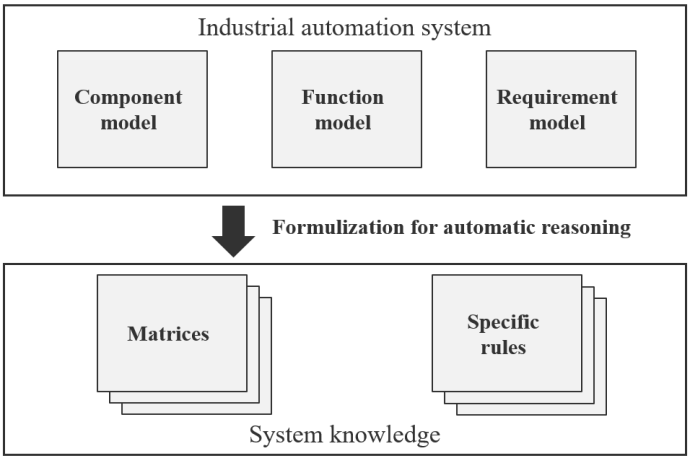



Figure 1.4: Schema of system model and system knowledge

**Chapter 5** introduces the concept of the dynamic fault handling and reconfiguration (see Figure 1.5). To handle faults in industrial automation systems, this concept supposes cooperating with an existing fault diagnosis system which can provide primary fault diagnosis results. In this thesis, two types of faults are distinguished: known faults, and new faults. The handling process is divided into major parts accordingly, namely handling known faults, and handling new faults. To handle known faults, three steps are supposed: analysis of fault information from the existing fault diagnosis system in the industrial automation system; identification of available functions for known faults with the help of fault knowledge; and determination of reconfiguration commands to conduct a system reconfiguration with available functions. For new faults, two additional modules are required to determine fault location and still-available functions. Moreover, fault handling knowledge, including fault knowledge, symptom knowledge and system knowledge, is required to deal with faults. Firstly, based on the symptom knowledge and the system model, the fault location can be determined. Secondly, a fault impact in the industrial automation system can

be determined by means of formulated system knowledge. Consequently, available functions can be identified. Based on available functions and resource information, the availability of ongoing tasks in the industrial automation system can be evaluated. For available functions and available tasks, reasonable reconfiguration commands can be integrated.

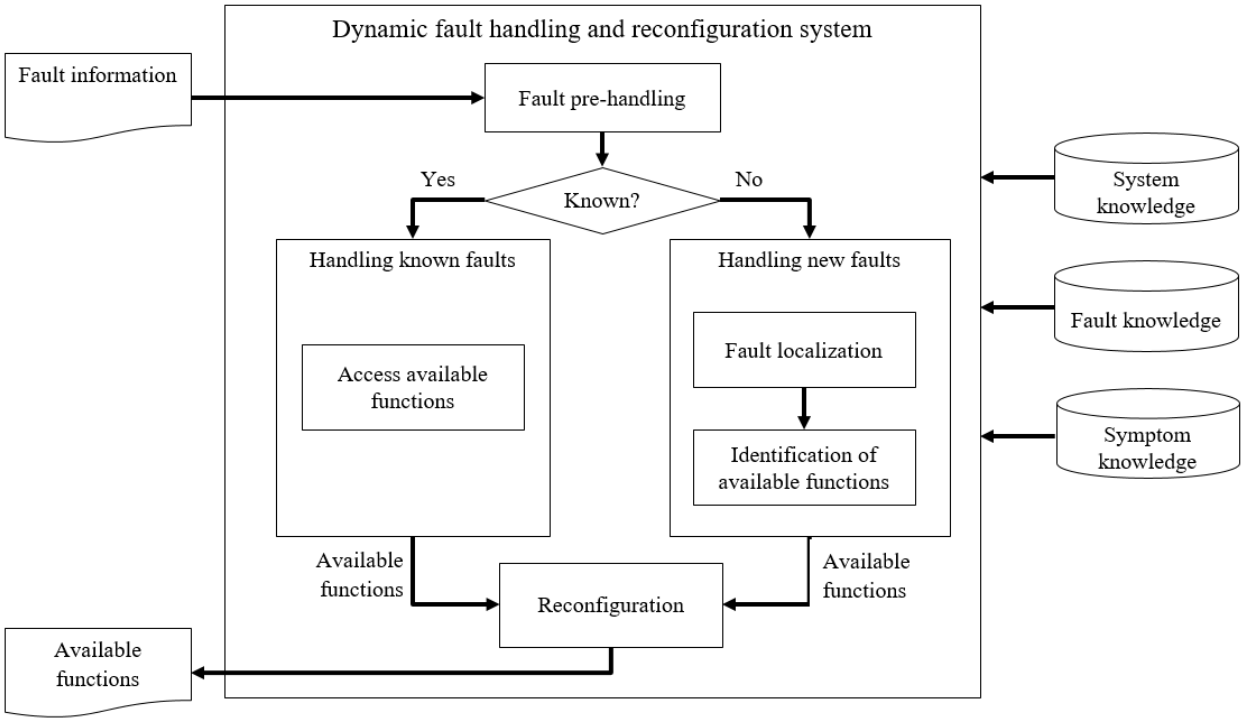



Figure 1.5: Conception of dynamic fault handling and reconfiguration

Following the introduction of the novel dynamic fault handling and reconfiguration concept, its realization and evaluation will be presented in **Chapter 6** (see Figure 1.6). The realization is introduced in terms of software architecture, fault handling knowledge and communication. To evaluate the system, three demonstrators (the two-tank system simulator, the coffee maker simulator, and the high-bay warehouse simulator) were developed. The evaluation consists of two major perspectives: the qualitative and the quantitative evaluation perspective. The qualitative evaluation is supposed to evaluate if basic functionalities of the dynamic fault handling and reconfiguration system are able to be performed correctly. The quantitative evaluation attempts to evaluate how much the availability can be improved with the help of the dynamic fault handling and reconfiguration system. For this purpose, 100 tests for each demonstrator were performed in comparison with the case with no dynamic fault handling and reconfiguration system. Finally, according to the defined requirements in **Chapter 2**, a general evaluation of the conception of dynamic fault handling and reconfiguration is highlighted.

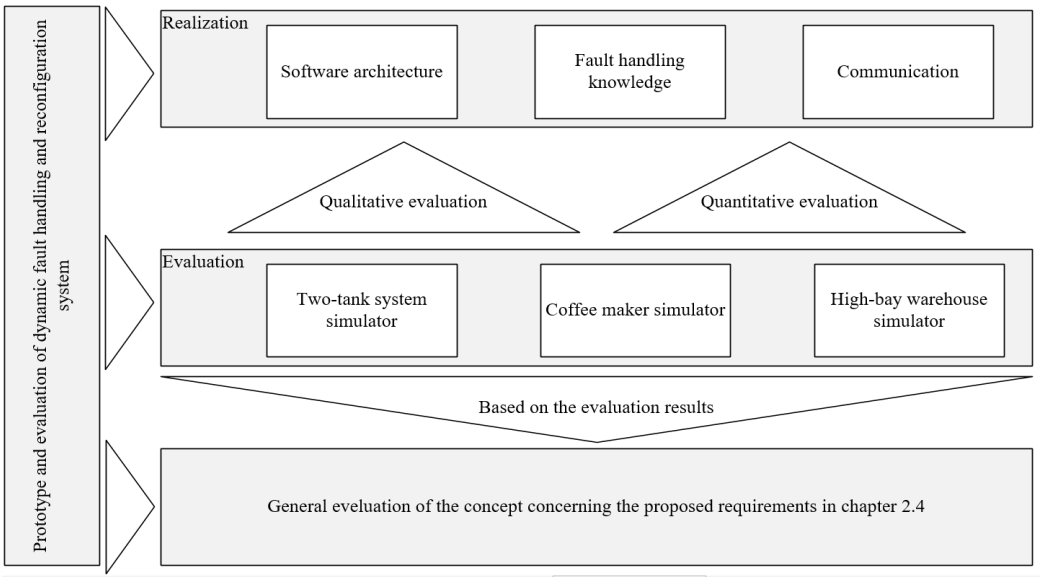



Figure 1.6: Prototype and evaluation of the conception of dynamic fault handling and reconfiguration

**Chapter 7** provides a summary of this thesis by underlining its important aspects. Moreover, the limitations of applying this concept are presented. Additionally, the possibilities that can be regarded in future work within this field are indicated, such as tools for the automatic formalization of the system knowledge.

# 2 Fundamentals of Fault Handling and Reconfiguration for Industrial Automation Systems

The complexity of industrial automation systems is a major reason why faults occur: The developer does not have a complete understanding of the complex system. Different reusable components are (possibly) being utilized in different domains in order to realize the functionality of the complex system. In this case, there is often insufficient time for a detailed and full test. It is therefore essential for fault handling to abstract the structure of industrial automation systems in order to provide an overview of the entire system for analyzing faults. In this chapter, the basics of the industrial automation system are presented. Additionally, the fundamentals concerning availability, fault handling and reconfiguration will be introduced.

## 2.1 Basics of Industrial Automation Systems

### 2.1.1 Definition of Industrial Automation Systems

An industrial automation system is a computer-based or microcontroller-based system, which is utilized to produce various products, to transfer energy or to handle information flow. According to [ZNM18] [WKS+17]], it consists usually of one or more processor units, such as microcontrollers, sensors, and actuators.

The processor unit interprets input signals, evaluates the input with its preset tasks via its own specific algorithms, and outputs the signals with the intention of adjusting the actuator behavior. The sensor converts the physical quantity, for instance, temperature, pressure, and velocities, into electrical signals and passes them to the processor units. The actuator receives the electrical signals delivered by processor units via the fieldbus system and performs the required actions with the intention of influencing the technical process. In addition, the technical process is the sum of the processes in which the physical quantities are recognized and influenced with technical methods. In this thesis, the physical individuals in the technical process are denoted as elements, e.g., liquid tank, pipes, etc.

In [ZNM18] [WKS+17], industrial automation systems concerning the different process variables are classified into three major types:

- Continuous processes for industrial automation systems are processes with time-dependent process variables that denote the time-dependent behaviors of physical state variables in the technical process, e.g., the chemical and heating processes. All control processes are denoted through continuous processes. To complete the description of time-dependent process variables, a mathematic model is suggested for one continuous process via differential equations, transfer functions, etc.

- Sequential processes (also called discrete event type processes) for industrial automation systems are processes that own various and distinguishable process states that perform systematically and consecutively. Examples for sequential processes can be the manufacturing process, start-up and shutdown process, etc. The transmission between these states can be assigned with binary events that can reflect or influence the discrete process states. To describe the sequential process, a state or event model can be a reasonable modeling approach.

- Discrete object type processes for industrial automation systems are processes in which single, identifiable objects that can be converted, transported or stored, can be assigned with their process variables. Examples of the discrete object type process can be the warehouse process, traffic process, etc. The object-oriented model can be applied in describing these processes.

In addition, an industrial automation system will usually have not only one process, but several processes. The manufacturing process includes, for instance, continuous processes, sequential processes and discrete object type processes, but the sequential process is the dominating process. Therefore, to fully describe an industrial automation system, the utilization of only one modeling approach is insufficient, requiring instead a combination of several models.

### 2.1.2 Availability

In relation to a technical product or a technical plan, availability means the property that a specific reaction follows for an input. One of the most important requirements with respect to technical products or technical plants is the continuity of services, that is, that the automation system can continuously provide necessary services, e.g. the manufacturing of products. Availability is an important criterion to judge the quality of an industrial automation system. High availability is also the objective of this research as mentioned in the last chapter. For clarity in terms of high availability in industrial automation systems, several definitions concerning availability are given below.

In [DIN10a], the term *availability* is defined as follows: “Ability of an item to be in a state to perform a required function under given conditions at a given instant of time or over a given time interval, assuming that the required external resources are provided”.

The above definition highlights a system being able to realize specific functions under given conditions successfully. Following this definition, the objective of the research can be thus indicated:

- Consideration unit: The entire system should continue to work during the appearance of a fault within a system component as well as resume work after an abort as soon as possible.

- Required functions: The functions required by the user as well as the available functions are able to be guaranteed by both isolating affected functions and further executing not affected functions.
- Certain time period: In this thesis, the selected observation time is the period of the occurrence of a fault in the industrial automation system.
- Without failure: Although a system component fails, the entire system is enabled to work by isolating the impact of occurred faults.

Furthermore, [Iser06] defines availability as the probability that a system or equipment can provide its functions effectively at any period of time. According to this definition, the case that failures and malfunctions occur and carry out repairs is taken into account by availability. As a probability, it proposes the equation

$$Availability = \frac{MTTF}{MTTF + MTTR}$$

to calculate the availability, where MTTF is the mean time to failure and MTTR is the mean time to repair. Obviously, to obtain a high availability, the research should attempt to increase the operation time MTTF and decrease the repair time MTTR, i.e. downtime. In other words, if the fault can be handled in the original repair time, and partial functions of the system can be performed immediately, then availability can be enhanced with a decreased MTTR and an increased MTTF.

### 2.1.3 System model

As discussed in the previous section concerning research goals, this research intends to guide automated products to reconfigure themselves in a certain range with the help of available functions when the automation system fails due to a component fault. Therefore, automated products are able to continue to work and to still provide available services to users.

The system model is the model that can describe and represent a system from different views and be considered as the result of system modeling. With the benefit of modeling techniques, the properties of an industrial automation system, the classes, architecture, behavior, and interactions, including either internal or external interactions with the environment, can be clearly demonstrated and shown. Different models developed by different modeling techniques play different roles: facilitating the development of algorithms, the system structure, the selection of hardware as well as software components, etc. Hence, the system model helps to simplify the analysis, design, and implementation in the development phase. Furthermore, the system model provides the opportunity for testing to verify the correctness of the system's behaviors. This can usually be applied for fault impact reasoning, such as fault tree analysis (FTA) and process model based fault diagnosis [Iser06].

However, if an automated system follows a wrong guide, not only is the desired objectives unachievable, but the wrong guide can also result in further fault impact expansion and it can even bring the whole system to its knees, e.g. through secondary failures. Therefore, a full and clear evaluation of the fault impact scope in the system's inner structure is a very important prerequisite for decision-making. Otherwise, the necessary constraints, like safety, ought to be considered. To achieve this purpose, system models are proposed which describe the system, like the structure and according attributes, and provide knowledge concerning the automated product in some definite form, e.g. text, graphics (including symbols), physical simulation and mathematical formulas [BIA11]. The classification of system modeling techniques can meanwhile be based on the various system information provided by specific models, such as context models, data flow models, state machine models, object models, semantic data models, etc. There is, however, a large subset of different types of models and associated modeling language to address various features of an automated system. The correct and suitable type of system modeling techniques should be selected for the intended purposes and scopes.

## 2.2 Fault Handling for Industrial Automation Systems

In this section, the fundamentals of fault handling regarding fault and symptom and fault handling for industrial automation systems will be introduced.

### 2.2.1 Fault

Functions of the industrial automation system ought to be executed successfully according to the requirements of users. In practice, unforeseen and undesirable situations may occur. In this case, the industrial automation system cannot perform its functions to achieve its target or, in an extreme case, is not functional at all. In the analysis of availability, faults, errors and failures play an important role [Gert15]. Hence, several basic terms concerning faults need to be indicated in this thesis.

In [DIN10b], the term fault is defined as the: "State of an item characterized by the inability to perform a required function, excluding the inability during preventive maintenance or other planned actions or due to lack of external resources". The term error is a "discrepancy between a computed, observed or measured value or condition and the true, specified or theoretically correct value or condition" [IEC61508] [Goll12]. In [ISO11], the term failure is a "termination of the ability of an element to perform a function as required".

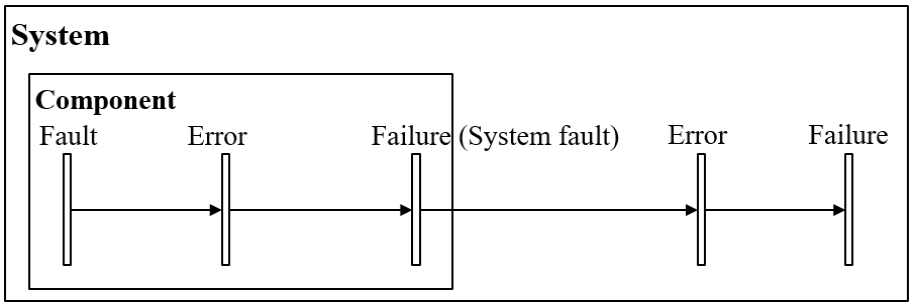



Figure 2.1: Relationship between fault, error and failure for component and system [Goll12]

Based on the above definitions, it can be concluded that a fault occurs first in a system component's physical layer, e.g. through the wear of actuators. As a result, the information level is affected as well. After a certain latency, the actual value cannot attain the desired setup value, i.e. error in component. The function of the component can then be lost, namely, a failure of the component. In a similar way, component failure can be propagated to the system level, which can result in a system fault. Due to the loss of the component function, the setup value of the system would no longer be achievable, and it would also result in a deviation, i.e. system error. In the end, a component fault can lead to the breakdown of the entire system, i.e. system failure [Goll12]. This process, as well as the relationship between fault, error and failure, is indicated in Figure 2.1. As discussed, because the aim of the research is to assure partial functions of an industrial automation system, this thesis concentrates on a system fault as well as the failure of a component. The defective component can be recognized as the location of the fault, i.e. the fault location that denotes the source of the fault in this thesis. Otherwise, the behavior or procedure of finding the fault location is called fault localization in this thesis. In addition, the defective function of the defective component is defined as malfunction.

Only component failure(s) are considered as fault(s) here. In terms of the number of appearances of a fault, faults in this research thesis are divided into known faults and new faults [WJG15a] [WJW15]. A known fault is a fault which has appeared at least once before and its fault diagnosis approaches are therefore known. In other words, when this fault reoccurs, it can be easily detected and diagnosed. If a fault appears for the first time, this thesis terms it a new fault. Such a fault will be diagnosed at first by finding its fault location.

Concerning various application cases, the term function can be comprehended in three different ways:

**Mathematical function** is a relation between one or a set of variables and another or another set of permissible variables, with the property that each variable is related to exactly another variable, e.g. $f(x) = ax + by$.

**Program function** is a part of code which can be performed by the computer with a specific running environment, e.g. *int function_add(int a, int b){int a, int b; int c = a +b; return c;}*.

**Conceptual function** is an action or a service in which an item or a system has to realize the demand of its user, e.g. *producing espresso* by the coffee maker.

In this thesis, when the term function appears alone, then it means the conceptual function.

### 2.2.2 Symptom

To detect and diagnose system faults, the observed event or variables will be utilized, i.e. symptoms. Generally, there are two types of symptoms: the analytic symptom for the automatic processing of measured variables, and the heuristic symptom for evaluating observed variables [Iser06].

The analytic symptom (also known as analytic knowledge) denotes the measurable and analytical information from the process of industrial automation systems. To attain this information, three major approaches can be applied via the process variables, which can be measured and processed based on the generated characteristic values. They are: limit value checking, signal analysis of directly measurable signals, and process analysis via mathematical process models. For them, different characteristic values ought to be recognized and generated, e.g. the measured sensor value with the setup threshold value, trend, variances, model parameters, state variables, etc. These parameters can be defined by various specific mathematical functions or methods, such as limit checking of absolute values [Iser06] with the mathematical function $Ymin < Y(t) < Ymax$, specific fault models, and process models.

Different from the analytic symptom, which can be generated by using quantifiable information, the heuristic symptom requires expert knowledge and experience with difficult to measure information [Iser06], for instance, noises, colors, smell, etc. Hence, the heuristic symptom is extremely dependent on the specific knowledge and is usually represented in a fuzzy form, i.e. linguistic variables: small, larger, approximate number, etc.

The analytic symptom is obviously a very reasonable choice for automatic symptom generation which can avoid the additional knowledge representation for human knowledge. By means of the symptoms, it can simplify the fault diagnosis procedure; in other words, establish reasonable functions or models for the symptom generation and an appropriate fault (location) -symptom- relations, such as the symptom matrix. Utilizing the reasoning approach, it can compare the current symptoms generated with the symptom matrix to complete the fault diagnosis as well as the fault localization.

### 2.2.3 Fault Handling in Maintenance

Maintenance as the last part of the life cycle of an industrial automation system is an important activity to keep the normal functionality of a system via handling faults. However, each year more than 60 million US dollars are lost due to ineffective maintenance management and faults

[Mobl02]. To perform effective maintenance, methods of maintenance ought to be designed and followed. Generally, two types of maintenance approaches are employed: corrective (also known as run-to-failure maintenance), and preventive.

Corrective maintenance is defined as "*activities undertaken to detect, isolate and rectify a fault so that the failed equipment, machine or system can be restored to its normal operable state*" [Alge10] [TYM10].

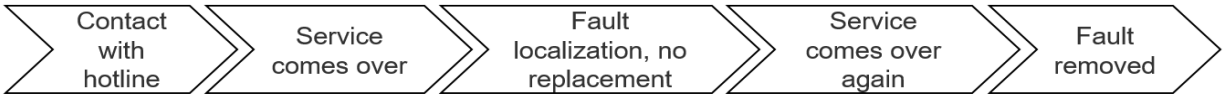



Figure 2.2: Corrective maintenance procedure by maintenance staff [Frie15]

As the definition highlights, corrective maintenance tries to handle faults that have occurred. The traditional handling approach is indicated in Figure 2.2. When a fault in an industrial automation system takes place, the following procedure will be carried out continuously, namely contacting the hotline and making an appointment, service coming over, fault localization with no replacement, service returning and fault removed with the replacement. The obvious advantage of this method is that faults can be eradicated with new replacements by means of skilled personnel, but the disadvantage is long downtime.

Preventive maintenance is defined as "*a series of activities undertaken to inspect system, detect, correct and prevent the incipient faults, before they become actual or major faults*" [KhDe11] [Wang02] [CCO12]. As a typical method of preventive maintenance, predictive maintenance is able to monitor, detect and diagnose the process condition of an industrial automation system [CCO12]. Benefits from preventive maintenance are the system's availability and reliability, and thereby the productivity of industrial automation systems can be increased [BCY03] [ÖFH15] [CCO12]. These two approaches aim to handle either faults that have occurred or have not appeared yet. As discussed in the last section, this research focuses on component failures as well as system faults.

Generally, in accordance with the development time of a fault, the interval of a fault can be divided into fault causation, fault detection, fault explanation, and fault elimination. Fault causation indicates the occurrence of a fault resulting from various factors, such as wear of an actuator. The fault can then be detected via various characteristic values [Iser06], namely fault detection. Subsequently, this fault impacts functions or behaviors of the subsystem, and even the entire system [WJW15]. In this phase, the fault impact can be further confirmed. However, with several specific actions, such as exchange replacement, the fault can be eliminated [Iser06].

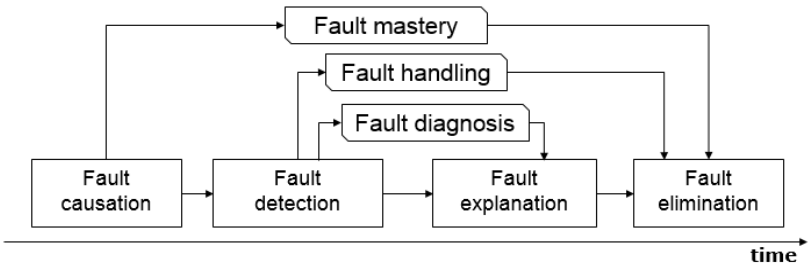



Figure 2.3: Process of fault mastery, fault handling, and fault diagnosis [Donl07]

Concerning the fault development phase, the techniques for processing faults can be defined as fault mastery, fault handling and fault diagnosis. Fault mastery aims to monitor the fault occurrence to the fault elimination in the entire process of faults. Fault handling covers the three phases, i.e. fault detection, fault explanation and fault elimination. Fault diagnosis allows for detection of a fault and its analysis with detected or observed characteristic values [Gert15].

In the operating phase, errors and failures lead to the stopping of the entire system. The handling of faults and failures plays a very important role in increasing availability. According to [Donl07] [WWW08], there are basically three strategies to increase the availability of industrial automation systems concerning the two maintenance strategies:

- Strategy of avoiding faults, also called perfection or intolerance strategy: This strategy attempts to prevent the causes of faults, errors and failures with the intention of arriving at a fault-free, perfect system. This includes, on the one hand, so-called fault prevention or error prevention [AlFu14] (prevention of the occurrence of faults), and on the other, fault or error disclosure (fault detection before the system is in operation, e.g. appropriate tests), such as fault prediction [ASF15] and fault elimination [MJKJ14].

- Strategy of avoiding the impact of faults, errors, and failures: When a fault cannot be completely avoided, this strategy enables a prevention or compensation of the fault effect, for instance, via the redundancy technique [Gert15]. Therefore, this strategy can be also referred as a fault tolerance strategy [ZhLy10].

- Strategy of reducing the fault, error, and failure effect: In this strategy, the individual participates in the treatment of faults, errors and failures in industrial automation systems in order to delimit their effects. Concerning occurred faults, errors as well as failures, it is sufficient that either the user, using the instructions of the industrial automation system (e.g. the user manual) tries to modify the technical system, or maintenance staff repairs the system to correct the error [CZJW16, WMSP17]. In addition, some technologies can support this strategy, such as fail-soft or fail-safe [Cool03] with the intention of guaranteeing partial operability of functions as well as reducing the fault effect [Bush14]. This strategy allows an industrial automation system to continue to work with maximum functionality during the existence of faults.

### 2.2.4 Principle Procedure of the Fault Handling

The previous section introduced three strategies of fault handling, and these three strategies follow the same principle procedure in Figure 2.4, namely feature generation, fault detection, fault diagnosis, fault evaluation and decision making [Iser06]. Feature generation can be attained by special signal processing, observed data from the process, state estimation, etc. With the generated data, it can be further processed towards fault detection in order to generate symptoms. Fault diagnosis tries to confirm the fault type, the fault reason and the fault location to employ the knowledge of analytic and heuristic symptoms. Both the classification and reasoning methods, in line with the relationship faults with symptoms, can be carried out for fault diagnosis; for instance, a fault-symptom-tree. Fault evaluation can be applied generally for the safety evaluation, such as evaluating the hazard level of the current fault to the industrial automation system. Depending on the fault diagnosis and fault evaluation results, decisions ought to be confirmed by the system itself or by the intervention of skilled personals [Iser06].

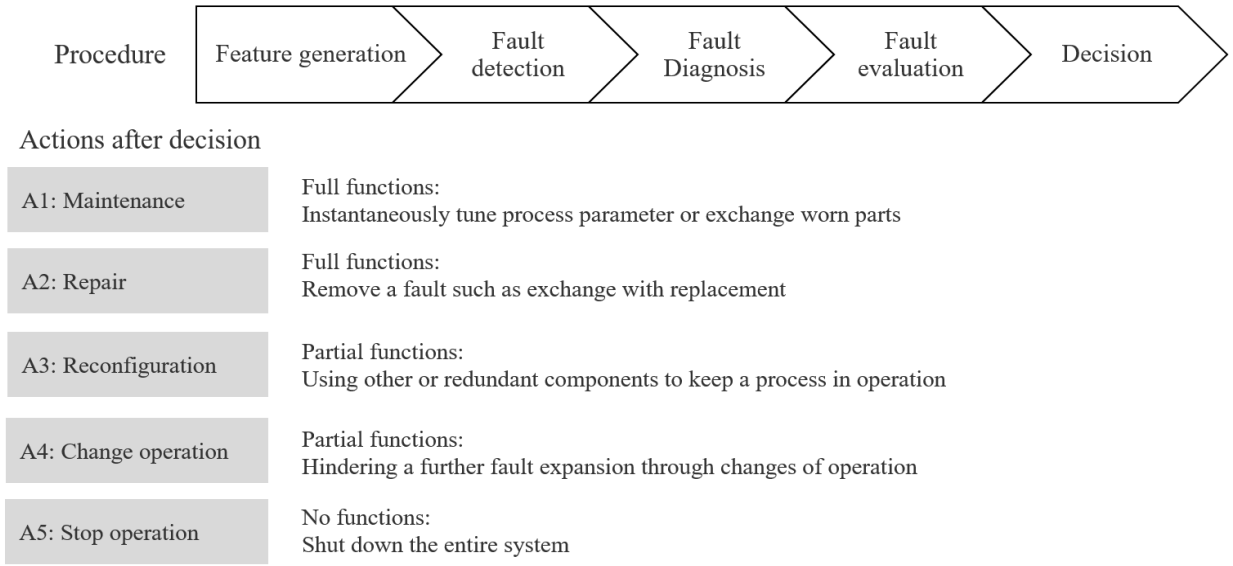



Figure 2.4: Principle procedure of fault handling and actions

On the basis of the series of fault diagnosis activities, five types of appropriate actions will be performed: regular maintenance while instantaneously maintaining or exchanging possible worn parts; repairing by removing a fault such as exchanging with a replacement; reconfiguring with other or redundant components to insure the normal operation of a process; changing operations with new operating ways to prevent further fault expansion; and stopping the operation by shutting down entire systems. Concerning the research's aim to reduce the MTTF, the first four actions can lessen the MTTF, specifically reworking in the original downtime. However, first action regular maintenance requires either financial input, including man-power and cost, or a very robust fault diagnosis system to predict and diagnose the occurrence of faults. The second action, repair, requires the intervention of maintenance staff. The MTTF can be reduced only through an optimized maintenance process. In this thesis, the action reconfiguration with redundancy and

change operation can be utilized for reducing the MTTF. To simplify the action types, these two approaches will be called reconfiguration, and either redundancy or change operation can be identified as two specific reconfiguration methods.

## 2.3 Reconfiguration of Industrial Automation Systems

In the case of the appearance of a fault in an industrial automation system, reconfiguration is an appropriate choice for suiting the change which is led by the fault [BPK06]. This subsection gives the basics of reconfiguration and the principle reconfiguration procedure for an industrial automation system. In addition, a survey of reconfiguration approaches concerning measures of fault handling will also be introduced.

### 2.3.1 Reconfiguration

In [Mate10], Matevska defined reconfiguration as follows: Reconfiguration represents the technical view of the process in which an already developed and operationally deployed system is changed to adapt new requirements, extend functionality, eliminate faults (effect) or improve quality features. Concerning the mentioned reconfiguration in the last subsection, this thesis focuses on reconfiguration when a fault has occurred. In line with the proposed definition, reconfiguration here means that developed and deployed industrial automation can change its system state to another operational state to eliminate the fault effect.

In order to perform a reconfiguration in an industrial automation system, the system requires the following major characteristics [MUK00]. Characteristic 1: Modularity, which means the system is made up of functional components, i.e. software and hardware components [GuGe04]. Characteristic 2: Inerrability, which means the system and its components are able to conduct either internal changes or to integrate future and further new technologies. Characteristic 3: Convertibility, which allows a quick changeover between existing and future products. Characteristic 4: Diagnosability, which means the industrial automation system has the specific module or functionality to identify the occurrence of a fault and the corresponding sources, i.e. fault location. Characteristic 5: Customization, which allows the deployed industrial automation system to match the application (product family) [KoCa00].

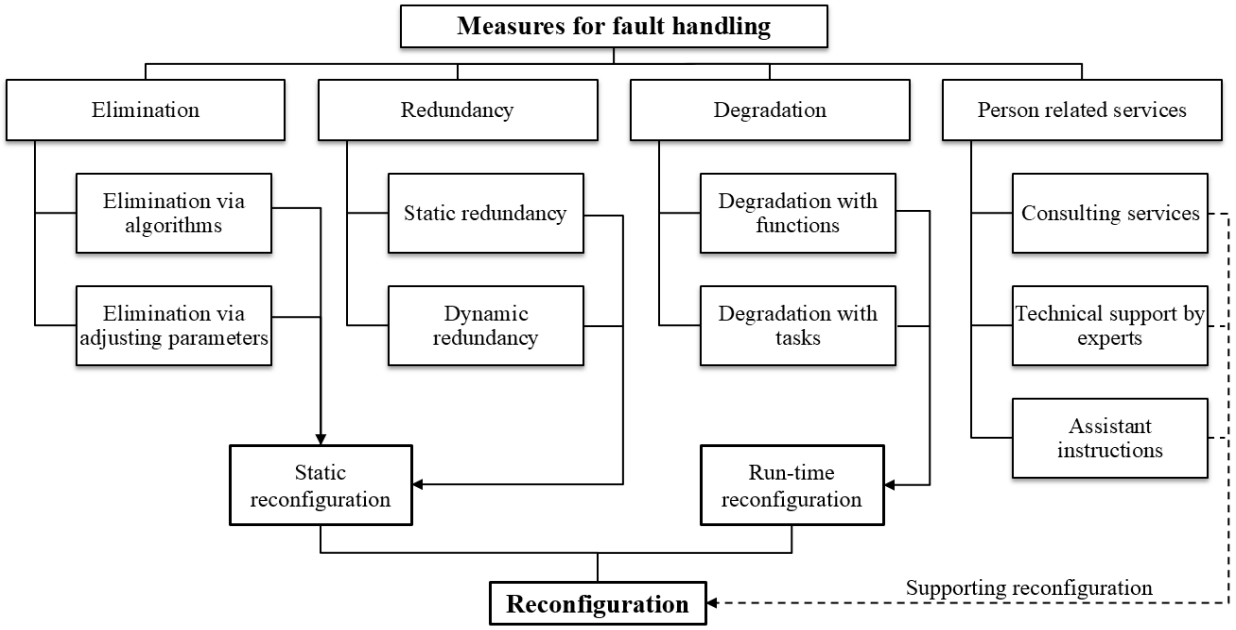



Figure 2.5: Survey - reconfiguration concerning measures of fault handling [Wang17]

As Figure 2.5 indicated, a survey of four major measures for fault handling and two main reconfiguration types is cited. Concerning measures of fault handling, possible approaches can be the elimination of the interference or error via adjusting mathematic algorithms [BKLS03] or adjusting parameters [NoJo09], a redundancy strategy via static redundancy or dynamic redundancy, graceful degradation with operational functions or tasks, and person-related services to support the reconfiguration such as consultation, technical support by experts and assisting instructions for users [Wagn14, Frit05, Böhl10]. In line with [WJW17], the first two approaches usually require a restart for adapting the configuration, such as activating the specific code for a specific algorithm or redundancy. Run-time reconfiguration (RTR) offers the capability of reconfiguring the system in run-time [BMS07, MHR03]. Typically, in the application of reconfigurable manufacturing systems, specific hardware is required for performing the reconfiguration, for instance, the deployment of FPGA [SiFe06].

### 2.3.2 Relationship between Reconfiguration and Fault handling for Industrial Automation system

For the purpose of utilizing reconfiguration as measures for fault handling, a fault detection and gnosis module and a reconfiguration module are deployed to rectify the fault effect in industrial automation systems, such as a value error due to an interference.

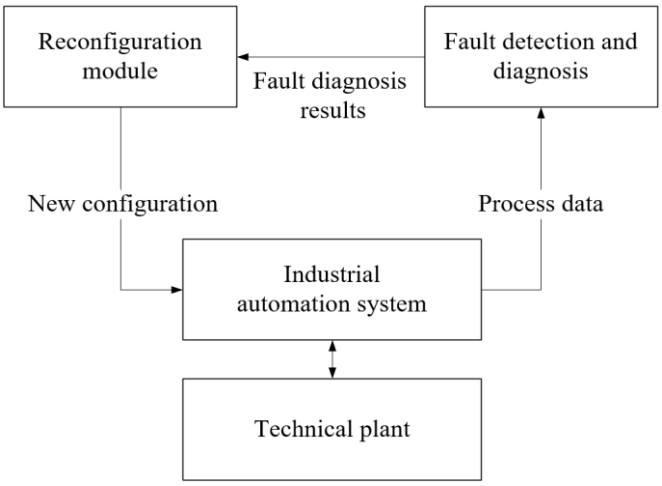



Figure 2.6: Control reconfiguration for industrial automation systems [CAA14] [Iser06]

In the present Figure 2.6, an interference or abrupt change can lead to failures in various components; the process data, e.g. state variables, can be detected by the fault detection and diagnosis module [CAA11]. By means of specific fault models, as well as process models, the faulty behaviors which occurred in the system should be tested or detected. In the fault case, the industrial automation system cannot afford the correct behavior. The corresponding diagnosis results, e.g. fault location and fault effect, are used as feedback for the reconfiguration module. On the basis of the results and its own knowledge, the reconfiguration module evaluates the fault effect and confirms possible measures to compensate for the error as well as the fault. Finally, the new configuration plan can be outputted to the industrial automation system to restructure the system architecture [PMD+17] or the control structure [VHBL15] to prevent a system level failure.

## 2.4 Requirement analysis of the Dynamic Fault Handling and Reconfiguration

As introduced in Chapter 1, the aim of this research is to enhance the availability of an industrial automation system as much as possible. In the case of the appearance of a fault, the objective can be explained as providing the still-available functions in the industrial automation system. To attain the ability of maintaining functionality when portions of a system break down, five proper requirements for fault handling, which are derived from the challenges in Chapter1, are necessary. The challenges were empirically determined with the following research: [BDW14] [ImSa13] [PSU13] [IPW10] [PDK15] [Roth10] [FrGö15] [FrGö15]. Additionally, more positive input came from the discussion with academic experts in different conferences and workshops, e.g. ICICM 2015. Similarly, visits to exhibitions, such as the Embedded World Exhibition, also provided suggestions about the challenges of industrial automation systems. Based on this input, five requirements for establishing the concept were derived. These requirements are further indicated as follows:

- **R1: Ability of enhancing the availability of the entire industrial automation system:** It requires endowing the automated systems with the continuously operating ability and lengthening the service time of the industrial automation system, i.e. increasing the mean time to failure (MTTF) and shortening the mean time to repair (MTTR). To achieve this, an industrial automation system ought to provide, and work with, partial functions when a component is out of order. It needs to be noted that this requirement concentrates on the overall system availability, but not partial function availability. In other words, some partial functions of an overall system are allowed to be sacrificed.

- **R2: Ability of automatic, reasonable and dynamic fault analysis:** The proposed concept ought to respond to faults that have occurred, send back information, and carry out a proper measure as soon as possible. At the same time, in terms of handling faults, the proposed concept should avoid the circumstance of fault duplication handling, for which it needs an information-share platform and mechanisms to process the known and new faults separately. This means that the proposed concept can pinpoint the fault location – defective components or subsystems – according to the fault information, and obtain the available system functions in line with the reasonable internal relationships of the automated system structure. Most obviously, the structure of an automated system should be broken down into different suitable views after considering the reasonable aspects, e.g. the physical aspects of components and the logical aspects of functions and dependencies of functions, such as safety, performance, security, etc.

- **R3: Ability of reconfiguration to maintain available functions remotely:** The proposed concept ought to provide an appropriate method to perform the reconfiguration, namely available functions and a guide for the reconfiguration in the industrial automation systems. For the former case, because of a wide distribution of systems, the activating of the new configuration is a serious problem to solve. Moreover, depending on different reconfiguration strategies (static and dynamic reconfiguration) and specific demands, the proposing concept will be able to provide enough reasonable measures for activating available functions and isolating unavailable functions. In some special cases, for instance, switch off some key valves by the user for activating redundancy, the guidance for the user is very important in completing corresponding actions.

- **R4: Ability of reducing the cost for implementation and in operation:** It is an important criterion to evaluate the proposed fault handling system. The development, establishment and running of the proposed concept should cost as little as possible. In addition, the proposed concept should attempt to avoid increasing the burden on the existing automated products, such as a large system change and large real-time data acquisition. The perspective of long distance and wide distribution of industrial automation systems can also influence the cost, i.e. providing an ***effective maintenance support by manufacturers remotely*** is a possible problem-solving approach.

- **R5: Ability of porting the conception for heterogeneous industrial automation systems:** The proposed concept ought to be ported for different types of industrial automation systems including continuous processes, sequential processes, and discrete object type processes. For the purpose of decreasing the porting cost, the communication approach and the required knowledge should be defined clearly and uniformly. Furthermore, the proposed concept should possess the ability of handling faults for all industrial automation systems of the same type at the same time.

This chapter has described the basics concerning the research on industrial automation systems, fault handling and reconfiguration. The composition of a system and three system types, i.e. continuous processes, sequential processes and discrete object type processes, were presented. Moreover, the definitions of availability and system model were introduced. On this basis, the research objective proposes to decrease the MTTR as well as increase the MTTF with the intention of increasing availability. Subsequently, the basics of fault handling, in line with fault, symptom, fault handling in the maintenance, and the principle procedure of the fault handling, were outlined. Three possible maintenance strategies concerning corrective maintenance and preventive maintenance were introduced. Based on these, the research aim was further limited in case of a system fault (component failure). After that, reconfiguration was defined and a survey regarding reconfiguration and fault handling measures was presented. Drawing from traditional fault tolerance control, the relation or cooperation method between reconfiguration and fault detection was introduced. Finally, five requirements for establishing a conception of dynamic fault handling and reconfiguration were outlined based on the challenges in Chapter 1. Hence, for a new fault, the fault location and fault impact, including available and unavailable functions, are required for analysis and confirmation.

# 3 Survey of Methods concerning Handling Faults and System Modeling

In the last section, the aim of the research was defined as improving or ensuring high availability of industrial automation systems by reducing MTTR as well as increasing MTTF. To achieve this aim, a proper approach for handling faults, as well as identification of fault impacts, is required. For this purpose, this section presents a review focusing on methods for handling faults and conceptions of system modeling. Four fault handling methods will be introduced. To identify and confirm the fault impact in an industrial automation system, it will be necessary to establish a system model; therefore, reasonable modeling concepts will be considered, too.

## 3.1 Survey of the Methods for Handling Faults

As [ZhLy15] [Dubr13] mentioned, there are typically four approaches or means for handling faults in industrial automation systems:

- Fault prevention of handling faults supports the prevention of the appearance or instruction of system faults.
- Fault tolerance of handling faults supports the assurance of the service of the system correctly in case of the presence of system faults.
- Fault removal of handling faults attempts to achieve the reduction of the occurrence of faults, including number and severity.
- Fault forecasting of handling faults is intended to appraise the presence, propagation, and possible consequences of system faults.

The meaning and research for each approach will be presented in the next four sections. An assessment of methods for handling faults will be outlined in Sub-section 3.1.5.

### 3.1.1 Fault Prevention of Handling Faults

Fault prevention is a significant issue for manufacturers to enhance the reliability and availability of a production line or an industrial automation system. Fault prevention generally works in tandem with fault diagnosis methods. Before a component fault develops into a system fault, it can be detected by various characteristic values, for example, temperature values, so that different corresponding measures can be applied to prevent the occurrence of the system fault [PaHa07]. System fault prevention helps to avoid the breakdown of the entire system and guarantee lower production costs and less waste [BoGö13].

Bordasch proposes a functional model and a hybrid abnormity identification concept to prevent the appearance of known faults [BoGö13] [Bord16]. For a new fault, the model provides a fault identification concept to identify characteristic values regarding limit and trend checking via a real-time monitoring, termed as abnormity. Fault and abnormity diagnosis are based on a reasonable process model. Based on these abnormities, Bordasch proposes to identify the development of known component faults before they develop into a system fault. If a component fault endangers the whole system's functionality, it can be reported to maintenance experts, who can intervene and correct the fault in time [BBG15] [WaWe16].

In [RFHG16], Rakyta proposed a maintenance support system for reconfigurable manufacturing systems to execute a quick response to a system abnormity, such as executing prevention repair. To overcome the limitation of long distances and to acquire run-time data for experts, Mori introduced a smartphone as a communication media between the local user and machine manufacturers [MoFu13]. Using it enables remote manufacturers to monitor the industrial automation system and provide reasonable maintenances [MRZ+13].

### 3.1.2 Fault Tolerance of Handling Faults

Fault tolerance is an approach that intends to contain the consequences or impact of faults and failures so that the system can still deliver correct functions to avert a system failure. To reach this goal, the most commonly utilized way is the redundancy of components, subsystems or even the whole system [KoKr07]. To realize fault tolerance, there are two methods: static redundancy and dynamic redundancy [Iser06].

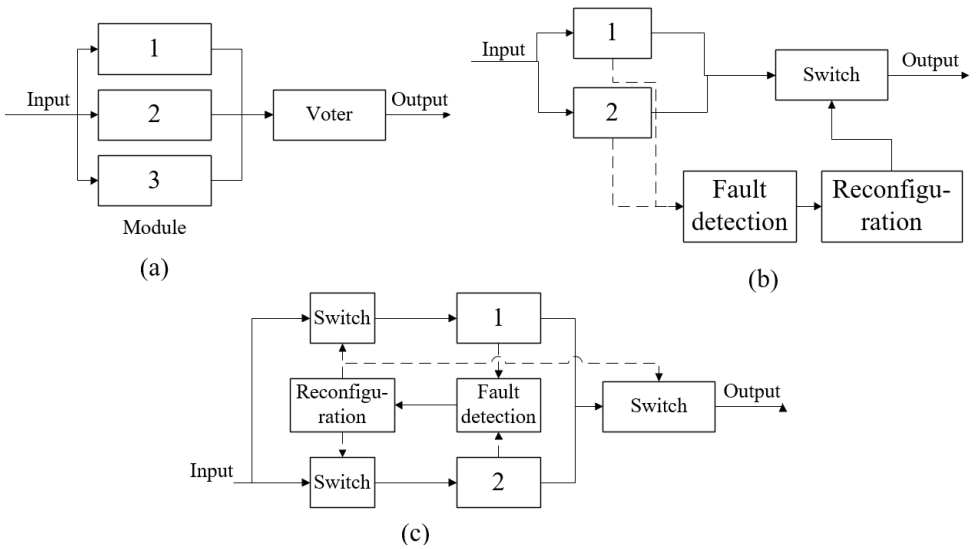



Figure 3.1: Schemes for fault tolerance [Iser06]

Figure 3.1 shows the schemes for fault tolerance in line with static redundancy and dynamic redundancy. For static redundancy (Figure 3.1a), a voter is established to compare three signals from three modules which receive the same input, and to choose the correct result as the output.

For dynamic redundancy, there are two additional modules for detection and reconfiguration: hot standby and cold standby. In hot standby (Figure 3.1b), fault detection is in charge of assessing the results from two modules. If one fails, reconfiguration instructs the switch to accept the correct result from the normal module. If one fails in cold standby, the reconfiguration module activates the switch of the redundancy module to get the correct signals and control the simultaneous switch in output point.

Some research in, and applications of, fault tolerance in industrial automation systems will now be reviewed. In [ESA07], Emmert suggested some fault-tolerant methods and a runtime reconfiguration for FPGA (field programmable gate arrays) logic blocks to realize online test, diagnosis, and reconfiguration for handling faults in defective blocks. He attempted to reuse defective logic blocks to increase the number of effective spares and extend the task life. More research concerning fault tolerance and the reconfiguration for FPGA can be outlined as follows: Lima proposed a method for transient fault detection and evaluation in SRAM-based (static random-access memory) FPGAs [LCR03, LNH+04].

Avizienis insisted that fault handling is an individual method in fault tolerance and belongs to corrective maintenance. Fault handling prevents known faults from being reactivated. Four steps are required to implement a fault handling: fault diagnosis, to identify the fault type and fault location; fault isolation, to isolate the defective component either from a physical or logical perspective; system reconfiguration, to control the switch logic, either by hard standby or cold standby; and system re-initialization, to check and update the new configuration [ALR01].

To prevent a system failure or safety consequence, graceful degradation is applied as the solving solution via maintaining limited functionality of a system. Different applications of graceful degradation are fault safe, to assure the safety functionality, and fail soft, to guarantee operational functionality [Iser06].

### 3.1.3 Fault Removal of Handling Faults

Fault removal can be performed either in the development phase or in the entire operational life of an industrial automation system. This aims to reduce the number of faults which remain in the system via a set of methods [Dubr13]. In the development phase, three steps are required: verification, diagnosis, and correction. During the operational phase, corrective maintenance, to remove reported component faults, and preventive maintenance, to uncover and remove component faults, are used to prevent a system error or system failure [ALR01].

[RaGo11] proposed a concept for the safety analysis of a railroad crossing's critical system based on the combination of FMEA (Failure mode and effects analysis) and fault tree analysis to implement the fault removal. Additionally, in order to overcome the wear and tear of the hardware component, industry generally proposes a regular check and maintenance, with a time schedule in which the possible or soon-to-be-defective components can be exchanged with new

replacements [RaHo14]. Additionally, along with the popularization of smartphones, a large number of research conceptions have been developed for fault removal [WaTs06] [VSG+12] [FrGö15] [MRZ+13]. [FrGö15] proposed a user-friendly diagnosis system based on a mobile device which contains an extended diagnosis app. By means of this system, the normal user is able to diagnose the fault and follow specific guidelines to remove faults.

Moreover, faults actually can be removed in the development phase via various tests, however, more faults will arise in the operation phase. Using the fault information to improve the continuous test of an industrial automation system is another opportunity to improve the systems availability. Abele [AbWe16] proposed a shared decision support system that provides benefit for both, the development department and the local maintenance team. This system can generate a value-add by using synergies by combining the support functions for fault diagnosis and the test management [AbWe17].

Afterwards, regardless of preventive or corrective maintenance, remote maintenance via the internet, e.g. e-maintenance [LLWY09], is a typical maintenance strategy, [HAA+10]. In this approach, the maintenance service center builds a real-time monitoring system and a simulation system to monitor the running industrial automation system. If a fault appears, the maintenance staff can detect the fault in time and carry out corresponding measures with the help of additional staff, such as an exchange replacement.

### 3.1.4 Fault Forecasting of Handling Faults

Fault prevention attempts to estimate the number of the remaining faults in an industrial automation system, the time of the next fault appearance and the consequence of a fault [Dubr13]. Fault forecasting can be deployed using both qualitative and quantitative approaches. The former approach intends to evaluate the system behavior through the failure mode and event. The latter refers to determining the probabilities concerning system quality attributes, either reliability or availability.

A considerable amount of research has been done regarding fault forecasting during the last decade [ZXL07] [Wang04]. [ZXL07] presented a methodology for forecasting device downtime by means of an auto-regressive moving average (ARMA) model. In this approach, the historical data is used to predict the future behavior of the system. This method can reflect the condition of an industrial automation system and cooperate with the fault removal approach to carry out proper maintenance measures for manufacturers.

### 3.1.5 Model-based Approaches

Besides the introduced four general approaches, there are different model-based approaches for handling faults in industrial automation systems to reduce or compensate the fault effect. A large

number of publications and research exist in the area of dealing with fault handling [WiPa16] [VRF+16] [AET11] [APA+16] [WaVo08] [SWLV13]. These researches are capable of maintaining the entire system availability in case of a component failure by means of self-healing, reconfiguration, restructuring, robust optimization, etc. [MSPB12]. They are named as fault tolerant control. As classified in [JiYu12] [AET11], fault tolerant control are generally divided into two categories: passive fault tolerant control systems and active fault tolerant control systems. The difference between them is, that the active fault tolerant control requires a fault diagnosis in order to perform reasonable actions, but the passive fault tolerant control compensates the fault automatically via fixed robust controllers [JiYu12]. Furthermore, the active control can be categorized into adaptive robust controllers, direct redundancy, analytical redundancy, and flexible scheduling.

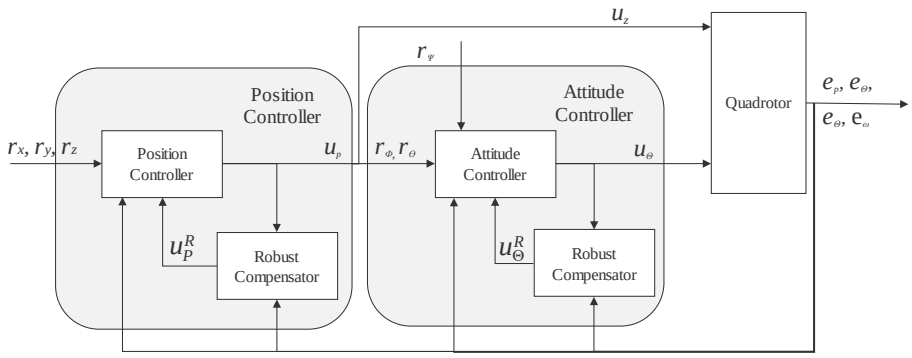



Figure 3.2: Schema of a robust controller for quadrotors [LZZZ17]

In order to deal with several parameters like uncertainty, noise and disturbances, the robust controller are designed to be robust automatically based on the robust control theory [Mack13]. Concerning the passive robust controller, Benosman [BeLu10] proposed a Lyapunov-based feedback controller to assure local uniform asymptotic stability of the system. This research tries to compensate the additive unknown bounded signals on actuators. In this case the faults are already predefined, and as a result, if an occurred fault is not considered in this scope, the stability and satisfactory performance of the industrial automation system cannot be guaranteed. Furthermore, to compensate the uncertainty and disturbance faults in actuators, a lot of active robust controller approaches have been proposed, for example, sliding mode control based designs [LGSZ14], learning based approaches [LWZ17], robust adaptive fault-tolerant compensation controllers [LiYa12], etc. These approaches suggested either to select a precomputed control law or redesign the robust controller online. With help of the robust controllers, the closed-loop systems are bounded and, as a result, the states converge asymptotically to zero. [LiYa12] proposes an adjustment of the controller parameters to estimate unknown lower or upper bounds, using adaptive laws driven by system response errors and to compensate then the errors automatically and adaptively. Against the time-varying sensor faults in wind energy conversion systems, Kamal proposes a fuzzy proportional-integral-observer to estimate the fault and an observer based dynamic fuzzy fault tolerant controller to compensate the fault effect via stabilizing the closed-loop system [KAGB12]. In order to function against the parametric

perturbations, nonlinear and coupled dynamics, external disturbances, state delays and input delays in quadrotors, Li proposes a robust cascade controller which includes an attitude controller, a position controller, based on a hierarchical control scheme and a robust compensating technique [LZZZ17]. The schema of the proposed concept is depicted in Figure 3.2. Without determining the fault location, the robust controller is a reasonable approach to compensate the occurred fault effect and turn the system to a stable state again automatically. However, this approach needs a very detailed mathematical process or system model. Realizing such an approach requires a lot of effort and computation time. The robust controllers are usually regarded as an attribute or ability of the industrial automation systems, and are integrated in the industrial automation control system. The knowledge about the system and faults is also integrated in the system. This makes the management of the knowledge very difficult. Due to its specific algorithms, it is hard to port it to another industrial automation system directly. This approach requires always a closed loop to compensate faults. However, if a component is defective, e.g. a faulty sensor, there is no more closed-loop control available for the controller, hence the robust controller cannot maintain the stability of the industrial automation system as well as its functionalities.

To overcome the weakness of the robust controller in case of failed components, redundancy becomes a very useful measure. Redundancy can be furthermore divided into direct redundancy and analytical redundancy [AET11]. The direct redundancy requires real physical redundancy of components such as sensors and henceforth the switching from a defective component to the redundant component has to be realized. Obviously, such physical one to one redundancy is very expensive and not economical for industry. Under the assumption that already many components, which perform similar tasks, exist in the industrial automation system and the data transmission is logically reconfigurable. Marcos proposes a fault tolerant component management platform over data distribution services to compensate the fault effect for industrial automation systems [AEM12]. He suggests a data distribution service as an efficient middleware to resign the communication way. In the assumed application scenario, the communication among different components can be established with different nodes. So if one of the nodes is shutdown, the industrial automation system is able to restart the affected components with other available nodes. Figure 3.3 shows the communication reconfiguration schema for two nodes. With concern to qualitative requirements, in [AME12] dynamic service reconfiguration and fault effect compensation are performed automatically based on backup data, nodes and component redundancy. But this approach is limited for the industrial automation system who owns plenty of components with similar functionalities, such as sensors [APEM14], communication nodes, etc. This limits its application range excludes systems which don’t have backup components. Furthermore, there is no consideration about a central knowledge for known faults. This can result in multiple analysis for the same faults. Because the concept is designed for a specific scenario, it is also very hard to transfer the concept to other industrial automation systems. Moreover, this approach is usually based on known fault locations, in which case the fault analysis can be executed. But if a new fault occurs, it is not mentioned how to identify the fault location.

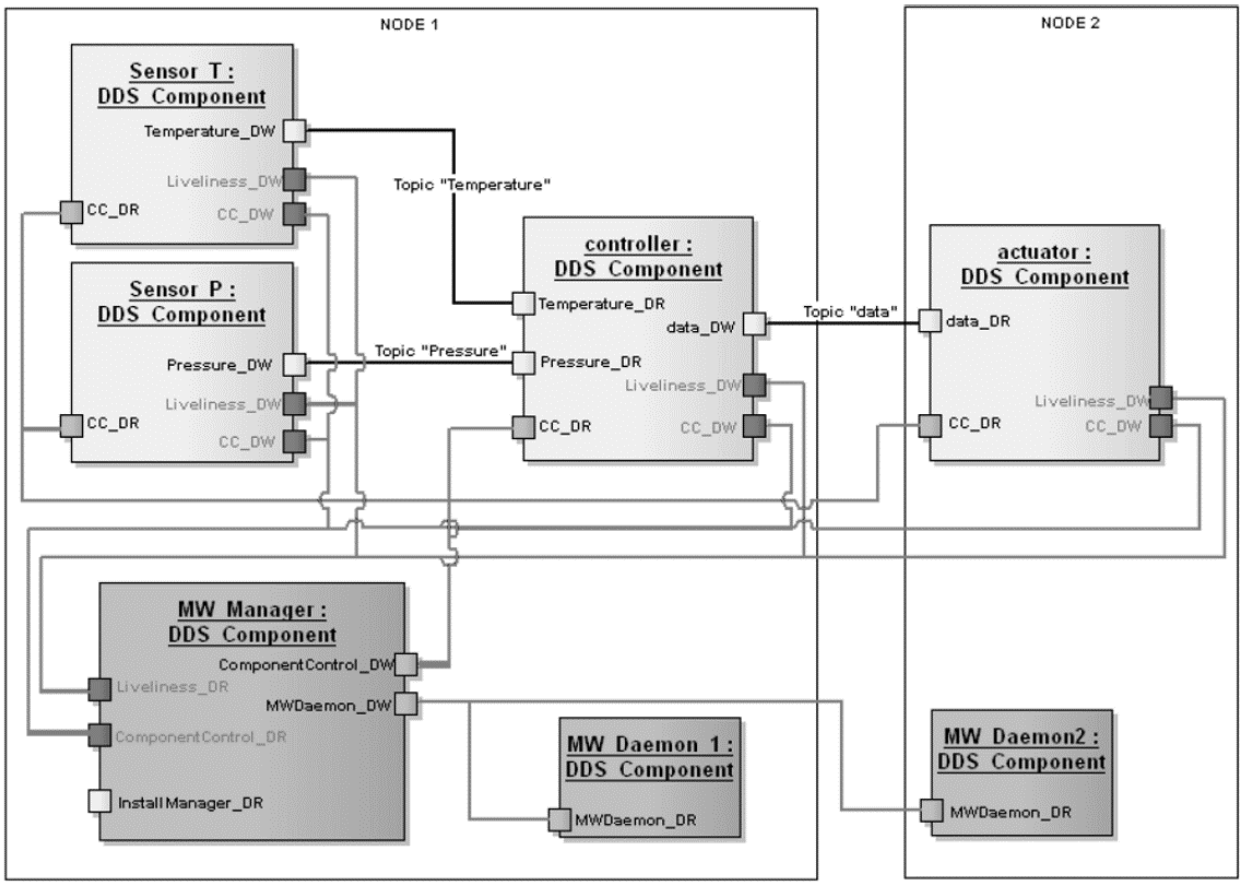



Figure 3.3: Example for direct redundancy of communication nodes [AEM12]

In order to overcome the high costs of direct redundancy, analytical redundancy which owns no real additional physical components but uses virtual components is proposed by different researches to rebuild a full control loop [PIGM17] [RNPB12] [PTA10]. Instead of using model matching approaches, as discussed with the robust controller, Rotondo designed a virtual actuator or a virtual sensor to replace the defective actuator or sensor in the closed-loop control against the noise in the control loop as well as sensor or actuator faults [BRPN14] [RNPB12] [RHWL11]. It assumes that the sensor or actuator is stuck or the gain of the faults by comparing the current sensor data and previous measurements in Figure 3.4. If there is a direct redundancy for the defective component, it can be directly activated. Conversely, it activates predesigned virtual components with its corresponding fault estimation module. Then it reconfigures the controller within the virtual component, and adjusts the fault estimation module in the virtual component on the basis of the feedback sensor data, until the system is in a stable state to compensate the fault of the defective components [OdSt12].

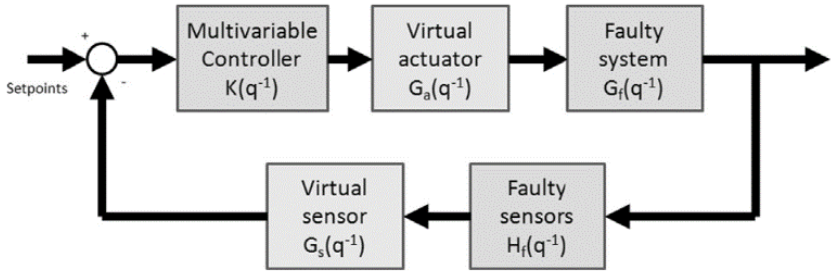



Figure 3.4: Analytical redundancy with virtual components concerning the robust controller [BRPN14]

However, if the sensor or actuator completely failed, there is no feedback data. The theoretical stability of the industrial automation system is not safe and the correctness of the performed actions cannot be assured. In order to avoid this situation, Wannagat proposed a multi-agent based approach, which is able to perform a runtime reconfiguration of the industrial automation system in order to fully compensate the component faults, such as failed sensors and actuators. To avoid an expensive downtime of plants, a failed physical sensor is replaced by a virtual sensor, which is seen as a redundancy for this physical sensor and is created dynamically via calculation of a measuring points, based on analytical dependencies to neighboring sensors [WaVo08a] [WaVo08b] [SWLV13]. This concept is used in a PLC-based industrial automation system. In its scenario, there are several sensors in one continuous process. They perform the measurement functions for one parameter, e.g. distance. Then, when one sensor is defective, an agent is activated as a virtual sensor to represent this defective sensor. The dependencies of sensors are illustrated in the Figure 3.5. Depending on the runtime constraints like material flow dependencies and accuracy, which are formulated in a redundancy matrix, the best neighboring sensors are chosen for the creation of a virtual sensor. The virtual sensor uses calculated values as the measurement values to rebuild the control loop. So that the industrial automation system is able to perform the function of the defective sensors as a normal sensor [Wann10] [VLL15] [Voge17a] [WSV13].

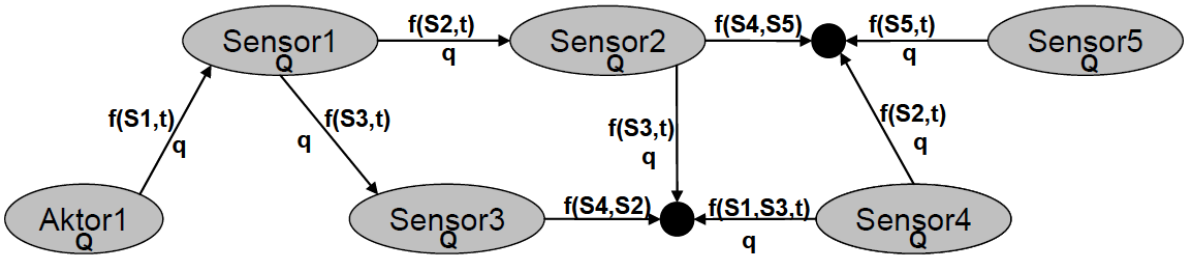



Figure 3.5: Analytical dependencies of sensors and actuators [WaVo08a]

Furthermore, to establish such a multi-agent PLC-based industrial automation system, Daniel proposed a model based development tool to realize such a virtual sensor as well as the required agents [ScVo13].

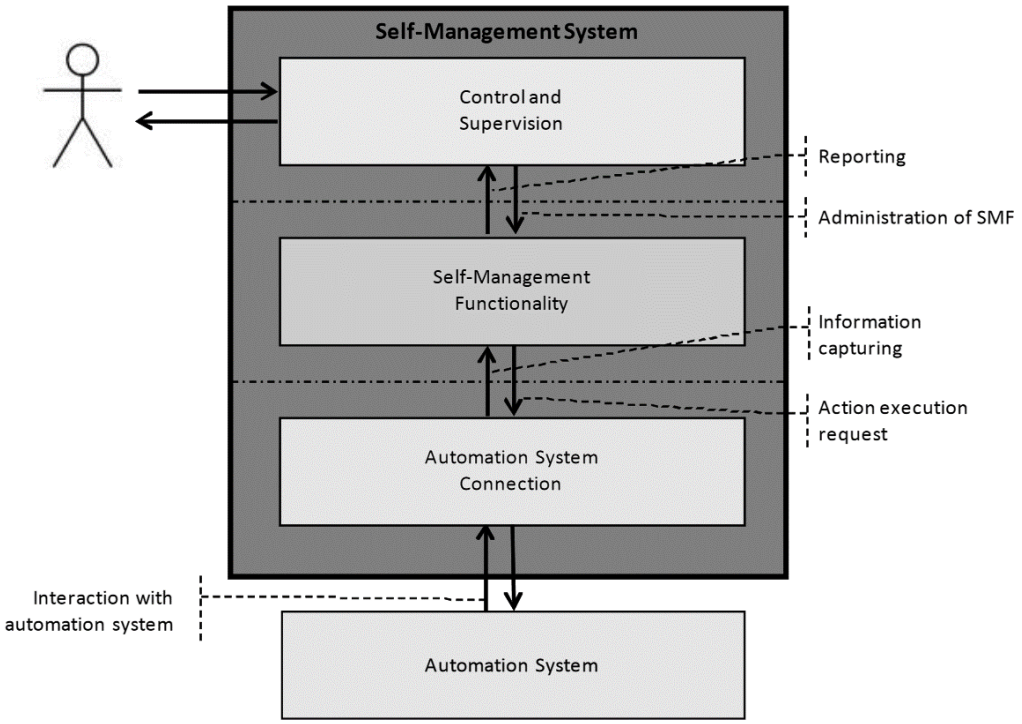



Figure 3.6: Basic architecture of the multi-agent based self-management system [MuGö11a]

Similarly, Mubarak proposed an agent-oriented approach for self-management of industrial automation systems to compensate the occurred faults during operation time [MuGö10a]. In this concept, an assistant system is proposed with six different agent types in three working levels, they are automation system connection level, self-management functionality and control supervision in Figure 3.6. The concept is shown on a lift control example with a defective position sensor, one of four position sensors. The negotiation agent coordinates with the remaining three correct sensor agents to determine the current position of the lift. This result can be transferred to industrial automation systems. Hence, the industrial automation system can use only three position sensors to control the lift stop in the corresponding floor [MuGö10b], so that the industrial automation system can still perform the functions smoothly though one sensor is defective. However, this approach in principle utilizes an information redundancy, which can be created by establishing mathematical calculation instead of direct physical redundancy. This assumption is very specific. If there is only one physical sensor for a parameter in one work station, this concept cannot be realized. Moreover, this concept is performed by agents. The knowledge ontology formulization is very complex [Bazg12]. Another disadvantage concerning communication and implementation of the multi-agent-system is its costs. With the increasing number of agents, there are plenty of messages to process and this result in a very high protocol complexity [Glav06]. Meanwhile, a large number of agents leads to reduced execution speed [Bazg12]. In [Wann10], the proposed concept is integrated into an industrial automation system to compensate fault effects. It requires an exact and correct system model and also redundancy information in the development phase. This makes it hard to adjust the system model and predefined evaluation requirements to

changes of the system. For example, in the development phase, two serial robots are accessing different screw types. After the calculation of accessing distance, one robot is able to take over the work for the defective one. But after the installation, the working range of this robot is too large which poses a threat to human safety. If the fault effect compensation mechanism is integrated in the industrial automation system, the system has to be redeveloped to adapt to the change in the industrial automation system. Another approach [MuGö10a] owns an additional self-management system utilizing a multi-agent-system. The disadvantages of utilizing agents has been discussed in the preceding part of the text. Furthermore, this self-management concept is in principle an integrated fault handling system, which has to be developed in the development phase in order to realize the communication between the self-management system and the industrial automation system. However, the upgradation of industrial automation system has to adapt extremely fast to the quickly changing demands of the customer. This requires a very limited development time budget. Parallel development of the industrial automation system and the self-management system, even completing the functional test of both, is also a hard work. Additionally, due to worldwide distribution, such an integrated fault effect compensation mechanism is very costly to maintain. Moreover, due to a lack of a central fault knowledge base, one same fault in the system as well as different systems has to be analyzed repeatedly. Due to no individual system knowledge, the development of these concepts is very specific. So that porting to different industrial automation systems requires a redesign of the fault effect compensation mechanism.

Besides redundancy, flexible scheduling on the ongoing orders in industrial automation system is another way to compensate or reduce fault effects [Pine16] [CTT+17] [TPB13+] [WWBF14]. Flexible scheduling is the ability of adapting to the increased demands for manufacturing facilities, typically, in the context of Industry 4.0 [LeVo17] [JBM+17]. That means, according to the changes of the customer demand, the industrial automation system is able to automatically adjust the receiving orders into different work stations as well as various different automation systems. It is necessary to note, that a production line may own different working machines from different suppliers and the internal system knowledge of an individual machine is unknown to the whole industrial automation system [JBM+17]. Concerning the fault effect compensation, the defective working station or component can be regarded as the change of a customer demand. The defective industrial automation system reschedules its work load to the rest of the working stations [PIGM17]. Priego supposes a customizable and extensible architecture to assure the fulfillment of the quality of service an industrial automation system should offer, which is depicted in Figure 3.7[PIGM17]. Based on a set of quality of service (QoS) requirements and distributed agents, the concept is able to reconfigure the microcontrollers with deactivation and activation of non-critical microcontroller states and reconfigure the orders for still available controllers. This concept can realize a system recovery with some quality of service and thereby realize partial orders in industrial automation systems. Furthermore, in [PAO+14] [Prie17], Priego suggests a model-based approach to assure the availability of control systems in spite of PLC-failures in an automation system. In this approach, a supervisor and its specific design methodology is able to

restore the functionality of a failed PLC into an already running PLC. Then the industrial automation system can still provide functionality of the whole industrial automation system to meet the requirements of the customers. In order to establish such a specific design, which includes reconfiguration mechanisms, Priego proposes a model-based tool to support the generation of the elements that compose the reconfiguration control system using model driven engineering (MDE) techniques and technologies [PAEM15].

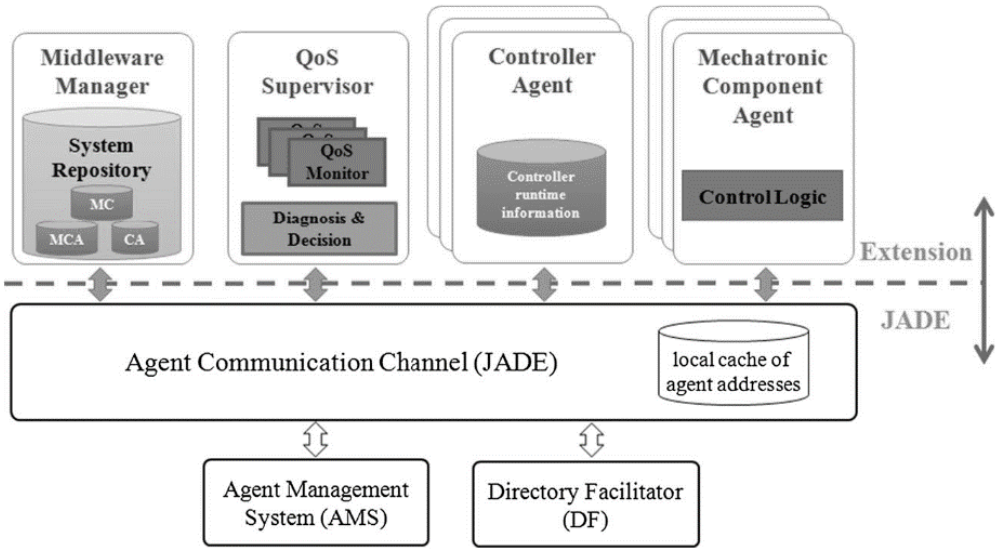



Figure 3.7: The structure of the multi-agent based middleware for managing different service [PIGM17]

Concerning operation states, similarly, Bareiß proposes a model-based failure recovery approach for automated production systems, combing sysML and industrial standards like IEC 61131-3 [BSP16+]. To describe the states of a process, this concept supposes to establish pre- and post-conditions for each operation state. On the basis of these states, the automated production system is able to realize the fault diagnosis as well as the fault localization. However, there are no further recovery measures mentioned than only to inform the maintenance to exchange the defective module. Moreover, Legat suggests configurable partial-order planning approach on the basis of a combination of an adapted goal-based planning formulation and its reformulation by means of linear programming techniques [LeVo17]. In order to increase the efficiency of flexible scheduling in the context of Industry 4.0 and cyber-physical systems [Voge17b] [ASS+17], a state space planning and a domain-specific layering is used for adjusting the ongoing orders in each work station, a small industrial automation system which completes some specific tasks with some specific functions and a fixed throughput (average output of a production process per unit time) for each order. Then the orders can be re-planned after determining the efficiency for each order and each work station. This leads to a maximal flexible situation, so that the possible free work station can work on the next partial work of the next order, avoiding down time for all work stations (similar works [GNYL15]). Obviously, a flexible industrial automation system is able to compensate a component fault via reassigning the works to other work stations or reassigning the

available order plans to compensate the fault effect of a defective work stations as well as a PLC station. However, these approaches are concentrated on the self-autonomy of an industrial automation systems via a specific designed methodology [BSP16+] or development technology [PAO+14] [Prie17]. But actually, in the actual factory, like a smartphone assembling factory, the entire industrial automation system is composed of many different small industrial automation systems from different companies to complete different specific works. A specific designed methodology is not fit for such a situation, as all machine supplier would have to use the same methodology, unless all work stations are designed by one company. The communication between them is through specific commands, which are defined by the work station supplier. In this situation, if one work station is broken due to a defective location sensor in the vertical direction of a robot, then the only rescheduling possibility is to isolate the broken work station and to reassign the orders without this work station. But it is possible that the broken work station can still provide some available functions in the horizontal direction. In addition, concerning the fault knowledge, the proposed concept considers only fault handling rather than a dynamic fault handling. That means, they have to analyze a reoccurring fault repeatedly. This leads to a waste of the fault knowledge. Due to no common fault knowledge base, the fault knowledge cannot be shared between similar industrial automation systems located in different factories.

The mentioned approaches to handle faults dynamically in industrial automation systems as well as the worldwide distributed work stations are concluded in the following Table 3.1 with regard to the in chapter 2 defined requirements.

Based on Table 3.1, each model-based approaches is discussed as followed. The model-based approach of robust controller is able to automatically compensate a parameter change fault which is brought by noise, disturbances, etc. However, the fluctuation shall be not over the bounds. Moreover, if any component, like a sensor, is completely out of order, the closed-loop control cannot be guaranteed anymore and the industrial automation system has to be stopped to avoid serious consequences. Due to a lack of a fault identification function, every fault has to be analyzed repeatedly as a new fault. Furthermore, because the reconfiguration of robust controllers requires a runtime feedback, a remote reconfiguration is hard to be realized due to the physical internet delay. A robust controller is based on very complex mathematical algorithms which requires very much effort, especially for a nonlinear industrial automation system. In addition, even if there are different tools to support the creation of robust controllers, a designed robust controller is very difficult to port to another industrial automation systems, even to the same system installed in a different environment, because a changed parameter or an added component can affect the whole mathematical robust controller.

The model-based approach of direct redundancy can compensate the fault effect when the defective component owns a backup redundancy. For these faults, the availability of industrial automation systems can be assured. The redundancies and their fault identification approaches are designed for the components in the development phase, but if a new fault occurs, the fault cannot

be reasonably analyzed and no measures exist for it. In addition, due to a lack of a central knowledge base, faults have to be handled repeatedly. Moreover, the reconfiguration mechanisms are integrated in the industrial automation systems and the activation of redundancy are generally activated locally. A remote reconfiguration is possible but most applications activate the redundancies via the industrial automation system or its fault diagnosis system. Direct physical redundancy requires not only extra effort as well as cost in the development phase, but also extra cost for maintaining the backup redundancy. A physical redundancy is always specific for an industrial automation system type. Along with the change of the application field, mechanical structure and layout, the same redundancy cannot be easily ported to different industrial automation systems.

Table 3.1: Comparison of the four presented model-based fault handling approaches with regard to the requirements

| | **Robust Controller** | **Direct Redundancy** | **Analytical Redundancy** | **Flexible Scheduling** |
|---|---|---|---|---|
| R1: Enhancement of the availability | ● | ● | ● | ● |
| R2: Automatic and dynamic fault analysis | ◐ | ◐ | ◐ | ◐ |
| R3: Remote Reconfiguration | ○ | ◐ | ◐ | ◐ |
| R4: Small costs | ◐ | ○ | ◐ | ○ |
| R5: Portability to heterogeneous industrial automation systems | ○ | ○ | ○ | ○ |

Where, ● means fulfilled, ◐ means partial fulfilled, and ○ not fulfilled.

Same as the direct redundancy, the model-based approach of analytical redundancy is able to compensate the fault effect when a virtual redundant component is automatically created for the defective component. However, though the concepts using analytical redundancy require no direct backup redundancy, they are based on an assumption that there exists information redundancy, that is, there are more similar components which can provide the same information or similar information. This assumption limits the application possibilities of the concept. With predefined

information sharing or calculating mechanisms, an automatic fault analysis can be performed. However, due to a lack of a central fault knowledge base, the faults have to be analyzed repeatedly. Moreover, these concepts are usually established on a specific technology, like multi-agents. The implementation of them depends on specific platforms and the communication costs for a lot of software agents are also very high in operation, because of a high message exchange.

The model-based approach of flexible scheduling compensates the fault effect via providing available functions as well as available orders in case of a faulty machine. This approach is usually not focused on one specific machine but on a higher level, like the entire production line, the entire factory or even the cooperation among different companies in the context of the Industry 4.0. But most concepts are based on an assumption that the whole manufacturing systems are implemented with their specific concepts as well as their specific technologies. They are difficult to be ported to different systems. The cost of implementing such a concept is very high, because the customer has to change the whole machines and they are not designed and implemented with a common conception and technology. However, the reality is, that the entire factory is made up of several different small individual industrial automation systems from different suppliers. They communicate with each other with specific commands, without knowing the detailed structure or behaviors of others. Besides, most researches of this approach are concentrated on the improvement of the whole industrial automation system itself. They just isolate the defective small industrial automation system directly, even if they can still provide some available functions.

### 3.1.6 Assessment of the Surveyed Methods

Using the defined requirements in the last chapter, four methods will be assessed as follows: Firstly, fault prevention is helpful in enhancing system availability by preventing a breakdown of the system [Bord16]. With the help of properly-defined estimation characteristic values, an automatic detection of fault and fault development can be developed and integrated in a fault diagnosis system. It generally supports the prevention of known system faults and cannot guarantee the avoidance of all software faults [Lyu07]. It fails to judge the algorithms for new faults. To prevent the breakdown in time, fault prevention requires real-time data from the industrial automation system. Because of the wide distribution of industrial automation systems, e.g. coffee makers sold by one manufacturer, it is hard to obtain real-time data with less cost in operation. With properly defined, specific algorithms, and estimation characteristic values for various systems, fault prevention can be smoothly ported for various industrial automation systems.

Fault tolerance can compensate the occurred system fault as well as component failures and eliminate the corresponding fault impacts to assure the whole functionality of an industrial automation system, until the defective component can be replaced. Availability can also be enhanced by this method. Fault-tolerant control with redundancy allows for automatic fault detection which can establish a reasonable process model and a reconfiguration with redundancy

methods to isolate the fault effect of a failed component. It enables the reconfiguration of available functions (e.g. activating cold standby redundancy) remotely via the internet. Because of the necessary redundancy, fault tolerance requires very high costs for development and implementation, especially for a large number of systems, as the costs can increase exponentially over time.

Fault removal depends on the help of additional maintenance staff to remove system faults, even though some specific technologies like a smartphone with a diagnosis application can minimize fault detection time and provoke a faster order of spare parts. That can have an effect on waiting time, if the breakdown of an industrial automation system cannot be avoided. Availability can thus only be enhanced to a certain extent. This approach can be ported for a heterogeneous industrial automation system, but requires a mass of skilled maintenance staff.

Fault forecasting can predict the number of faults and fault occurrences. By means of additional workers, predicted faults can be removed in advance. This can enhance the availability of an industrial automation system. Regarding the qualitative strategy, the system can be modeled by process models to estimate the system behavior by gathering data and generating necessary characteristic values, for instance, the prediction of the fault location. Concerning the quantitative strategy, the analysis process needs historical data to predict the next possible fault presence time. This approach doesn't take care of reconfiguration. Developing and running a conception of this approach requires much time for a complete and accurate prediction. Furthermore, it is hard to port the conception for different types of systems, due to the wide distribution of industrial automation systems.

In the last section, the assessment of the model-based approaches for fault handling has been introduced in detail. Based on the comparison of these approaches, a brief decision and their limitations are given as follows. This thesis tries to enhance the availability of an industrial automation system via providing still available functions and available tasks. But the robust controllers are not considered due to the physical internet delay in the remote mode. This thesis considers not only known faults but also new faults, which shall be detectable like changed specific parameters in specific individual industrial automation systems. With a central fault base, a new fault will be analyzed just once. But a high level fault effect compensation, like in the context of Industry 4.0 [WKS+17], is not considered. It can be regarded as an extension of the concept of this thesis, as it just requires more factors like runtime efficiency. Because it is very expensive and very difficult to complete a monitoring function for worldwide industrial automation systems, the communication with the integrated fault diagnosis system is very important in order to access empirical data as well as the current state of the industrial automation system. For a remote reconfiguration, this thesis shall consider using reconfiguration commands to activate the integrated reconfiguration mechanisms in industrial automation systems. So to reduce the implementation cost, the reasoning process for fault handling shall be a common process and can be implemented after the complement development of the industrial automation

system. The formulization of system knowledge shall be completed by experts, but a formulization tool is useable to reduce the porting difficulty.

As a result, fault handling in fault tolerance is a possible approach which meets all requirements. Thus, the formalization of system knowledge and the high cost for redundancy are challenges for the development of a conception of fault handling. Concerning the aims of this research, the conception can utilize fault detection approaches in fault prevention and fault tolerance, with the intention of detecting the fault location and establishing a reasonable system model for determining the fault impact on the industrial automation system. The system model can be formalized as the system knowledge, that is, the analysis knowledge of the experts or maintenance staff. With the proposed concept, the industrial automation system can prevent a system breakdown and reconfigure the logical system structure in the case of a component failure, as well as a system fault.

## 3.2 Survey of the Conceptions of System Modeling

As the last section mentioned, a proper system model is required in order to complete the fault detection and fault effect analysis. The aim of the system modeling is to establish a simplified representation of a real system via graphic or textual description.

Table 3.2: Classification of the conception of system modeling [Goll12]

| Existing Modeling concept | **Process-oriented system modeling concept** | **Data-oriented system modeling concept** | **State-oriented system modeling concept** | **Object-oriented system modeling concept** |
|---|---|---|---|---|
| Associated modeling approaches | Input- and output-modeling | Information structure oriented modeling | Discrete state and state transition modeling | Entity relationship modeling |
| | SADT method | Information flow oriented modeling | Continuous state and transition modeling | Object- and class-oriented modeling |
| | Function-oriented modeling | | Combination between state modeling and data-oriented modeling | |
| | Rule-oriented modeling | | Petri nets | |

However, along with the accuracy of the system modeling, the complexity of modeling will be greatly increased. This leads to a contradiction between the accuracy of the modeling and its comprehensibility: more accurate and harder to understand, less accurate and easier to understand.

Hence, it is very important to establish a proper system model along with the demands. Generally, there are four modeling conceptions for industrial automation systems in Table 3.2. The four approaches are: process-oriented system modeling, data-oriented system modeling, state-oriented system modeling, and object-oriented system modeling [Goll12]. Different corresponding approaches are listed in the table. One of them, object- and class- oriented modeling approach, is extended as the proposed modeling approach for describing industrial automation systems. And the consideration process is discussed as follows.

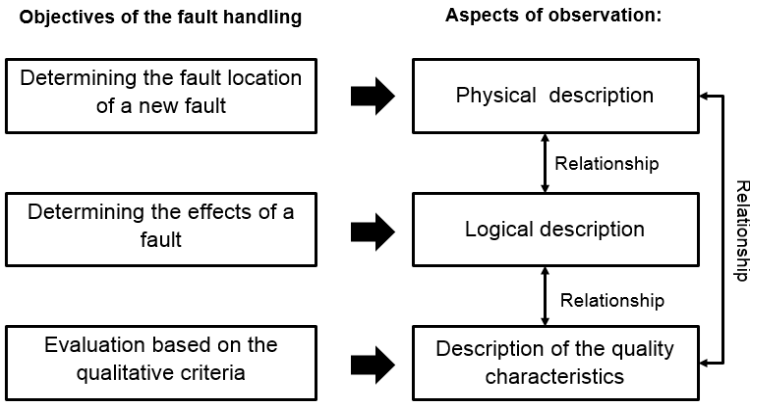



Figure 3.8: Representation perspective of an industrial automation system

Faults are divided into known faults and new faults. The establishment of the system model is aimed at handling a new fault. In order to identify the available function in case of a fault, the following three functionalities ought to be considered: determining the fault location of a new fault to know where the fault is; determining the effects of a fault to know which functions are affected due to the defective function(s) and which functions are still functionally available; and then an evaluation based on the qualitative criterion to identify which non-affected functions fulfilled the predefined criterion.

Determining the fault location of a new fault requires finding out which component(s) is/are defective. As mentioned above, the component is located in the physical structure of an industrial automation system. Hence, determining the fault location can be interpreted as the identification of defective parts in the physical structure, i.e. a ***physical description*** is required in the modeling of an industrial automation system.

Moreover, determining the effects of a fault refers to the identification of the direct consequence of the defective component on system functions – the term of available functions is included in the objective of this research. Thus, to determine the affected and not affected functions, a ***logical description*** with respect to the **relationships** between components and functions and the relationship between functions ought to be denoted in the system model.

An evaluation based on the qualitative criterion is supposed to estimate whether unaffected functions can be performed under the condition of satisfying the qualitative constraints which are defined in the development phase. The requirements of an industrial automation system that

denotes the nonfunctional constraints ought to be highlighted. Hence, a ***description of the quality characteristics*** is required in the system model.

As a result, it is worth mentioning that the four aspects, i.e., physical description, logical description, relationship, and quality characteristics, ought to be considered for modeling an industrial automation system in order to realize the required functionality in this research, namely determining the fault location and determining fault effect, as shown in Figure 3.8.

### 3.2.1 Process-oriented System Modeling

Process-oriented system modeling takes running processes in industrial automation systems as objectives to establish system models, in order to complete deductive or constructive models. To build a process-oriented system model, four major types are required: input- and output-models, modeling based on SADT method, function-oriented model, and rule-oriented model.

Concerning the input-output-model, behavior of a system is described through the relationship between input and output in the system, in other words, input variables and output variables, utilizing the mathematic functions to describe the relationship. The input and output variables refer to time-dependent parameters, such as temperature, pressure, flow rate, etc. Figure 3.9 outlines a basic concept for modeling a dynamic technical system. The input and output variables are described respectively: x1(t), x2(t), x3(t) … xn(t), and y1(t), y2(t), y3(t) … yn(t). Based on these variables, a concrete mathematical function can be developed and transformed via Laplace's approach [Iser05]. [ThJa10] proposed a process model based approach for detecting and predicting faults in nonlinear multiple-input-multiple-output discrete-time systems.

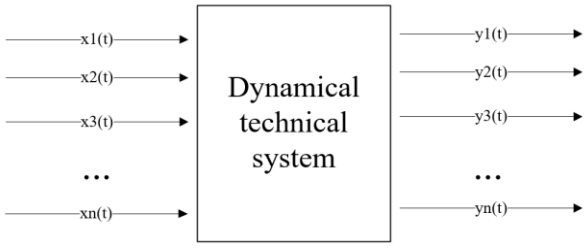



Figure 3.9: Modeling a dynamical technical system via Input-output-model [Bequ03]

The structured analysis and design technique (SADT) method refers to an approach for requirement analysis in the developmental phase rather than as a modeling conception. As in Figure 3.9, it also consists of inputs, including setup value and measured value, outputs, including output value, and in the middle are the control algorithms. One of the most important roles of the SADT method is to build a clear hierarchical structure via step-by-step refinement [DHJ17].

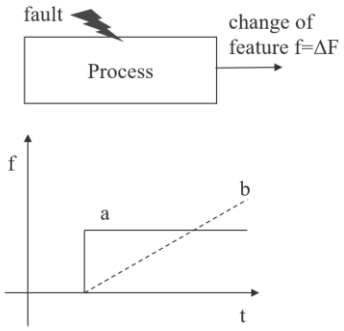



Figure 3.10: Time dependency of faults in processes [Iser05]

Function oriented models (also named behavior oriented models) describes functions that can be performed, and under which conditions and in which sequence [BRU00]. If the function describes the execution of an algorithm via the computational unit, the sequence of executing the function is named control flow. The structure of a function oriented model consists of functions and conditions. If one function has more than one sub-function in a fixed order, every sub-function can possess only one input and one output. Behaviors of a system can be clearly described with graphs. This is also helpful for the development phase to catch the main control flow of the entire system.

Rule oriented modeling can be considered as an extension of function oriented models. This modeling approach describes the sequential lineal control structure with the graphic and textual methods. Regarding the formalization of rules, a series of “if-then-rules” is implemented to represent the system, for instance, “If condition is fulfilled, then function1, else function2”.

Many have carried out research studies on the application of the process model for fault handling. Generally, based on the process model, the industrial automation system can be divided into various processes and signals. Faults can be detected by means of a process model, fault modeling, parameter estimation, and observer techniques. Figure 3.10 shows an example of detecting time-dependency faults using a process model [Iser05]. Likewise, the rule oriented model enables the estimation of the fault location of the functions via the function-condition-relationship. On the basis of proper algorithms in process models, the approach to fault control reconfiguration can overcome, or compensate for, the impact of interference [YJSZ15].

### 3.2.2 Data-oriented System Modeling

Data-oriented system modeling (also named: product-oriented system modeling) describes the information and information process in an industrial automation system instead of the input-output-process, such as the storing and retrieving of packs in a high-bay warehouse. Data-oriented system modeling can be classified by data structure and data flow oriented modeling.

Data structure modeling supposes modeling the presented data in an industrial automation system and deploys a hierarchical structure to all data and their relationships. Three main relationships

are used for describing the relationship: sequence, selection, and iteration. For example, a measured parameter in high abstract level consists of temperature and pressure in the low abstract level.

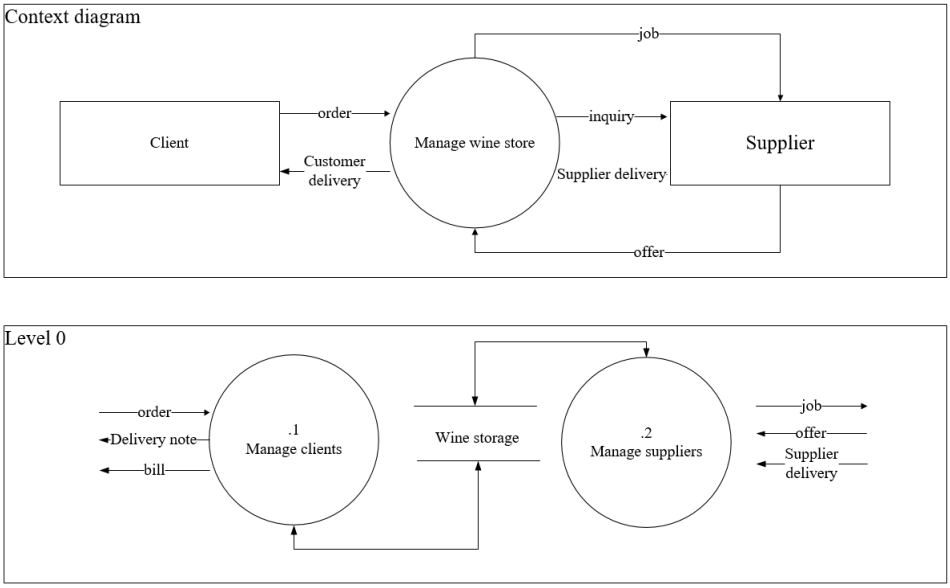



Figure 3.11: Example of data flow diagram for a wine store management system

Another widely used approach is the information as well as data flow oriented modeling concept. This concept describes a system with four terms, namely: data flow, data flow transformation (also can use “function”), memory for storing necessary information, and terminator for external information source. With these terms, the internal data exchange of an industrial automation system can be clearly defined. Generally, this conception uses the data flow diagram to describe the system graphically. Figure 3.11 indicates an example of the data flow diagram for a wine store management system. The context diagram shows all terminators, the entire system, and the necessary data between the terminator and system. It is worth mentioning that there is no data exchange between terminators. This depends on the hierarchical structure of the functions; the system represented with functions and data will be decomposed further. In the example, there are two necessary functions: managing clients and managing suppliers. Data will be also decomposed, such as client delivery data with the delivery note and bill data. Memory stores all data about wine. Additionally, minimal specs are required for specifying the input and output data.

The data-oriented modeling approach is usually used for a structural analysis of a system in the development phase. With regard to fault detection and fault impact analysis, the developer is able to verify the developed system structure and system requirements in line with the developmental approach of a V-Model [LaOv11] with the intention of determining the fault location and impact. This is so that the developer can remove faults in the development phase in advance.

### 3.2.3 State-oriented System Modeling

State-oriented system modeling (also named service oriented system modeling) depicts states and a transition among states of industrial automation systems from the internal logical perspective [Bell08]. This conception is able to describe discrete states and a state transition, and dynamic behaviors of a time-continuous or time-discrete system. As Table 3.2 shows, there are four major types of state-oriented system modeling, i.e., modeling via discrete states and states transition, modeling for a time-continuous or time-discrete system, modeling via combination with data flow oriented modeling conception, and modeling with a Petri net.

Modeling via discrete states and a state transition attempts to decompose the activities of a system into various discrete states with various time points, and the transition between states is influenced by events. It provides a graphical method (state chart model) and mathematical method (finite state machine) to represent the states. Figure 3.12 outlines two modeling methods with examples. Figure 3.12a shows an example of the state chart model of an elevator. It contains two states, namely "elevator stops" and "elevator running". With different events, different actions can be activated to complete the transition between them. Figure 3.12b indicates a general finite state machine in a mathematical format, in which x indicates the state in time t, u indicates the input data in time t, and y indicates the output data in time t. Based on these, a mathematical function can be created, i.e. $x(t) = \delta\big(u(t), y(t)\big)$. In addition, in the real application, a decision table can also be utilized for representing a state-oriented model.

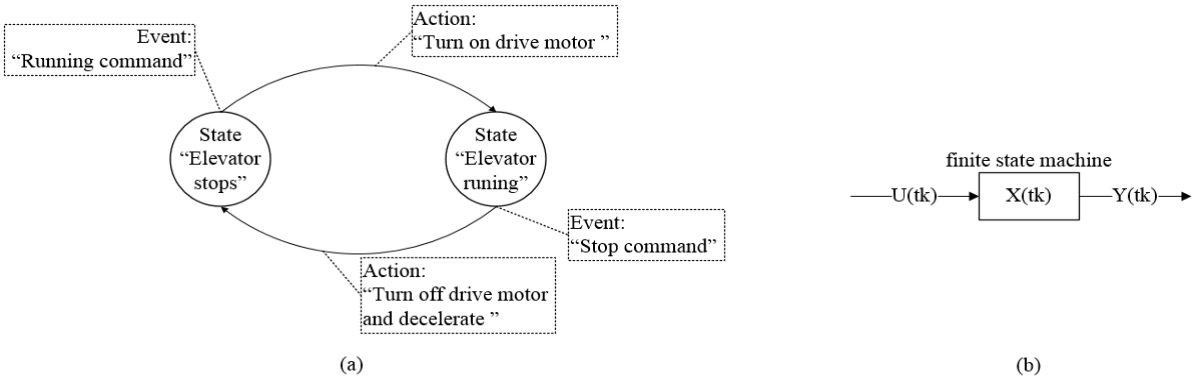



Figure 3.12: Examples of state chart model and finite state machine

Similarly, time-continuous or time discrete system dynamic behaviors can be modeled through a state chart model. In contrast to only one input and one output functions in modeling via discrete states and states transition, a set of input variables, a set of state variables and a set of output variables are presented with the vectors, specifically input vector, state vector, and output vector, for example, *u(t) = [u1(t), u2(t), ... u3(t)]*. To apply them, a recurrence relation (also named: difference equation) method is required.

The combination with the data flow oriented modeling approach is usually utilized for graphical representation of the design, since the two modeling approaches have almost the same graphic

diagrams. In the combination modeling approach, the states are utilized as the input instead of data in the data flow diagram.

A Petri net can be considered as an extension of the state chart model, in which two additional features are added; in other words, a transition between two states to represent the action in Figure 3.12a, and a point in the cycle of state to represent the current state. In addition to the change of graphical description, the mathematical representation is also altered. A Petri net can be described with a 6-tuple, i.e. P = (S, T, F, K, W, M0), where S is a finite set of places, T is a finite set of transitions, F are the backward and forward incidence functions, K is the capacity of S, W is the valuation of F, and M0 is the initial marking [SMBG02].

In general, service-oriented modeling is used for service-oriented architecture (SOA) of software products as well as IT systems to provide services [Yang06]. Referring to the functional correctness of SOA, plenty of service oriented research regarding fault detection and diagnosis has been carried out. [AlBo09] proposed a model-based conception for monitoring the execution of events in SOA and detecting the appearance of a fault online. Likewise, Hanemann has suggested a hybrid architecture that includes a rule-based reasoning module and a case-based reasoning module via the service-oriented event correlation to identify resource failures which are used for impact analysis [Hane06] [HSS05]. In [CLT04], a method for modeling intermittent faults and their resets in the context of discrete event system models has been introduced. Similarly, Cabasino presented an approach for diagnosing fault events and regular unobservable events via labeled Petri nets [CGS14].

According the popular software architecture AUTOSAR, in [SZW17], Schmidt proposed a model-based and service-oriented architecture to describe the industrial automation systems with different predefined functional modules. In this concept, an industrial automation system is divided into three levels, a basic software level to communicate with the hardware directly, a real-time communication level, and an application level. The functions as well as their inputs, outputs and communications are defined in the application level. This concept allows the development of software with a specific template, which can be used by Matlab in order to create runnable codes directly without the dependence on hardware IO ports [SZW16].

In addition, Klein proposed a cloud-based and service-oriented e-Production system to establish ad-hoc networks of industrial automation systems to produce individual products in the context of Internet of Things and Services, and smart products [KJW17]. This concept does not only consider the industrial automation system itself but also the orders from customers, the design of the product, the scheduling of the entire production line, the logistics of the production and even the delivery of the final product to the customer.

Due to the fault impact, the system cannot fulfill the original requirements, which can be considered as the change of requirements. To meet the change requirements, Rastogi has presented a quality of services (QoS) based approach for multiple faults in SOA [RSS16]. On the

basis of specific QoS values, services in SOA can be reconfigured by means of the concept of finding the optimally shortest path from source to destination. Concerning web services (WS), Tsai supposed a services-oriented dynamic reconfiguration framework for dependable distributed computing [TFCP04].

### 3.2.4 Object-oriented System Modeling

Object-oriented system modeling, or objected-oriented modeling, is an approach used for modeling systems by utilizing the object-oriented paradigm during the entire development phase. Object-oriented modeling allows the modeling of dynamic behaviors and also of static structures. The object-oriented system modeling conception is derived from the entity-relationship-modeling conception, which contains three important features to represent a system: real units with "entities", relations between these units with "relationship", and properties of the relationship with "attributes". This modeling conception can describe the internal static structure of all units in an industrial automation system in detail.

Object-oriented system modeling conception has been applied and extended for a long period of time in three fields of activity: object-oriented programming, object-oriented design, and object-oriented analysis. In order to analytically describe a system, design and implementation as required, several diagrams are necessary to represent the system structure and system behaviors: For example, objects (classes) for the description of components, functions, requirements, interaction between objects for control flow, state flow. The most popular language for object-oriented system modeling is the Unified Modeling Language (UML), which attempts to afford a standard way to visualize the design of a system [ViTr17]. Seven diagrams can be created to describe the structure: the class, component, composite structure, deployment, object, package, and profile diagrams., Seven diagram types are available to describe the behaviors of a system: the activity, communication, interaction overview, sequence, state, timing, and use case diagrams [RuQu12].

As an extension based on the UML 2.0 [KiCh13], the SysML (Systems Modeling Language) was created as a continuation and expansion of systems engineering applications. SysML contains nine diagram types. Concerning behavior, sequence, state machine and use case diagrams, they are the same as in the UML, but the activity diagram is modified. Regarding the structure and package diagrams, they are the same as in UML, whereas the block definition and internal block diagrams are modified, and a new "parametric diagram" is added. Additionally, a requirement diagram is added on the same level as the behavior diagram and the structure diagram [Tolk12]. This allows an industrial automation system to be abstracted from three perspectives: requirements, structure, and behaviors. Obviously, SysML, with three views, can better represent an industrial automation system [KeVo13].

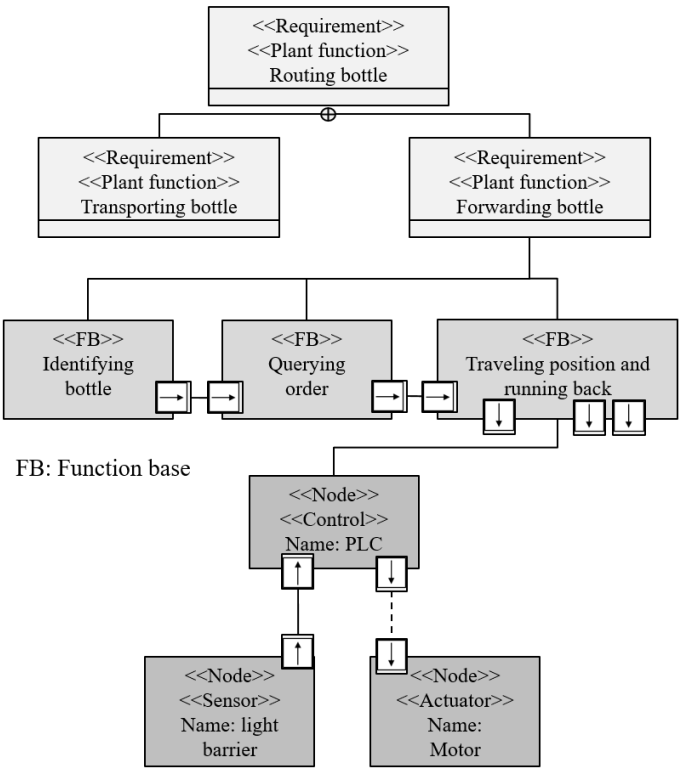



Figure 3.13: An example of object-oriented modeling based on SysML [FSV13] [KeVo13]

[FSV13] proposed, based on the application of SysML as well as object-oriented modeling technique in industrial automation systems, to represent an industrial automation system from four views: the process view, software view, hardware view and deployment view. This thesis supposed that the process view could be modeled via the piping and instrumentation diagram (P&ID). Furthermore, the requirements described are the same as those in SysML and are included in the process view. The software view describes the functional behaviors of the industrial automation system using functions named function bases. The hardware view models controllers, sensors, and actuators in the industrial automation system. An example that models a technical production system is depicted in Figure 3.13. In this thesis, functional and nonfunctional requirements are described as boundary conditions.

Considering the relationship of product, process and resources, the guideline VDI/VDE proposes a formalized process description for the modeling of processes based on this relationship. Marks proposed five categories to describe such an industrial automation system [MHWF18]. They are component-based parameter space, structure-based parameter space, process-specific parameter space, software-based parameter space and feasible parameter space. Here, the parameter spaces represent resources, processes and products [HMWF17].

In [WZS+16] Weyrich proposed a new object-oriented smart components concept for modeling an industrial automation system in the context of IoT networks. In this concept, smart components are able to independently coordinate with each other to realize the production. With a loose coupling of these components, using the mentioned service-based architecture [SZW16], a high

flexibility of the configuration of IT-systems for the production can be realized in the future [Weyr18].

With respect to fault detection and fault diagnosis, Huang et al. presented a diagnosis application case for a vehicle infotainment system. Based on fault symptoms and object oriented model structures, the fault location can be localized as subsystems, functions, or operations [HMDJ08]. Furthermore, to determine the fault impact, Kurtoglu attempted a functional-failure identification and propagation framework to estimate potential faults and their propagation paths with a graphical schema [KuTu08]. With a confirmed fault or failure impact, the industrial automation system is able to reconfigure its functions based on specific techniques, such as those that are multi-agent based, with the intention of compensating for the fault impact. For instance, in [FSV13], the function of a defective sensor can be replaced by the function of other still available sensors to enhance the availability.

### 3.2.5 Assessment of the Surveyed Modeling Methods

To describe an industrial automation system properly, this subsection compares the introduced system modeling conceptions by means of reasonable criteria. The first four criteria describe the completeness of describing an industrial automation system as presented in Subsection 3.2.

Additionally, concerning the requirements of lower cost, less complexity, and automatic reasoning, three additional requirements have been added:

- ***Managing the complexity*** of the interior of an industrial automation system: It requires that the modeling techniques describe the internal four aspects of an industrial automation system in a clear and simple way.
- ***Possibility of implementation*** of the system modeling conception: It means that the established models by the system modeling conception can support the implementation in either a mathematical or semantic method rather than only in graphic method. This is because the first two methods enable the assistance of automatic reasoning for decision making instead of depending on persons.
- ***Lower cost and high efficiency*** in establishing models: Based on the last two requirements, this requirement demands lower cost and high efficiency in establishing a system model, which is the needed model which ought to be created as soon as possible in a cost effective manner.

Table 3.3 shows the comparison of the four system modeling conceptions presented with seven criteria, where the notation used indicates if the criterion is completely satisfied (++), satisfied (+), partially satisfied (0), or not adequately satisfied or not considered (-).

As depicted in Table 3.3, process-oriented modeling can significantly describe the physical process, modeling the control flow in the process including the sensor, actuator, controller and their input variables, as well as output variables. Moreover, with the defined mathematical functions (control algorithms), the logical description can also be satisfied. But the requirements of the system are not included. In the process model, the mathematical functions process the variables of sensors, actuators, and controllers in the industrial automation system. Hence, the relationship between physical and logical descriptions is satisfied. With a clear mapping relationship, complexity regarding a relationship can be controlled clearly, but complexity is very high due to a multitude of input variables, output variables and their transition with mathematical functions. Hence, the requirement of managing complexity by process-oriented modeling is partially fulfilled. By means of mathematical transition approaches, process models can be simply implemented as mathematical functions, which can further be implemented in a program smoothly. Nevertheless, mathematical functions require much human power and time to be created to establish process models. In addition, adjusting the accuracy of the functions is also a hard and time-consuming task, which means lower cost and high efficiency cannot be satisfied.

Table 3.3: Comparison of four presented system modeling methods with seven criteria

| | **Process-oriented system modeling** | **Data-oriented system modeling** | **State-oriented system modeling** | **Object-oriented system modeling** |
|---|---|---|---|---|
| Physical description | + | - | + | ++ |
| Logical description | + | 0 | ++ | ++ |
| Quality characteristics | - | - | - | ++ |
| Relationship | + | - | ++ | ++ |
| Managing complexity | 0 | + | 0 | 0 |
| Possibility of implementation | + | - | ++ | + |
| Less cost and high efficiency | - | ++ | ++ | 0 |

Data-oriented system modeling does not refer to a physical description, but rather to the logical information structure and information flow diagram (specifically the data flow diagram). The data flow diagram provides a good hierarchical structure of functions and data via the context diagram, and further data flow diagrams in decomposed levels. However, it only gives general data

structure and data exchange. Furthermore, the quality characteristics are also not considered in the data flow diagram. Due to the missing physical description and quality characteristics, the relationships between different models are not considered. Since the data flow diagram is a single diagram for determining the data exchange with different functions, the complexity can be satisfied. The data flow diagram is only created in the form of a graphic diagram. The implementation of mathematical functions and semantic methods are not satisfied because of the simplicity of the data flow diagram. At the same time, the simplicity allows the cost and efficiency criteria to be satisfied optimally.

State-oriented system modeling conception can describe parts of the system architecture, that is, the owner of each service and, thereby, the owner structure following the hierarchical structure of the services. The requirement of the physical description can be satisfied. The service-oriented model can represent the behaviors of the interior of an industrial automation system in detail with its states, actions and events. Thus, the logical description can be completely satisfied, whereas the quality characteristics are neither considered in the diagram nor in the models. Due to the mapping relation between the physical description with the owner of services and logical description with services, the relation can be satisfied. However, when the system is a time-continuous system and possesses plenty of mathematical functions and input variables as well as output variables, this can lead to an infinite number of states for every time point, meaning the complexity is partially satisfied. As introduced in Subsection 3.2.4, this allows the transformation of graphic diagrams into mathematical functions. Currently, , service-oriented modeling, such as modeling with Matlab [SZW17], can be performed quickly with less cost by means of the development of various tools. The requirement “possibility of implementation” of the state-oriented modeling conception can be completely satisfied. Furthermore, lower cost and high efficiency can also be completely fulfilled via a properly defined state model and the necessary tools.

The object-oriented system modeling approach refers to the development phase, from requirement analysis to design and implementation phase. With the modeling techniques, as well as diagrams, the internal structure and behaviors of an industrial automation system can be completely described using physical and logical descriptions. As introduced in [FSV13], the requirements, including functional and nonfunctional, are included in the SysML modeling method. This allows the requirement of description of quality characteristics to also be completely fulfilled. Similarly, the relationship between different diagrams can be clearly defined. Since there are 14 diagrams in UML and 9 diagrams in SysML, the number of relationships and coupling between various diagrams is very high. This evaluation is a general assessment. Nevertheless, some researchers have also attempted to simplify the original SysML and describe the system from four views with a unified notation [FSV13]. In this case, the complexity and relationship are managed in a limited scope. Hence, the requirement of managing complexity is partially satisfied. Depending on the field of the applications, different diagrams can be formalized with semantic methods [Lano09]

[FSV13], such as ontologies [KhPo15]. The object-oriented diagrams can be partially realized with tools, which can increase the efficiency of the implementation. However, more manpower is required to formalize the models in a semantic format. Hence, the requirement of lower cost and high efficiency is partially satisfied.

In summary, there is no system modeling conception which can satisfy all requirements. The object-oriented modeling, especially the proposed system modeling approach based on SysML in [FSV13], however, can satisfy the great number of proposed criteria. The formalization of the system model to system knowledge has to be considered in the establishment of a new system model instead of only in the ontology to adopt the different industrial automation systems and running platforms. In addition, the manufacturing capability of machines have to be considered under four perspectives: set of different operations, parameter range of different operations, set of feasible sequences of operations, and range of output quantity [HMWF17] [VoNe17]. These characteristics should also be checked in the system model, like the function model is used to describe operations, their sequences and also the requirements for these operations.

A survey regarding the methods of handling faults and conceptions of system modeling was presented in this chapter. In the first part, four methods of handling faults, specifically fault prevention, fault tolerance, fault removal, and fault forecasting, were depicted in detail. Fault tolerance and fault removal are supposed to remove either the fault or fault effect, and they are two approaches of corrective maintenance. The other two methods, fault prevention and fault forecasting, are attempted to eliminate faults before the appearance of the faults via monitoring abnormities or by predicting the presence of the next fault, including fault location and fault consequence. In line with requirements of the proposing conception, fault tolerance is the one of four methods of handling faults that can fulfill almost all requirements. In the approaches of fault tolerance, fault handling was proposed with four steps: fault diagnosis, fault isolation, system reconfiguration, and system re-initialization. In the second part, four conceptions of system modeling (namely process-oriented, data-oriented, state-oriented, and object-oriented system modeling) were introduced in detail. Different approaches are supposed to represent various aspects of an industrial automation system, e.g. dynamic behaviors, static structures, and requirements. Then these four approaches were compared using seven criteria. On the basis of the comparison table, the object-oriented approach can fulfill most of the set criteria, in which the proposed system modeling approach based on SysML in [FSV13] represents the system from the perspective of requirements, functions and components. Hence, it provides a very good basis for establishing a system model with the intention of determining fault location and fault impact. The proposed system model will be introduced in the next chapter.

# 4 Modeling of Industrial Automation Systems

This chapter shows the decisions for the overall design of the fault handling conception in consideration of the system requirements, and how to establish the system model of an industrial automation system as system knowledge. With regard to the overall system design, basic decisions will be introduced with respect to primary fault diagnosis, the method of handling a new fault, the automatic reasoning, the knowledge base, the execution of the reconfiguration, and system knowledge. Additionally, as the basis of identifying the fault location and determining the fault effect, the system model will aim to describe an industrial automation system in a proper way.

## 4.1 Representation of an Industrial Automation System from three Perspectives

This section describes a system modeling method of an industrial automation system from three perspectives, e.g., components, functions and requirements [FSV13]. As the requirement in Chapter 2.4 mentioned, the requirement “Ability of automatic, reasonable and dynamic fault analysis” needs a reasonable description of the industrial automation system, so that a computational unit is able to determine the fault location and available functions. Automatic analysis depends intensively on input data from the fault diagnosis system and a systematic reasoning process. This is because it can derive the available functions from the interior structure of the industrial automation system in case of the appearance of a fault. To attain the correct fault location and the precise fault effect, the worker who performs the reasoning must not only be extremely familiar with the system’s physical structure, but also the logical process structure. Here, there needs to be a comprehensive system knowledge, including the physical structure, logical structure, and some possible specific constraints for automatic analysis, that is, automatic reasoning. However, establishing a system model is a common method for representing or describing an industrial automation system from different perspectives. In addition, a correct and reasonable format of the system model also plays an increasingly important role for automatic reasoning. Later, if the automatic reasoning can be realized on a computer, it would also reduce the cost during the operation phase.

Therefore, two main preparation tasks for the automatic analysis ought to be completed before establishing the fault analysis conception: Firstly, a rational system model from at least physical and logical views ought to be established to provide integral knowledge for the proposed reasoning. Secondly, it is essential to formulate the system model into a proper format, for instance rules and matrices, so that the analysis process can be achieved by the computer rather than a human being via the formulated knowledge. Hence, to design a proper system model, the model should be also suitable for the demands of the fault analysis process. In addition, different kinds of industrial automation systems should also be considered in the process of defining and creating

the system model so that the universality of fault handling for industrial automation systems can be validated. At the same time, the interface between the industrial automation system and the fault handling system should be clearly and uniformly defined to achieve the information exchange to be able to achieve compatibility with the existing industrial automation system.

As discussed in the previous chapter, the goal of this research is to guide an automated system to reconfigure itself within a certain range, with the help of available functions when the automation system fails due to a component failure. Therefore, the industrial automation system must be able to continue to work and still provide available services for users. However, if an automated system follows a wrong guide, not only are the desired objectives not achieved, but also the wrong guide can result and a further fault impact expansion, and it even can bring the whole system to its knees, like a secondary failure with a system level short circuit. Hence, fully and clearly evaluating the fault impact scope in the inner system structure is a very important prerequisite for the decision-making. Otherwise, some necessary constraints, like safety, ought to be considered to evaluate the availability of every function. Since a system model empowers the user (here, a user can be a computer system) to filter out unnecessary internal complexities of an automated system, the user is able to directly access to the extremely important parts of the industrial automation system.

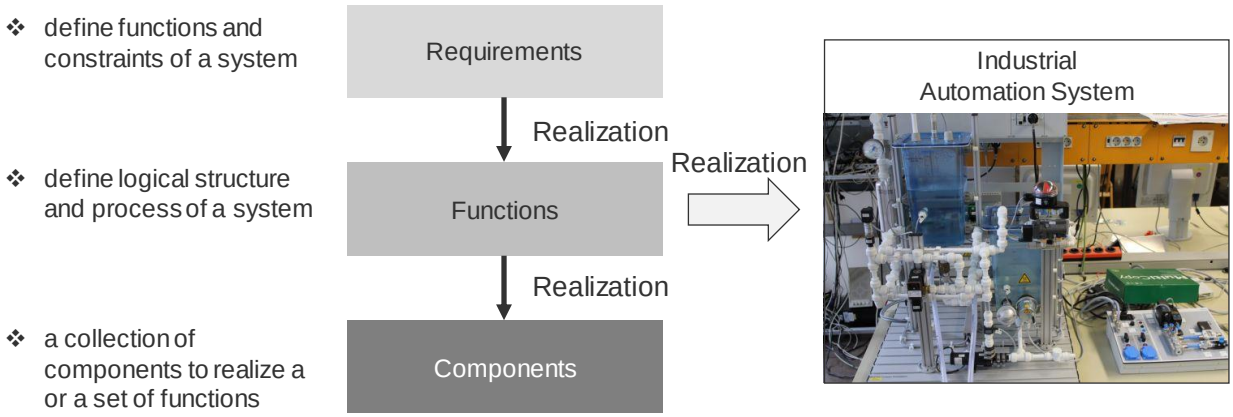



Figure 4.1: Development of an industrial automation system via components, functions and requirements

Figure 4.1 shows the general development process of an industrial automation system. After coordination with the customer, the manufacture formulates the requirements with respect to functional requirements, which indicate the concrete functions or services of an industrial automation system, and non-functional requirements that indicate the constraints or quality attributes for availability, performance, reliability, etc. Secondly, the manufacture analyzes, customarily in the system design phase, the requirements to determine the possible functionalities, and then decomposes these into further small unit functions with necessary information inputs and outputs. To achieve the information exchange, the information flow between functions has to be defined. Generally, after the system's logical design, the system physical structure is supposed to define the functional subsystems and the connection interfaces as well as the reasonable

components that can realize various unit functions of each subsystem. Furthermore, according to the designed architecture, interfaces, and information, the developer is able to convert the logical description into a concrete system, that is, the realization of the system with components and subsystems. The developer is therefore able to realize the defined functions from hardware and software. Here, each component means either the concrete hardware, such as a temperature sensor, or the encapsulated software, which can realize the function of measuring temperature. Afterwards, via system integration and system test, an industrial automation system, such as a coffee maker, can be realized.

As mentioned above, an industrial automation system can be identified in three major views: physical view (component model), logical view (function model), qualitative requirements view (requirement model, dependencies or constrains between functions), and the mapping relationship among different views.

### 4.1.1 Physical Description (Component Model)

The component model outlines the physical objects of an industrial automation system, as an object-oriented (as well as assembly-oriented) decomposition. Firstly, the component model describes all the physical elements, components and their connections, of an industrial automation system [7], such as a microcontroller, sensor, and an actuator. In addition, a system includes different subsystems, each subsystem being connected by different components to achieve some specific tasks.

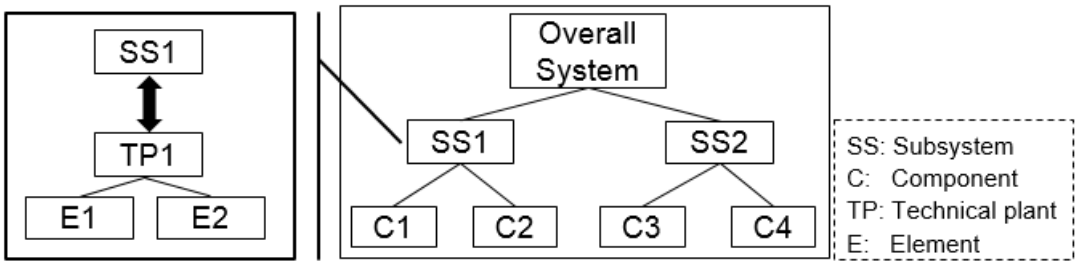



Figure 4.2: Schema of the component model

Figure 4.2 presents a schema of the proposed component model. The right tree shows an overall system decomposed into a hierarchical structure with two subsystems and four components. The left box shows the relationship of a subsystem with a technical plant [ZNM18] and two elements. The far-right dashed-line box illustrates the abbreviation of the subsystem, component, technical plant, and element.

- Element: It means that all physical entities in a technical plant are influenced by the technical system and directly transforming or changing the material, information and energy [DIN 66201], for instance, a part of a water pipe, a tank, a part of an electric wire, etc. Here, because of the complex mechanism in the machine field, it is hard to determine which element is defective without the help of manual intervention. Similarly, due to complex mechanical

engineering, the affected component of an industrial automation system is also hard to determine. Hence, the element fault is considered as a technical plant fault, i.e., its related subsystem fails, and ought to be confirmed by the maintenance service via a field investigation.

- Component: It describes the entities of an automation system, including the controller, the sensor, and the actuator. A controller is located in the core of a subsystem to provide the desired system responses between components. An actuator influences the behavior of the physical system. Some examples of actuators are motors, pumps, valves, and switches. A sensor measures different properties of the technical plant, such as temperatures, pressures, positions, liquid levels, etc.

- Subsystem: It consists of a certain number of components which form a special configuration to realize one or more functions. In addition, a subsystem is usually divided according to either the component distribution or the function's influences. However, every subsystem must contain at least one controller.

- Overall system: Here, it means the overall industrial automation system—which is made up of all subsystems.

- Component-tree (i.e. Hierarchy): It presents the hierarchical structure of all the features in an industrial automation system, including components, subsystems and the overall system (see feature model [BaHa10] [WXH+10]). In addition, the proposed component model tries to number each level from bottom to top, from small to large: Figure 4.5 shows an example, the component level being level 1, the subsystem level, level 2, and the overall system level, level 3.

Afterwards, as an additional attribute, subsystems have different specific symptoms, namely that subsystem failure can result in one or more specific symptoms, which are also a part of the fault diagnosis result.

Figure 4.3 shows a general description of the component, as well as the subsystem, on the left side: parameters, hierarchy and relationship with the function model. The upper right tree illustrates the hierarchical structure of the component model. The lower right frame gives an example of a component, in this case the temperature sensor, with the general attributes: general information, parameters, connected function, and connected subsystem. In the presented system model, components can only be sensors, actuators or microcontrollers.

- Sensor: It measures the physical values of the technical plant, converts them into electrical variables, and transfers the variables to the computational unit, e.g. a temperature sensor.

- Actuator: It receives commands from the computational unit and influences the technical plant, e.g. a heater.

- Computational unit: carries out the tasks that are assigned by the user. The input value (i.e. set value) will be processed with one or more specific algorithms and sent to the actuators, e.g. a microcontroller.

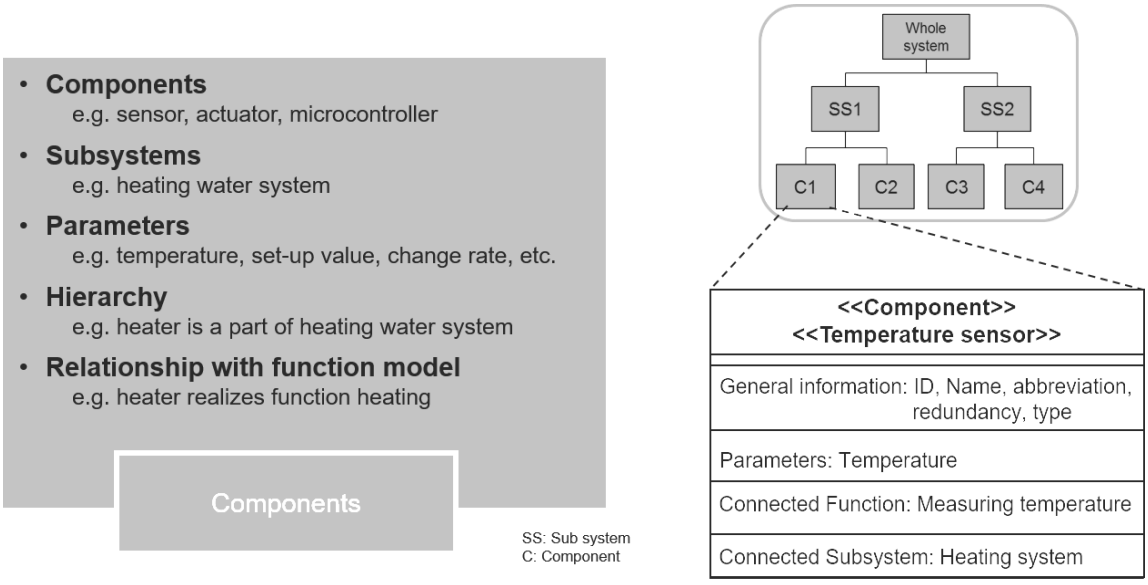



Figure 4.3: Attributes of each feature in the component model, and an example of a temperature sensor

As Figure 4.3 shows, some attributes of a temperature sensor is illustrated. The details of each attribute are presented as follows:

- ID: It indicates the identification number of each unit in the component model, such as the ID of the temperature sensor, which is 1.
- Name: It gives the name of the component, e.g. temperature sensor.
- Abbreviation: It gives each unit in the component model one or more letters and a number, e.g. the abbreviation of a temperature sensor is “C1”, for a heating subsystem it is “SS1”.
- Type: It indicates the type of the described item. It can be a temperature sensor, a heater, a heating subsystem, etc.
- Redundancy: It shows whether the described item is a redundancy of another item, e.g. there are two temperature sensors. One of them is working and the other one is on standby. When the working temperature sensor is out of order, the redundant temperature sensor can replace the defective sensor to provide the required service.
- Parameter: It denotes all the parameters of each component or each subsystem. It could be a certain parameter such as a measured temperature value of a temperature sensor. It can also be the input data and the output data of a subsystem as well as the predefined data, which is the user’s set value.

- Connected functions: It means that functions can be realized by the component or system and indicate the mapping relationship with the function model. For instance, a temperature sensor realizes the function of “measuring the temperature”. A heating subsystem realizes the function of “Heating water to X°C”, where X is the value of the temperature and can be defined by the user.

- Connected subsystems: It shows ownership in the component model. There are two types of ownership: For the higher level, it means that the connected subsystems possess the described component or subsystem. For the lower level, it means connected subsystems belong to the described subsystem.

Afterwards, as an additional attribute, subsystems have different specific symptoms, which correspond to the mentioned parameter, e.g. the tendency of a temperature. A subsystem fault can result in one or more specific symptoms, which can also be identified by the existing integrated fault diagnosis system in the industrial automation system as a part of the fault diagnosis result, providing the principle for the fault localization. With the help of symptoms, it is able to fix the defective area in the system, for instance, the defective subsystem.

### 4.1.2 Logical Description (Function Model)

The function model describes all functions (activities, actions, processes and operations) from the logical standpoint, and their connections to an automation system [HMWF17]. With the help of these functions, an automation system can produce plenty of products or realize various services, for example, producing different kinds of coffees, transporting specific bins to some specific slots, etc.

However, a single process usually cannot realize a final service or manufacture a final product by itself in an industrial automation system. The production processes of an overall system enable the processing of different raw materials, to obtain the majority of intermediate products, and finally to manufacture the final product. Based on this production procedure, the functions of an industrial automation system can be divided into basic functions, sub functions and main functions. Certainly, there can be more than one sub function level between the basic and main function levels, such as sub-1 function, sub-2 function … sub-n function.

- Each basic function represents the corresponding component with its behavior pattern. A basic function provides the original material or information, such as providing the cold water in a coffee maker.

- Similarly, each sub function is the mapping of its corresponding subsystem with one of its behaviors. A sub function can supply an intermediate product through the connection of some basic function, such as producing hot water after heating the cold water. Here the sub function

"Producing hot water" is the cooperation of basic functions "Providing cold water" and "Heating water" in a certain order.

- The main function represents one of the final products or services that can be provided by the overall system, and are usually directly required by the user. For instance, a cappuccino is one of the final products of an automated coffee maker. However, a main function could also be a part of the other main function.

In addition, it is necessary to illustrate that the final product required by the user can also be provided by a sub function, for instance, the espresso can be a demand by the customer and can also be the material for the cappuccino product. Hence, it ought to be noted that the division of functions follows the basic principle of correspondence with the component model. The division of functions is flexible and can also be defined in light of the demand of the developer.

Figure 4.4 illustrates the schema of the function model. As the left grey frame shows, there are some characteristics in the function model: the function types, the relationship between functions, the hierarchical structure, and the relationship to the component model and requirement model. The function types are already interpreted above. The functions can be divided into two major types:

- Function relationship regarding the automation system: It shows the relationship of the function according to the internal information flow of an industrial automation system. The internal information flow denotes the inherent information transmission sequence to realize the basic functionality successfully. For instance, a temperature sensor with a function "measuring the temperature" sends the measured temperature to the microcontroller with a function "controlling the temperature". This information flow consists of a dependency: the function "controlling the temperature" depends on the function "measuring the temperature".

- Function relationship regarding the technical process: Besides the inherent internal dependency, there is another logical functional dependency in line with the technical process. As noted above, a function aims to perform an activity or a process. Along with the application of a function model in the field of system engineering, a function in the function model depicts the objective of the transformation of the material, the information, or the energy. In accordance with this, the division of functions can be subdivided into material, information, or energy. Hence, it can generate another logical dependency higher than the inherent functional relationship. With this logical dependency, functions follow a defined combination and perform a sequence to produce a specific product or provide a specific service, which is the ultimate goal of the user. For example, there are three functions of a coffee maker: the function "producing cappuccino" needs the function "producing espresso" and the function "producing milk foam". Apart from the dependency that the first function depends on the realization of the other two functions, there is an additional potential dependency, so that without the function "producing espresso", the significance of the function "producing milk

foam” no longer exists. This dependency of functions is hidden in the function performing sequence.

- Hierarchy (function tree): It presents a hierarchical structure of different functions. Depending on the relationship of the functions, they are divided into different levels and each level numbered from bottom to top, from small to large, for instance, as Figure 4.7 shows, the basic function level is level 1, the sub function level, level 2, and the main function level, level 3.

- Relationship with the component model: It provides the relationship between the function model and the component model, i.e., the described functions are realized by the corresponding components, subsystems and the overall system. As an assumption, each basic function can be mapped to a component. The computational unit is, however, a special component and can be called upon by different subsystems. The computational unit can have only one, but also more control tasks. Hence, the computational unit can afford more than one basic control function. Otherwise, either a subsystem or the overall system can also provide more than one function that depend on the system’s logical design.

- Relationship with the requirement model: It shows the relationship between the function model and the requirement model. This means that the described function has a relationship with the (qualitative) requirement. This function can be the condition of the requirement, through which the requirement can be fulfilled, and can also be the consequence of whether the function can be fulfilled under the condition of the fulfillment of the requirement.

As Figure 4.4 shows, the function tree and an example of a basic function “measuring temperature” can be interpreted as follows:

- ID: It indicates the identification number of each function in the function model, such as the ID of function “measuring temperature” is 1.

- Name: It gives the name of a function, e.g. “measuring temperature”.

- Abbreviation: It gives each function in the function model one or more letters and a number, e.g. a basic function “measuring temperature” is “BF1”, a sub function “Heating water to X°C” is “SF1”.

- Type: It indicates the type of the described item. It can be a basic function, a sub function, a main function, etc.

- Commands: It means the reconfiguration command which can control the availability of the function in the logical level. It can be considered as the program function in the software with a flag variable. For instance, the function “measuring temperature” has a flag “Flag_Measure_Temp”. If the “Flag_Measure_Temp = True”, then the program function of “measuring temperature” can execute, in other words, the function “measuring temperature” is activated. Conversely, if the “Flag_Measure_Temp = False”, then the program function of

"measuring temperature" can no longer be executed, i.e. the function "measuring temperature" is deactivated. In addition, if this function is a redundant function, not only the activation command, but also a part of specific program, are needed for a reconfiguration in certain circumstances.

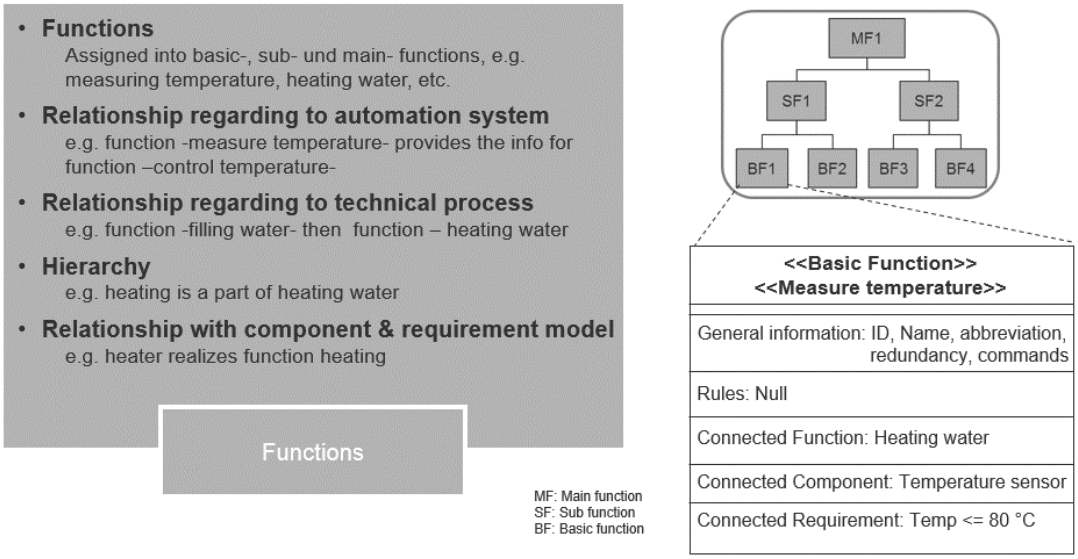



Figure 4.4: Schema of the function model

- Redundancy: It shows that whether the described item is a redundancy of another function.
- Rules: It denotes the relationship with the connected functions that can influence the availability of the described function; with the same example of a coffee maker: *availability of "producing cappuccino" = "producing espresso" AND "producing milk foam"*. Obviously, if the availability of two functions behind the equation is known, then the availability of the function "producing cappuccino" can be easily calculated. Therefore, a general rule to estimate the availability of a function can be concluded: **If the calculating result of the described function is *TRUE*, then the described function is available**. **Conversely, if the calculating result of the described function is *FALSE*, then the described function is unavailable.**
- Connected functions: It means that the described function has the relationship with the other functions. It can be the function across different levels: for example, the function "measuring temperature" being connected with the function "heating water" in the logical relationship.
- Connected components: It means that the described function can be realized by a component or system in the component mode: for instance, the function "measuring temperature" is realized by the temperature sensor.
- Connected requirements: It denotes that the described function has the relationship with a requirement. This function can be a tenable condition or a tenable consequence of the requirement.

- Parameter: It denotes the parameter of the function. It could be a certain parameter, such as a measured temperature value. It can be the input data and the output data, as well as the predefined data of the function, for instance, the function “Heating water to X°C”, where X is the value of the temperature and can be defined by the user.

### 4.1.3 Description of Qualitative Requirements (Requirement Model)

As previously mentioned regarding the general development procedure, a requirement can be realized by one or more functions. The functional requirements are actually implemented in the function model. It is unnecessary to describe the functional dependency again in the requirement model. But the non-functional requirements, which describe the qualitative constraints of the system, ought to be described in the requirement model. Hence, in the following chapters, the term requirement is used instead of the term non-functional requirement.

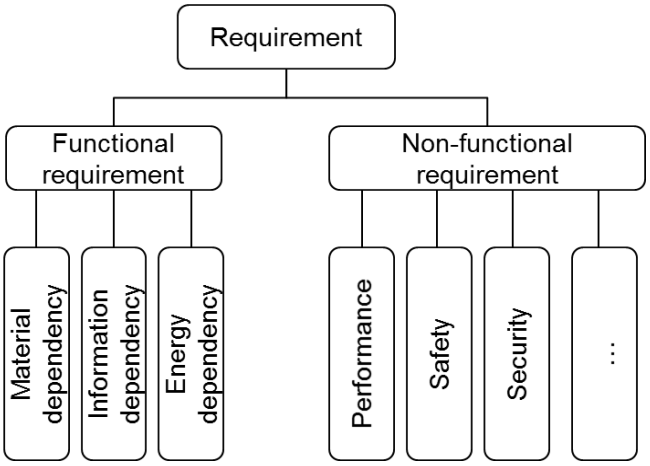



Figure 4.5: Classification of the requirements in functional und non-functional types

The requirement model gives a specific description of qualitative constrains for functions and components (see Figure 4.5): the requirement model here does not consider the functional requirements, which are concluded in the function model, but non-functional requirements. The [Part12] has defined 13 non-functional requirements: performance, interface, operational, resource, verification, acceptance, documentation, security, portability, quality, reliability, maintainability, and safety requirements. As mentioned in the literature [Part12], three non-functional requirements are clarified as follows:

- Security: The inoperative function and the not-affected functions should not threaten the privacy data and operation. When a memory that stores the system’s root password is defective, not all functions can be activated until the repair addresses the security of the privacy of the user.

- Safety (Survivability): The rest of the functions must not threaten the safety of system as well as of the user. For instance, the pressure in a bottling plant must be validated within a safety range to avoid a potential hazard.

- Performance: A specific requirement in [Part12] concerns response time, speed, volume, rates, etc. According to the performance requirement, the result of the function should be defined in a specific scope for meeting the user's demands.

Beside the mentioned non-functional requirements, specific user requirements ought to be included in the requirement model.

Afterwards, an additional requirement about the main component, which refers to a subsystem or the entire system, and plays a key role as the core element, such as a microcontroller, should be included. If the main component is defective, its subsystem or the entire system can be out of order. This is helpful in decreasing the handling process. The functions' analysis can begin directly from the subsystem level instead of the component level.

- Non-functional requirement: The details of the non-functional requirements are described above.

- Promise condition: It defines the estimation logic of a requirement. With the help of this promise condition, it enables either the fulfillment of the higher level requirements or the consequence of the functions to be estimated. Generally, in an **IF-THEN** logic, in order to perform a reasonable reasoning, the primary judgment is to determine the IF statement. Depending on this result, the THEN consequence can be fixed.

- Consequence: It denotes which actions ought to be activated corresponding to the analysis result of the promise condition. For instance, if the system safety is true, then the function "heating water" can be performed, or else the function "heating water" cannot be activated.

- Hierarchy: It presents a hierarchical structure of different requirements. Different from the component model and function model, the highest level of the requirement tree is the entire system requirement that consists of all different requirements. The second level shows every specific requirement, such as safety, security, etc. Then these specific requirements are decomposed into more sub requirements in the requirement tree. In addition, the requirement tree is numbered at each level from bottom to top according to the principle of small to large, e.g. the basic requirement level is level 1, the sub requirement level is level 2, and the system requirement level is level 3.

- Relationship to the function model: It shows the relationship between the requirement model and the function model. It means that the described requirement has relationship to different functions. These functions can be either the condition or the consequence of this requirement.

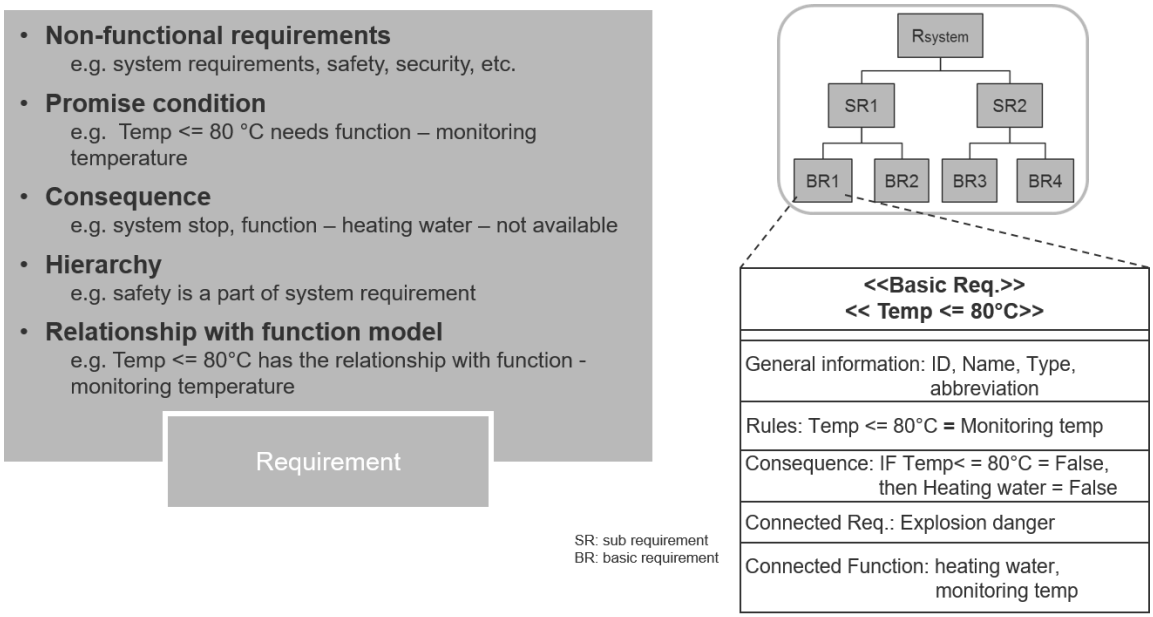



Figure 4.6: Schema of the requirement model

Figure 4.6 shows the schema of the requirement model, the conceptual hierarchy, and an example of a basic requirement “Temp <= 80°C”. The first two are already interpreted above. The details of the example are described as follows:

- ID: It indicates the identification number of each requirement in the requirement model, such as the ID of the requirement “Temp <= 80°C” is 1.
- Name: It gives the name of a requirement, e.g. “Temp <= 80°C”.
- Abbreviation: It gives each requirement in the requirement model one or more letters and a number, for example, a basic requirement “Temp <= 80°C” is “BR1”, a sub requirement “Safety of the system” is “SR1”, the entire system requirement is “ESR1”.
- Type: It shows the type of the described requirement such as entire system requirement, sub requirement, basic requirement.
- Rule (Promise condition): It provides a formal conditional statement that describes the relationship with the corresponding functions or the other requirements in a mathematical equation. The described requirement is located on the left side of the mathematical equation and the corresponding functions or requirements described on the right side. With the help of the calculation of the right side, it enables the determination of the value of the described requirement. The rule will later be used in the IF side of the IF-THEN logic. For instance, “Temp<=80°C” = “Monitoring temp”, it means that if the function “Monitoring temp” is available with the value of TRUE, then the requirement “Temp<=80°C” is available with the value TRUE. Otherwise, if the function “Monitoring temp” is not available with the value of FALSE, then the requirement “Temp<=80°C” is also not available with the value FALSE.

- Consequence of the described requirement: It defines different actions according to the decision results of the promised condition, i.e., the consequence is the left side, thus THEN of the IF-THEN logic. In general, there are two possibilities in this thesis, either TRUE or FALSE. If the result of the promise condition is TRUE, then the limited corresponding functions fulfill the requirements and are available; otherwise, if it is FALSE, then the limited corresponding functions are not available.
- Connected requirements: It denotes that the described requirement has the relationship with the other requirements.
- Connected functions: It means that the described requirement has the relationship with functions.
- Connected components (optional): It indicates the principal components of a system. The failure of one principal component can lead directly to a breakdown of one or more subsystems and even stop the entire industrial automation system.

This section described the categories of the non-functional requirements, as well as the major aspects of modeling the non-functional requirements of an industrial automation system. Moreover, the concrete details of the attributes of a non-functional requirement together with specific examples were also indicated above.

## 4.2 Formalization of the System Model via Matrices and Rules as System Knowledge

This section gives the formalization principle of the system model via matrices, which are used to describe simple relationships, and via rules, which are used to describe complex relationships and utilize them as the system knowledge, which is utilized for identifying the fault location and the available function in case of a fault. In this thesis, a simple relationship denotes the relationship between two items, and a complex relationship describes a relationship that has three or more items. If the state of an item is decided by two other items, a 3-dimensional matrix will be proposed rather than a 2-dimensional one. Along with the increase in the number of items, the dimension of the matrix will increase, so that not only will the difficulty of describing the relationships increase, but also the difficulty of future operations. To avoid a high dimensional matrix, this thesis proposes to utilize a rule to describe a relationship type of more than three items.

### 4.2.1 Formalization of Simple Relations via Matrices

A matrix can be used to describe the relationship of m rows and n columns of various items. In traditional math applications, each position in the matrix can be used with different values. In addition, different operations can be performed on matrices, if particular rules are met. Depending

on the application conditions, a matrix can be utilized with particular limitations to meet the purpose of computing or reasoning, like a 0/1 matrix, i.e., the value of each position in the matrix is either 0 or 1.

There are three typical interactions between two features in Figure 4.7:

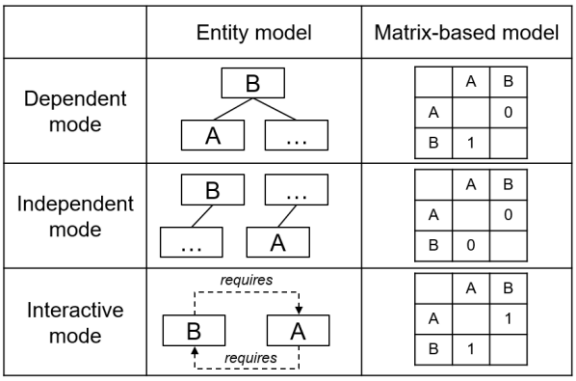



Figure 4.7: Three interactions between two features

- Dependent mode: One item depends on, or is part of, another item, e.g. A is part of B or B depends on A, then the value of the position BA is 1. For instance, component A is part of subsystem B.
- Independent mode: It denotes that there is no relationship between the two items, e.g., the position AB and BA are set to 0, as shown in the figure.
- Interactive mode: It means that two items depend on each other, e.g., as the above figure shows, position AB and BA are set to 1. For instance, a temperature sensor in the component model performs the function measuring the temperature in the function model. In this case, the interactive mode can be used to describe the mapping relationship between different models.

These basic modes can be used to describe the relationship between any two items in the system model. Nevertheless, in the above description, a confusion between the dependent mode and the interactive mode can occur, if both A depends on B and B depends on A and, therefore, AB and BA are 1. Hence, in order to better distinguish relationships and serve reasoning, a **one-way matrix** will be applied in this thesis, i.e. either the row of the relying item or the high-level item will be set to 1.

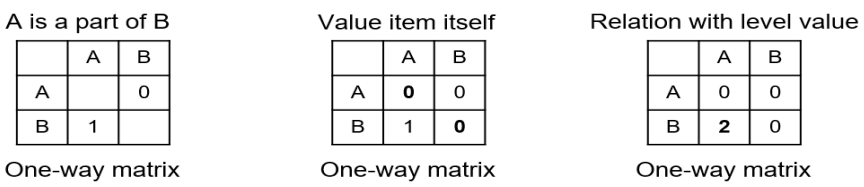



Figure 4.8: Specific rule example for the application of the matrix

Afterwards, to simplify the reasoning cost, the relationship of the item to itself will be set to 0.

In addition, for the complex relationship where an item can be decided by more than 2 items, simply whether a relationship exists can be described by a simple matrix, i.e., 1 with relation, or 0 without relation. In addition, different values are applied to replace the simple 1 to better mark the locating level of the described item. The rule is that the lowest level is 1, is the second level is 2, up to the top level N. There is no level 0, as 0 indicates that there is no relationship. For instance, in the function tree, the function “heating water” SF1 requires the function “heating” BF1, and the function “heating water” SF1 is one level higher than the function “heating” BF1. Then, in the matrix description, the position SF1BF1 will be set to 2, and BF1SF1 will be set to 0.

Typically, there are six relations for internal relationships and relationships across models. The first three matrices are used to formalize the relationship across models, and the remaining three matrices depict the simple internal relationships of each model. In addition, a matrix of redundancies has to be established.

- Matrix between components and functions: It describes the mapping relationship between components and functions. It includes the independent mode and the interactive mode. There are thus usually two values: 0 indicates no relationship, and 1 indicates a relationship.

- Matrix between functions and requirements: It describes the mapping relationship between functions and requirement. It includes the independent mode and interactive mode. So there are usually two values: 0 indicates no relationship, and 1 indicates a relationship.

- Matrix between components and requirements: It shows the mapping relationship between components and requirement. It includes the independent mode and the interactive mode. There are thus usually two values: 0 without relationship, and 1 with relationship. In addition, this is only used to depict the principle components with the requirements. If the principle components are defective, then one or more subsystems, or even the entire system, can be out of order. With the help of this matrix, the reasoning process of identifying available functions can be simplified.

- Internal matrix of components: It depicts the simple owner-member relationship between components and subsystems, subsystems and systems, as well as elements and subsystems. It includes the dependent mode, independent mode, and interactive mode. In addition, it is also a one-way matrix. The first two relationships show relations inside of the component model. The relationship between elements and subsystems indicates the necessary mechanical parts of the technical process which do not belong to the can-controlled automation system, but could impact the workflow of a part or the entire system.

- Internal matrix of functions: It shows the simple owner-member and influence relationship between functions. It includes not only the three modes, but also the one-way matrix and relations with the level value. As Figure 4.8 shows, the bigger the number, the higher the level in the function tree. In addition, not only can the relationship’s different levels be indicated,

but also the relationship on the same level. For instance, the function “controlling heating water” demands the function “measuring the temperature” to keep the water at a concrete temperature. They are, however, at the same level, and parts of the sub function “heating water”. Hence, such a control flow relationship is also indicated in this matrix.

- Internal matrix of requirements: It depicts the simple owner-member and influence relationship between requirements. It includes the three modes and is a one-way matrix and relations with the level value. However, it is necessary to note that the level of the requirement is not the same as that of the functions. Since the highest level in the requirement model is system requirements, and then system requirements are decomposed into various specific non-functional requirements, and then those specific non-functional requirements are further decomposed into concrete requirements, which are generally realized by specific functions.

- Redundancy matrix: It indicates the redundancy relationships in an industrial automation system of the redundant components, but also those of the redundant functions. This is because, in the real industrial automation system, the utilization of redundancies is still a very common approach to insure the normal working ability of an industrial automation system when a fault occurs. Hence, in this thesis, redundancy has to be considered to replace functions as well as the defective components. This matrix is specific: one column shows the original components or functions, and another column shows the redundant components or function.

In addition, it is important to define the application of the term *component*, which can denote the component itself, for instance, a sensor, and can also denote any item in the component model, for example, a subsystem. To avoid misunderstanding in the following sections, this thesis utilizes the term *basic component* to indicate the first meaning, i.e. the component itself.

### 4.2.2 Formalization of Complex Relations via Rules

A set of mathematical formulas equivalent to the system model is used to describe the logical structure of the industrial automation system. There are four typical relations, in general, among the three features in the feature model [BMC05] [WNW15]:

- **Mandatory**: It means that a junior feature is required by the superior feature in a feature tree. Formalized in a semantic format, the state of the superior feature is equal to the state of the junior feature, namely, *SUPER_Feature = JUNIOR_Feature*. When there is more than one junior feature, the state of the superior feature is equal to the state of the junior features in term of a logical conjunction, i.e. *SUPER_Feature = JUNIOR_Feature1* ***AND*** *JUNIOR_Feature2*.

- **Or-relation**: It depicts a superior feature that requires at least one of its junior features. Formalized in a semantic format, the state of superior feature is equal to the state of junior features in term of a logical disjunction: SUPER_*Feature = JUNIOR_Feature1* ***OR*** *JUNIOR_Feature2*.

- **Alternative**: It means that the superior feature requires exactly one of the junior features, i.e. it does not allow the activity of two junior features at the same time. Formalized in a semantic format, the state of the superior feature is equal to the state of junior features in terms of an exclusive disjunction: SUPER_*Feature = JUNIOR_Feature1* ***XOR*** *JUNIOR_Feature2*. To develop a new product in the development process, it is necessary to indicate the relation of the alternative for the system design and system implementation. However, for a developed product, if a single junior feature cannot be realized, it will not influence the execution of the superior feature. This is because the alternative relation in this thesis can be simplified as the or-relation, namely, SUPER_*Feature = JUNIOR_Feature1* ***OR*** *JUNIOR_Feature2*.

- **Optional**: It shows that the superior feature can be performed either with a junior feature or without the junior feature. That is, the optional junior feature exerts no influence on the superior feature. Due to this, this relation will be not considered in this thesis.

These four relations are indicated in one feature tree branch. Actually, the relations exist across tree branches and across levels. For example, a function of an air pressure switch requires the function of an air supply system. Hence, a new relation is introduced to describe the relationship of the cross tree:

- **Require**: It means that one feature requires one or more features of the tree. In the case of one required feature, in order to formalize it with the semantic format, the state of one feature is equal to the state of the required feature, that is, *Feature1 = Feature2*. If there is more than one feature, in order to formalize it, the state of the feature is equal to the state of the required features in term of a logical conjunction or a logical disjunction, that is, *Feature1 = Feature2* ***AND*** *Feature 3*; *Feature1 = Feature2* ***OR*** *Feature3*.

For the sake of judging the availability of each function and each requirement, the rule for functions and requirements should be formulated:

**Principle of building up function rules**

As discussed in Chapter 4.1.2, there are three function relationships: function constitution relationship, function relationship regarding the automation system, and function relationship regarding the technical process. In addition, this includes the function tree of each branch and across branches. The principle of building function rules is defined as follows:

**Objective_Function = Functions in the branch AND Functions across branches**, i.e.

**Objective_Function = (Consitution_Functions_In_One_Branch AND**
**Infomation_Constraint_Functions_In_One_Branch) AND**
**(Information_Constraint_Functions_Across_Branches AND**
**Material_Constraint_Functions_Across_Branches AND**
**Energy_Constraint_Functions_Across_Branches)**

With the help of this principle, it is possible to build a rule for each function in the function model. For instance, there are three functions of a coffee maker: the function MF1 “producing cappuccino” needs the function SF1 “producing espresso” and the function SF2 “producing milk foam”. And these two required functions are combined functions. Then, the rule for the function “producing cappuccino” is: *MF1 = SF1* ***AND*** *SF2*.

For the utilization of the function rule, this thesis assumes that there are two states exclusively: true and false. Obviously, if the value of the state is true, then the analyzed function is available. Conversely, if the value of the state is false, then the analyzed function is unavailable.

**Principle of building up the requirement rule**

As discussed in Chapter 4.1.3, there are two major requirement relationships: relationships with functions, and relationships with other requirements. Considering the requirement tree of one branch and across branches, the principle of building up requirement rules is defined as follows:

**Objective_Requirement = Functions AND Other requirements**, i.e.

**Objective_Requirement = (Functions) AND (Requirements_In_One_Branch AND Requirements_Across_Branches)**

With help of this principle, it is possible to build a rule for each requirement in the requirement model. For instance, there is a requirement BR1 “Temp <= 80°C” and a demanded function SF2 “Monitoring temp”. Then, the rule for the requirement “Temp <= 80°C” is: *BR1 = SF2*.

For the utilization of the requirement rule, this thesis assumes that there are a total of two states for each requirement: true and false. Obviously, if the value of the state is true, then the analyzed requirement is available. Consequently, along with the predefined consequence for the true-state, the corresponding actions ought to be performed. Conversely, if the value of the state is false, then the analyzed requirement is unavailable. Similarly, with the help of the predefined consequence for the false-state, the related function will not be executed, for instance, some corresponding functions should be not performed anymore and the state of these functions should be set to false, even keeping the stop state of the entire system due to a safety reason.

In summary, this chapter has proposed a system model to describe an industrial automation system and the formalization of the proposed system model via matrices and rules as system knowledge. Firstly, the establishment of a specific system model was proposed from three perspectives, i.e. physical description with component model, logical description with the function model and quality characteristics with the requirement model. For each model, its specific attributes with a concrete simplified example were introduced. Differently to the normal UML notation, the proposed model utilizes a tree structure to represent various features in the model. Finally, the formalization of the system via matrices and rules were presented to establish the system

knowledge, which can easily and directly be implemented on a computation platform such as a computer. Simple relations are represented via matrices among components, functions, requirements and themselves. For complex relations, specific rules for all functions are represented. Principles for building function rules and requirement rules were also outlined.

# 5 Conception of Dynamic Fault Handling and Reconfiguration

An industrial automation system can be represented by the system model from three perspectives, namely: the component model, function model, and requirement model. With the purpose of utilizing the system model as the system knowledge, the system model is formalized by various matrices and several rules. In this chapter, the conception of the dynamic fault handling and the reconfiguration based on knowledge will be presented. Firstly, this chapter introduces the conception and different kinds of fault handling knowledge. Consequently, the approaches of handling known faults and new faults will be presented. Finally, in approaching the handling of new faults, three major operations with respect to fault localization will be indicated, thereby identifying the available functions and the process of the reconfiguration. Through this chapter, the concept of dynamic fault handling and the automated reconfiguration of industrial automation systems can be observed in detail.

A missing or inadequate empirical foundation is an often-occurring problem in scientific work [PRK12] [ScHv18]. Despite the choice of a topic of practical interest, the relevance of scientific work with regard to practice is often inadequate, so that ultimately the scientific findings are not applied in practice [ScHv18] [PRK12]. To solve this problem **Design Science Research** was developed.

Design Science Research is a recognized method for gaining knowledge of the information system [Trep15] [PRK12] and provides an iterative approach to both the scientific foundations (rigor) and the practical relevance (through empirical studies) of the research results [ScHv18]. The Design Science Research sees itself as a construction-oriented method that focuses on the creation and evaluation of IT artifacts [ScHv18] [HeCh10] [GrHe13]. Its cyclical and iterative structure makes it flexible and enables the consideration of short-term changes as well as new insights. The main idea is, to develop an artifact and then to evaluate the developed artifact with empirical data, to then improve the artifact in a new cycle of the development of the artifact. The definition of an artifact is given as a construct, model, method or an instantiation. By those iteration, a constant verification of the concept by empirical data is guaranteed, which improves the quality and the practical relevance of the concept [PRK12].

Therefore, the concept of this dissertation was developed by using the Design Science Research and the required empirical investigations.

The final objective of this research is to realize a dynamic fault handling and reconfiguration concept for various industrial automation systems. Following the approach of the Design Science Research the research process can be divided into three phases for the *concept development* and the *concept evaluation* based on a reasonable demonstrator. In the first phase, a simple concept

for the dynamic fault handling and reconfiguration was developed via a simple relation matrix of components, functions and requirements. It was the start point for gathering scientific results. To identify available functions, known fault locations for new faults were analyzed. In order to better analyze, whether the developed concept is feasible, a coffee maker simulator was introduced as the object of practice testing. By manually adding the fault component in the background, the fault handling system can infer the available functions using the simple relationship matrix, and then activate the available functions of the coffee machine. Through the establishment of a background database of the fault information, the fault handling system can read the available functions from the database for known faults. After verifying the feasibility of the developed concept, the dynamic fault handling and reconfiguration concept was expanded in two ways. They were to construct a more detailed system model, to describe the industrial automation system from the perspective of components, functions and requirements, and to construct a reasonable inference machine, with the purpose of identifying available functions. This inference machine contains not only the reasoning logic, but also the required logic resources, integrated in the dynamic fault handling and reconfiguration system. In addition to the implementation on the coffee maker, this concept was implemented on the newly introduced high-bay warehouse system, which includes more components, like more than 50 sensors. This verified the possibility of porting the fault handling concept on diverse, complex systems. At the same time, in the new test demonstrator, a more complex monomial matrix and therefore more complex rules were introduced for the individual functions, to analyze the influence of other functions by the defective functions and to evaluate the influence on basis of their materials, information and energy relationship. After testing, the developed dynamic fault handling and reconfiguration concept can easily infer the available functions of the coffee maker and the high-bay warehouse as well as create the reconfiguration commands for available functions. Since the fault location of the new fault may be unknown, and therefore to identify the fault location became an important functionality for the third development stage for the concept. In addition, taking into account the diversity of industrial automation systems, an inference machine for a particular industrial automation system cannot be well ported to a new industrial automation system, so in the third stage of the concept development, an individual inference machine with separated knowledge, including fault knowledge and system knowledge, was required. In order to verify the feasibility of the developed concept, whether it can better infer the fault location of the industrial automation system and can be transferred to the new industrial automation systems, in the third stage, a new test object, namely the two-tank system, is introduced. For the new system, a new fault generation approach was brought from the perspective of variables via changing the parameters during the operation of the two-tank system by fault injection, so that the dynamic fault handling and reconfiguration concept can be tested, if any arbitrary fault can be identified and handled. Meanwhile, in the two-tank system, the required knowledge was formulated in a specific format and stored in the database instead of being implanted into the inference engine. In this way, it can achieve the liberation of the inference machine and can be more efficient to transfer the concept to diverse industrial automation systems.

This porting ability was evaluated by another two demonstrators after the implementation of the coffee maker and the high-bay warehouse.

## 5.1 Overview of the Conception of Dynamic Fault Handling and Reconfiguration

This section gives an overview of the conception of dynamic fault handling and automated reconfiguration. With the help of this conception, the complexity of the fault handling process can be controlled via the automated fault handling, rather than the manual fault diagnosis. Consequently, the fault can be quickly handled and the downtime of industrial automation system could be reduced. The general objective of the research is to increase the availability of the industrial automation system by means of performing the still available functions in case of a fault. To realize this goal, three functionalities can be outlined as follows:

- Identifying available functions of either known faults or unknown faults: The fault handling system is able to classify faults which occur into categories of known and unknown and to provide available functions. For known faults, it can directly provide the available functions, as well as reconfiguration commands. In addition, for a new fault, it enables the determination of the fault location, the available functions, and the integration of the reconfiguration commands.

- Automatic and remote handling of faults: it ought to perform the fault identification, fault localization and address fault effect, namely the affected functions, automatically.

- Guiding the reconfiguration for the industrial automation system: it ought to evaluate the reconfiguration types, integrate the reconfiguration commands and determine the related necessary measures to perform the reconfiguration successfully.

To realize the above the functionalities, an outline of handling the fault in an industrial automation system is given (see Fig. 5.1). This conception of dynamic fault handling and reconfiguration consists of the following main modules: fault pretreatment, handling known faults, handling new faults, reconfiguration, and a knowledge base.

Industrial automation system: this is an individual system that has its own inherent control module, execution module and data collection module. It can independently complete certain functions and services required by the operator.

Fault diagnosis system: this system is usually integrated in the industrial automation system. The fault diagnosis system has to monitor a system, gather process data, identify the appearance of a fault, and determine the fault location.

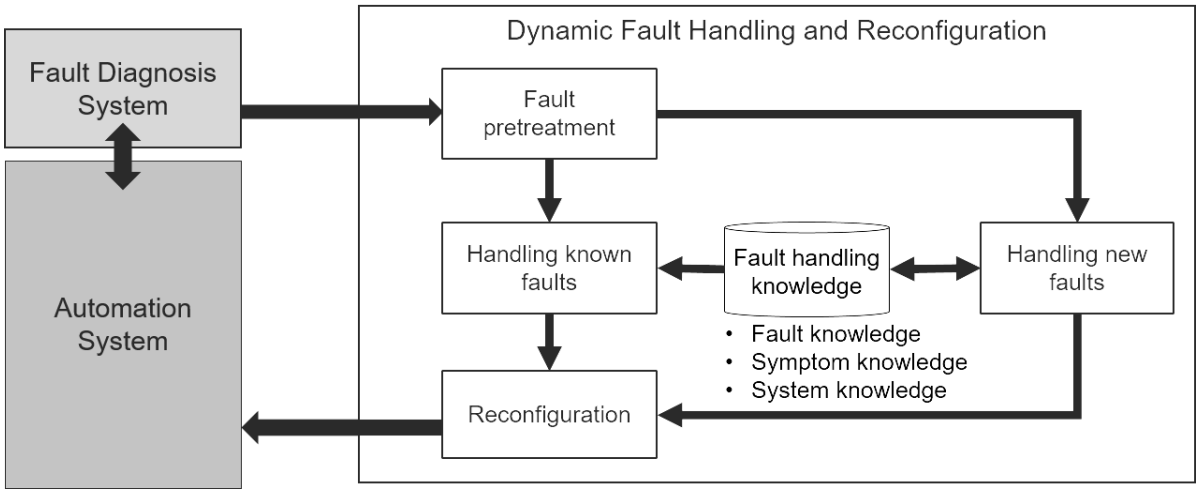



Figure 5.1: Overview of the conception of dynamic fault handling and reconfiguration

Local fault knowledge: it stores the known fault knowledge, which will be regularly updated with the fault knowledge in the server. In addition, the local knowledge base stores the sensor data and system preset data within a certain time interval before the fault occurs.

Interfaces: this consists of a local interface that is located in the industrial automation system, and a server interface, which is located in the server. The local interface supplies the possibility of sending the fault message and historical data from the fault diagnosis system to the server and receiving the reconfiguration commands. The sever interface is meant to receive fault messages and historical data and send the reconfiguration commands.

Fault pretreatment: this module serves to analyze the fault message and assign the fault information to the related fault handling based on the fault type, i.e. known faults, to the module of handling known faults and new faults to the module of handling new faults. If the fault is unknown, the historical data will be required and gathered from the local industrial automation system.

Handling known faults: with the help of the *fault ID*, this module accesses the fault knowledge to get the available functions and sends this information to the reconfiguration module.

Handling new faults: this module aims to identify fault impacts as well as available and not available functions by two steps, namely, fault localization, and identifying available functions.

Reconfiguration: this module aims to evaluate the reconfiguration possibilities (e.g. available functions, available tasks in the industrial automation system, and several specific corresponding measures) and creates the specific reconfiguration commands.

Fault handling knowledge consists of the fault knowledge, symptom knowledge, and system knowledge. The detail of this knowledge will be presented in the next section.

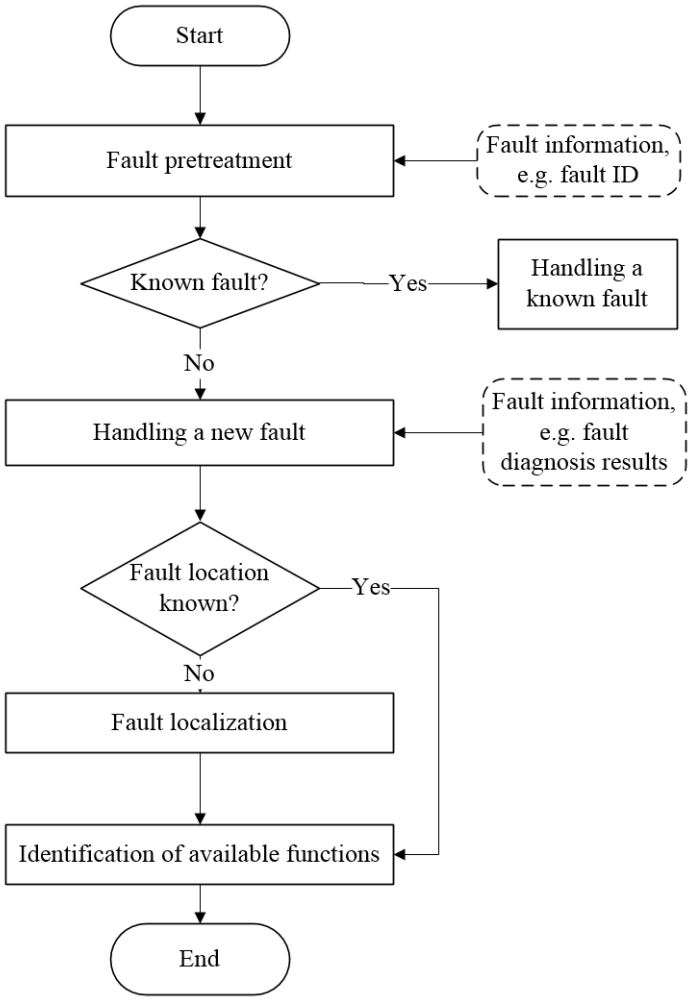



Figure 5.2: General process of handling a fault in the dynamic fault handling and reconfiguration system

In addition, it is necessary to highlight the general process of handling a fault in the dynamic fault handling and reconfiguration system (as shown in Figure 5.2). Firstly, the fault information will be previously processed. If the fault is known, then it will be assigned to the module of handling a known fault. Otherwise, if it is a new fault, it will be assigned to the module of handling a new fault. If the fault location is already included in the fault diagnosis results, i.e. known, then it will be directly assigned to the module of identification of available functions. Otherwise, the fault location ought to be determined first.

The procedure for fault handling through remote fault handling and reconfiguration is introduced as follows. In the local part, when an industrial automation system is out of order due to the failure of a component, the integrated fault diagnosis system performs its test cases, or several diagnosis approaches, such as process-identification methods [Iser06], with the intention of identifying the fault. Then it creates the fault message for the dynamic fault handling and reconfiguration system. Here, if the fault is known in its local fault knowledge, the fault information will be created with a fault message including the *fault ID*, states of the resources, and current tasks. Conversely, if it is a new fault, the fault information will be created with a fault message including a specific *fault*

*ID* and previous fault diagnosis results, states of the resources, current tasks, and historical data including a time stamp. In addition, it is necessary to point out that the local fault knowledge will be synchronized with respect to a certain time interval. Theoretically, this thesis assumes that the number of known faults in the local fault knowledge base is equivalent to the number of known faults in the server.

In the server, with the help of the fault message, the module of fault pretreatment in the dynamic fault handling and reconfiguration system identifies the fault type and assigns the fault information to a different module. The information is fed to a current module of reconfiguration. If the fault is known, the module “handling known fault” accesses the fault knowledge to determine the available functions. Otherwise, if this is a new fault, the module “handling new faults” can handle the fault through fault localization with the aid of the symptom knowledge and historical data, and identifying the available function by means of the system knowledge. Afterwards, based on the available functions, the module “reconfiguration evaluation” identifies the available tasks as well as changing the priorities of each task, and creates the related commands for available functions and tasks. Finally, the remote fault handling and reconfiguration system sends these commands to the industrial automation system, thereby guiding the industrial automation system to complete the reconfiguration.

## 5.2 Knowledge for the Dynamic Fault Handling and Reconfiguration

This section is intended to outline the detail of the fault handling knowledge which consists of symptom knowledge, fault knowledge, and system knowledge.

Symptom knowledge consists of symptom ID, fault location, features regarding parameters.

- Symptom ID indicates the unique identification of each symptom.
- Fault location depicts the possible defective components or subsystems which can result in this symptom.
- Features regarding parameters that show the analytical parameters, which can be the threshold, errors, parameter change tendency, parameter change rate, etc.

In accordance with the system level, the symptom knowledge, namely, the symptom table in the database, is divided into N-1 types. Here, N is the highest number of the system level. For instance, if there are three levels of a system and two subsystems, then there will be two types of symptom tables. Three symptom tables ought to be given later: one table for the system for determining defective subsystems, and two tables for the two subsystems for identifying the defective components.

Fault knowledge contains the fault ID, fault type, fault location, fault effect, available functions, reconfiguration commands for functions, and corresponding measures for reconfiguration.

- Fault ID indicates the unique identifier for each fault.
- Fault type shows the type of the fault such as an abrupt fault, incipient fault or intermittent fault.
- Fault location depicts the defective components or subsystems.
- Available functions are still available functions described with abbreviations.
- Reconfiguration commands are specific program commands and can be interpreted by the industrial automation system, which can be reconfigured with the help of these commands.
- Corresponding measures for reconfiguration are specific actions or instructions like inserting a small piece of code, operation instruction for the user, etc.

System knowledge is made up of component knowledge (formalized by the component model), function knowledge (formalized by the function model) and requirement knowledge (formalized by the requirement model). In the area of practical application, this knowledge will be redesigned and redefined with different matrices and rules (see Chapter 4). Furthermore, it is necessary to note that all tasks which have the **relationship with functions and resources** should be clearly defined. For example, a task *"Heating 4 liters water at 45*°C*"* requires the function "*Injecting X liter water*" and the function "*Heating water to Y*°C". Regarding the resources in this task, it demands *4 liters* water from tank1, which provides tank2 with heated water.

In addition, another necessary factor is process models or fault models for each parameter in the symptom tables. These models support the processing of the historical data reasonably and identify the defective component, thus creating the required symptoms for the current historical data.

## 5.3 Handling a Known Fault

As discussed above, faults in an industrial automation system are known as a known fault which has taken place at least one time either in the test phase or in the operating phase. Figure 5.3 shows the procedure of handling a known fault.

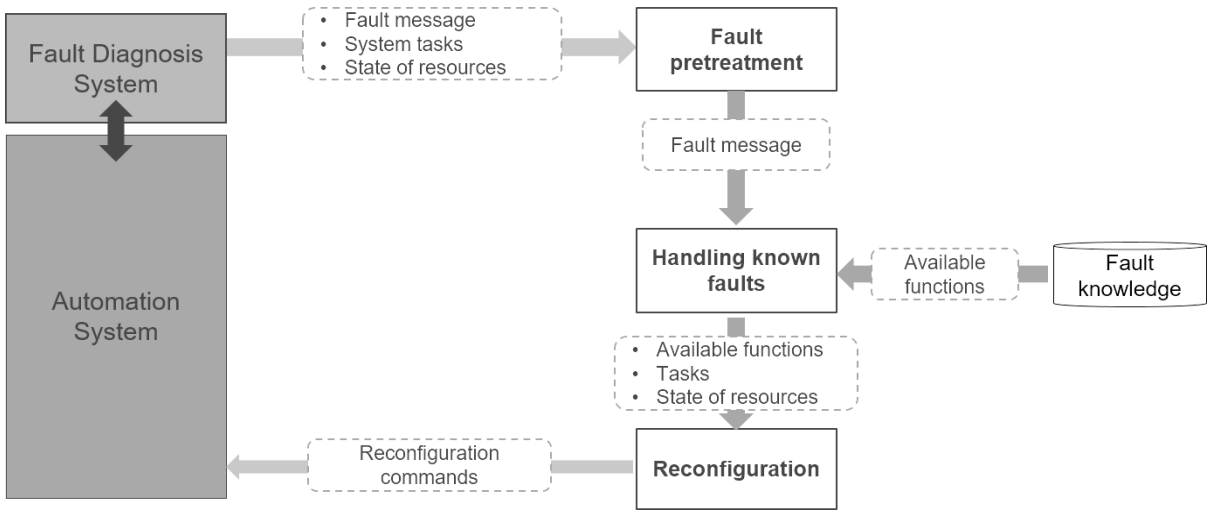



Figure 5.3: Procedure of handling a known fault

When a known fault in the industrial automation system occurs, the existing fault diagnosis system detects the fault, and then it sends the fault information, including *fault ID*, system tasks and state of resources. The module of the fault pretreatment then recognizes the *fault ID*, which is a normal fault identifier, and assigns the *fault ID* to the module of handling known faults. Subsequently, it accesses the fault knowledge and reads the available functions, as well as the reconfiguration commands for available function, after comparison with the *fault ID* in the database. This information will be sent to the reconfiguration module. The module of the reconfiguration analyzes the availability of tasks in the industrial automation based on the available functions and the states of resources. Then it generates the reconfigure commands for tasks and functions, and sends them to the industrial automation system with the intention of guiding the reconfiguration.

An example of interpreting the process of handling a known fault is introduced as follows: In a two tank system, there are three tasks with the following sequence, i.e. task1 “Heating 4 liters water at 45°C”, task2 “Injecting 3 liters water”, and task3 “Cleaning”. There are three functions: function1 “injecting X liters water from tank1 to tank2”, function2 “Heating water to Y°C”, and function3 “Outputting Z liters water from tank2 to tank1”. Hence, when a fault occurs, the integrated fault diagnosis system detects the fault and sends the fault diagnosis results, i.e. the fault ID, with *0x0001*, and fault location to the *temperature sensor*. In terms of the fault ID, the module of the fault pretreatment identifies this fault as a known fault. Likewise, with the help of the fault ID, the module of handling known fault accesses the fault knowledge, identifies the available functions as well as the reconfiguration commands, i.e. function1, function2 and 0x110. Subsequently, the module of the reconfiguration analyzes the availability of the current tasks from two aspects: the availability of functions and the availability of resources. Due to the malfunction of function2, task1 cannot be completed. But the other two functions are still available. Then, considering the availability of resources here, which is the capacity of water in the tank1, it assumes that tank1 has 6 liters of water. Hence, there are enough resources for task2 and task3. This module generates the commands for the tasks, i.e. 0x011. Finally, the dynamic fault handling

and reconfiguration system sends the reconfiguration command to the industrial automation system to perform the reconfiguration.

## 5.4 Handling a New Fault

Due to the weakness of the existing fault diagnosis system, it cannot determine the fault location for a new fault. Hence, in contrast to the known fault, there is no existing knowledge about available functions of a new fault; even the fault location is unknown. In order to solve this difficulty, this thesis proposes two steps in terms of the symptom knowledge and system knowledge respectively: the fault localization, to detect the fault location, and the identification of available functions.

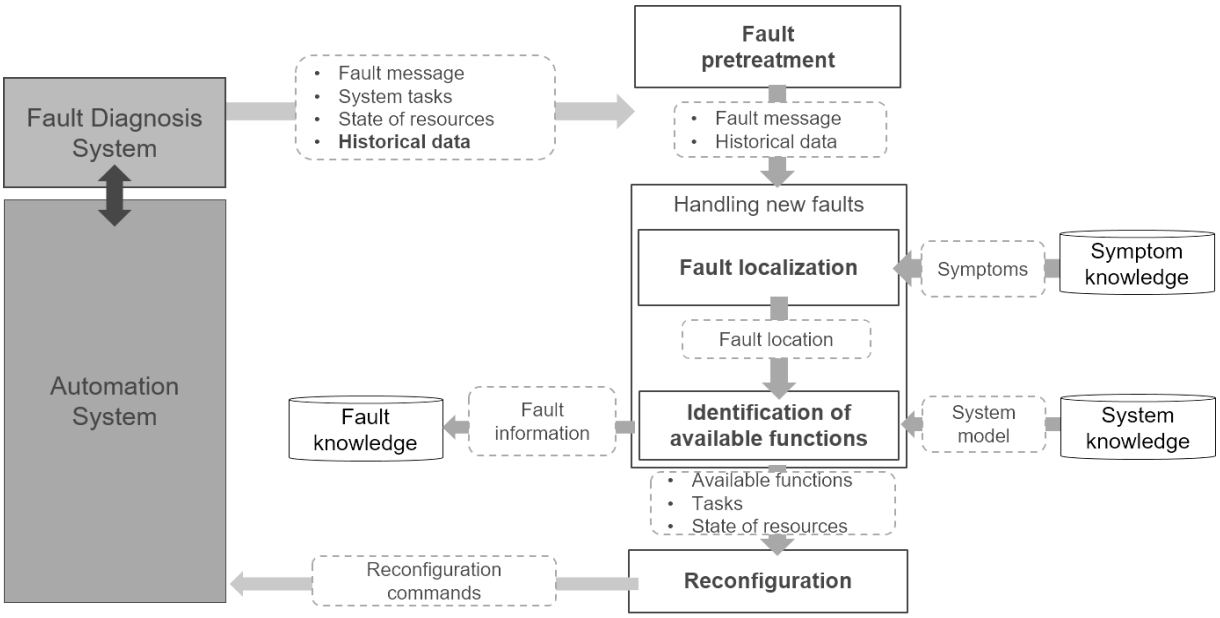



Figure 5.4: Procedure of handling a new fault

Figure 5.4 shows the procedure of handling a new fault in the industrial automation system. When a new fault in the system occurs, the automation system is out of order. With the help of monitoring, the existing fault diagnosis system detects the fault. Due to the lack of enough test cases or algorithms, the fault location cannot be confirmed in this case. The FDS later sends all generated existing fault information to the dynamic fault handling and reconfiguration system. The fault information includes a specific *fault ID* like 0x0000, previous fault diagnosis results, states of the resources, current tasks, and historical data including a time stamp. By means of the specific fault ID, the module of the fault pretreatment identifies this fault as new. It then assigns the fault with the proper information to the module of handling a new fault, that is, the fault message and historical data. Afterwards, with the benefit of various fault models and process models, the historical data can be analyzed and different features generated. Then, the generated features are compared with features in the symptom knowledge to determine the fault location. By means of the fault location and the system model, the available function can be identified

through two steps respectively: identifying the not-affected function via a function model, and identifying available functions via the requirement model. Details of the fault localization and identifying available functions will be introduced in the Sections 5.5 and 5.6 sequentially. In terms of the identified available functions, similar to handling a known fault, the module of the reconfiguration analyzes the availability of tasks in the industrial automation based on the available functions and the states of resources. Then it generates the reconfigure commands for tasks and functions and sends them to the industrial automation system with the intention of guiding the reconfiguration. An example for handling a new fault will be included in the following subsections.

## 5.5 Fault Localization for a New Fault

As mentioned in the last section, the dynamic fault handling and reconfiguration system enables the handling of a new fault in an industrial automation system. Following the general maintenance approach (Wang et al. 2015a), this research proposes two major steps to handle a new fault: fault localization, and the functional analysis. This section attempts to introduce the identification of the fault location with the help of historical data and symptom knowledge.

The main principle of fault localization is to determine the fault location from top to bottom in the component model. That is to say, the research determines the approximate scope of the fault location, namely possible defective subsystems. By means of inspecting all components belonging to the determined subsystems, the exact fault location can be ascertained. Apart from the fault information, including the fault message, previous fault diagnosis result and historical data, two kinds of knowledge play an important role in this principle: symptom knowledge and the component model. The component model divides the entire industrial automation system from top to bottom into various levels and groups.

Figure 5.5 outlines the conceptual procedure of the fault localization compatible with the component model from top to bottom. Firstly, the fault message and historical data are processed as input to generate the possible symptoms for the entire system in the component model. Then, by means of the symptom knowledge regarding to the entire system, it can inspect which of the subsystems of the entire system is the fault location. It is worth noting that here the system to analyze refers to the possible scope of the fault. It aims to check the subsystems of the system to analyze is the fault location. For instance, if a two-tank system is analyzed, then the top subsystems, i.e. injection subsystem, heating subsystem, and drawing out subsystem, are the possible fault locations.

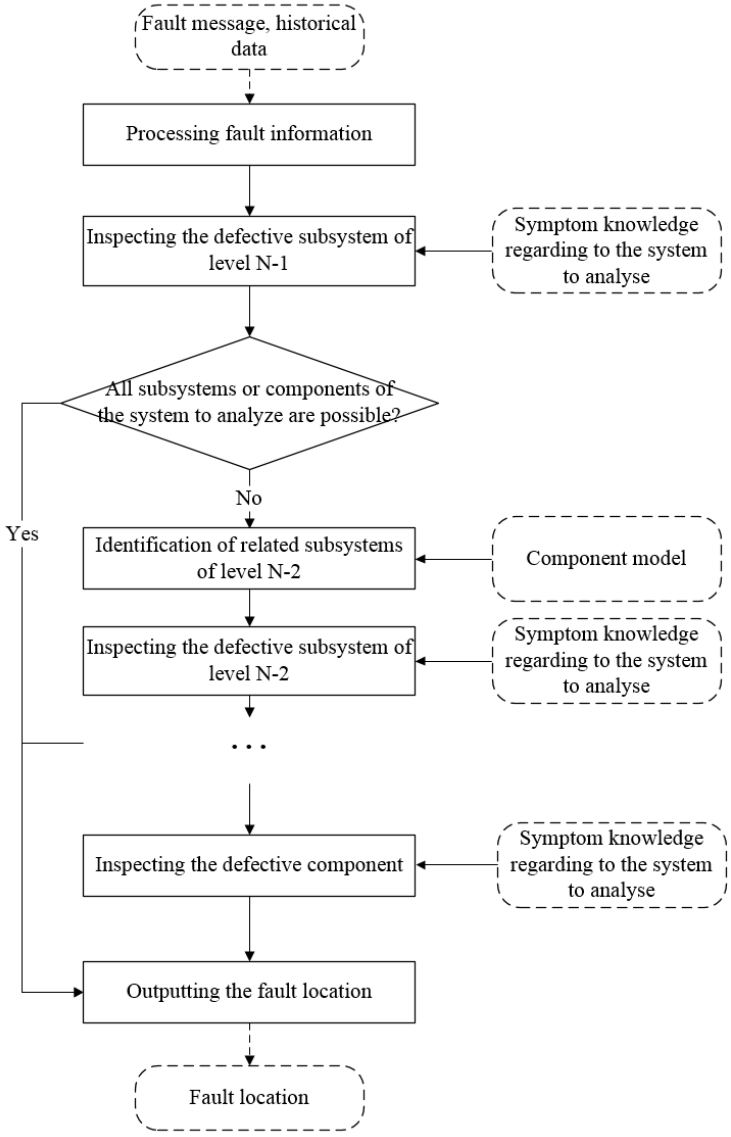



Figure 5.5: Conception of the fault localization, where N means the number of the top level of the component model

After the comparison between present symptoms and symptom knowledge, the fault location can be identified. If the fault location is all subsystems, then there is no necessity for further inspection, for it outputs the fault location to the entire subsystem. If the fault location is one subsystem, then this subsystem will be outputted to the next analysis procedure for a further inspection. If it involves more than one subsystem, but not all subsystems, then all the inspected subsystems will be checked for the further inspection. On the basis of the component model, the corresponding inferior subsystems, which have the relationship with the possible fault location, can be assured for the further analysis. Then the process of generating symptoms and comparison symptoms will be repeated again and again until it reaches the lowest subsystems. At that level, the components of the possible subsystems will be inspected. Finally, the fault location can be determined at the component level and will be outputted for the next module in identifying available functions. Now

that the main conceptual procedure of the fault localization is presented, the concrete conception of processing the fault message and historical data will be introduced in the next section.

### 5.5.1 Principle of Generating the List of Symptoms

Fault models and process models are utilized for the purpose of establishing the list of symptoms which is based on the data of an industrial automation system. The historical data will be used as input for the various models and is further analyzed. As a result, the list of analytical symptoms will be generated.

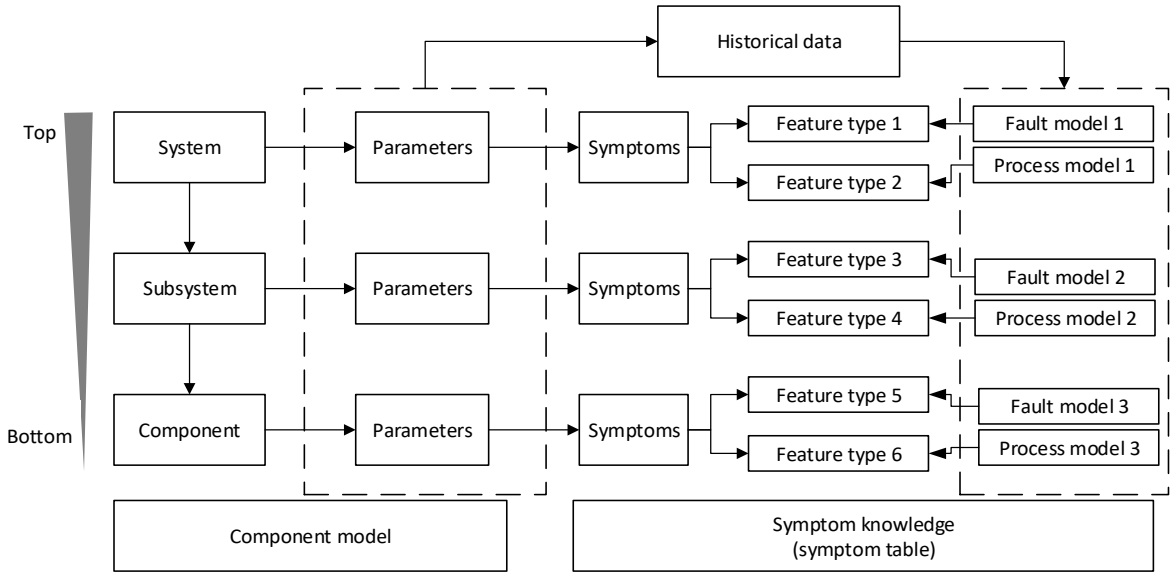



Figure 5.6: Relationship of various terms for the fault localization

Each industrial automation system is composed of different subsystems. The subsystem owns input values, output values and state parameters. The subsystems consist of various components, e.g. temperature sensors, heater, etc. The subsystems represent different functions of the industrial automation system. When a component of the system is out of order, this can result in certain abnormal changes in technical processes, such as variables, sensor data, etc. And these abnormal changes are described in the traditional fault diagnosis system as symptoms. A defect in different subsystems or components leads, further on, to different corresponding symptoms. Therefore, if the symptoms can be clearly identified, and there is a symptom table, which stores all known symptoms and their related fault locations, it is simple to identify the source of the fault, or fault location. Hence, to determine the fault location, the important problem is how to generate the symptoms. There are two ways this goal can be achieved. The previous fault diagnosis results of the existing fault diagnosis system can be utilized as an information source for symptoms. Furthermore, because the historical data of the parameters are known, and with the help of the historical data and reasonable mathematical models, such as fault models and process models, it is possible to derive various required features which make up the symptoms. Feature means a specific variable or parameter in the symptom table, such as an abnormal parameter, an error with

the set-up value, a change tendency, a change rate, etc. Symptoms here consist of: identifier, feature types, fault / process model, and related location. Fault models and process models are certain specific mathematic models: for example, a process model checks the error with the set-up value, $f(x) = TempSetupValue - TempCurrentValue$. These models are actually recognized as completed diagnosis components. They are developed by the producers and can perform the diagnostic function individually. These relationships are indicated in Figure 5.6.

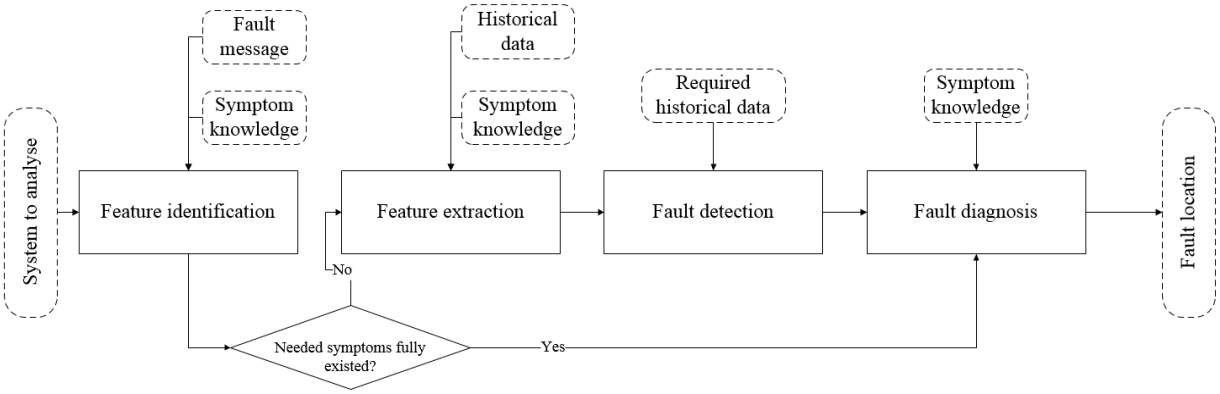



Figure 5.7: General principle of generating symptoms and localizing the fault

Figure 5.7 outlines the general working principle of generating the list of symptoms and localizing the fault. For the sake of generating symptoms and localizing the fault, this thesis proposes the following principle: feature identification, feature extraction, fault detection, and fault diagnosis. Feature identification intends to access the correlative symptom table, fetching the feature types in the symptom table and analyzing the fault message. Feature extraction aims to generate the necessary fault and process models, as well as features from the historical data, such as the temperature value with the time stamp. Fault detection proposes to process historical data, i.e. executing the calculation function, and generating the symptoms. Fault diagnosis is intended to compare the generated symptoms with the chosen symptom table with the intention of determining the fault location. The general working principle of generating symptoms and localizing the fault is as follows: Firstly, based on the system to be analyzed, the corresponding symptom table will be obtained from the symptom knowledge. After comparing the fault message and symptoms in the symptom table, if the required symptoms already fully exist in the fault message, it will directly go to the step of fault diagnosis. If not, it ought to generate the required symptoms initially and proceed to the step of feature extraction. In feature extraction, using symptom knowledge, the required data, as well as the features, are derived from the historical data. Subsequently, this extracted data will be further processed with the help of various specific fault models and process models to generate the required symptoms. Finally, with the benefit of the generated symptoms, the available symptoms will be chosen from the symptom table in line with the generated symptoms. It is natural that the possible fault location will be determined in the symptom table.

### 5.5.2 Fault Localization Procedure

As mentioned regarding fault localization, the aim is to identify the possible subsystems or components and the processing of the historical data, which is accessed from the fault diagnosis system in the pretreatment step.

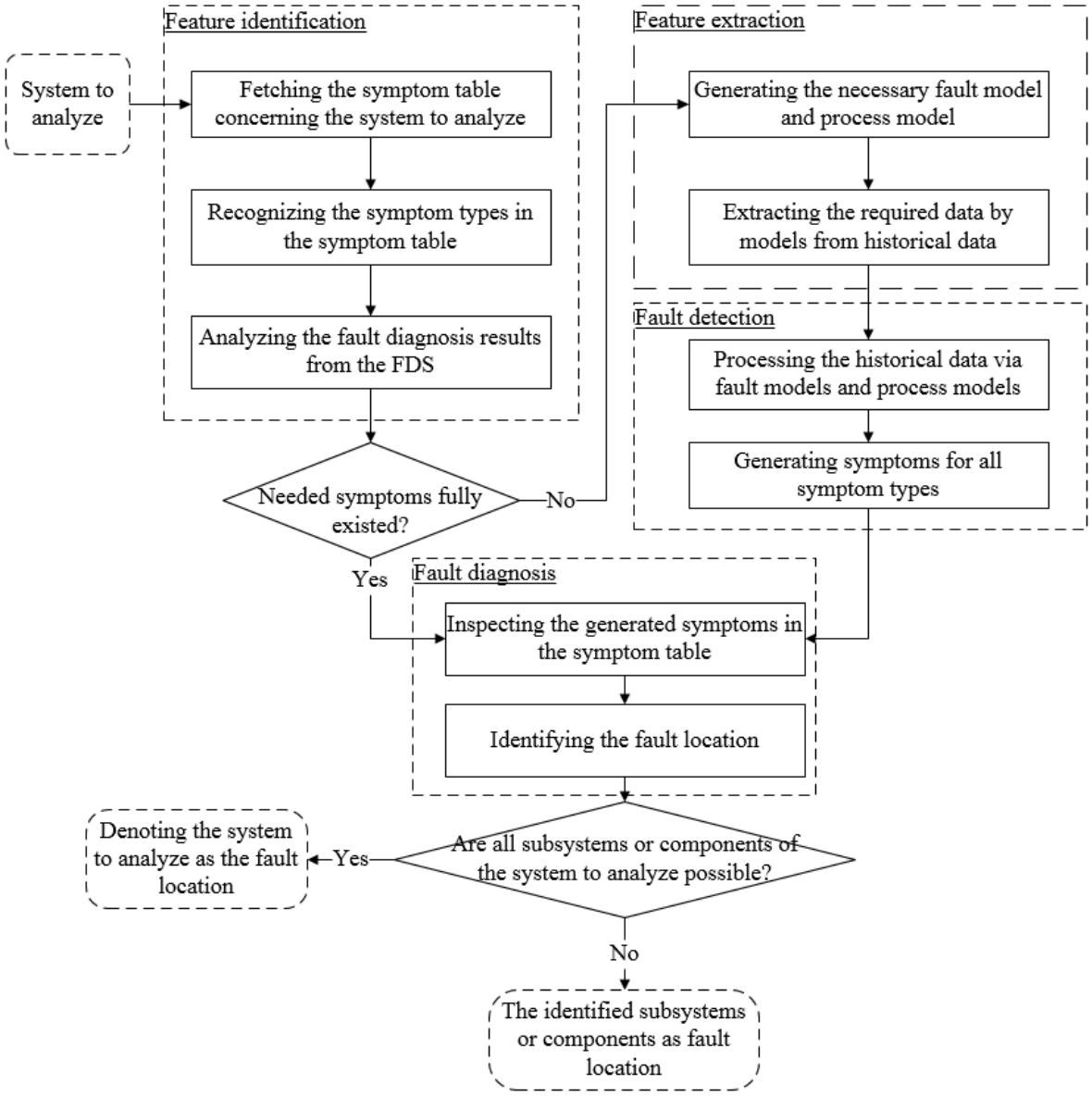



Figure 5.8: Processing the fault information to identify the fault location

As Figure 5.8 shows, this section proposes the procedure for processing the fault information to identify the fault location. Firstly, the system to be analyzed is inputted to the feature identification step. The symptom table will be obtained with regard to the system to be analyzed from the symptom knowledge, which consists of an amount of the symptom table for the entire system and different subsystems. Next, the symptom type will be recognized in the symptom table. The fault diagnosis of the fault message from the FDS is then analyzed. If the symptoms are already included in the fault message, the next step is to perform the fault diagnosis. Otherwise, fault

models and process models for all symptom types are generated. Here, the symptom types are the same as the feature types, which means abnormal behaviors in the industrial automation system. Then, the required data from the models will be extracted from the historical data in the next step.

Subsequently, the extracted historic data can be processed via each fault model and each process model. Then symptoms for all the required symptom types are generated. For example, if the error of the temperature is 5°C, two symptoms can be generated: the abnormal parameter is temperature, and the error is 5°C. The generated symptoms are then matched to the symptoms in the symptom table to determine which symptoms are fulfilled. Subsequently, the fault location is fixed in the symptom table. However, one symptom can be the result of more than two different locations. For example, if the temperature is abnormal, the reason for this symptom can be a broken temperature sensor or a broken heater. In such cases, all possible locations are considered as the fault location. As in the above example, if there are no further exact symptoms, the temperature sensor and the heater are denoted as the fault location. If all subsystems or components of the system to be analyzed are the possible fault location, then the system to be analyzed as the fault location is denoted. Finally, the identified subsystems or components are outputted as the fault location.

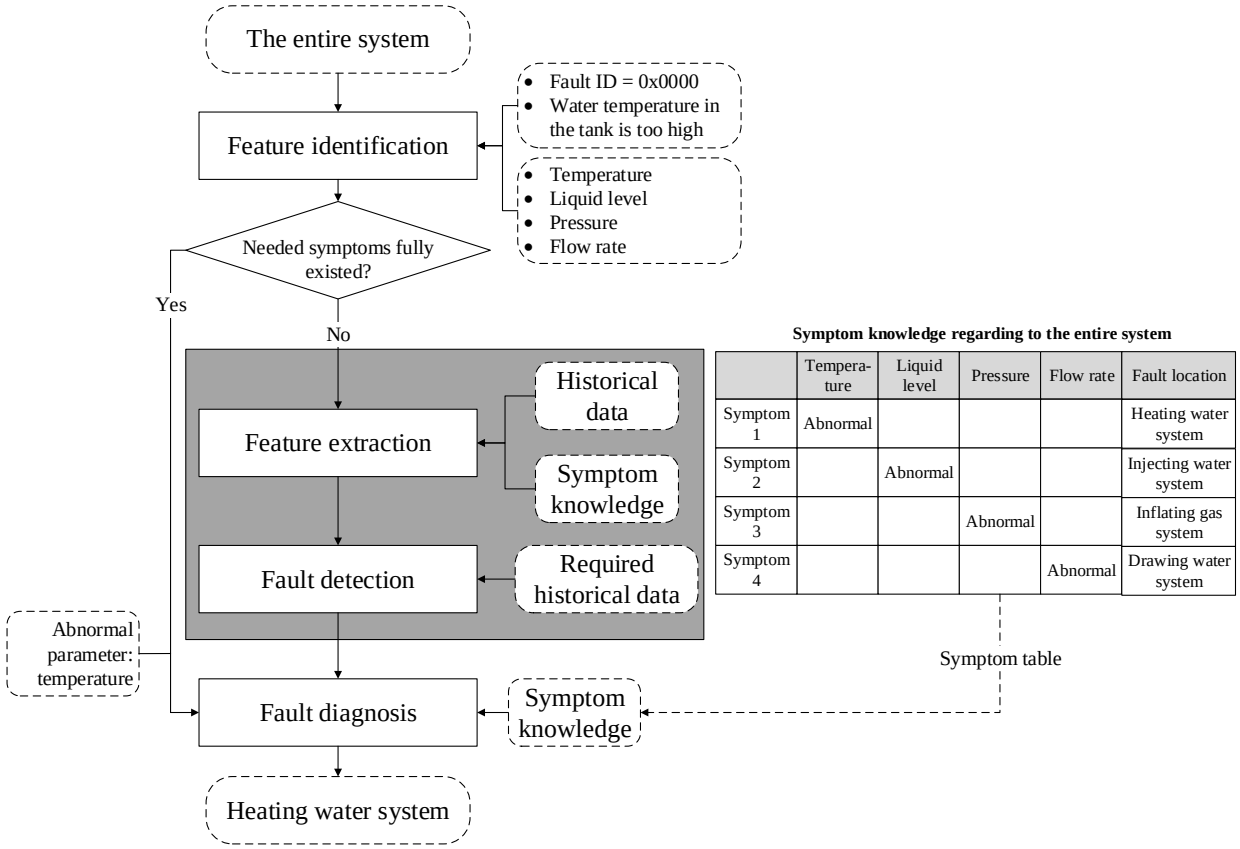
## 5.5.3 Example of the Fault Localization

This section provides an example of fault localization. Here, it is assumed that a two-tank system has three levels in the component model: the entire system, subsystems, and components. It consists of four subsystems: an injecting water system, a heating water system, an inflating gas system, and a drawing water system. There are four different typical parameters for these four subsystems respectively: the liquid level, the temperature, the pressure and the flow rate. The heating water system consists of a temperature sensor, a microcontroller, and a heater. As introduced in the section where the procedure of the fault localization is described, when a new fault of the heater in the heating water system occurs, the dynamic fault handling and reconfiguration system proposes two major steps to determine the exact fault location: identifying the defective subsystems, and then identifying the defective components. These two steps will be presented in the next two sections.

### 5.5.3.1 Identification of the Defective Subsystem

This section gives the example of identifying the defective subsystem (see Figure 5.9). It assumes that the fault information is as follows: "*0x0000; the temperature of water in the tank is too high; temperature 15, 20, 30…; liquid level, 1.0, 1.2, 1.4…* ". As Figure 5.9 shows, the symptom knowledge regarding the entire system is indicated in the table. There are four symptoms: Firstly, symptom1 consists of the abnormal temperature and the fault location of the heating water system. Symptom2 possesses the abnormal liquid level and the fault location of the injecting water system. For symptom3, the abnormal parameter is the pressure and the fault location is the inflating gas

system. In symptom4, the abnormal parameter is the flow rate and the fault location is the drawing water system.




| | Tempera-ture | Liquid level | Pressure | Flow rate | Fault location |
|---|---|---|---|---|---|
| Symptom 1 | Abnormal | | | | Heating water system |
| Symptom 2 | | Abnormal | | | Injecting water system |
| Symptom 3 | | | Abnormal | | Inflating gas system |
| Symptom 4 | | | | Abnormal | Drawing water system |

Figure 5.9: Procedure of identifying the defective subsystem of a two-tank system

Firstly, the dynamic fault handling and reconfiguration system assures that the entire system is to be analyzed. In step1 of feature identification, the fault ID is extracted from the fault information, i.e. *0x0000*. It determines that this is a new fault. Then it accesses the symptom knowledge regarding the entire system and fetches the symptom types, including the state of the four parameters, i.e. the temperature, the liquid level, the pressure, and the flow rate. Afterwards, it checks the fault information to ascertain whether the previous fault diagnosis results includes the corresponding symptoms. Here the fault message shows “Water temperature in the tank is too high”. So the required symptom type already exists in the fault message. Hence, the steps of feature extraction and fault detection can be skipped. It can be confirmed that the abnormal parameter is the temperature. Then it turns to the fault diagnosis step. In this step, it compares the abnormal parameter – the temperature – with the symptom knowledge concerning the entire system. Obviously, only symptom1 in the symptom table can be fulfilled with the abnormal temperature. Finally, it is able to figure out that the fault location is the heating water system. In the component model, the heating water system consists of a heater and a temperature sensor that are supposed to be checked in the next step.

### 5.5.3.2 Identification of Defective Components

The four steps of processing fault information will be repeated again, but with various data and symptom knowledge concerning the heating water system to identify the defective component in the heating water system. It can be certain that the abnormal parameter is the temperature. Hence, the data of the temperature should be further derived and processed. In addition, to simplify the complexity, it is assumed that there are two known symptoms in the symptom table. Symptom1 has the following attributes: the working process is the heating process, the value of the temperature smaller than 25°C, the error between the setup value and the real value is smaller than 15°C, the change tendency rate is 0 and the fault location is the heater. Symptom2 has the following attributes: the heating process, the value of the temperature is 0, the error is more than 21°C, the change tendency rate is 0, and the fault location is the temperature.

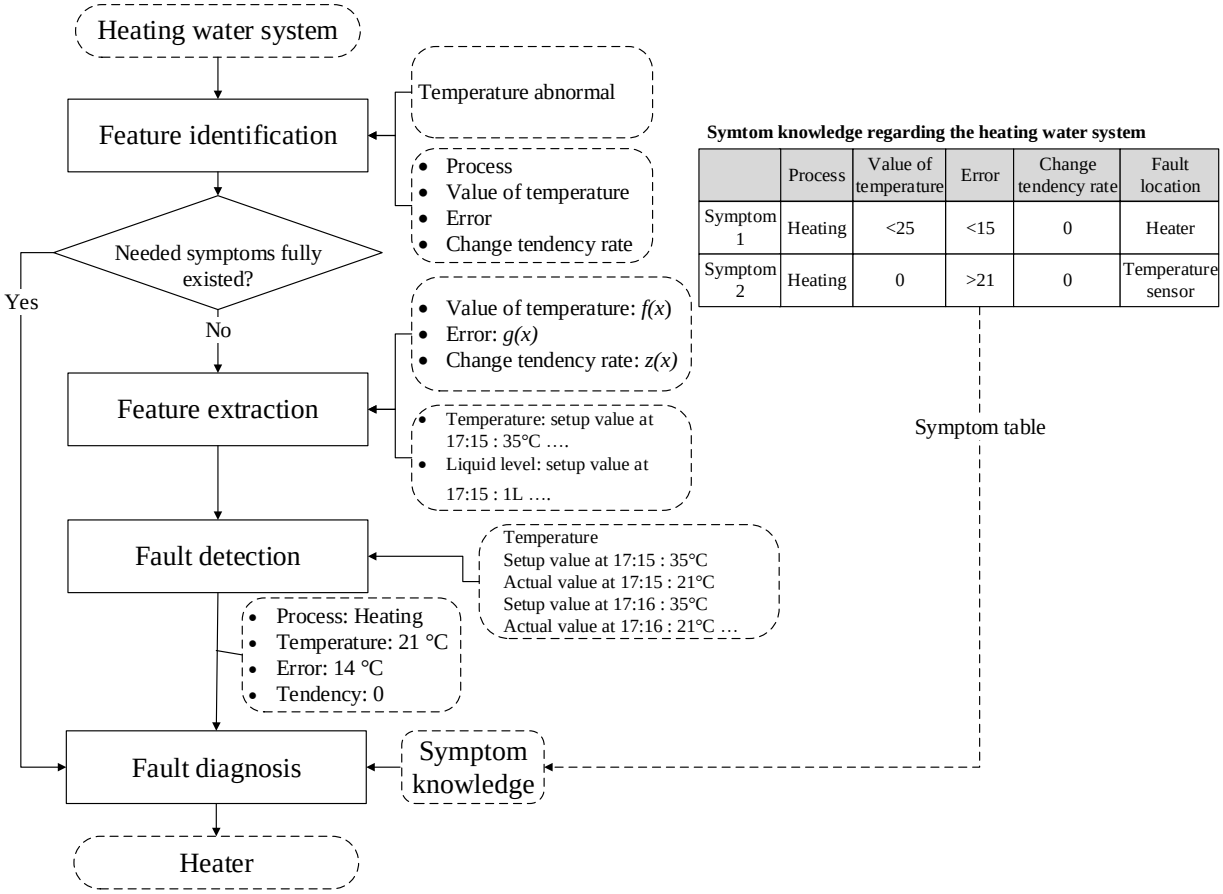



Figure 5.10: Procedure of identifying the defective components of a two-tank system

Figure 5.10 indicates the procedure of identifying the defective components. Firstly, in the feature identification step, the symptom knowledge regarding the heating water system is derived from the knowledge base. And the required feature types, that is, symptom types, are identified as the process, the value of the temperature, the error of the temperature, and the change tendency rate. However, in the existing fault message, only the previous fault diagnosis result that the temperature is abnormal is presented. It is visibly certain that this symptom cannot handle the demands for analyzing the fault location any more. Hence, the system turns to the step of the

feature extraction. In this step, the necessary fault models and process models regarding the heating water system ought to be extracted from the system model: the value of the temperature with *f(x)*, the error between the setup temperature and the real temperature with *g(x)*, and the change tendency rate with *z(x)*. In addition, the required data regarding the temperature will be filtered and extracted from the historical data. In Figure 5.10, the original historical data consists of the temperature data, the liquid level data, and so on. After this step, the data on the temperature is extracted as the input for the next step of the fault detection. The extracted temperature data is: *Setup value at 17:15: 35°C, actual value at 17:15: 21°C, setup value at 17:16: 35°C, actual value at 17:16: 21°C,* etc. Subsequently, in the fault detection step, this data is determined in the process of the heating water process. Then, the value of the temperature is 21°C with the help of the mathematical function f(x). The error between the setup value and the actual value is 11°C by means of the mathematic function g(x), e.g.

$$\mathrm{g(x)} = \sum_{1}^{N} \frac{SetupValue - ActualValue}{N}.$$

On the basis of the mathematic function z(x), the change tendency rate of the temperature can be obtained with 0. Ultimately, based on the determined symptoms and the symptom table for the heating water system, it is simple to figure out that symptom1 can conform to the analyzed results from fault detection step: the working process with the heating process, the value of the temperature with 21°C, the error of the temperature with 11°C, and the change tendency rate with 0. Hence, the heater is identified as the fault location for the current fault. And this result will be the output for the next step to identify available functions.

## 5.6 Identification of Available Functions

Following the identified fault location, this section proposes to analyze the fault impact in the logical scope of the overall industrial automation system, namely, the identification of available functions. As the requirement supposed, an automatic analysis is required for solving the problem of identifying the available functions. This thesis proposes a knowledge-based approach in order to realize an automatic reasoning for this objective, but without intervention in the reasoning process; in other words, in order to identify the available function, the dynamic fault handling and reconfiguration system is able to be trigged by a specific input, own the knowledge in a formal format, utilize automated algorithmic calculation with the help of that knowledge, and follow the instruction of a specific logic.

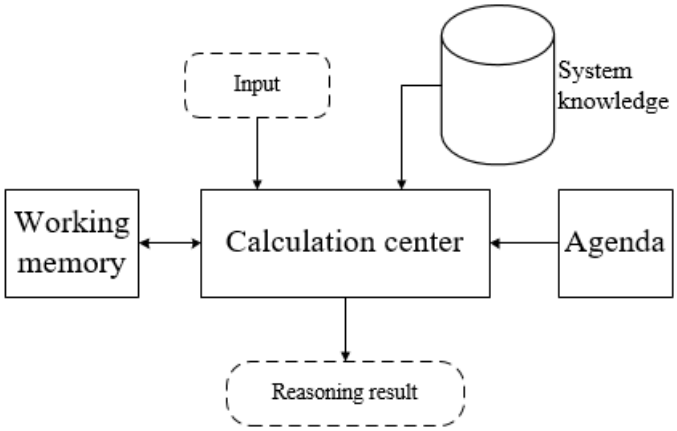



Figure 5.11: General approach of reasoning with knowledge

Figure 5.11 indicates the general approach of reasoning with knowledge. The input is the information which can trigger the reasoning process; in this research, the trigger is the fault location. The reasoning result is the output of the reasoning process; in this research, the output is the available functions. The system knowledge is the knowledge in the knowledge base; in this research, it is the formalized system model; in other words, the various matrices and rules mentioned in Chapter 4. The working memory stores the intermediate results of the reasoning: the states of functions and requirements, the availability of functions and requirements, functions to check, etc. The agenda provides the performing instructions for the reasoning. The calculation center executes the central reasoning functionality: triggering the reasoning process, accessing the system knowledge, reading and storing the intermediate results in the working memory, following the instruction from the agenda, and outputting the reasoning result.

In addition, in order to achieve the calculation of the reasoning, the availability of functions and requirements are denoted into TRUE and FALSE, 1 and 0, respectively. The value “TRUE” means that the function to be analyzed is available. Conversely, the function to be analyzed is not available.

### 5.6.1 Overview of Identification of Available Functions based on the Fault Location

Figure 5.12 outlines the overview of the identification of available functions based on the fault location and the system model. It performs the two steps of the identification of not-affected functions and the evaluation of not-affected functions: verification and validation.

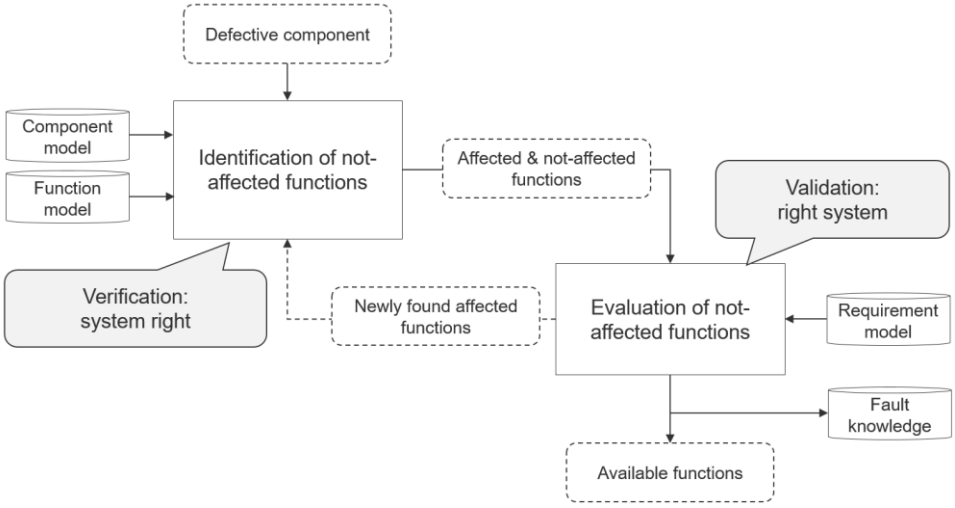



Figure 5.12: Overview of identification of available functions based on the fault location

On the one hand, the former receives the fault location, e.g. the defective basic component, reasoning with the component model and function model, and outputting the not-affected functions. The former is supposed to check if the system to be analyzed can meet all functional demands, and identifies the functions which cannot be achieved any more, in order to verify if the functions of the system to be analyzed can still work correctly internally. On the other hand, based on the requirement model, the latter attempts to inspect whether the system to be analyzed can meet all the specific requirements, e.g. safety, and identify the available functions from the not-affected functions, to validate that the functions of the system to be analyzed are still the correct functions which are needed by the customer. Moreover, if some new affected functions are found in the evaluation step of the non-affected functions, those newly found affected functions are transferred into the former step to identify whether some functions can be affected by those newly found affected functions.

Finally, if no newly affected functions are found in the evaluation step of unaffected functions, all unaffected functions are denoted as available functions. Subsequently, the available functions are transferred to the next step of the reconfiguration. The available functions will also be stored in the fault knowledge as one attribute of a new fault.

Before introducing the two steps in detail, it is necessary to give an overview of the reasoning methodology. There are a total of seven matrices in the system model which can be grouped in three types. They are in charge of two abilities: addressing, i.e. finding the individuals that need to be analyzed and mapping, and finding individuals in different models corresponding to the individual. The three types are the mapping matrices which can realize the decision conversion across different models through each matrix, addressing matrices which can indicate the relationship between items in each model, and a redundant matrix which can outline redundant relationships. System knowledge also includes plenty of rules, and consists of two parts: the conditional part, the quantitative analysis of the effectiveness of the individual, and the operational

part, the consequences of the conditional part. Functions own only the computational part. But requirements possess these two introduced parts.

With the help of the system knowledge, this thesis proposes the following methodology for identification and evaluation. Firstly, it maps the defective component to the function model via the matrix between components and functions. Then, it identifies the functions to be analyzed and values the basic functions with 0. Subsequently, by means of the rule of each function, it calculates the value of each function. Then it transfers the value of the functions to the requirement model. Finally, it evaluates each requirement and outputs the available functions.

### 5.6.2 Identification of not-affected Functions via Function Model

This section introduces the approach for identifying not-affected functions with the help of the function model. It consists of four main steps (see Figure 5.13): step 1, mapping defective components to functions; step 2, valuation of basic functions; step 3, identification of related functions of the malfunction; and step 4, estimation of related functions.

For step 1, the dynamic fault handling and reconfiguration system utilizes the matrix between the components and functions, the relationship between the component model and the function model, and determines the malfunction which can be mapped from the defective component. For instance, the malfunction is the basic function of “measuring the temperature”, when the defective component is the temperature sensor.

In step 2, to prepare for the further automatic reasoning, the dynamic fault handling and reconfiguration system values basic functions on the basis of the malfunction. If the malfunction is a basic function, then the basic function will be valued as 0, and the other basic functions are valued as one. If the malfunction is a sub function, then the sub function will be valued as 0 and all its basic functions will be valued as 0, too. Here, 0 means unavailable, and one means available. Continuing with the example in the previous paragraph, the basic function of “measuring the temperature” is valued as 0.

In step 3, the related functions in the function tree are recognized via the matrix between functions. The dynamic fault handling and reconfiguration system identifies the related functions as directly related to the malfunction, at first. Reflected in the function tree, these related functions have the direct upper and lower relationship with the malfunction. Then, based on the newly recognized functions, it continues to search in the functional matrix to find their related functions until there are no longer any relevant functions. For instance, the malfunction “measuring the temperature” has the relationship with the sub function “heating water to X°C” and the basic function “heating”.

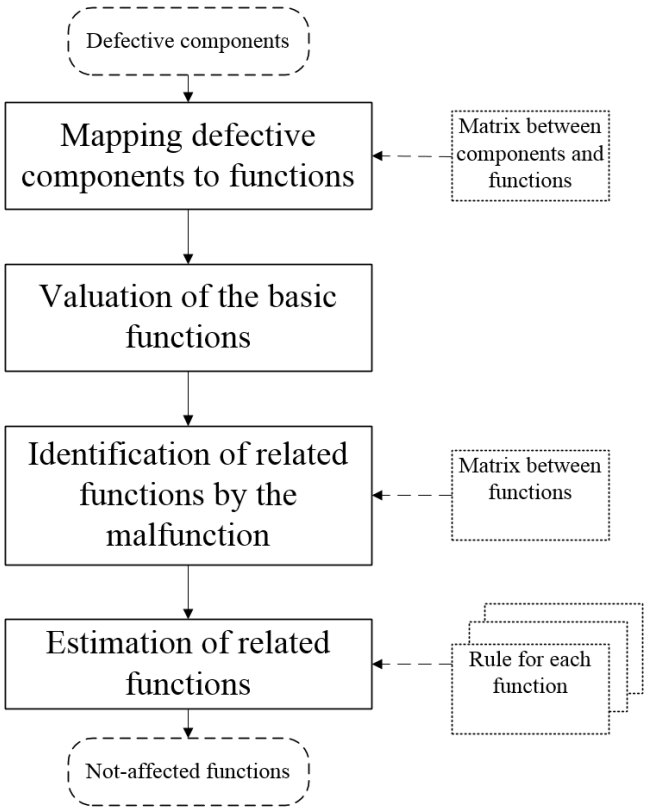



Figure 5.13: General procedure of identification of not-affected functions

Step 4 evaluates the availability of the recognized related functions by means of the rules for each function. In accordance with the hierarchical structure of the function tree, each function has its own level. The reasoning logic is, according to the number of levels in the function tree, from the largest number, namely the highest level; each function is evaluated in line with its rule equation following the principle from top to bottom until all related functions are analyzed. Subsequently, because the unrelated functions are unaffected by the malfunction, they will still be available and set to one. For instance, for the malfunction BF1 “measuring the temperature” with the value of 0, the related sub function SF1 “heating water” needs this function BF1. If the rule for this sub function is $SF1 = BF1\ \boldsymbol{AND}\ BF2$, it assumes that the BF2 “heating” is available with the value of one. Then, after the calculation, the value of the sub function is 0. Hence, the sub function “heating water to X°C” is affected.

#### 5.6.2.1 A Single Basic Component Fault

This section introduces an example of identification of not-affected functions with a single basic component fault, i.e. a defective temperature sensor in a two-tank system. After the fault localization, the fault location is determined with the temperature sensor following the procedure for the identification of available functions.

Firstly, it maps the fault location of the temperature sensor with the help of the matrix between the components and functions, (see Figure 5.14). In the matrix, the connected function BF1 “measuring the temperature” is indicated.

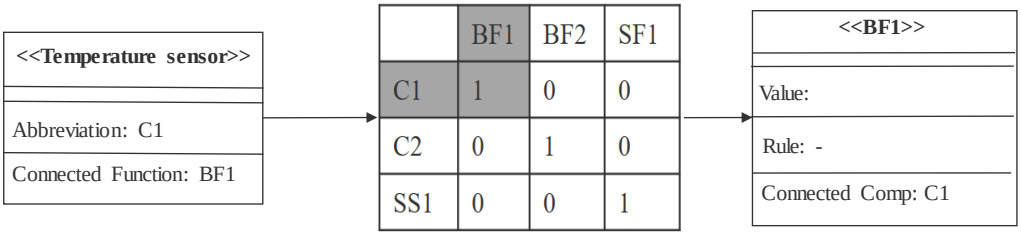



Figure 5.14: Mapping the defective component to the function model

Secondly, it attempts to give a value to each basic function. In this example, there are three functions, a sub function “heating water to X°C” with the abbreviation SF1, a basic function “measuring the temperature” with the abbreviation BF1, and a basic function “heating” with the abbreviation BF2. The BF1 is then set to 0 and the BF2 is set to 1. These results are transferred to the intermediate function matrix, (see Table 5.1).

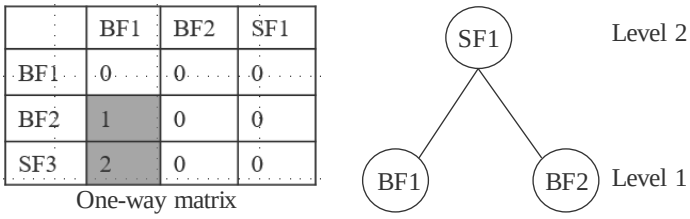



Figure 5.15: Matrix between functions in the two-tank system and the function tree

Thirdly, with the help of the one-way matrix between functions as shown in Figure 5.15, the related functions are identified in the matrix. SF1 needs these two basic functions to attain the heating water functionality. BF2 has the relationship due to the information flow, i.e. heating provides the objective of monitoring the temperature. Similarly, heating is based on the ability to measure the temperature as a prerequisite. Hence, in line with the malfunction BF1, it is simple to recognize that SF1 and BF2 are related. SF1 is located in the high level with level 2, and BF1 is located in the low level with level 1. These two functions will be transferred to the matrix to store the intermediate results of the functions in the reasoning process as an analytical function buffer, e.g. Functions_to_analyze = (SF2, BF1).

Table 5.1: Matrix to store the intermediate results of the functions in the reasoning process

| **ID** | **Availability** | **State** |
|---|---|---|
| BF1 | True | Checked |
| BF2 | True | To check |
| SF1 | Null | To check |

Finally, it evaluates the availability of the function SF1 and BF2. Following the principle from top to bottom, it searches for the analytical route in the function tree.

To perform the search, this thesis proposes the depth-first-search (DFS) approach, which is a recursive algorithm for searching a tree structure [WXH+10]. It starts searching from the high level to low level and moves forward as far as possible. In this process, if there are two possible

traversing nodes, one node is suspended, and it continues the search with another node. When there are no more nodes in the current path, it moves backwards, until it meets a node which has two possible traversing node. Then the search will be continued along with the suspended node. And it repeats this process until all nodes of the tree to be analyzed are covered. This method is usually used in the navigation system to help the driver to find a new route.

In this thesis, the DFS approach is supposed to be used in the process of evaluating the availability of functions. In the function tree, all basic functions are assumed with values in step 2. To assess the functions of the upper layer, the DFS approach selects a function to analyze in the highest level, e.g. SF1. In line with its rule equation, such as *SF1=BF1*BF2*, the required functions will be identified. In the function tree, it moves forward until it meets the basic functions. After the recalculation of basic functions, the value of basic functions is reset. Then it moves backwards with the value of basic functions until all required functions have been verified with its rules, which is presented in the right of the rule equation.

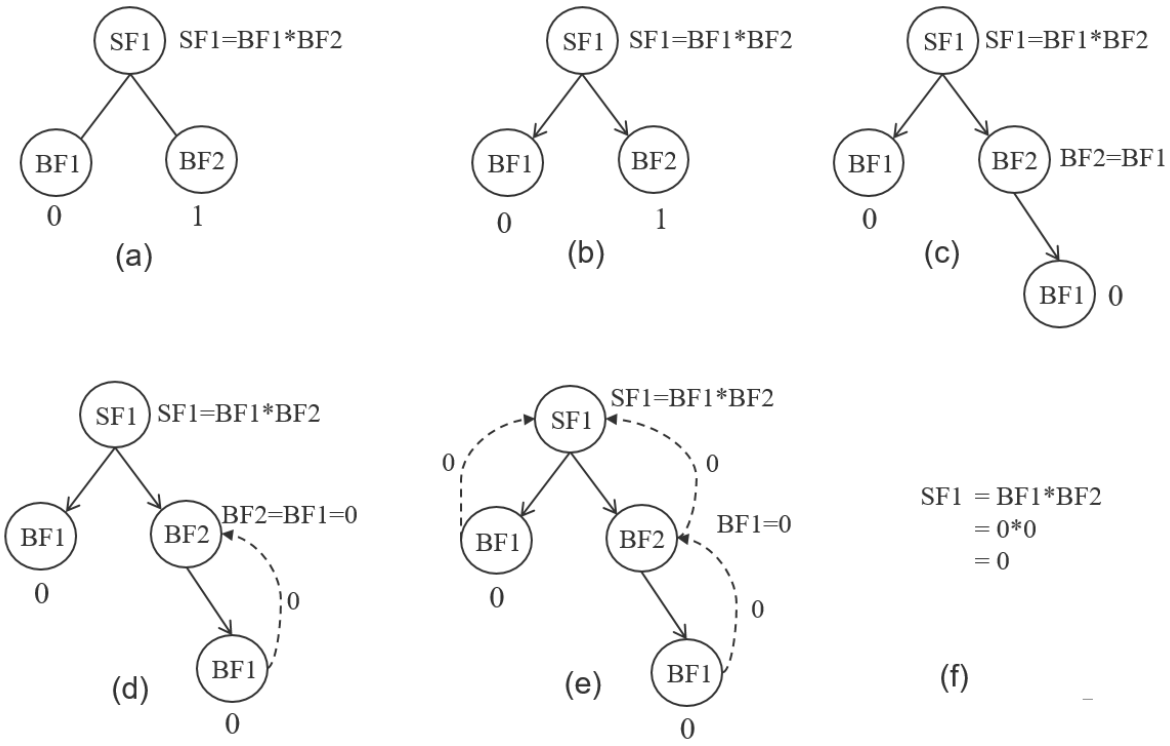



Figure 5.16: Evaluation of functions with the help of the DFS approach

For the example of the two-tank system, Figure 5.16 shows the procedure for evaluating functions with the help of the DFS approach. Subfigure a shows the original function tree with three functions: SF1 with its rule equation, BF1 with the value 0, and BF2 with the value 1. Subfigure b outlines the application of the DFS to calculate the SF1. The function tree will be transferred with direction, from top to bottom. As a previous search result, BF1 and BF2 are required for the further calculation. Subfigure c shows that BF2 will be sought further due to it rule equation which has the connection with the function BF1. Because the function BF1 is 0, there is no need to search

further. The deep search process is finished for now. Then backtracking ought to be started. As subfigure d shows, the first backward step is from BF1 to BF2. Then, based on the rule equation of BF2 = BF1, the value of BF2 is reset to 0. Subfigure e indicates the second backtracking, from BF1 to SF1 and from BF2 to SF2. The values of the functions BF1 and BF2 are transferred to the function SF1. Finally, with the help of the determined BF1 and BF2, the function SF1 is evaluated with equation SF1 = BF1 * BF2 = 0 (see subfigure f). As a result, the functions to be analyzed, i.e. SF1 and BF1, are evaluated in terms of the DFS approach. Due to the negative value, both functions, i.e. SF1 and BF1, are identified as affected functions.

It is worth noting that the remaining functions which are not covered by the related functions and the malfunction ought to be marked as unaffected; in principle, these functions are functionally available. In another words, the states of those functions not covered are set to 1.

#### 5.6.2.2 Multiple Basic Components Fault

The previous section introduced an example with a single basic component fault. However, in a complex industrial automation system, the fault diagnosis result usually not only provides a single basic component fault, but multiple fault locations. There are many reasons for such results. For instance, it may be a fact that a number of basic components are out of order. It is also possible that one basic component is out of order due to the second damage of the defective component, such as a defective heater burning out the temperature sensor because of the too high temperature of the liquid. In most instances, the fault diagnosis result is subject to the ability of the fault diagnosis system. It cannot be completely accurate in terms of a specific point of the fault, but can only provide a scope of the fault location or parts of all components as the fault location. Hence, as mentioned in Section 5.5, there are still another two possibilities for the fault location: the multiple basic components fault, and a subsystem fault.

When it is a multiple basic component fault, the steps of identifying not-affected functions are the same as for a single component fault. There are two possibilities to attain the objective of the identification of not-affected functions: processing one by one, and processing them at the same time. The drawback of the first possibility is that the same basic component fault will be assigned twice so, while increasing the amount of repeated calculations, the same function to be analyzed may be set as different results. Hence, this thesis proposes dealing with multiple basic component faults in the meantime. That is to say, in the first step of identification of not-affected functions, all defective basic components will be mapped to the function model. And the value of these basic functions will be set to 0. The remaining reasoning process is the same as the process of dealing with a single basic component fault.

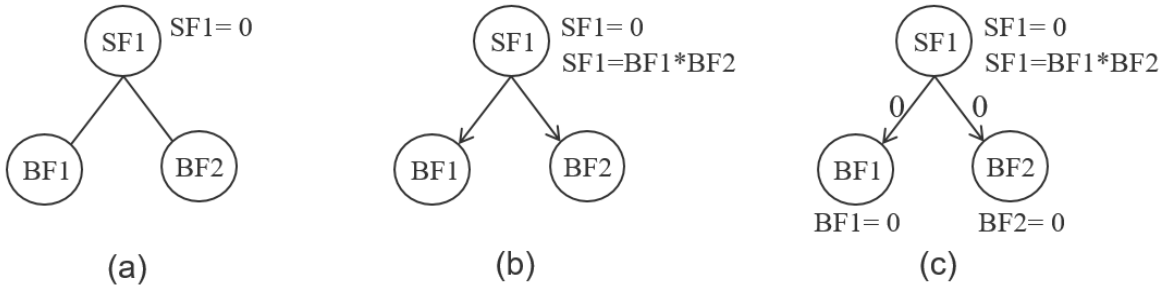



Figure 5.17: Evaluation of related functions for a subsystem fault

But if it is a subsystem fault, there are some differences when comparing it with the basic component fault. Firstly, it maps the subsystem to the function model. Then the related sub functions are denoted as defective, and are set to 0. Subsequently, based on the relationship matrix between functions, the related functions are determined. Due to the lack of clarity of the fault location, in step 4, before evaluating from the highest level, the functions which are included by the defective sub functions are identified and set to 0. For example, as Figure 5.17 a shows, the function SF1 is defective and has two basic functions, BF1 and BF2. Figure 5.17 b indicates that there are only two basic functions that can affect the state of the sub function SF1. Then these two basic functions will be set to 0 (see Figure 5.17 c). The evaluation process later is the same as dealing with a single component fault.

### 5.6.3 Identification of Available Functions via Requirement Model

In the previous section, the identification of not-affected functions has been introduced, namely, that the system has been correctly verified in case of a fault. However, not-affected functions cannot be assured of still fulfilling every reasonable requirement. By evaluating a requirement, it denotes not only the states of the requirement itself, but also the state of the corresponding functions. It might be that a safety function is affected by the malfunction. In this case, the function restricted by this safety function cannot be activated. For instance, the safety requirement requires the highest temperature under a safe value. Otherwise, there is the danger of an explosion.

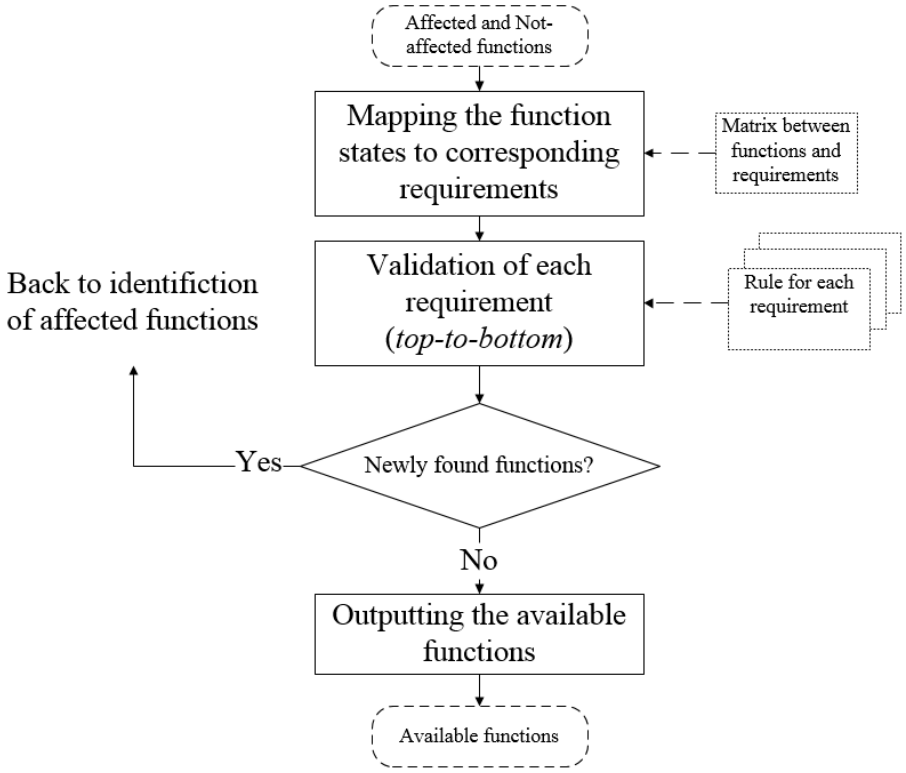



Figure 5.18: Procedure of the identification of available functions via requirements

The procedure of identification of available functions via requirements, specifically the evaluation of not-affected functions, is presented in Figure 5.18. It is made up of three main steps: mapping the function states to corresponding requirements, validation of each requirement, and outputting the available functions.

In the last step, the affected functions and not-affected functions are identified and also marked with the values 0 and 1, respectively. Generally, these states are stored in the intermediate function matrix. Firstly, the states of all functions ought to be transferred to the corresponding requirements. These function states enable the evaluation of the availability of specific requirements. It must also be highlighted that the requirement has a relationship with functions. In this thesis, the requirement itself is not considered, but rather the relation to its necessary functions, so if the necessary functions exist and are not-affected by the malfunction, then the requirement can be assured and performed.

Secondly, to assure the completeness of the requirements, it is necessary to check the availability of all of them. The DFS approach starts from the top system requirement in the requirement tree. Then it identifies all needed requirements in its rule. To search further, specific requirements, which make up the system requirements, ought to be checked, e.g. safety, security, etc. In choosing one specific requirement, this search process continues until the lowest level of requirements. Then these requirements are evaluated with the help of the states of their required functions. Subsequently, it begins the process of the backtracking, transferring its value to the upper level. However, in contrast to the backtracking process of functions, the consequence of the

requirement ought to be performed before transferring the value to the upper level. The consequence of a requirement defines the change of some corresponding functions. In addition, the consequence of the requirement states of these newly found functions in the intermediate function matrix is set to 0 until all specific requirements and the system requirement are inspected.

If there are no newly found functions, i.e. the states of functions are not changed, then all not-affected functions are denoted as available functions. Otherwise, if there are some newly found functions, then these newly found functions ought to be transferred to the last step, to identify whether these newly found functions can affect the other functions. Finally, these available functions are outputted to the next step to generate suitable reconfiguration commands.

#### 5.6.3.1 An example of the evaluation of not-affected functions

To begin with, there are five requirements in the example: system requirement (SR), safety requirement (SaR), security requirement (SeR), Safety requirement considering the temperature (SaR1), and Safety requirement considering the pressure (SaR2). This example considers only two functions: sub function “heating water to X°C” (SF1) and sub function “monitoring the temperature” (SF2). One matrix of intermediate states of functions records the function states after the identification of not-affected functions. The relationship matrix between functions and requirements shows the relationship between two sub functions and two safety requirement. One requirement tree depicts the hierarchical structure of the five mentioned requirements. To simplify the complexity of the calculation, there are two assumptions concerning states of functions and requirements: *SF1 = 0, SF2 = 1, SaR2 = 1* and *SeR = 1*.

The analysis procedure of identifying an available function via the requirement model is simplified in Figure 5.19.

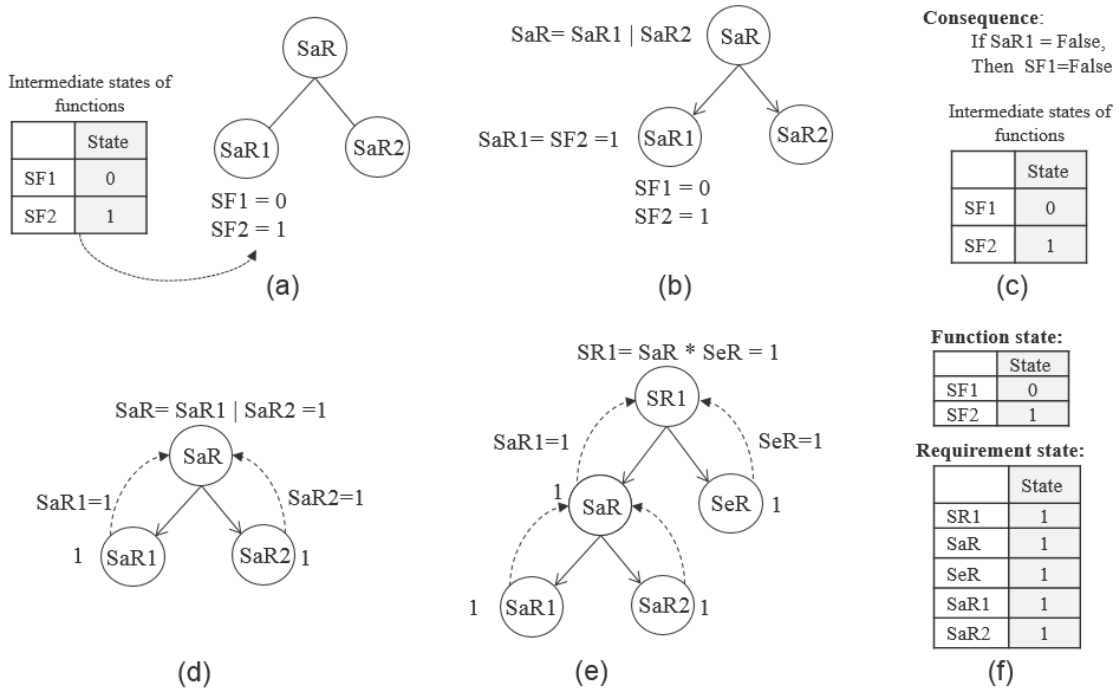



Figure 5.19: Identification of available function via the requirement model

As Figure 5.19 a shows, stats of functions are mapped to the requirement model. It shows that the requirement SaR1 has a relationship with two functions, SF1 and SF2. In this example, the process of searching in the tree begins from the specific safety requirement SaR, which has the rule equation SaR = SaR1 | SaR2. The symbol "|" stands for the relationship OR. In accordance with the DFS approach in Figure 5.19 b, it recognizes that two safety requirements, SaR1 and SaR2, are needed to evaluate the availability of the requirement SaR. To evaluate the availability of the requirement SaR1, the figure depicts the state of SaR1 to be equal to the state of the function SF2. And because the state of the functions SF1 is positive, the state of the requirement SaR1 is 1, meaning this requirement is available. Before further searching or backtracking, the consequence of this requirement ought to be evaluated (see Figure 5.19 c). The consequence provides that If SaR = False, then SF1 = False. Due to the positive state of the requirement SaR, it performs no change of the state of functions. Figure 5.19 d outlines the backtracking in the branch of the specific safety requirement SaR. With the help of the rule equation, i.e. SaR = SaR1 | SaR2 = 1 | 1 = 1, the state of SaR is set to 1, meaning this requirement is also available. Figure 5.19 e highlights the backtracking until the top of the requirement tree. The state of the system requirement SR1 is later evaluated with the rule equation, i.e. SR1 = SaR * SeR = 1 * 1 = 1. Hence, the requirement SaR is confirmed as an available requirement. Finally, states of functions and requirements are showed in Figure 5.19 f. All requirements are available and there are no function that have changed their states in the analysis process. Therefore, there is no need to analyze the availability of functions again. The function SF1 is then determined as an unavailable function, and the function SF2 is confirmed as an available function.

In addition, it is worth noting that this analysis process can be stopped at any step, when a rule of a requirement is fulfilled, and the consequence of the requirement indicates that the industrial

automation system ought to be shut down, i.e. all functions should be deactivated. For example, the loss of a safety requirement can result in an explosion. Going back to Figure 5.19 e, if the requirement SaR is unavailable, it provides that the system be shut down to avoid a potential injury to personnel. In this case, all functions will be directly set to negative, i.e. 0. The entire analysis process of handling a new fault will be stopped during this time, the next reconfiguration step is skipped.

## 5.7 Reconfiguration based on the Available Functions

In this section, the method of reconfiguration in terms of available functions is presented. In general, there are three reconfiguration possibilities. Hardware reconfiguration [ShJh02] changes the physical structure or physical connection among physical cell devices. Software reconfiguration changes the context of the logical structure, like as the parameter, the control algorithms, the logical structure among applications, etc. [SZW17]. The third possibility is the combination of hardware and software reconfiguration. However, it is hard to reconfigure the physical structure of an industrial automation system in practice without the help of experts because the change of a developed industrial automation system requires detailed knowledge about the system, and very professional skills. In addition, another limitation of hardware reconfiguration is that a physical reconfiguration usually relies on a specific hardware type, e.g. FPGA.

The objective of the reconfiguration is to lead the defective industrial automation system into a new working or partially working mode, so the fault effect will be isolated. This thesis proposes a reconfiguration approach to change the logical structure of an industrial automation system, i.e. changing function states in the industrial automation system by activating available functions and isolating unavailable functions. Needless to say, an industrial automation system, especially in the field of industry, has more than one task, each of which is arranged in advance. Hence, the ongoing tasks in the system ought to both be reevaluated and rearranged.

To attain a reasonable and successful reconfiguration, this thesis proposes two steps:

1. Consideration of the availability of ongoing tasks: Based on the available functions and necessary aspects, the availability of ongoing tasks will be considered.

2. Generation of reconfiguration commands for functions and tasks: This requires the dynamic fault handling and reconfiguration system to generate reconfiguration commands, which includes not only the commands for functions and tasks, but also the possible corresponding measures, such as activating specific codes for a redundancy. On the basis of step 1, necessary commands are created for a defective industrial automation system.

### 5.7.1 Estimation of the Reconfiguration Types and Verification of Current Tasks

Based on the available functions, this thesis suggests four reconfiguration possibilities in the scope of the logical structure (see Figure 5.20), i.e. functions and ongoing tasks:

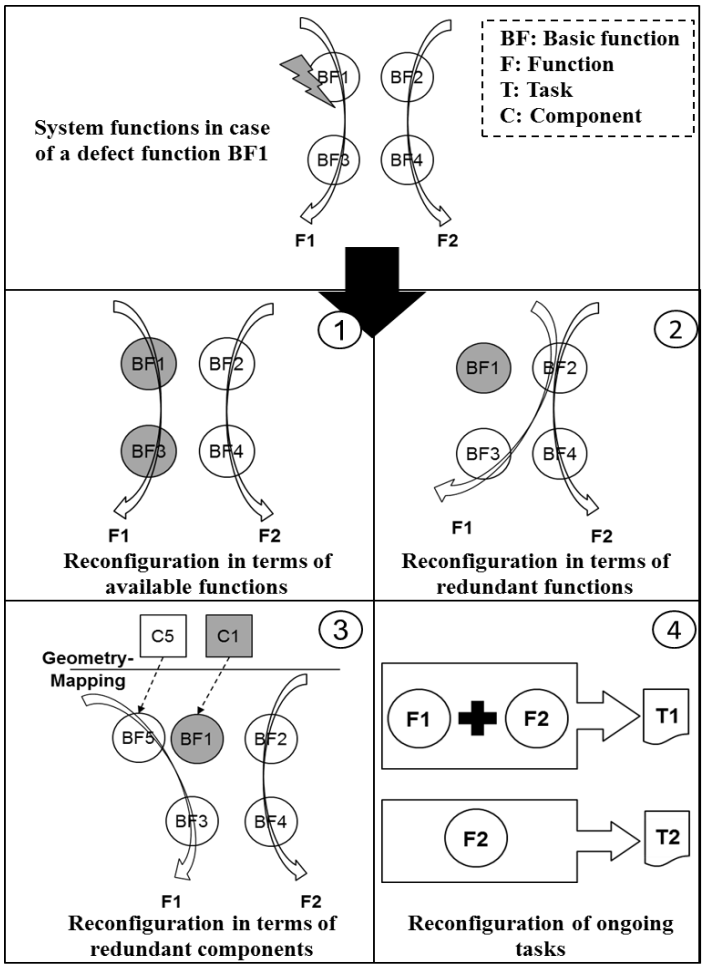



Figure 5.20: Four reconfiguration types for the industrial automation system

***1. Reconfiguration in terms of available functions:*** All affected functions are disabled, which are marked as 'FALSE' in the reconfiguration commands, and available functions are enabled again, which are marked as 'TRUE' in the reconfiguration commands, like "Function:0x010010".

***2. Reconfiguration in terms of redundant functions:*** This changes the geometry between functions, as well as in the logical view, i.e. using one function to replace another function. For example, there are two liquid level sensors in the water tank: an ultrasonic sensor can measure the value of the water level, and a photoelectric switch can monitor a specific water level. When the latter fails, this function "monitor a specific water level" cannot be performed and its function can be replaced by the former function. Nevertheless, a specific command is also required, e.g. function.switch.level.high = (function.ultrasonic.level =3L).

***3. Reconfiguration in terms of redundant components:*** This changes the geometry between components and functions, as well as from a physical view to a logical view [KeVo13]. For instance, there are two valves between two tanks, and one is the redundant valve for the other. In this situation, when the valve is out of order, then the redundant one can replace its work to transport the water.

***4. Reconfiguration of ongoing tasks:*** This attempts to verify the availability of all ongoing tasks and change the priorities of all tasks. To obtain this goal, three aspects must be considered: available functions, relationship between functions and tasks, and the states of all resources, such as the volume of water. After the evaluation of these conditions for every task, the available tasks will be chosen and enabled, so the industrial automation system can reconfigure its functional structure with available functions to complete the available tasks in a specific prioritized sequence.

### 5.7.2 Procedure of the Reconfiguration based on Available Functions

The reconfiguration procedure is proposed in this section, along with the four different reconfiguration types, where available functions are the key for reconfiguration. Figure 5.21 indicates the main procedure of the reconfiguration with the available functions. Based on the result from the previous step, the available functions, as well as unavailable functions, are recognized. Here, six steps are required to complete the reconfiguration:

- Identification of available functions: Concerning the result of identification of available functions, which are stored in the intermediate matrix of function states, these functions with their states will be divided into different groups in accordance with the function levels, such as basic functions, sub functions, main functions, redundancy, etc.

- Identification of the redundancy: This attempts to identify the activated redundancy to identify corresponding measures, such as the state of restart, specific code, and assistant help by the user.

- Verification of current tasks: This suggests an evaluation of the availability of the ongoing tasks from two aspects in this thesis respectively: the availability of required functions, and the availability of required resources.

- Support service for the reconfiguration: This is supposed to confirm the necessary measures which need the help from the maintenance service or user, e.g. operation instruction, service contact information, etc.

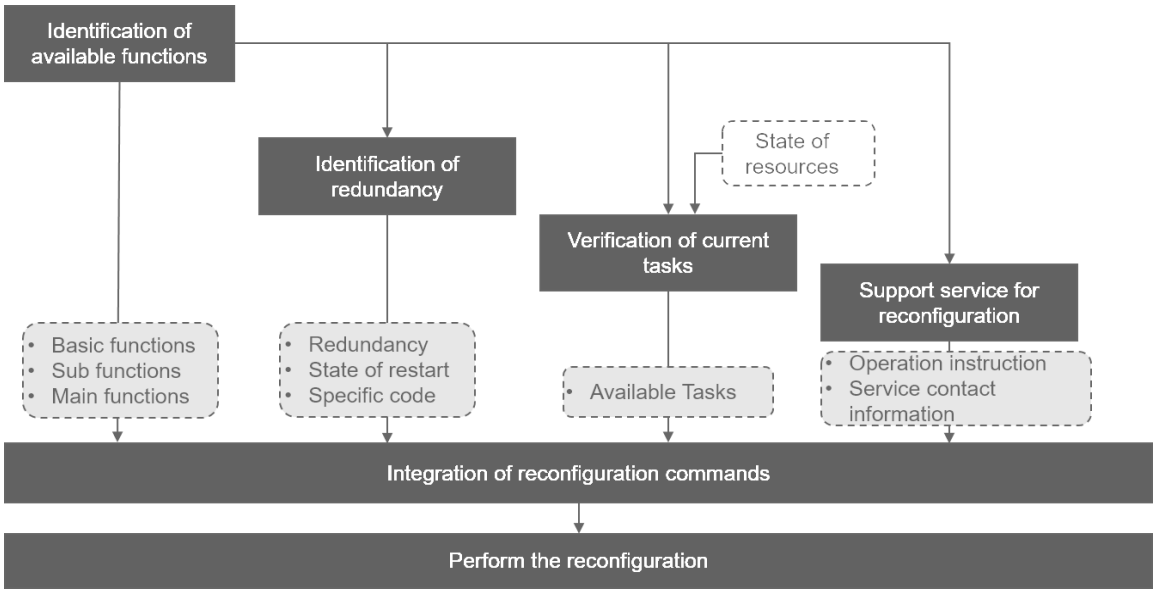



Figure 5.21: Overview of the reconfiguration based on available functions

- Integration of reconfiguration commands: This generates the reconfiguration commands for functions, ongoing tasks, specific code, state of restart and support service in a specific format. In addition, if it is a new fault, the fault information with fault ID and symptoms will be also integrated together.

- Performing the reconfiguration: This undertakes the reconfiguration in the industrial automation system in practice. The corresponding fault information will later be stored in the database with a new fault ID, symptoms, available functions and reconfiguration commands for functions. The fault name and fault description can be manually replenished by the experts. With the help of integrated reconfiguration commands, the industrial automation system interprets the reconfiguration commands locally and performs the reconfiguration to transfer the system from a shutdown mode to a working mode again. The existing fault diagnosis identifies the fault information during this time and stores this fault information in its local fault database as a known fault.

### 5.7.3 An Example for the Reconfiguration

In order to interpret the concept of the reconfiguration, an example for the two-tank system will be presented in this section. Figure 5.22 shows that there are three functions, i.e. injecting X liter water from tank1 to tank2, heating water in tank2 at Y°C and drawing Z liter water from tank2 to tank1. The original water resource in tank1 is 10 liters. Due to a detective heater, the function “heating water” can no longer be performed, but the other two functions are still available. There are still three ongoing tasks respectively: task1 “heating 3L water to 45°C”, task2 “cleaning”, and task3 “Injecting 4L water”.

Firstly, with the state of functions from the previous step, the availability of the functions is identified. Here in the example, only the main functions are depicted. Then, the activation of redundancies ought to be recognized because there is neither redundancy for the heater nor for the function “heating”. No redundancy is thus activated in this case. The availability of ongoing tasks will be concurrently reevaluated. For one thing, from the perspective of the availability of functions, on the basis of the relationship between tasks and functions (see Figure 5.20), namely that task1 demands all three functions, the task2 requires functions “water injection” and “drawing water”, and task3 requires the function “water injection”, task1 cannot be performed, and the other two tasks can be adopted in practice. Furthermore, from the perspective of the availability of resources, task1 and task2 will be further reevaluated. Task2 “cleaning” needs 5L water to complete the cleaning of the water pipes and the tank2. Task3, “injecting 4L water in tank2”, requires 4 liters of water as the resource. So 10 liters of water is enough for both task2 and task3.

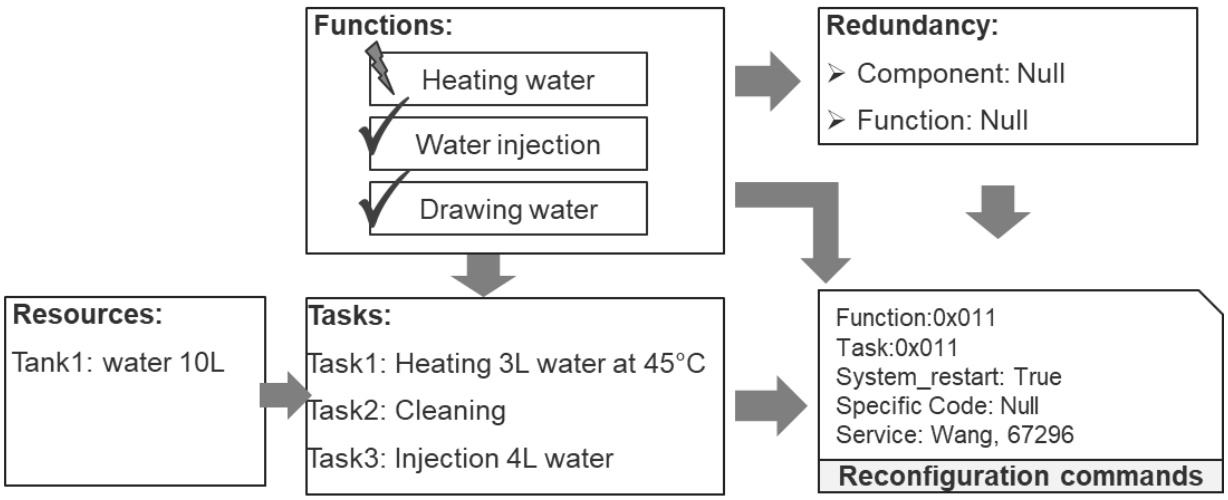



Figure 5.22: An example of the reconfiguration

Finally, the reconfiguration commands are integrated for functions with 0x011 and tasks with 0x011. The code 0x011 is hexadecimal for 0000 0001 0001 and means that the functions F1 “water injection”, and function F5 “drawing water”, can be executed. To calibrate the initial state of the water level in the two tanks, the system is in need of a restart, i.e. system_restart: true. Due to the non-activation of redundancies, specific codes are not demanded for the reconfiguration. As additional information for the local user, the normal service information will be also integrated, e.g. Wang, 67296. Subsequently, this information will be sent to the industrial automation system with the intention of performing the reconfiguration.

## 5.8 System Recovery after the Reparation

System recovery is concerned with applying reasonable repairs to eliminate the fault source and bring the industrial automation system back to normal operation. Due to the reconfiguration of the logical structure in the industrial automation system, namely the change in the software,

although the hardware component has been replaced, its corresponding software is also in need of new update.

A unified recovery command ought to be created. Firstly, all functions can be activated, and redundant functions brought back to the standby state, e.g. 0x111, which makes it unnecessary to analyze the availability of ongoing tasks. It can integrate a simple command to activate all tasks in the industrial automation system, e.g. “@task:AllTrue”. To avoid potential recovery faults, this thesis suggests that a restart is always required after the system recovery. The specific code for some specific functions will have been deactivated and the assistant instruction eliminated. Actions for the reconfiguration that have been manually performed by the user, such as reopening the closed valve in the water pipe, will, on the other hand, also be performed.

In order to summarize the entire conception of this chapter, Figure 5.23 outlines the working sequence of dealing with faults. When a fault in the industrial automation system occurs, the existing fault diagnosis system detects the fault, stops the industrial automation system, and performs the previous fault diagnosis. Then it connects with the dynamic fault handling and reconfiguration system and sends the fault information to it. The EFDS shows the user the occurrence of a fault at the same time. The user can contact maintenance service to repair the system.

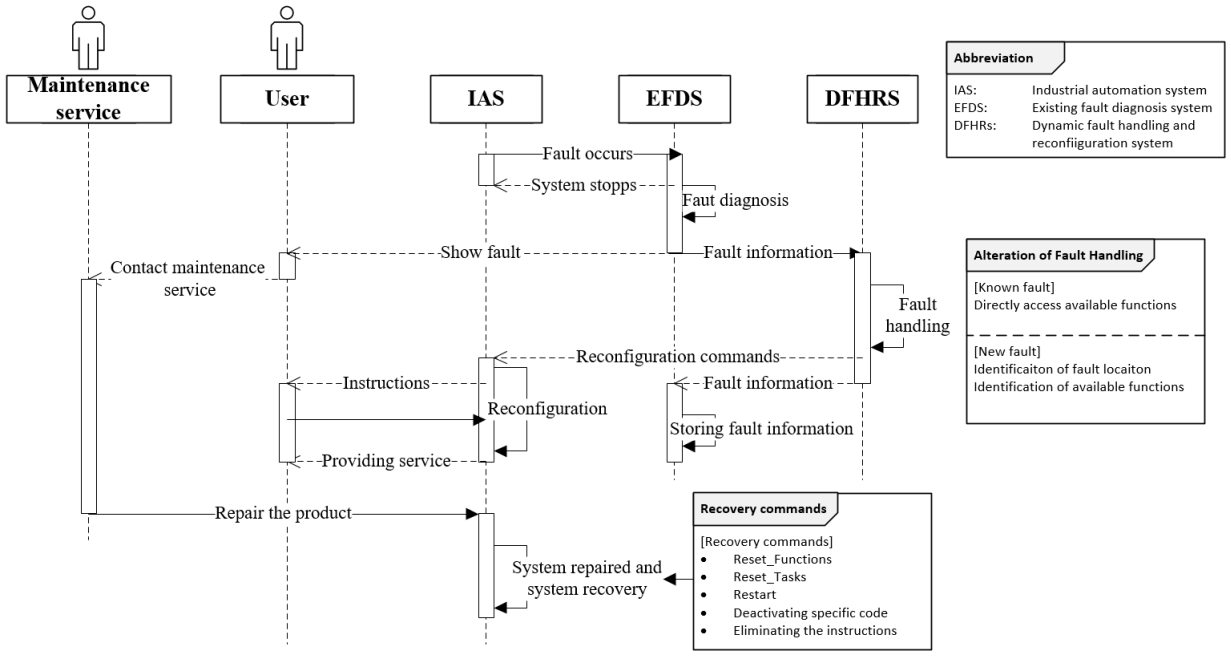



Figure 5.23: Sequence Diagram for handling faults via the dynamic fault handling and reconfiguration system

With regard to the fault type, i.e. known fault or new fault, different analytic processes will be performed to identify available functions. If it is a known fault, it accesses the fault knowledge directly and generates the reconfiguration commands. If it is a new fault, two main steps have to

be taken: identification of fault location with the help of symptom knowledge and fault information including previous fault diagnosis results and historical data, and identification of available functions by means of the system knowledge including component model, function model, and requirement model. Then the reconfiguration commands will be generated for functions, ongoing tasks and other necessary information after the appraisal of the availability of ongoing tasks and the necessity of the redundancy and service information. Afterwards, the new fault information will be stored in the fault knowledge database. In the meantime, the reconfiguration commands and the new fault information will be sent to the local. With help of the interpretation of the reconfiguration commands, the industrial automation system can perform the reconfiguration with still available functions and provide further services for the user. In addition, some reconfiguration requires the help of the user who can follow the provided instruction to do certain specific actions. After the repair by the maintenance service, the industrial automation system extracts the recovery commands, which includes resetting functions, resetting tasks, restarting, deactivating specific code and eliminating the instructions, and performing the recovery of the logical structure to bring the industrial automation system back to normal operation mode.

# 6 Realization and Evaluation of the Conception

The aim of this chapter is to describe the technical realization of the conception of the dynamic fault handling and reconfiguration system, to evaluate the conception with the help of three demonstrators, and to assess the conception on the basis of the requirements. To describe the realization of the conception, the next section will explore this topic from the following perspectives: system architecture, software architecture, and realization of the fault handling knowledge in a database. The conception will be evaluated by qualitative aspects with regard to the correctness of required functionalities as well as quantitative aspects concerning the increasing availability of industrial automation systems. Afterwards, the three demonstrators will be presented with reference to the simulators, the combination and evaluation. Finally, the conception of dynamic fault handling and reconfiguration will be qualitatively evaluated concerning the predefined requirements.

## 6.1 Realization of the Conception

Referring to the realization of the conception, this section attempts to present it from the following aspects: system architecture, data type, and realization of the knowledge including the local fault knowledge as well as the knowledge that is stored on a server. This contains symptom knowledge, fault knowledge and system knowledge. In the following section, the realization of various functionalities in the dynamic fault handling and reconfiguration system will be outlined with specific examples and figures.

For the purpose of evaluating the proposed conception, three applications were developed. These were evaluated by empirical ascertained results, allowing the conception of the dynamic fault handling and reconfiguration system to be realized. The following student works in Appendix A have been performed during the research process. These works can be classified as investigation works, conception test works, system development works based on specific demonstrators, and further improvement and application works.

### 6.1.1 Overview of the System Architecture

Due to features like robustness, platform independence and security, the implementation of the conception of dynamic fault handling and reconfiguration was based on the programming language Java. As mentioned in Chapter 4, this thesis proposes a conception of handling faults automatically, making it possible to serve all distributed industrial automation systems of the same type worldwide. This thesis suggests a server as the platform for the implementation of the presented conception because a server can be installed at the location of the manufacturer to facilitate easy maintenance. With help of a server, a centralization of control can be attained.

The Apache Tomcat is an application server that provides Java Servlet, JavaServer Pages (JSP), the Java Expression Language, and Java WebSocket technologies. Due to the features of open source, it affords a pure Java HTTP webserver environment. The Tomcat server supports the HTTP protocol that allows information exchange through the Internet as well as a JSP engine (named Jasper) which can compile JSP files into Java code. Because of its web technologies, it provides plenty of possibilities of extension with the intention of the utilization of web access, e.g. for maintenance. To increase the security of the server, the Tomcat server provides an additional layer of security [McHa02]. Hence, this thesis uses the Tomcat server as the platform to run the Java applications.

To store historical data, local fault knowledge, fault knowledge in server, symptom knowledge and system knowledge, this thesis utilizes a MySQL database. Due to its security and reliability, the MySQL database is very suitable for a server application. Additionally, the MySQL database is a relational database which is helpful in building the database for the system knowledge.

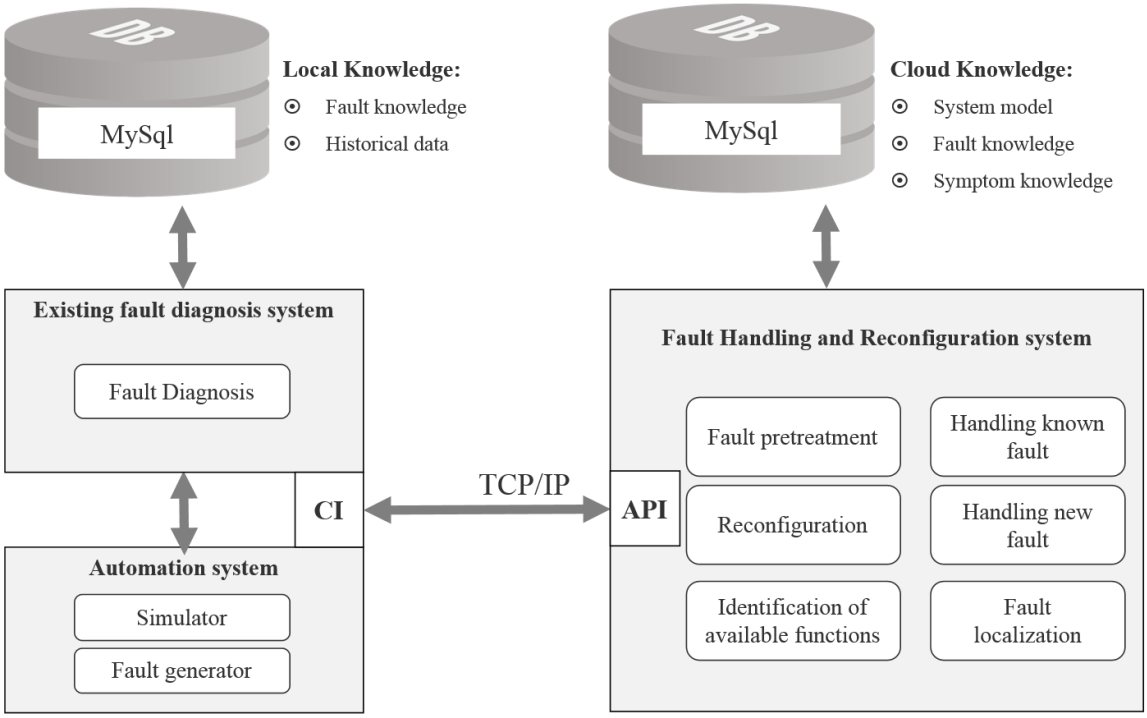



Figure 6.1: System architecture of the deployment

Figure 6.1 shows an overview of the system architecture of the dynamic fault handling and reconfiguration system. The system architecture of the deployment consists of the application of the dynamic fault handling and reconfiguration system, the application of an automation system, the application of existing fault diagnosis system (EFDS), the local knowledge, and the server knowledge. To deal with a fault, either known or new, the server part of the dynamic fault handling and reconfiguration system consists of six main modules, as well as an application programming interface (API) which is necessary for the communication with the local communication interface (CI) via the TCP/IP protocol. The EFDS application realizes the fault diagnosis functionalities of

detecting faults and creating the previous diagnosis results. The application of the automation system is able to simulate the behaviors of a real industrial automation system as well as to generate various faults. The communication between the EFDS and the automation system is based on an internal API.

### 6.1.2 Software Architecture

Figure 6.2 depicts the software architecture of the dynamic fault handling and reconfiguration system which contains the services of the server. The example regarding the software architecture of Figure 6.2 has already been simplified. To realize the software application, eight major classes are designed and developed. Four classes realize the functionalities of handling faults. The class “MainDFHRS” is the core processing junction for the different classes and is responsible for the previous handling of a fault, identifying the fault type (either known or new), communicating with different information sources to acquire different information, transferring information for different classes, and calling on the other three assistive classes to complete the reasoning work. For a known fault, the available functions will be directly acquired via the program function “JSONObject handleFault()”. For a new fault, the program function will access the class “FaultLocalization” and the class “FunctionAnalysis” with the intention of achieving the reasoning work. The required knowledge for reasoning is transferred by the class “MainDFHRS”. The class “ReconfCommandGenerator” takes charge of generating reconfiguration commands after confirming available functions and available tasks.

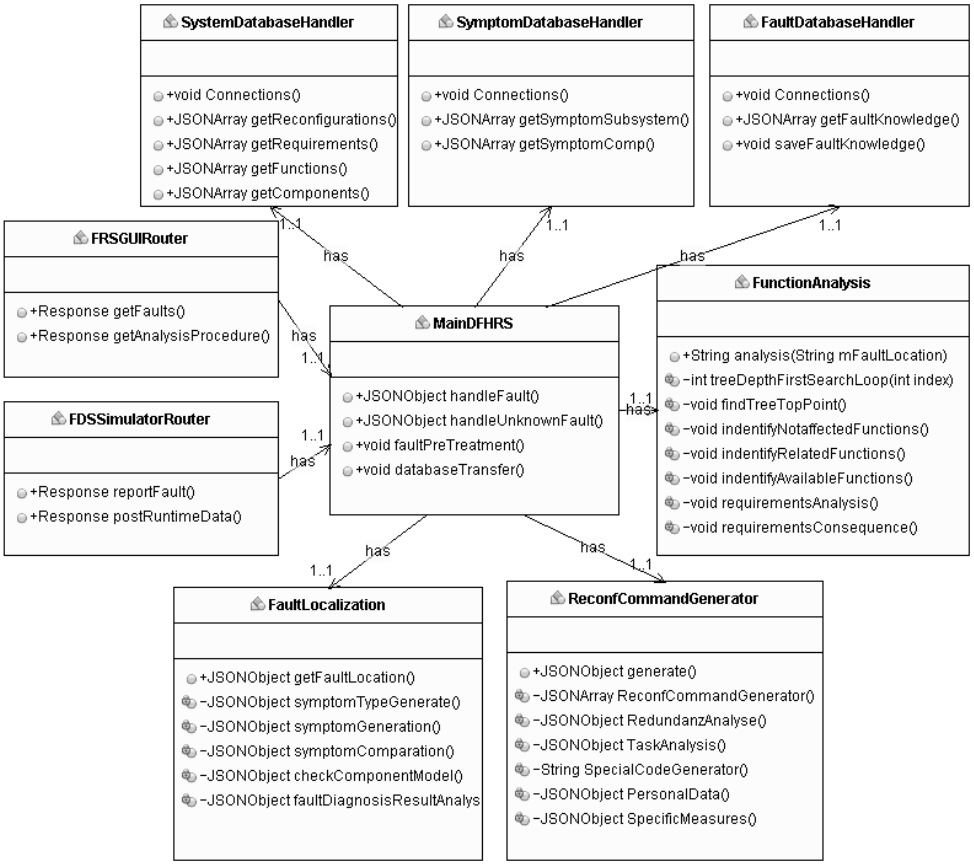



Figure 6.2: Software architecture of the realized dynamic fault handling and reconfiguration system

The class “FRSGUIRouter” and the class “FDSSimulatorrouter” are APIs for transferring information. “FRSGUIRouter” transfers information for the class “MainDFHRS” and the class “GUI” which is not shown in the example. “FDSSimulatorrouter” is in charge of exchanging information with external systems including EFDS and industrial automation systems. The classes “SystemDatabaseHandle”, “SymptomDatabaseHandler” and “FaultDatabase Handler” attempt to access the database to acquire system knowledge, symptom knowledge and fault knowledge, as well as to update the fault knowledge in case of a new fault. Beside the class “GUI”, another useful class “Demonstration” is also included in the figure. These two classes depict the fault processing procedure and visualize the procedure in the form of animations.

### 6.1.3 Realization of Fault Handling Knowledge

As presented in Chapter 5, the fault handling knowledge includes three knowledge types (KType), i.e. system knowledge (SSK), symptom knowledge (SMK), and fault knowledge (FK). Appendix B lists the necessary tables of the fault handling database.

In addition, it is worth mentioning that the number of symptom tables depends on both the hierarchy of the component model and the number of the subsystems. Appendix B denotes the symptom table regarding only three layers and several subsystems.

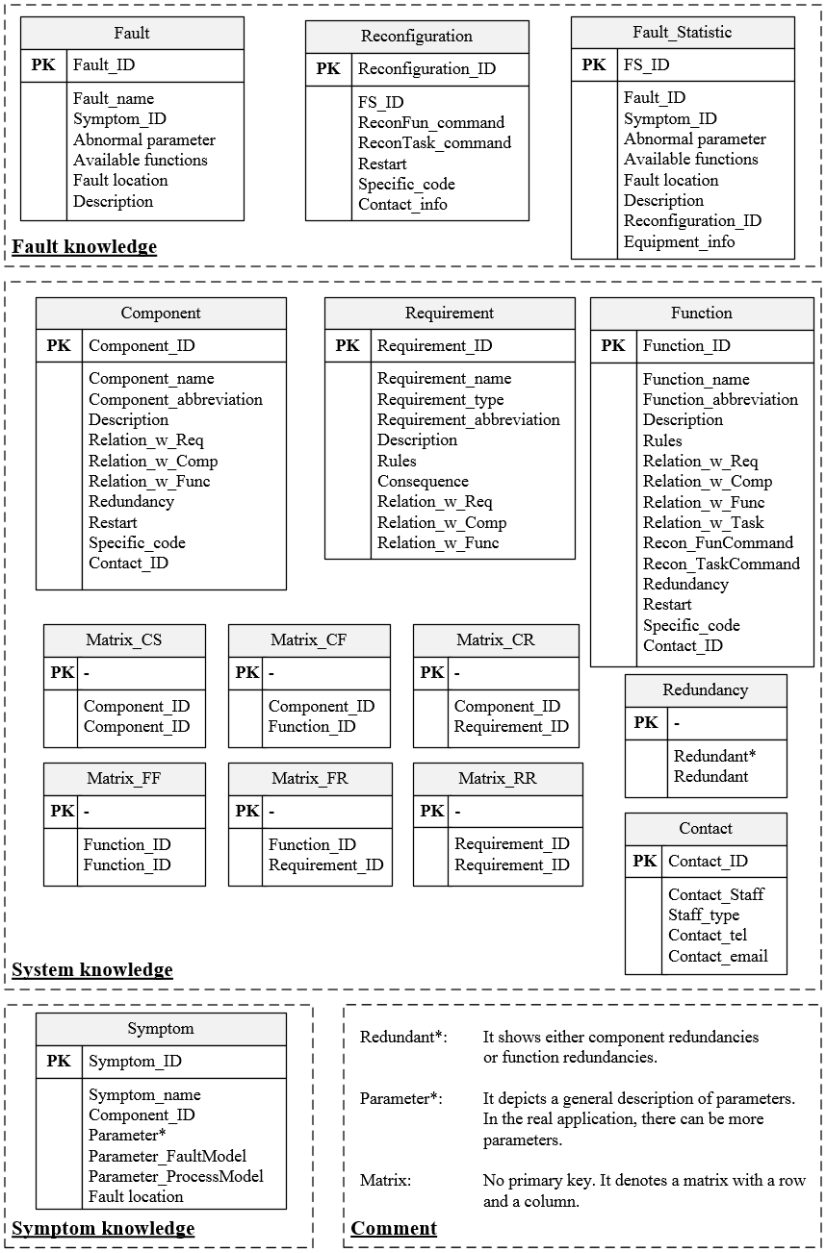



Figure 6.3: Implementation of the fault handling knowledge via MySQL

Figure 6.3 shows the fault handling knowledge in form of database tables which are realized with the MySQL software. The attributes of each table are listed in the figure. This figure contains 15 tables in total. However, due to an uncertain number of symptom tables, this number can be more than 15 tables in the realization. Additionally, the attribute “Parameter*” in the symptom table is a generic term for parameters. It denotes various parameters, such as temperature, liquid, pressure, etc. Furthermore, it denotes various features for one parameter, such as abnormal parameter, error, change tendency, change rate, etc. In a real application, the attribute “Parameter*” ought to multiply depending on the specific application.

### 6.1.4 Realization of Data Format and Communication Type

As described above, the dynamic fault handling and reconfiguration system utilizes the Tomcat server as the application platform to handle service requests from multiple industrial automation systems. However, for handling faults, especially a new fault, a large amount of information is transferred in the process of the communication among EFDS, dynamic fault handling and reconfiguration system and industrial automation systems. For instance, this includes historical data which comprises a mass of sensor data in a specific time interval. In addition, due to distance between the server and local industrial automation systems, and in light of the wide popularization of the internet nowadays, the internet is a worthy choice as the communication media [Shyr12]. The realization of the information exchange, the data format and the communication type are discussed in this chapter.

This thesis chooses JSON (JavaScript Object Notation) as the data format to represent the information and data of the dynamic fault handling and reconfiguration system. JSON is a lightweight data exchange format. On the basis of a subset of the ECMA Script, it utilizes a text format completely independent of the programming language to store and represent data. JSON is recognized as an ideal data exchange language for this purpose because of its simple and clear hierarchical structure. The data in the JSON format is extremely easy to comprehend and can be generated by either humans or machines. Due to its lightweight feature, the utilization of JSON can effectively improve network transmission efficiency [DuSi16]. Figure 6.4 shows two simplified examples of the application of JSON as a data format in the dynamic fault handling and reconfiguration system. The left figure depicts the reconfiguration commands, and the right shows the data of a temperature sensor at a specific time.

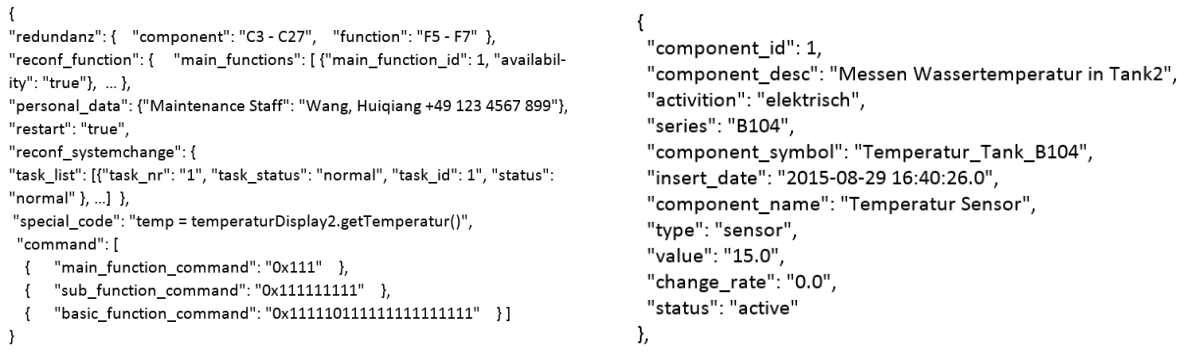



Figure 6.4: Reconfiguration commands (left) and historical data (right) in the JSON format

In order to realize the communication with the dynamic fault handling and reconfiguration system server via TCP (transmission control protocol), the following three possible communication types can be considered:

- WebSocket: This is a full duplex communication protocol based on TCP connection. It can realize data exchange between clients and servers and allows the server to push data to clients

initiatively. To build the connection between one server and one client, it requires only a handshake with an acknowledgement (ACK), allowing the client and the server to directly establish a permanent connection to transfer data to each other until the connection is initiatively broken by one of them [BaMa14].

- HTTP: This is an application-layer protocol for distributed collaborative, hypermedia information system [BMR17]. HTTP is a request-and-response standard for clients and servers. Used in the context of the internet, a client pulls a request and builds the connection with a specific UDP port of the server. Once the server receives the request, the server responds with a status and the required information.

- HTTPS: HTTPS is similar to the HTTP protocol. The superiority of HTTPS is the higher security which is guaranteed by a certification mechanism and a data encryption technology.

WebSocket demands a permanent connection, which can result in other industrial automation systems being blocked. To assure a secure and unblocked communication, this thesis chooses HTTPS as the communication type. With the help of an HTTPS protocol, two request methods from eight specified by HTTP/1.0 [BFN45] and HTTP/1.1 [FiRe14] are chosen for the realization of the communication between the dynamic fault handling and reconfiguration system server and local EFDS, as well as the local industrial automation system. The chosen request methods are GET and POST. The former can be used for communication in which the request does not pack any resource data. The latter can be applied for communication when some data needs to be packed in the request, for instance, historical data by EFDS. The application format of the two methods are as follows: *GET - Http://<IP_DFHRS>/FRS/status; POST - Http://<IP_DFHRS>/FRS/reportFault; <Request > →JSON Data packet {fault_ID: "0X00000"}; <Response > →JSON Data packet {Recon_Function: "0X10011111"}.*

### 6.1.5 Development of Interfaces between local Systems and a Server

It is necessary to establish a generic interface within the local system to realize the communication. The communication interface (CI) consists of three modules: CI_EFDS to realize the communication with the EFDS, CI_CentralControl to realize the communication with the central control module of the industrial automation system and CI_DFHRS to achieve the data exchange with the dynamic fault handling and reconfiguration system. Subsequently, the CI is in charge of the data exchange between different systems. In consideration of different protocols, the CI is able to package data in a specific manner and interpret protocols [FA2722] such as JSON.

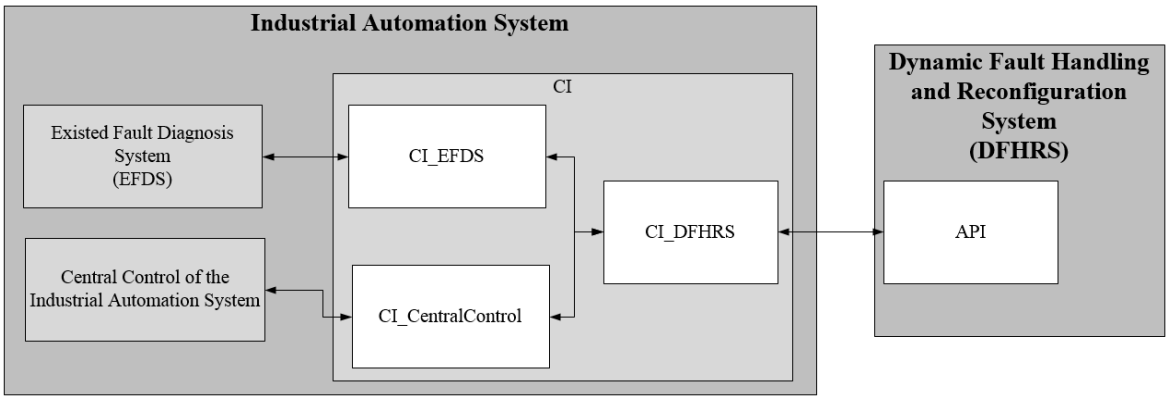



Figure 6.5: Establishing a communication interface (CI) [FA2722]

To complete the data exchange, i.e. handling faults, the communication is divided into two major steps in this thesis. To establish the connection between CI and API via HTTPS, an initialization process is performed (see Figure 6.6). The local CI receives the connection request from the EFDS and sends a specific command, which utilizes the GET method, i.e. *GET - Http://<IP_DFHRS>/FRS/status*, to inquire about the status of the dynamic fault handling and reconfiguration system, if it is running and free at the moment. If the dynamic fault handling and reconfiguration system is running and free, then the dynamic fault handling and reconfiguration system gives a response to the local system with the following information, *Response – JSON: {status: "running"}*. Conversely, if the dynamic fault handling and reconfiguration system is busy, it sends a response to the local system with the following information, *Response – JSON: {status: "busy"}*.

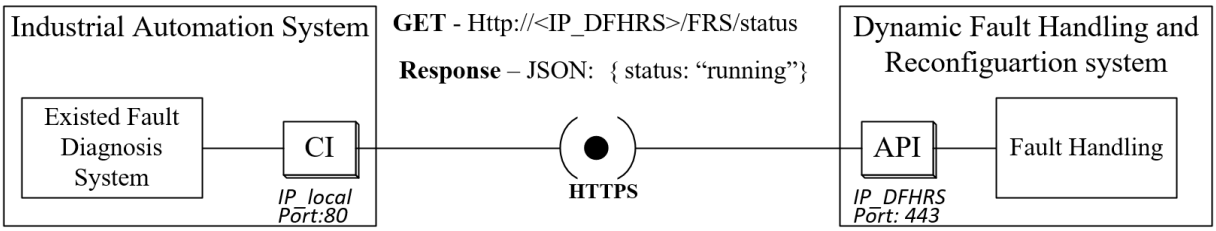



Figure 6.6: Initialization of the connection between CI and APT via HTTPS

On the basis of a successful connection, the fault handling process will be carried out as follows (see Figure 6.7): Depending on the fault type, the exchange information is differentiated. In case of a known fault, the local EFDS posts the fault information with the fault ID to the dynamic fault handling and reconfiguration system. Subsequently, after analysis of the fault, the dynamic fault handling and reconfiguration system returns a response with reconfiguration commands, e.g. *<Response > →JSON Data packet {Recon_Function: "0X10011111"...}*. In case of a new fault, the local EFDS sends the fault information with a specific fault ID and the previous fault diagnosis result, such as "fault ID = 0x00000". Because of a new fault, the dynamic fault handling and reconfiguration system responds with status information "success" to the EFDS and waits for the historical data from the EFDS. Having received the response, the EFDS packages the historical data and reports it to the dynamic fault handling and reconfiguration system. With the historical

data and fault information, the fault handling process can be carried out. In the response package, there are two parts: the fault information with the new fault ID, and the reconfiguration commands. With the help of the former information, the local fault knowledge base will be updated. The latter is helpful for guiding the reconfiguration of the industrial automation system.

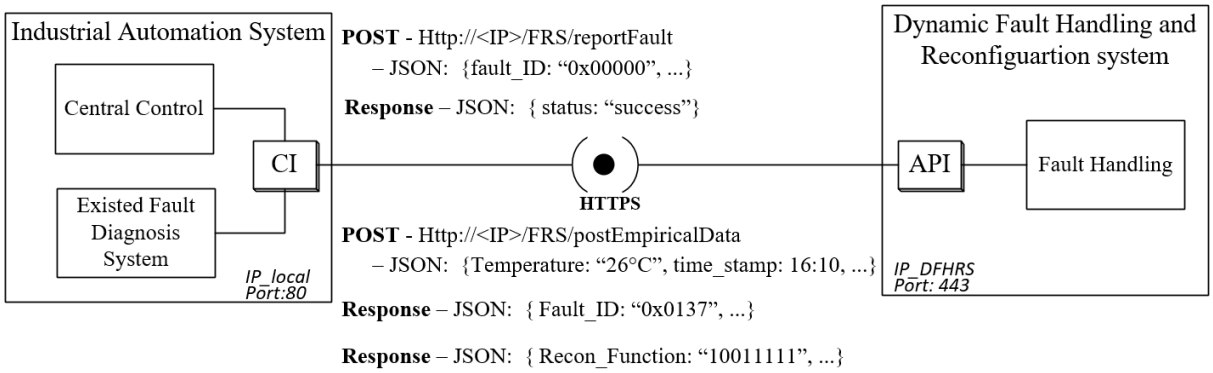



Figure 6.7: Communication via CI and API for handling a fault

Additionally, it is worth noting that the initialization of the connection does not mean that a long term connection is built up. Actually, the connection is closed after the response. The advantage of such a connection is that it can avoid an overlong occupation of the communication channel by one industrial automation system. Thus, it can avoid some potential faults, such as long time suspension of the system, and improve the work efficiency of the dynamic fault handling and reconfiguration system.

### 6.1.6 Prototype of the Conception

To realize a user-friendly interaction with the system, a flexible interface is needed. Thus, this section outlines the realization of various user interfaces of the dynamic fault handling and reconfiguration system and explains the realization of the corresponding functionalities.

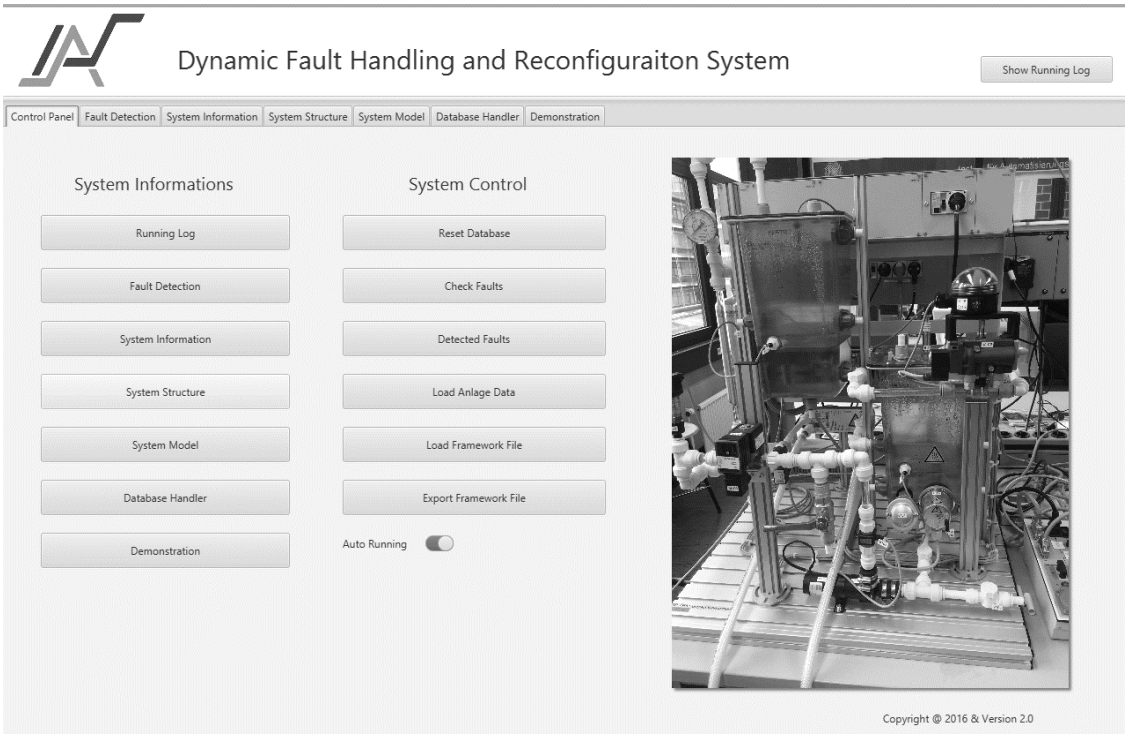



Figure 6.8: User interface of the dynamic fault handling and reconfiguration system [MA2800] [MA2913]

Figure 6.8 depicts the user interface of the dynamic fault handling and reconfiguration system. This prototype is based on the fault handling for the two-tank system. The flexible design of the UI allows experts to check the status of the dynamic fault handling and reconfiguration system, such as the procedure of the fault localization or the procedure of the identification of available functions.

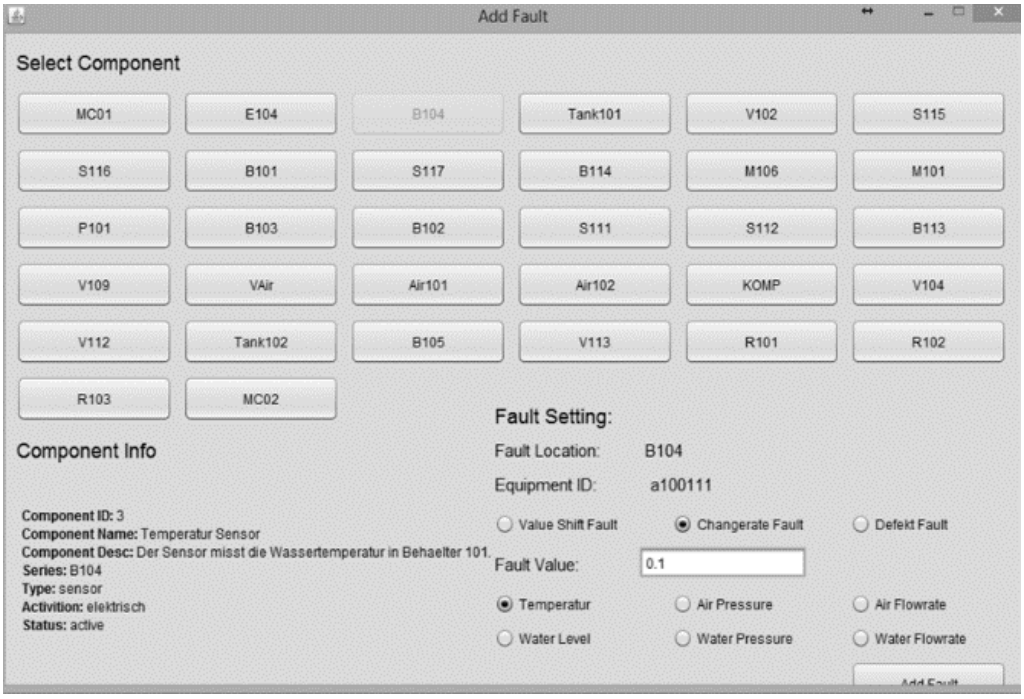



Figure 6.9: Adding faults into the industrial automation system

Simulating a fault requires the tester to add the simulated fault into the industrial automation system, such as the two-tank system simulator. As Figure 6.9 illustrates, the interface provides a choice panel for every component in the two-tank system. After choosing a component, the tester is able to specify the characteristics of the fault by using different attributes, making it possible to realize, for instance, a change rate fault for the temperature in case of the fault of a temperature sensor.

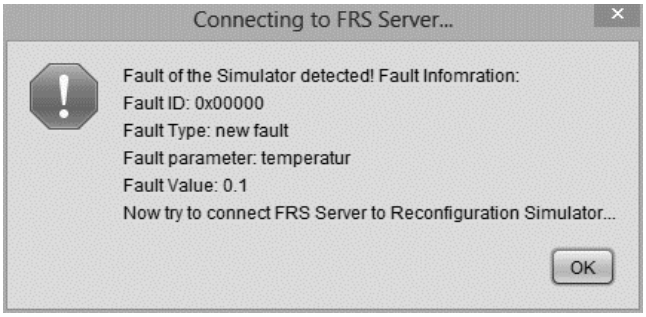



Figure 6.10: Primary fault diagnosis results of the EFDS

The local existing fault diagnosis system usually monitors the industrial automation system continuously. If a fault occurs, the EFDS performs the fault diagnosis approaches [Frie15] to detect the fault, and depicts primary fault diagnosis results, such as fault ID, fault parameters, etc. Figure 6.10 highlights the fault diagnosis results.

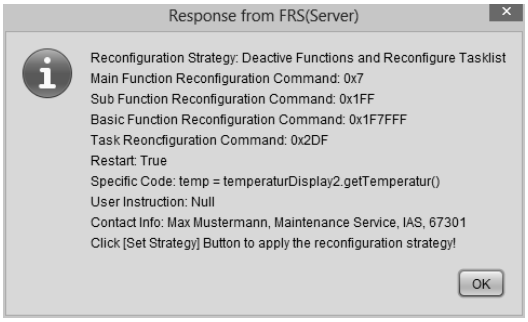



Figure 6.11: Simplified Reconfiguration commands

With the received fault information and system run-time tasks, the dynamic fault handling and reconfiguration system analyzes the fault, identifies available functions and tasks and generates the appropriate reconfiguration commands for functions and tasks. Additionally, necessary information (see Figure 6.11), such as user instructions [Frie15] and contact information of the maintenance staff, are displayed for the user of the system.

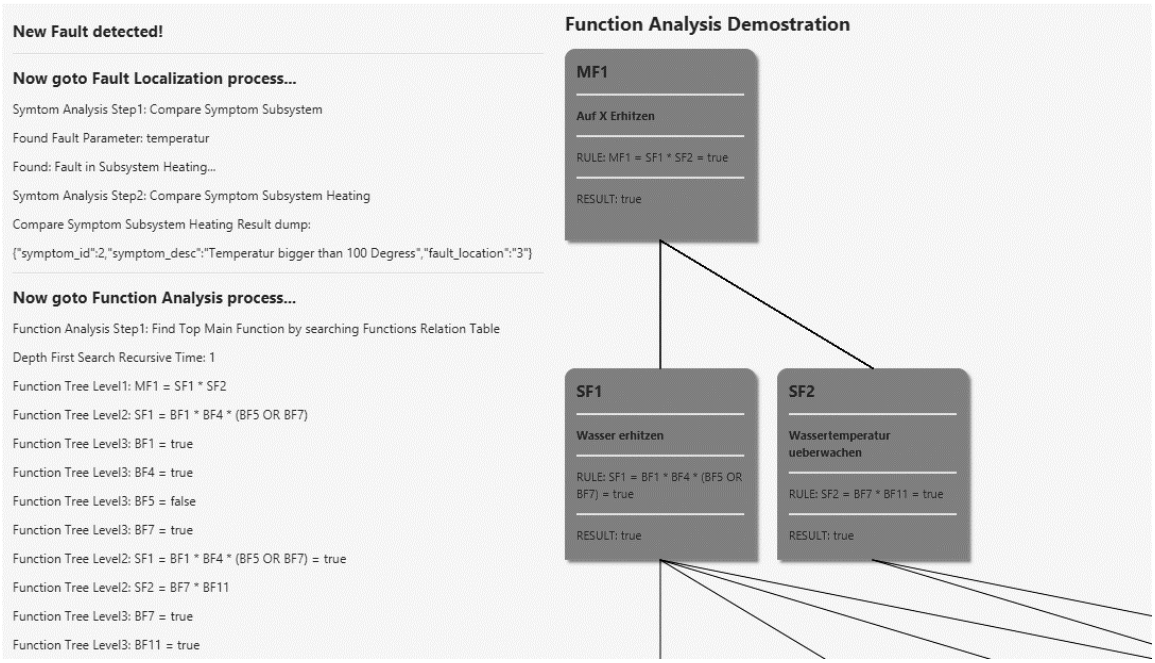



Figure 6.12: Fault handling procedure with the demonstration

Figure 6.12 shows the fault handling procedure of a new fault. The user is able to verify whether the analysis has been performed accurately, and if the result of each step is correct. The left side of the illustration shows all the steps carried out while handling a new fault (see Chapter 5.6). The corresponding analysis results, such as the defective subsystem, and the fulfilled symptoms for current faults, are depicted accordingly. The right side gives a dynamic animation of the reasoning process with the function tree and the requirement tree by using the depth-first-search approach. In total, four colors are utilized to emphasize the current status: not checked with white, available with green, under analysis with brown, and not available with red.

### 6.1.7 Evaluation of the Conception

To evaluate the conception of the dynamic fault handling and reconfiguration system, a qualitative as well as a quantitative evaluation approach is conducted in the scope of the present thesis. The evaluation process is based on a concrete industrial automation system (simulators in this thesis). In addition to the evaluation of the dynamic fault handling and reconfiguration system, the correctness of the simulators ought to be checked as well.

**Qualitative evaluation**

The qualitative evaluation attempts to perform systematic testing to confirm the quality of projects and software. In this thesis, it aims to verify the dynamic fault handling and reconfiguration system by using specific test cases and by examining whether the dynamic fault handling and reconfiguration system was capable of performing its functionalities correctly. It is not feasible to test the dynamic fault handling and reconfiguration system by itself. Thus, it is necessary to add

at least one corresponding industrial automation system as an execution object, referred to here as the simulator.

- Evaluation of the simulator: A simulator is a program which reproduces the behavior of an industrial automation system. The following aspects should be evaluated: simulation of the processes of the industrial automation system, fault simulation, adding various faults, fault diagnosis by the local fault diagnosis system and performing the reconfiguration.

- Evaluation of the dynamic fault handling and reconfiguration system: Depending on the fault handling steps, the following aspects ought to be verified: identification of fault type (known or new), identification of fault location, identification of available functions and generation of reconfiguration commands.

- Interaction: There are two possibilities: the interaction between dynamic fault handling and reconfiguration system and the local system, and the interaction with the corresponding database. The former refers to the interaction between the dynamic fault handling and reconfiguration system and the existing fault diagnosis system, as well as the interaction between the dynamic fault handling and reconfiguration system and the simulator. The latter indicates the interaction between the dynamic fault handling and reconfiguration system and its server database, as well as the interaction between the existed fault diagnosis system and its local database.

**Quantitative evaluation**

Quantitative evaluation is the evaluation which refers to the goal of the research with mathematical characteristics, i.e. to improve the availability of industrial automation systems and assess the amount of increase in availability. The quantitative evaluation is used to determine the availability. With reference to [Stap09], the mean time to failure (MTTF) plays an important role in the calculation of the availability. Hence, this thesis presents the following two equations to calculate the availabilities:

Original availability (OA)

$$\mathrm{OA} = \left( \sum_{1}^{N} \frac{MTTF}{MTTF + MTTR} \right) \Big/ N$$

The OA attempts to calculate the average value of the original availability without the help of the dynamic fault handling and reconfiguration system; specifically, there is no reconfiguration in case of a fault, where N stands for the number of tests.

Real availability (RA)

$$\mathrm{RA} = \left( \sum_{1}^{N} \frac{MTTF}{MTTF + MTTR} + \frac{TBRBF}{MTTF + MTTR} * Naf \right) \Big/ N$$

The RA addresses the average value of the real availability with the help of the dynamic fault handling and reconfiguration system, i.e. the industrial automation system reconfigures with available functions in case of the occurrence of a fault. TBRBF is the time before repair between faults after reconfiguration. Naf is a binary parameter pointing out whether there are available functions or not. Naf equals 1 in case of available functions and 0 in case of no available functions.

In the interval of the MTTF, all functions of an industrial automation system are available. However, in the interval of the TBRBF, only the available functions are activated. The RA cannot highlight the weight of the number of the available functions for the availability. Hence, in this thesis the proportion of the available functions is used to calculate the availability for the interval of the TBRBF. The real availability concerning the proportion of available functions is calculated as follows:

$$\mathrm{RA_{pf}} = \left( \sum_{1}^{N} \left( \frac{MTTF}{MTTF + MTTR} + \frac{TBRBF}{MTTF + MTTR} * \frac{Maf}{Mf} \right) \right) \Big/ N$$

Maf is the number of available functions and Mf stands for the total number of functions.

To highlight the change in the availability, the difference value of the availability (D_Availability) is additionally calculated using the following equation:

$$\mathrm{D\_}Availability = RApf - OA.$$

In order to simplify the calculation process, the following assumptions have been made: 1 hour for the MTTF, 0.5 hour for the MTTR, 5 minutes to replace the defective component, and 25 minutes for working with available functions. Due to the fact that the fault handling time is very short, the time span between the appearance of a fault and the reconfiguration can be ignored.

As introduced in 2.1.1, industrial automation systems are classified into three types. To check the universality of the proposed conception (see Requirement 5 in Chapter 2), this conception was implemented with three demonstrators:

- A two-tank system simulator simulating the two-tank system and representing the continuous process type of industrial automation systems.
- A coffee maker simulator simulating the coffee machine and representing the sequential process type of industrial automation systems.
- A high-bay warehouse simulator simulating the high-bay warehouse and representing the discrete object type process type of industrial automation systems.

## 6.2 Evaluation of the Conception on the Two-Tank System Simulator

For the evaluation, three simulators for I, II and III were developed in line with the research. These simulators were combined with the developed dynamic fault handling and reconfiguration system and were tested in many test cases.

I. A two-tank system simulator, simulating the two-tank system and representing the continuous process type of industrial automation systems.

II. A coffee maker simulator, simulating the coffee machine and representing the sequential process type of industrial automation systems.

III. A high-bay warehouse simulator, simulating the high-bay warehouse and representing the discrete object process type of industrial automation systems.

In this subsection, the realization of the dynamic fault handling and reconfiguration system on the basis of a two-tank system simulator will be presented. Subsequently, the realization of the connection between the dynamic fault handling and reconfiguration system and the simulator will be outlined. Finally, the evaluation of the system will be presented.

**Realization of the two-tank system simulator**

The two-tank system simulator recreates the real two-tank system which includes two tanks, a heater, a temperature sensor, four photoelectric liquid level sensor, one ultrasonic liquid level sensor, an air pressure switch, a flow transducer, a pump, a pressure sensor, an electrical proportioning valve and some pipes. The simulator is illustrated in Figure 6.13.

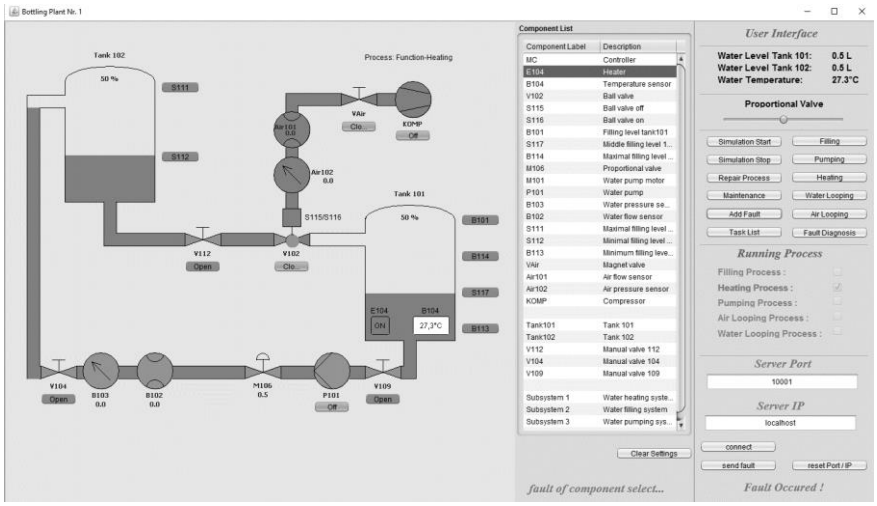



Figure 6.13: Simulator of the two-tank system [MA2800]

The process in the original two-tank system is controlled by a specific Java application which runs on an industrial computer [Bord16]. To maintain consistency with the assumption in the

conception, a microcontroller is simulated as the central controller, replacing the industrial computer. Additionally, four processes are implemented in the simulator as follows:

- Process1 (injecting process): It attempts to inject the water from tank 102 to tank 101 at a specific volume. Between two tanks, the air pressure switch (v102 in figure 6.13) is responsible for controlling the flow through the pipe. It provides the sub function “Injecting water”.
- Process2 (air inflating process): It provides the air for the air pressure switch which keeps the pressure at 6 Pa. It offers the sub function “Providing air”.
- Process3 (heating process): It tries to heat the water in tank 101 to a specific temperature according to the demand of the user. It affords the sub function “Heating water”.
- Process4 (water draining process): It is responsible for draining water from tank 101 to tank 102 because there is no other output pipe to keep a continual simulation. The discharge of the draining is equal to the volume of the injecting water by default.

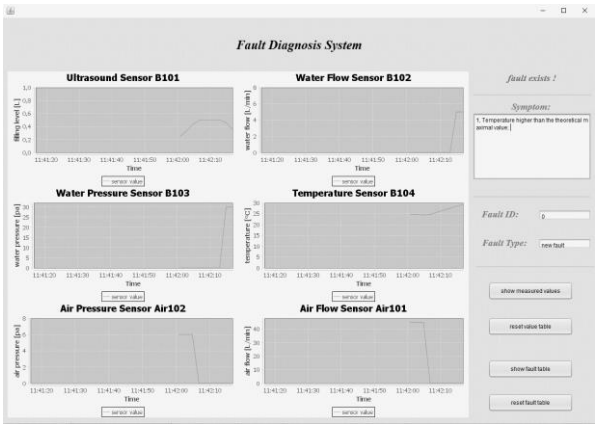



Figure 6.14: Simulator of the existing fault diagnosis system [MA2800]

To realize the fault diagnosis and the monitoring of the system, a Java application for a fault diagnosis system is developed. Figure 6.14 depicts the GUI of the developed fault diagnosis system. The application of adding a fault was shown in Figure 6.9. In the case of the appearance of a fault, the EFDS can detect the fault and generate the fault information (see Figure 6.10).

The simulator, which includes the two-tank system and the fault diagnosis system, creates the basis for the further evaluation.

**Combination of the dynamic fault handling and reconfiguration system and two-tank system simulator**

In the realization, the Java applications of the dynamic fault handling and reconfiguration system and the simulator run on two computers individually. For the purpose of establishing the connection between the dynamic fault handling and reconfiguration system and the two-tank

system simulator, an Internet router is utilized as communication media. It runs three individual applications, i.e. the two-system simulator, adding faults, and the existing fault diagnosis system (EFDS). The EFDS monitors the real-time data from the two-tank system simulator and stores it in the database as historical data. In addition, the fault knowledge affords the necessary verification knowledge for the fault diagnosis. With the help of the Java application of adding faults, it provides the possibility to add various faults.

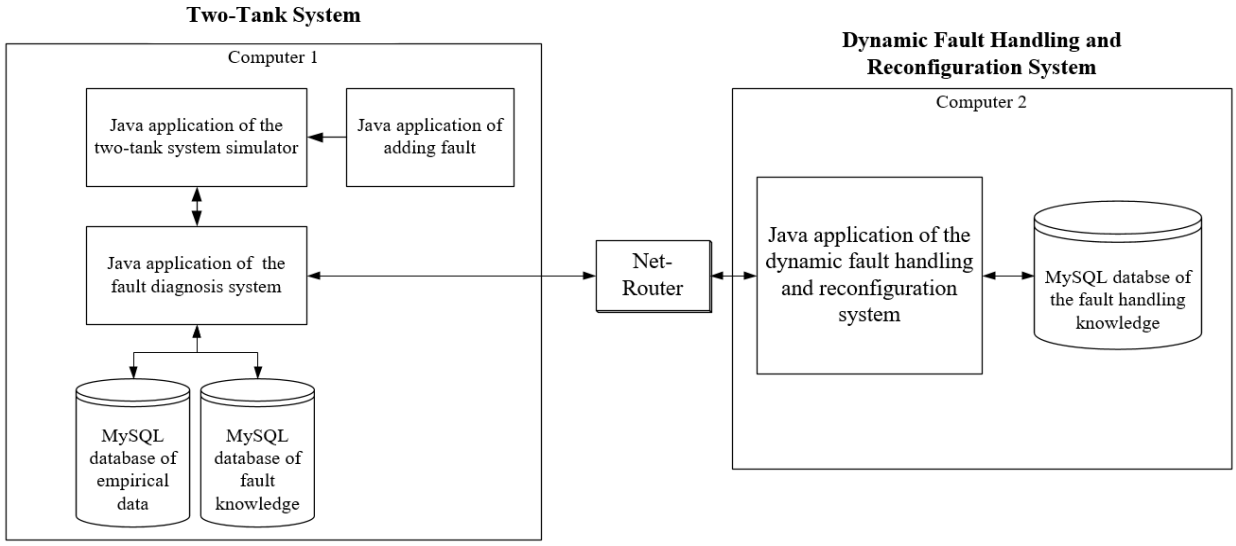



Figure 6.15: Combination between the dynamic fault handling and reconfiguration system and the two-tank system simulator [MA2800]

Moreover, the computer furnishes the operation system to run the Tomcat server as the platform for the DFRHS application. The fault handling knowledge including fault knowledge, symptom knowledge and system knowledge is simultaneously stored in the MySQL database.

**Evaluation of the conception on the two-tank system simulator**

In order to evaluate the functionalities of the dynamic fault handling and reconfiguration system on the basis of the two-tank system simulator regarding the qualitative aspect, 12 test cases were defined and carried out to evaluate the correctness of the two-tank system simulator (TSS), its corresponding fault diagnosis system (FDS) and the dynamic fault handling and reconfiguration system (DFHRS).

Table 6.1: 12 Test cases for the evaluation of the developed software [MA2800, pp. 54-68]

| Test case | Test object | Test objective |
|---|---|---|
| **Client initialization** | TSS, FDS, DFHRS | Testing the correctness of GUI of TSS and FDS; testing the connection between local and server |

| | | |
|---|---|---|
| **Server initialization** | DFHRS | Testing the correctness of GUI of the DFHRS; testing connection between DFHRS and its database |
| **Behavior of the two-tank system** | TSS, FDS | Testing if the simulation of the behavior of TSS is correct; testing the correctness of recording historical data |
| **Fault detection** | FDS | Testing the correctness of process monitoring, data extraction and fault diagnosis by FDS |
| **Adding fault** | TSS, FDS | Testing the correctness of the data change of TSS and diagnosis result of FDS |
| **Sending historical data** | FDS, DFHRS | Testing the correctness of the data exchange between FDS and DFHRS |
| **Sending reconfiguration data** | DFHRS, TSS | Testing the correctness of the data exchange between DFHRS and TSS |
| **Reconfiguration** | TSS | Testing the correctness of performing the reconfiguration by TSS |
| **Updating the fault knowledge** | DFHRS, DFS | Testing the correctness of fault knowledge exchange between DFHRS and FDS; testing the correctness of updating local fault database |
| **Handling a known fault** | DFHRS | Testing the functionalities of DFHRS for handling known faults, including accessing the fault knowledge, identification of available ongoing tasks and generation of reconfiguration commands |
| **Handling a new fault** | DFHRS | Testing the functionalities of DFHRS for handling new faults, including fault localization, identification of available functions, automatic reasoning, identification of available ongoing tasks and generation of reconfiguration commands |

With the help of the introduced test cases, the basic functionalities of the software were tested successfully [MA2800], providing a basis for performing a quantitative test in which 100 random faults were fabricated, including 40 single component faults for 26 components, 10 single subsystem faults for 4 subsystems, and 50 multiple faults. As introduced in Chapter 6.17, due to the objective of the research, the two-tank system simulator can be reconfigured based on the available functions. The reconfiguration is conducted in the time span from the appearance of a fault to its repair. However, for some faults, the dynamic fault handling and reconfiguration system cannot provide available functions, since the defective component can cause a loss of all functions, such as a microcontroller, etc. After the test, 83 faults could be processed, i.e. the two-tank system simulator was able to be reconfigured with the available functions. The rest could be

processed so that the two-tank system simulator had to maintain the stop status. This thesis makes assumptions concerning the following parameters: 1 hour for MTTF, 0.5 hour for MTTR and 5 minutes for replacing a defective part. Based on the equation in Chapter 6.17, the calculation result of the availability of the two-tank system simulator is presented below.

Table 6.2: Availability of the two-tank system simulator

| Availability | Percent |
|---|---|
| Original availability (OA) | 66.67% |
| Real availability (RA) | 89.72% |
| Real availability concerning the proportion of available functions (Rapf) | 87.46% |
| D_Availability | 20.79 % |

Table 6.2 shows the availability of the two-tank system simulator. Moreover, by means of the integrated timers in the program, the average fault handling time from receiving the request to sending a response is 465.72ms. Hence, the following conclusion can be derived: the dynamic fault handling and reconfiguration system can handle a fault very rapidly.

As a result, the three software tasks perform flawlessly based on the qualitative and quantitative test. Each functionality of the dynamic fault handling and reconfiguration system achieved its expectation, as evidenced by the test cases. Not only the known faults but also new faults were located successfully. The two-tank system can be reconfigured smoothly and availability can evidently be enhanced. According to the evaluation results of the two-tank system simulator, an industrial automation system of the continuous process type can be successfully maintained and reconfigured by the dynamic fault handling and reconfiguration system with the available functions. The objective of the research, i.e. an increase in availability, can be achieved.

## 6.3 Evaluation of the Conception on the Coffee Maker Simulator

After the two-tank system simulator, the dynamic fault handling and reconfiguration system conception was realized with the coffee maker simulator, which simulates the industrial coffee maker at the Institute of Industrial Automation and Software Engineering, i.e. the WMF CombiNation S [MT2782] [SA2861][SA2721]. The coffee maker simulator is presented below. The concept of the dynamic fault handling and reconfiguration system will be evaluated later based on the coffee maker simulator.

**Description of the coffee maker simulator**

The coffee maker simulator represents industrial automation systems of the sequential processes type. The coffee maker enables the production of different products, for instance cappuccino, espresso, coffee with milk, hot water, etc. The structure of the coffee maker is outlined in the following figure.

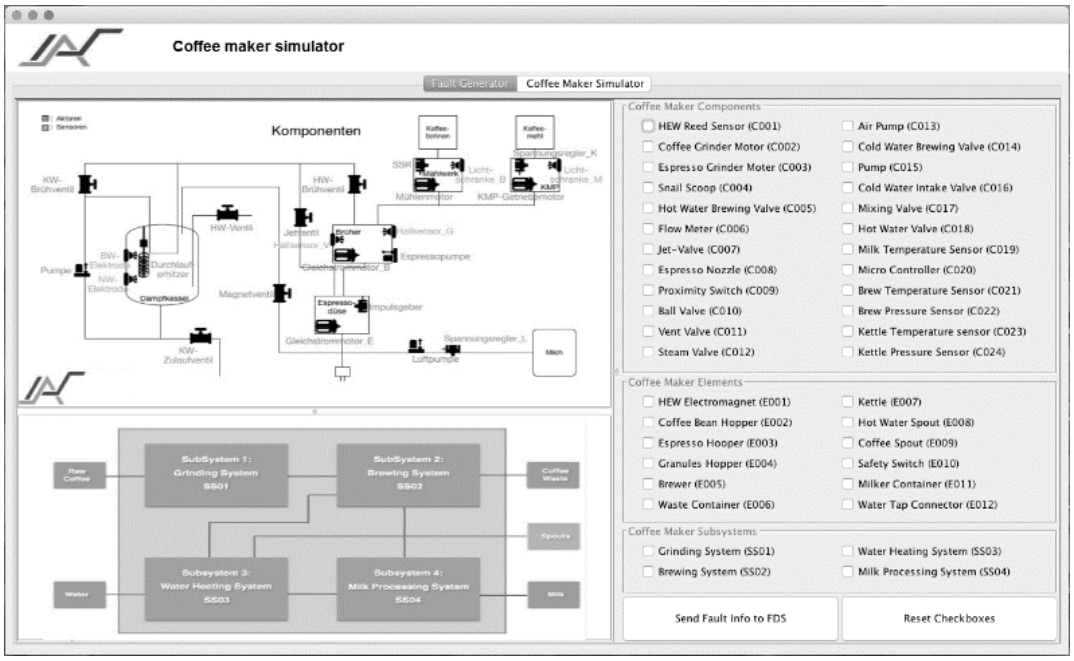



Figure 6.16: System structure of the coffee maker simulator [MT2782] [SA2721]

The coffee maker consists of four subsystems.

- Subsystem1: The grinding system is in charge of grinding the coffee beans into coffee powder for the further brewing.
- Subsystem2: The brewing system is responsible for controlling the brewing temperature, the brewing pressure, the proportion of coffee powder and hot water for specific coffee types, and the collection of coffee residue.
- Subsystem3: The water heating system is responsible for pumping the water from the water tap and heating water in the kettle.
- Subsystem4: The milk processing system is in charge of drawing milk from the milk tank, heating milk and producing milk foam.

The coffee maker includes 24 components, 12 elements, 4 subsystems, 9 main functions, 9 sub functions and 36 basic functions [MT2782]. The simulator, including the simulation of the coffee maker and its fault diagnosis system, creates the basis for the connection with the dynamic fault handling reconfiguration system and further evaluation.

**Combination of the dynamic fault handling and reconfiguration system and the coffee maker simulator**

Figure 6.17 indicates the combination of the dynamic fault handling and reconfiguration system and the coffee maker simulator. To show the production process intuitively, this research realized the combination between the coffee maker simulator and the real coffee maker through the USB-CAN adapter [FA2722] [SA2861]. In this realization, the Java applications of the dynamic fault handling reconfiguration system and the simulator ran on two computers individually. For the purpose of the establishment of the connection between the dynamic fault handling reconfiguration system and the coffee maker simulator, an internet router was utilized as communication media. Computer 1 ran three individual applications, i.e. an application of the coffee maker simulator, an application of adding faults and an application of the existing fault diagnosis system (EFDS). The EFDS monitors the real-time data from the coffee maker system simulator and stores it in the database as historical data. In addition, the fault knowledge affords the necessary verification knowledge for the fault diagnosis. With the help of the Java application of adding faults, it provides the possibility to add various faults in the simulator.

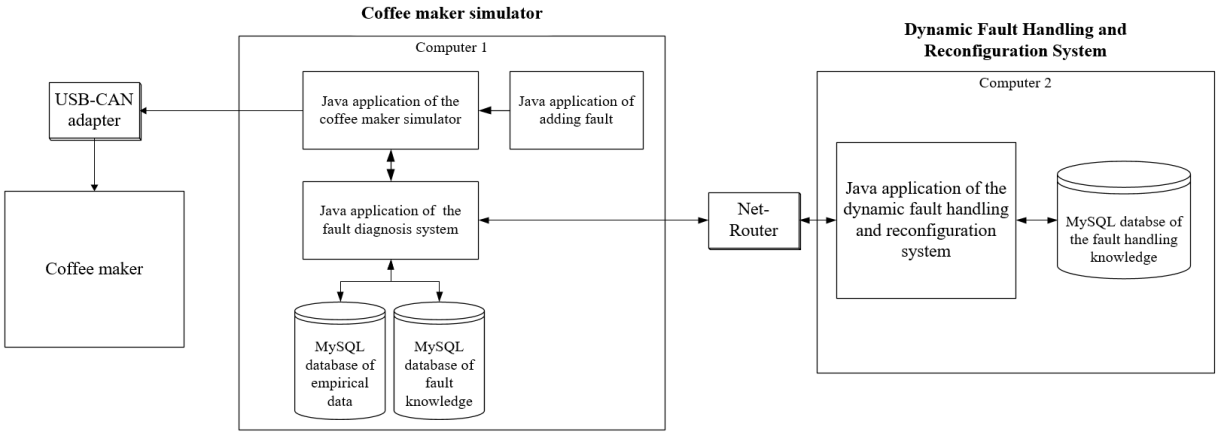



Figure 6.17: Combination between the dynamic fault handling reconfiguration system and the coffee maker simulator [MT2782] [SA2861]

Computer 2 furnishes an operation system to run the server as platform for the application of the dynamic fault handling reconfiguration system. The fault handling knowledge, including fault knowledge, symptom knowledge and system knowledge, was implemented in the MySQL database.

**Evaluation of the conception on the coffee maker simulator**

In order to evaluate the functionalities of the dynamic fault handling and reconfiguration system on the basis of the coffee maker simulator, 5 test cases were defined and carried out to test the correctness of the coffee maker simulator (CMS), its corresponding fault diagnosis system (FDS), and the dynamic fault handling and reconfiguration system (DFHRS) in Table 6.3.

Table 6.3: Test cases for evaluating the developed software [MT2782, pp. 50-62]

| Test case | Test object | Test objective |
|---|---|---|

| | | |
|---|---|---|
| **Coffee producing simulation** | TSS, FDS, | Testing the correctness of simulating the process of coffee production, including GUI of TSS and FDS; testing the correctness of resource state simulation |
| **Fault occurrence simulation** | TSS | Testing the correctness of the application of adding faults |
| **Determining fault type** | FDS | Testing the correctness of the diagnosis result of the FDS |
| **Handling a known fault** | TSS, FDS, DFHRS | Testing the correctness of handling a known fault; testing the correctness of reconfiguration |
| **Handling a known fault but new in local** | FDS, DFHRS | Testing the correctness of updating the fault knowledge; testing the correctness of reconfiguration |
| **Handling Fault New Locally and Remotely** | FDS, DFHRS | Testing the correctness of handling a new fault, including fault localization and identification of available functions; testing the correctness of reconfiguration |

Following this, with the help of the introduced specific test cases, the basic functionalities of the three software applications were later tested and they ran correctly and flawlessly [MT2782], providing a basis for performing a quantitative test, in which 100 various random faults, consisting of 40 single components faults and 60 multiple faults, were fabricated. As introduced in Chapter 6.1.7, the coffee maker simulator can be reconfigured with available functions in the time span from the appearance of a fault to its repair. After the test, 96 faults could be processed, i.e. the coffee maker simulator could be reconfigured with the available functions. The rest could be processed, so that the coffee maker simulator had to maintain the stop status. Here, this thesis assumes: 1 hour for MTTF, 0.5 hour for MTTR, and 5 minutes for replacing a defective part. Based on the equation in Chapter 6.17, the calculation result of the availability of the coffee maker system simulator is shown below.

Table 6.4: Availability of the coffee maker simulator

| Availability | Percent |
|---|---|
| Original availability (OA) | 66.67% |
| Real availability (RA) | 93.33% |
| Real availability concerning the proportion of available functions (Rapf) | 85.40% |
| D_Availability | 18.73 % |

Table 6.4 shows the availability of the coffee maker simulator.

As a result, based on the qualitative and quantitative test, the three software applications performed flawlessly. Each functionality of the dynamic fault handling reconfiguration system had achieved its expectation as defined in the test cases. Not only the known faults but also new faults were successfully dealt with. The coffee maker was reconfigured smoothly and its availability evidently enhanced. According to the evaluation result of the coffee maker simulator, industrial automation systems of the sequential process type can be successfully maintained and reconfigured by the dynamic fault handling reconfiguration system with the available functions. The objective of the research to increase the availability is achieved.

## 6.4 Evaluation of the Conception on the High-bay Warehouse Simulator

In the last two subsections, the realization and evaluation based on the two-tank simulator and the coffee maker simulator were presented. Those applications were based on the Java programming language. For the purpose of verifying the universality to implement the conception of dynamic fault handling and reconfiguration, the high-bay warehouse simulation and its fault diagnosis system, as well as the dynamic fault handling reconfiguration system, were realized based on the program language C# instead of the Java program language. The dynamic fault handling reconfiguration system utilizes the self-defined TCP Server with the communication method of Sockets rather than the Tomcat server. Based on the realization of the three applications, the conception of the dynamic fault handling reconfiguration system was evaluated with regard to qualitative and quantitative aspects.

**Description of the high-bay warehouse simulator**

The high-bay warehouse represents an industrial automation system of discrete object process type. It is able to store packaged products automatically.

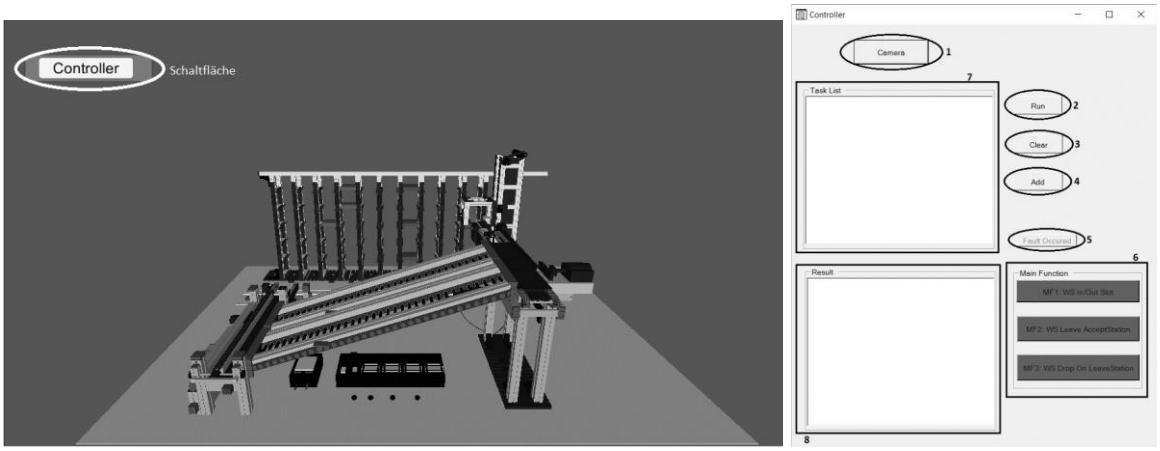



Figure 6.18: Overview of the high-bay warehouse simulator [MA2801]

As Figure 6.18 depicts (left side), the simulator stores a package in the rack and removes a specific package from the rack. To realize a better visualization, the behavior of the high-bay warehouse was implemented based on 3D modeling software, i.e. Unity, which provides mature and standardized developed modules. Its integrated development environment cannot afford a completed control simulation and cannot satisfy the further deployment of other functions, such as the simulation of faults. Hence, the control application (right side in Figure 6.18) was developed individually and an interface was established to complete the communication between the control application and the 3D simulator. The high-bay warehouse and its simulator consist of three stations, i.e. three subsystems:

- Station1: The shelf control system is responsible for controlling the conveyor cage. It can build the physical connection with the input and output stations to receive the packages and output the packages respectively. It then stores the packages into specific slots and removes packages from specific slots, respectively. This subsystem is realized with X-axis sensors, Z-axis sensors, a presence sensor, a telescopic conveyor, a horizontal track, and a vertical track.

- Station2: The input station transfers the packages from the guide ramp to the storage location which can build the connection with the conveyor cage with the help of the conveyor belt. This subsystem is made up of a storage sensor, a unit motor, two analytical sensors, an end-position sensor, a conveyor motor, and a light barrier for the removal position.

- Station3: The output station is able to access the package from the conveyor cage in the storage location by means of the conveyor belt and puts the package on the guide ramp. This subsystem includes a light barrier for the delivery position, a conveyor motor, and an electromechanical bolt.

The high-bay warehouse simulator provides 3 main functions, 71 sub functions and 49 basic functions. Based on the high-bay warehouse simulator, the combination of the dynamic fault handling and reconfiguration with the simulator will be presented in the next section.

**Combination of the dynamic fault handling and reconfiguration system and high-bay warehouse simulator**

Since the realization of the high-bay warehouse simulator was based on the programming language C# instead of Java, as well as the self-defined TCP server rather than Tomcat server, the communication method between server and client was replaced by the socket. But the principle of communication was still internet-based and only the communication approach was different. JSON was, in terms of the data type, the same as in the other two applications. The overview of the combination between the dynamic fault handling reconfiguration system and the high-bay warehouse simulator is depicted in Figure 6.19.

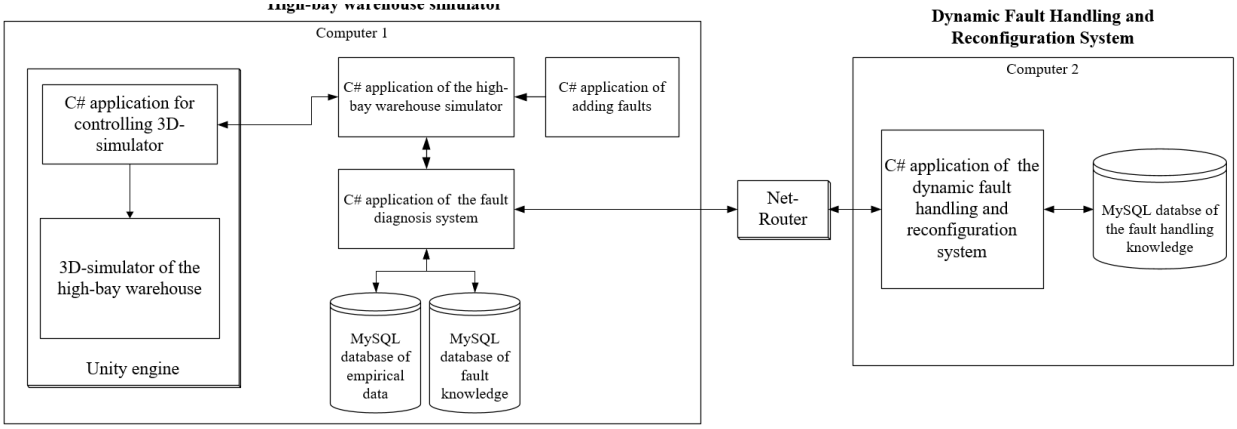



Figure 6.19: Combination between the dynamic fault handling reconfiguration system and the high-bay warehouse simulator

In the realization, the C# applications of the dynamic fault handling reconfiguration system and the simulator run on two computers individually. For the purpose of the establishment of the combination between the dynamic fault handling and reconfiguration system and the high-bay warehouse simulator, an internet router was utilized as the communication media. Computer 1 ran five applications, i.e. the application of adding faults and the application of the existing fault diagnosis system (EFDS), and the Unity engine, which supported the C# application in controlling the 3D-simulator and the 3D-simulator of the high-bay warehouse. The C# application provided the possibility to add various faults. The EFDS monitors the real-time data from the high-bay warehouse simulator and stores it in the database as historical data. Computer 2 furnished the operating system to run the server as the platform for the application of the DFRHS. In addition, the fault knowledge afforded the necessary verification knowledge for the fault diagnosis.

**Evaluation of the conception on the high-bay warehouse simulator**

This was done to evaluate the dynamic fault handling and reconfiguration system on the basis of the high-bay warehouse simulator. To evaluate the functionalities of the developed software from the qualitative aspect, 12 test cases were defined and carried out to evaluate the correctness of the high-bay warehouse simulator (HWS), its corresponding fault diagnosis system (FDS), and the dynamic fault handling and reconfiguration system (DFHRS).

Table 6.5: 12 Test cases for evaluating the developed software [MA2801, pp. 57-77]

| Test case | Test object | Test objective |
|---|---|---|
| **Changing the camera positions** | HWS | Testing the correctness of the observation view of the 3D-Simulator |
| **Server initialization** | DFHRS, HWS | Testing the correctness of starting the DFHRS and the combination between DFHRS and HWS |

| | | |
|---|---|---|
| **Setting tasks** | HWS | Testing the correctness of adding tasks for HWS |
| **Deleting tasks** | HWS | Testing the correctness of deleting tasks of HWS |
| **Performing tasks** | HWS | Testing the correctness of performing tasks with the 3D-Simualtor |
| **Adding faults** | HWS, FDS | Testing the correctness of adding faults; testing the correctness of diagnosing faults by FDS |
| **Presenting the reconfiguration results** | DFHRS, HWS | Testing the correctness of receiving the reconfiguration commands from DFHRS and showing the result in a log field |
| **Local combination** | HWS, FDS | Testing the correctness of the combination and the data exchange between HWS and FDS |
| **Sending fault information** | FDS, DFHRS | Testing the correctness of the data exchange between DFHRS and FDS |
| **Handling faults** | DFHRS | Testing the correctness of handling known and new faults with the help of the fault ID, symptom knowledge and system knowledge |
| **Presenting reconfiguration** | DFHRS | Testing the correctness of the graphical representation of the reconfiguration results |
| **Recovery** | HWS, DFHRS | Testing the correctness of resetting the system states of HWS and DFHRS when the fault is removed |

With the help of the introduced specific test cases, the basic functionalities of the three software applications were tested and ran correctly and flawlessly [MA2801]. Hence, it provided a basis for performing a quantitative test in which 100 various random faults were fabricated, including 42 single components faults and 58 multiple faults. As introduced in Chapter 6.1.7, the high-bay warehouse simulator could be reconfigured with available functions in the time span from the appearance of a fault to its repair. However, for some faults, the dynamic fault handling reconfiguration system could not provide the available functions because the defective component could have resulted from any one of all the functions being out of order, such as the microcontroller, etc. After the test, 88 faults were processed, i.e. the high-bay warehouse simulator could be reconfigured with the available functions. The rest could not be processed, so that the high-bay warehouse simulator had to maintain the stop status. Here, this thesis assumes: 1 hour for MTTF, 0.5 hour for MTTR, and 5 minutes for replacing a defective part. Based on the equation in Chapter 6.17, the calculation result of the availability of the high-bay warehouse simulator is presented below.

Table 6.6: Availability of the high-bay ware house simulator

| Availability | Percent |
|---|---|
| Original availability (OA) | 66.67% |
| Real availability (RA) | 91.11% |
| Real availability concerning the proportion of available functions (Rapf) | 85.07% |
| D_Availability | 18.41 % |

Table 6.6 shows the availability of the high-bay warehouse simulator.

As a result, based on the qualitative and quantitative tests, various applications can be performed flawlessly. Each functionality of the dynamic fault handling reconfiguration system has achieved its expectation as defined in test cases. Not only known faults but also the new faults could be successfully dealt with. The high-bay warehouse could be reconfigured smoothly and the availability evidently enhanced. According to the evaluation result of the high-bay warehouse simulator, industrial automation systems of the discrete object process type can be successfully maintained and reconfigured by the dynamic fault handling and reconfiguration system with available functions. The objective of the research, to increase the availability, is achieved.

## 6.5 Summary of the Demonstrators

In this section, the mentioned the demonstrators including the two-tank system, the coffee maker and the high-bay warehouse are about to be summarized.

Demonstrator 1: Coffee maker

The introduced coffee maker consists of a grinding system to grind coffee beans into coffee powder, a water heating system to pump water and heat water, a milk processing system to heat milk and produce milk foam, and a brewing system to mix water and coffee powder. This coffee maker can produce hot water, espresso, cappuccino, latte, milk coffee, etc. In the simulator, the resources and the process are simulated by the mathematical model in the background. It provides the possibility to use the fault injection approach, for example, it is easy to change a parameter, like temperature, into an abnormal state as well as directly set a component into a defective state. In addition, a fault diagnosis system for the coffee maker is developed, which can monitor, not only the volume of the resources, like water, milk, coffee beans but also the intermediate parameter in the coffee producing process like water flow rate, water adding time, the weight of the coffee beans, etc. Hence, the fault diagnosis system can identify the presence of a fault.

In order to simulate various faults in the coffee maker, an additional fault injection panel was developed, which can change the state of a component, a subsystem as well as arbitrary parameter in the process. Hence, a single temperature sensor, a single weight sensor or both can be simulated into a defective state.

To evaluate the developed dynamic fault handling and reconfiguration system for the coffee maker, different test cases were designed and implemented. Firstly, the developed coffee maker simulator can correctly simulate the coffee maker to produce different products through the value of the final parameters of the products. The coffee maker simulator can successfully receive the reconfiguration commands and perform the reconfiguration, like deactivating the not available functions. Secondly, the developed fault diagnosis system can monitor the coffee producing process correctly, such as reading the ongoing parameter value correctly, identifying the abnormal parameter via the predefined process model and assigning the presented faults to known fault correctly. Thirdly, the communication between the developed fault diagnosis system and the dynamic fault handling and reconfiguration system was established successfully. The fault diagnosis system can successfully send the fault information to the dynamic fault handling and reconfiguration system and also receive the new fault information as well as the reconfiguration commands. Finally, the functionality of the dynamic fault handling and reconfiguration system for the coffee maker can be successfully executed. The presented faults can be correctly identified as known faults or new faults. The available functions can be successfully and correctly accessed from the data base, namely the fault knowledge. It was possible to correctly identify the fault locations of the coffee maker. The inference machine can perform the process of fault localization and identification of available functions correctly. The available functions of the coffee maker can be correctly identified via two processes, identification of affected functions and identify available functions, respectively. The ongoing tasks in the coffee maker can be successfully identified, based on the available functions and ongoing capacity of water, coffee beans and milk. Additionally, the speed of the entire fault handling time is fast.

Demonstrator 2: High-bay warehouse

The introduced high-bay warehouse consists of a shelf control system to control the conveyor cage for building the physical connection between the input and output stations to receive the packages and output the packages from and to the specific slots, an input station to transfer the packages from the guide ramp to the storage location, and an output station to access the packages from the conveyor cage in the storage location. The high-bay warehouse contains axis sensors, tracks, motors, etc. and can store a package in the rack and remove the package from a specific rack. In the high-bay warehouse simulator, the capacity of the slots with packages, the speed of the motor and the signal of each position sensor can be simulated. Moreover, the processes of transporting a package and storing it in a specific slot were simulated by the mathematical model. So it provides the possibility to use the fault injection approach, for example, it is easy to change a parameter, like the state of an X-axis sensor in the shelf into a defective state. It is necessary to

point out that the main sensors of the high-bay warehouse are Boolean type variables, such as a light barrier for the delivery position. Moreover, a fault diagnosis system for the high-bay warehouse is developed, which can monitor not only the capacity of the free slots in the shelf but also the sensor states. Hence, the fault diagnosis system can identify the presence of a fault from the component view directly, for example, if no signal of a light barrier in a specific time is received.

In order to simulate various faults, a fault injection panel was also developed, which can change the state of a component and a subsystem, like a motor. Hence, a single unit motor, a single end-position sensor or both can be simulated into a defective state.

To evaluate the developed dynamic fault handling and reconfiguration system for the high-bay warehouse, different test cases were designed and implemented. Firstly, the developed high-bay warehouse simulator can correctly simulate the real high-bay warehouse to transport a package to the storage position, to store the package into a specific slot, to get out the package to the output position, and to access the package from the output position. The high-bay warehouse simulator can successfully receive the reconfiguration commands and perform the reconfiguration like deactivating the not available functions. Secondly, the developed fault diagnosis system can monitor the package transporting and storing process correctly, such as reading the ongoing sensor value correctly, identifying the abnormal parameter via the predefined process model and assigning the presented faults to known fault correctly. Thirdly, the communication between the developed fault diagnosis system and the dynamic fault handling and reconfiguration system was established successfully. The fault diagnosis system can successfully send the fault information to the dynamic fault handling and reconfiguration system and also receive the new fault information as well as the reconfiguration commands. Finally, the functionality of the dynamic fault handling and reconfiguration system for the high-bay warehouse can be successfully executed. The presented faults can be correctly identified as known faults or new faults. The available functions can be successfully and correctly accessed from the data base, namely the fault knowledge. It was possible to correctly identify the fault location of the coffee maker. The inference machine can perform the process of fault localization and identification of available functions correctly. The available functions of the coffee maker can be correctly identified via two processes, identification of affected functions and identify available functions, respectively. The ongoing tasks in the high-bay warehouse can be successfully identified based on the available functions and ongoing capacity of free slots in the shelf and the volume of the rest packages with specific colors. Additionally, the speed of the entire fault handling time is fast.

Demonstrator 3: Two-tank system

The introduced two-tank system consists of an injection system to inject water from one tank to another in a specific volume, a heating water system to heat water to a required temperature, a water draining system to drain water from one tank to another, and an additional air inflating

process to provide air for the air pressure switch. In the simulator, the resources and the processes are simulated by the mathematical model in the background. It provides the possibility to use the fault injection approach, for example, it is easy to change a parameter, like temperature into an abnormal value as well as directly set a component into a defective state. In addition, a fault diagnosis system for the two-tank system simulator is developed, which can monitor not only the volume of the resources like water and temperature but also the intermediate parameter in the pumping process like the water flow rate and the degree of the air pressure switch. Hence, the fault diagnosis system can identify the presence of a fault.

To simulate various faults in the two-tank system, a fault injection panel was developed, which can change the state of a component, a subsystem as well as arbitrary parameter in the process. Hence, a single temperature sensor or the value of the safety water level sensor or both can be simulated into a defective state.

To evaluate the developed dynamic fault handling and reconfiguration system for the two-tank system, different test cases were designed and implemented. Firstly, the developed two-tank system simulator can correctly simulate the two-tank system to produce different volumes of water with different temperatures. The two-tank system simulator can successfully receive the reconfiguration commands and perform the reconfiguration like deactivating the not available functions. Secondly, the developed fault diagnosis system can monitor the water injecting and heating process correctly, such as reading the ongoing parameter value correctly, identifying the abnormal parameter via the predefined process model and assigning the presented faults to known fault correctly. Thirdly, the communication between the developed fault diagnosis system and the dynamic fault handling and reconfiguration system was established successfully. The fault diagnosis system can successfully send the fault information to the dynamic fault handling and reconfiguration system and also receive the new fault information as well as the reconfiguration commands. Finally, the functionality of the dynamic fault handling and reconfiguration system for the two-tank system can be successfully executed. The presented faults can be correctly identified as known faults or new faults. The available functions can be successfully and correctly accessed from the data base, namely the fault knowledge. It was possible to correctly identify the fault location of the two-tank system. The inference machine can perform the process of fault localization and identification of available functions correctly. The available functions of the two-tank system can be correctly identified via two processes, identification of affected functions and identify available functions, respectively. The ongoing tasks in the two-tank system can be successfully identified based on the available functions and ongoing capacity of the water volume in both tanks. Additionally, the speed of the entire fault handling time is fast.

The following three tables shows the evaluation results with three demonstrators intuitively.

Table 6.7: Demonstrator of the Coffee maker simulator

| Structural Design | Usage | Evaluation Actions |
|---|---|---|
| Realization of the coffee maker simulator:<br>• Realization in Java<br>• Uses MySQL database<br>• Includes 5 classes and 2900 LoC<br>• USB-CAN-Bus based communication with the real coffee maker | • Simulate the coffee maker functions<br>• Remote connection to server<br>• Simulate unknown faults<br>• Local connection with the fault diagnosis system<br>• Simulate parameter faults for components<br>• Reconfigure available functions | • Simulated 9 products as well as 45 functions<br>• Simulated 10 known faults including 5 single component faults and 5 multiple faults<br>• Simulated 90 new faults including 35 component faults and 55 multiple faults<br>• Received reconfiguration commands from server<br>• Reconfigured available functions |
| Realization of the fault diagnosis system:<br>• Rees MySQL database<br>• Includes 9 classes and 3700 LoC | • Monitor the parameters of the coffee maker<br>• Record the process data in the data base within 10 minutes<br>• Remote connection to server<br>• Fault diagnosis | • Recoded the process data within 10 minutes<br>• Monitored the change of the parameter in the coffee maker<br>• Identified the known faults with predefined process model and symptom knowledge<br>• Identified new faults and provided fault diagnosis results for server<br>• Received the new fault information and stored it in the local fault knowledge base |
| Realization of the dynamic fault handling and reconfiguration system:<br>• Realization in Java<br>• Uses MySQL database<br>• Base on Apache server<br>• Includes 12 classes and 11500 LoC | • Remote connection with the coffee maker<br>• Identify fault type<br>• Handle known faults<br>• Handle new faults<br>• Create reconfiguration commands<br>• Identify available ongoing tasks<br>• Fault analysis with the inference machine | • Connected coffee maker 100 times<br>• Connected with the fault diagnosis system 100 times<br>• Identified known faults 10 times<br>• Identified new faults 90 times<br>• Accessed available functions from fault knowledge base 10 times<br>• Accessed the symptom knowledge and specific mathematical model correctly 90 times<br>• Inferred fault location 90 times<br>• Inferred affected functions 90 times<br>• Inferred available functions 86 times<br>• Checked ongoing tasks 96 times<br>• Created reconfiguration commands 96 times<br>• Average response time 513.83ms |

Table 6.8: Demonstrator of the High-bay Warehouse

| Structural Design | Usage | Evaluation Actions |
|---|---|---|
| Realization of the high-bay warehouse:<br>• Realization in C#<br>• Based on the Unity-3D<br>• Using MySQL database<br>• Includes 16 classes and 4300 LoC | • Simulate the high-bay warehouse functions<br>• Remote connection to server<br>• Simulate unknown faults<br>• Local connection with the fault diagnosis system<br>• Simulate parameter faults for components<br>• Reconfigure available functions | • Simulated 3 services as well as 123 functions<br>• Simulated 10 known faults including 5 single component faults and 5 multiple faults<br>• Simulated 90 new faults including 37 component faults and 53 multiple faults<br>• Received reconfiguration commands from server<br>• Reconfigured available functions |
| Realization of the fault diagnosis system:<br>• Realization in C#<br>• Uses MySQL database<br>• Includes 3 classes and 2100 LoC | • Monitor each component state of the high-bay warehouse<br>• Record the process data in the data base within 10 minutes<br>• Remote connection to server<br>• Fault diagnosis | • Recoded the process data within 10 minutes<br>• Monitored the change of the parameter in the high-bay warehouse<br>• Identified the known faults with predefined process model and symptom knowledge<br>• Identified new faults and provided fault diagnosis results for server<br>• Received the new fault information and stored it in the local fault knowledge base |
| Realization of the dynamic fault handling and reconfiguration system:<br>• Realization in C#<br>• Uses MySQL database<br>• Based on Apache server<br>• Includes 12 classes and 12000 LoC | • Remote connection with the high-bay warehouse<br>• Identify fault type<br>• Handle known faults<br>• Handle new faults<br>• Fault analysis with the inference machine<br>• Create reconfiguration commands<br>• Identify available ongoing tasks | • Connected the high-bay warehouse 100 times<br>• Connected with the fault diagnosis system 100 times<br>• Identified known faults 10 times<br>• Identified new faults 90 times<br>• Accessed the available functions from fault knowledge base 10 times<br>• Accessed the symptom knowledge and specific mathematical model correctly 90 times<br>• Inferred the fault location for 90 times<br>• Inferred affected functions 90 times<br>• Inferred available functions 78 times<br>• Checked ongoing tasks 88 times<br>• Created reconfiguration commands 88 times<br>• Average response time 571.26ms |

Table 6.9: Demonstrator of the Two-Tank System

| Structural Design | Usage | Evaluation Actions |
|---|---|---|
| Realization of the two-tank system simulator:<br>• Realization in Java<br>• Uses MySQL database<br>• Includes 6 classes and 6000 LoC | • Simulate the two-tank system functions<br>• Remote connection to server<br>• Simulate unknown faults<br>• Local connection with the fault diagnosis system<br>• Simulate parameter faults for components<br>• Reconfigure available functions | • Simulated 52 functions<br>• Simulated 10 known faults including 5 single component faults and 5 multiple faults<br>• Simulated 90 new faults including 45 component faults and 45 multiple faults<br>• Received reconfiguration commands from server<br>• Reconfigured available functions<br>• System stopped in the presence of a fault and system restarted after reconfiguration |
| Realization of the fault diagnosis system:<br>• Realization in Java<br>• Uses MySQL database<br>• Includes 5 classes and 7000 LoC | • Monitor the parameters of the two-tank system<br>• Record the process data in the data base within 10 minutes<br>• Remote connection to server<br>• Fault diagnosis | • Recoded the process data within 10 minutes<br>• Monitored the change of the parameter in the two-tank system<br>• Identified the known faults with predefined process model and symptom knowledge<br>• Identified new faults and provided fault diagnosis results for server<br>• Received the new fault information and stored it in the local fault knowledge base |
| Realization of the dynamic fault handling and reconfiguration system:<br>• Realization in Java<br>• Uses MySQL database<br>• Based on Apache server<br>• Includes 12 classes and 16000 LoC | • Remote connection with the two-tank system<br>• Identify fault type<br>• Handle known faults<br>• Handle new faults<br>• Fault analysis with the inference machine<br>• Create reconfiguration commands<br>• Identify available ongoing tasks | • Connected two-tank system 100 times<br>• Connected with the fault diagnosis system 100 times<br>• Identified known faults 10 times<br>• Identified new faults 90 times<br>• Accessed the available functions from fault knowledge base 10 times<br>• Accessed the symptom knowledge and specific mathematical model correctly 90 times<br>• Inferred fault location for 90 times<br>• Inferred affected functions 90 times<br>• Inferred available functions 73 times<br>• Checked ongoing tasks 83 times<br>• Created reconfiguration commands 83 times<br>• Average response time 465.72ms |

As showed in the introduced tables, the dynamic fault handling and reconfiguration system is able to handle various faults. They are known faults, including a single component fault and multiple components fault, new fault parameter faults, like temperature abnormally increased (still working but no not correctly), a single component fault (not working), two-components-fault (combination, one working and another not working, both working but without correct results, both not working), a subsystem fault, a subsystem and component fault, and more combinations. Based on the evaluation results, the correctness of every functional module in the dynamic fault handling and reconfiguration system and the correctness for handling various faults in different demonstrators were proven. However, available functions can also not always be provided in case of a defective main component.

## 6.6 Assessment of the Dynamic Fault Handling and Reconfiguration System regarding the Requirements

In the last four sections, the dynamic fault handling and reconfiguration system was evaluated by means of the two-tank system simulator, the coffee maker simulator, and the high-bay warehouse simulator. This sub section proposes to estimate the conception from qualitative aspects.

**R1: Ability of enhancing the availability of the entire automation system**

With the help of the dynamic fault handling and reconfiguration system, industrial automation systems can be transferred into another operation mode via the reconfiguration in case of a fault has occurred. This is because the dynamic fault handling reconfiguration system enables the industrial automation system to be kept in operation before the fault is removed. Moreover, from the perspective of the operation time, the original mean time to repair is reduced and the mean time between faults is increased. According to the definition and the discussion in Section 6.1 concerning availability, the availability of industrial automation systems can be extremely enhanced. This was proved in the last three sub sections.

**Result:** The average availability of the implemented industrial automation systems is enhanced by more than 18% as shown in the experiments.

**R2: Ability of automatic, reasonable and dynamic fault analysis**

By means of the defined system model, the description of an industrial automation system can be formalized in the database as the system knowledge, such as the relationship matrix of the components, the functions and the requirements, specific rules for each function, and specific rules for each requirement, as well as their consequences. The formalized system knowledge provides the opportunity for determining the fault effect. In addition, with the help of the formalized symptom knowledge for the component model, the dynamic fault handling reconfiguration system can localize the fault location for either single or multiple faults. Moreover, based on the fault location, the function tree, and the requirement tree, the dynamic

fault handling reconfiguration system can identify either affected or unaffected functions, and identify available functions automatically through the depth-first-search approach. In addition, the ongoing tasks can be evaluated with the relationship between functions and tasks. Finally, the reconfiguration commands can be generated and integrated to perform the reconfiguration in the industrial automation system, because all these analyses and reasoning procedures are performed automatically with the specific defined reasoners.

**Result:** The fault location of new faults and corresponding available functions can be identified based on the specific system models.

**R3: Ability of reconfiguration with the available functions**

As mentioned above, with the aid of the available functions and available ongoing tasks, the dynamic fault handling reconfiguration system can generate appropriate reconfiguration commands, with which the industrial automation system can carry out its available functions. For one thing, if it requires no specific commands or additional actions for isolating not available functions, i.e. the system requires no restart, the industrial automation system can reconfigure itself. For another thing, the dynamic fault handling reconfiguration system affords specific codes for activating specific functions and provides user instructions to the user for isolating unavailable functions. For instance, to isolate the function of a valve in a pipeline in the two-tank system, the user ought to switch off the manual valve which is in the front of the defective valve in the pipeline. Hence, the dynamic fault handling reconfiguration system can directly help the industrial automation system to complete the reconfiguration with available functions.

**Result:** The dynamic fault handling and reconfiguration system does not only allow specific reconfiguration commands to be generated, but also its necessary corresponding measures in order to assure the operation of industrial automation systems with available functions.

**R4: Ability of reducing the cost for implementation and in operation**

In order to implement the dynamic fault handling reconfiguration system, instead of developing a new local fault diagnosis system, this thesis proposes to cooperate with the existing fault diagnosis system via an adaptive communication interface. Moreover, this thesis attempts to run the dynamic fault handling reconfiguration system on a remote server to handle faults via the internet, rather than one dynamic fault handling and reconfiguration system for each industrial automation system, so that the associated fault handling knowledge is stored in the server and faults are handled by the server uniformly. Therefore, the local industrial automation system has no need to add additional costs to increase storage space and computing power such as faster, more powerful processors and a larger memory. Hence, the entire implementation cost can be further limited. Due to a server system, the thesis enables the managing of the dynamic fault handling reconfiguration system and fault handling knowledge remotely with the help of professional experts. To avoid resource waste, enhance the processing speed, and overcome the weakness of

existing fault diagnosis systems, the dynamic fault handling reconfiguration system can analyze the fault diagnosis results at first and localize the fault location with its own integrated fault diagnosis methods. Subsequently, benefiting from automatic reasoning, the dynamic fault handling reconfiguration system can support the 24 hours computer-based automatic fault handling without the intention of additional maintenance services. The labor cost for this service can be saved. Depending on the analysis above, the cost of implementing the dynamic fault handling reconfiguration system can be reduced exponentially, or at least to a limited extent.

**Result:** The predefined common communication interface assures a low development cost and provides a high portability for more various industrial automation systems. Moreover, the automatic server platform makes it possible to keep costs for handling faults low, reducing or eliminating the need for additional maintenance services.

**R5: Ability of porting the conception for heterogeneous industrial automation systems**

Concerning the heterogeneity of industrial automation systems, this conception of the dynamic fault handling and reconfiguration has been defined as a uniformed communication interface (see 6.1.5) to suit the communication of different systems. Furthermore, this thesis has attempted to use a common uniform data type (JSON) and communication type (HTTPS). Moreover, with the help of formalized system knowledge and the reasoning logic for identifying available functions, the dynamic fault handling reconfiguration system can be easily transferred and implemented in other industrial automation systems. As evaluated in the last three sections, the conception of the dynamic fault handling and reconfiguration was implemented for three types of industrial automation systems. The realization of the dynamic fault handling reconfiguration system is application platform-independent. The conception of dynamic fault handling and reconfiguration can be simply ported to heterogeneous industrial automation systems.

**Result:** With the help of the predefined uniformed communication interface, data type and communication type, the dynamic fault handling and reconfiguration system can be developed efficiently for heterogeneous industrial automation systems.

As a result, the approved dynamic fault handling and reconfiguration system is able to handle faults in industrial automation systems successfully and efficiently. Additionally, with the help of the proposed universal interface, the conception can be easily integrated into new industrial automation systems. Since the execution is done without the intervention of users, the cost for running the proposed dynamic fault handling and reconfiguration system is greatly reduced. Moreover, the availability of industrial automation systems can be improved to a great degree with the proposed dynamic fault handling and reconfiguration system.

In summary, the prototype and the evaluation of the dynamic fault handling and reconfiguration systems were presented in detail in this chapter. The entire system structure, and the major parts

of the software structure via the class diagram were outlined. The fault handling knowledge, including system knowledge, symptom knowledge, and fault knowledge, were listed. And all attributes of these tables were depicted in detail. Subsequently, the adaptive communication interface for industrial automation system was outlined from the conceptual perspective. Afterwards, based on the two-tank system simulator, the realized dynamic fault handling reconfiguration system was presented as follows: the user interface of the dynamic fault handling reconfiguration system, the GUI of adding faults, the fault diagnosis results of the existed fault diagnosis system, the reconfiguration commands, and the procedure of identifying available functions with the visual demonstration, with which the analysis process in the function tree and requirement tree could be displayed step by step. Then, the evaluation methods concerning the qualitative aspect and the quantitative aspect were defined. It was shown that former attempts to evaluate the correctness of major functionalities and the latter tries to evaluate if the dynamic fault handling reconfiguration system can really enhance the availability of industrial automation systems with the three proposed calculation equations. Afterwards, on the basis of the demonstrators, i.e. the two-tank system simulator, the coffee maker simulator and the high-bay warehouse simulator, the structure of the simulator in combination with the dynamic fault handling reconfiguration system and the results of the evaluation were presented. Finally, the predefined requirements for the conception, the proposed conception of the dynamic fault handling and reconfiguration were evaluated. As a result, the proposed conception can fulfill the requirements, the industrial automation system can perform the available functions smoothly and the availability of industrial automation systems can be increased to an extreme degree.

# 7 Conclusion and Future Work

In line with the requirements presented in Chapter 2.4, the conception of dynamic fault handling and reconfiguration were evaluated in the last chapter. In this chapter, the results of this research will be recapitulated. Then two limitations on the conception will be presented. Finally, some possible future work will be outlined.

## 7.1 Summary and Contribution of the Research

Along with the widespread use of industrial automation systems, the continuous working ability of industrial automation systems plays an important role in daily production processes and life. Likewise, availability has become an extremely important indicator for an industrial automation system, and cannot be ignored. Due to the global expansion of the sales area, however, the traditional maintenance mode, e.g. manual door-to-door service, has been unable to fully adapt to that change. Therefore, within this context, ways to solve the issue of helping manufacturers increase their availability of produced industrial automation systems and to provide users with smoother services are needed urgently. To solve this problem, and support the maintenance for manufacturers, a novel dynamic fault handling and reconfiguration approach was developed and introduced. With the help of the dynamic fault handling and reconfiguration system, the current fault, as well as its effect, can be quickly identified, and the industrial automation system can still work with the available functions. If the fault is known, its effect can be directly identified by accessing the fault knowledge base, so that the fault can be handled rapidly. If the fault is new, partial functions can be identified by means of the dynamic fault handling and reconfiguration system based on the system model. The industrial automation system can perform the partial functions via the reconfiguration, even if the fault still exists. Hence, the **availability** of the industrial automation system can be improved.

The result of the research is a dynamic fault handling and reconfiguration system able to cooperate with the existing fault diagnosis system to deal with faults in an industrial automation system. Fault diagnosis systems provide fault diagnosis results for the dynamic fault handling and reconfiguration system when a fault is present. Based on the fault diagnosis results, a dynamic fault handling and reconfiguration system analyses the fault results, identify the functions still available, and generates corresponding reconfiguration commands to guide the reconfiguration of industrial automation systems. In this way, the industrial automation system can supply available services as well as partial functions for users before the fault is removed. The dynamic fault handling and reconfiguration system is a fixed constituent for the industrial automation system that runs in parallel to its execution. The benefit of the cooperation between the existing fault diagnosis system and the dynamic fault handling and reconfiguration system is that the manual fault diagnosis and maintenance activities can be replaced by an **automated** diagnosis and a reconfiguration with the available functions. Therefore, it can reduce the maintenance time and

assure the availability of partial subsystems. This helps the customers to save a great deal of time and money, as well as improving or assuring a high degree of **trust** in the industrial automation system.

In this conception, faults are handled according to their being known or new. Known faults and the corresponding available functions can be simply identified with the fault identifier. For new faults, fault locations can be diagnosed initially with the help of historical data and various fault models as well as process models. By means of various models, features are extracted from the historical data. The server system utilizes symptom knowledge to compare them with the extracted features to assess the fault location, i.e. defective components. Afterwards, with help of the system knowledge, the server system searches for the fault impact within the function tree via the depth-first-search approach to identify affected and unaffected functions. By means of the same search approach, the unaffected functions can be evaluated with the requirement tree to identify available functions. Corresponding available functions and their relationship with tasks, which are provided by industrial automation systems, and the availability of ongoing tasks, can be further evaluated. Finally, reconfiguration commands for available functions and available tasks are generated for industrial automation systems. This new fault, with its diagnosis symptoms and available functions, is then stored in the database as fault knowledge. The fault knowledge, except for the available functions, is also updated in the local fault knowledge, making it possible for the existing fault diagnosis system to detect this fault when it appears.

In order to adapt the specific requirement of system knowledge, a system model and its formalization have been proposed to represent an industrial automation system. A system model consists of three major parts: a component model to describe the physical structure, a function model to describe the logical structure, and a requirement model to represent all quality requirements. These three models can be mapped with each other. Furthermore, the system model has been formalized with eight matrices and a multitude of rules for functions and requirements to perform automatic reasoning. With the help of rules and matrices, a fault impact can be identified in the perspective of available functions.

A structure of a unified interface has been defined to adapt the quick development and different communication interfaces of industrial automation systems. The communication between local systems and server can be realized through the internet. With the defined data format, it allows quick exchange of large quantities of data. By means of the defined communication interface, the two-tank system, the coffee maker, and the high-bay warehouse can complete data exchange with the dynamic fault handling and reconfiguration system. Moreover, the diagnosis process and the maintenance process are very complex, including: dealing with empirical process data, identifying major features, etc. These activities are much cost and time intensive, as well as requiring very professional knowledge and tools. With the help of the proposed dynamic fault handling and reconfiguration concept, the user can be greatly **supported**, and the complexity can be very successfully controlled.

The proposed concept of the dynamic fault handling and reconfiguration for industrial automation systems provides the following benefits:

- Initially, the cooperation with existing fault diagnosis system and automatic fault impact analysis, based on the system model, can control the complexity of the fault diagnosis for the user.
- And then, when a fault appears in industrial automation systems, the downtime of an industrial automation system can be reduced by running the still available functions.
- Then, known faults can be processed directly by accessing the fault knowledge base. This helps the user to save time and money.
- Finally, new faults can be automatically analyzed by means of the established system model. This provides the still available functions as the analysis result.

With the identified available functions, the industrial automation system can be reconfigured to provide the still available services for the user, so the availability of an industrial automation system can be improved. Hence, all these are propitious for improving and enhancing the users' trust degree in the industrial automation systems.

## 7.2 Limitations of the Concept

The proposed conception of dynamic fault handling and reconfiguration is a novel approach for handling faults in industrial automation systems. But this conception has some limitations, too. In order to not only achieve the above design decisions efficiently and correctly, but also to fit the actual situations of an industrial automation system, three conditions enclosed this research: the targeted industrial automation system have to have integrated an existing fault diagnosis system; the industrial automation system has to be a component-based system; and each function has to be able to be individually activated.

- Existing fault diagnosis system in the industrial automation system: A fault diagnosis system should be integrated in the industrial automation system. It can realize real-time monitoring of the industrial automation system, detecting as well as diagnosing faults that have appeared. Moreover, the fault diagnosis system ought to recode real-time monitoring data for a certain period of time as historical data. This can provide a basis for further fault handling of the server system. In addition, the integrated fault diagnosis system can complete the fault diagnosis for a new fault according to the specific symptoms provided by the dynamic fault handling and reconfiguration system.
- Component-based industrial automation system: The industrial automation system should be a component-based system. Each component can be specified with its hardware and software to complete a specific function. In this case, the industrial automation system can be analyzed

and represented with the proposed system model. Furthermore, each function ought to be able to be activated and deactivated with a specific command. This allows for an automatic reconfiguration by activating commands.

## 7.3 Future Work

In this research, a conception was developed to maintain industrial automation systems for manufacturers via reconfiguring industrial automation systems with available functions in the case of the appearance of a fault. For future work, three perspectives can be considered.

Tool-support for formalization of system knowledge: As described in Chapter 4, a specific system model to describe an industrial automation system was presented. However, in practice, such a system model does not yet exist in the development phase. To complete such a system model, a great deal of time will be required and it cannot adapt to today's rapid research and development of industrial automation systems. Hence, a tool to formulate the existing graphical system model, such as UML and SysML, can be developed to generate the existing graphical system model in the proposed system model, even transferring the system model into system knowledge.

Web-based assistance for maintenance service: In the process of fault removing, if there is no replacement, the maintenance staff should return with replacements to remove faults. In this case, web-based assistance can be considered, which can realize the communication between maintenance staff with the dynamic fault handling and reconfiguration system remotely. The server can provide advice for a replacement, i.e., detect the fault location through the server system for the maintenance staff to improve effectiveness. Furthermore, maintenance staff can provide the exact fault location through a manual fault diagnosis, this allows the dynamic fault handling and reconfiguration system to reconfigure the industrial automation system with the exact fault location and provide more available functions before the fault is removed.

# Bibliography


[AbWe16] **S. Abele and M .Weyrich:** *A combined fault diagnosis and test case selection assistant for automotive end-of-line test systems*. In Industrial Informatics (INDIN), 2016 IEEE 14th International Conference on (pp. 1072-1077). IEEE, 2016, pp. 1072-1077.

[AbWe17] **S. Abele and M. Weyrich:** *Decision Support for Joint Test and Diagnosis of Production Systems based on a Concept of Shared Knowledge*. IFAC-PapersOnLine, Jg. 50, Nr. 1, 2017, pp. 15227-15232.

[AEM12] **A. Agirre, E. Estevez and M. Marcos:** *Fault tolerant component management platform over Data Distribution Service*. IFAC Proceedings Volumes, Jg. 45, Nr. 4, 2012, pp. 218-223.

[AET11] **H. Alwi, C. Edwards and C.P. Tan:** *Fault tolerant control and fault detection and isolation*. In: Fault Detection and Fault-Tolerant Control Using Sliding Modes. Springer, London, 2011, pp. 7-27.

[AlFu14] **N. Alrajei and H. Fu:** *A survey on fault tolerance in wireless sensor networks*. In Proceedings of the ASEE North Central Section Conference. American Society for Engineering Education, 2014. http://people.cst.cmich.edu/yelam1k/asee/proceedings/2014/Paper%20files/aseencs2014_submission_138.pdf

[Alge10] **G. B. Algelin:** *Maritime Management Systems*. Chalmers University of Technology, Gothenburg, Sweden, 2010.

[ALR01] **A. Avizienis, J. C. Laprie and B. Randell:** *Fundamental concepts of dependability*. University of Newcastle upon Tyne, Computing Science, 2001.

[AME12] **A. Agirre, M. Marcos and E. Estévez:** *Distributed applications management platform based on Service Component Architecture*. In Emerging Technologies & Factory Automation (ETFA), 2012 IEEE 17th Conference on. IEEE, 2012, pp. 1-4.

[APA+16] **A. Agirre, J. Parra, A. Armentia, E. Estévez and M. Marcos:** *QoS aware middleware support for dynamically reconfigurable component based IoT applications*. International Journal of Distributed Sensor Networks, Jg. 12, Nr. 4, 2016.

[APEM14] **A. Agirre, J. Parra, E. Estévez and M. Marcos:** *QoS aware platform for dependable sensory environments*. In 2014 IEEE International Conference on Multimedia and Expo Workshops (ICMEW). IEEE, 2014, pp. 1-5

[ASF15] **G. Abaei, A. Selamat and H. Fujita:** *An empirical study based on semi-supervised hybrid self-organizing map for software fault prediction*. Knowledge-Based Systems (2015), Elsevier Publishing, vol. 74, pp. 28-39.

[ASS+17] **T. Aicher, D. Schütz, M. Spindler, S. Liu, W. A. Günthner and B. Vogel-Heuser:** *Automatic analysis and adaption of the interface of automated material flow systems to improve backwards compatibility*. IFAC-PapersOnLine, Jg. 50, Nr. 1, 2017, pp. 1217-1224.

[BaHa10] **S. Baerisch and W. Hasselbring:** *Domain-Specific Model-Driven Testing*. Vieweg+ Teubner Verlag/GWV Fachverlage GmbH, Wiesbaden, 2010

[BaMa14] **A. Bahga and V. Madisetti:** *Internet of Things: A hands-on approach*. Published by Arshdeep Bahga & Vijay Madisetti, 2014.

[Bazg12] **A. Bazghandi:** *Techniques, advantages and problems of agent based modeling for traffic simulation*. International Journal of Computer Science Issues (IJCSI), Jg. 9, Nr. 1, 2012, pp. 115.

[BBG15] **M. Bordasch, C. Brand and P. Göhner:** *Fault-based identification and inspection of fault developments to enhance availability in industrial automation systems*. In Emerging Technologies & Factory Automation (ETFA), 2015 IEEE 20th Conference on, IEEE, September 2015, pp. 1-8.

[BCY03] **R. Bris, E. Châtelet and F. Yalaoui:** *New method to minimize the preventive maintenance cost of series–parallel systems*. Reliability engineering & system safety (2003), Elsevier in Press, Jg. 82, Nr. 3, pp. 247-255.

[BDW14] **Z. Bi, Da Xu L. and C. Wang:** *Internet of things for enterprise systems of modern manufacturing.* IEEE Transactions on industrial informatics, IEEE, 2014, Jg. 10, Nr. 2, pp. 1537-1546.

[Bell08] **M. Bell:** *Service-oriented modeling (SOA): Service analysis, design, and architecture*. Publisher: John Wiley & Sons, 2008.

[BeLu10] **M. Benosman and K. P. Lum:** *Passive actuators' fault-tolerant control for affine nonlinear systems*. IEEE Transactions on Control Systems Technology, Jg. 18, Nr. 1, 2010, pp. 152-163.

[Bequ03] **B. W. Bequette:** *Process control: modeling, design, and simulation*. Prentice Hall Professional, 2003, pp. 3.

[Bhas13] **B. Bhasker:** *Electronic commerce: framework, technologies and applications*. Published by Tata McGraw-Hill Education, 2013.

[BKLS03] **M. Blanke, M. Kinnaert, J. Lunze and M. Staroswiecki M.:** *Diagnosis and fault-tolerant control [M]***.** Berlin, Heidelberg: Springer-Verlag, 2003, pp. 281-291.

[BMC05] **D. Benavaids, P.T. Martin-Arroyo and A.R. Cortes:** *Automated Reasoning on Feature Models*. In: CAiSE. vol. 5, No. 3520, 2005, pp. 491-503.

[BMR17] **L. M. Bach, B. Mihaljevic and A. Radovan:** *Exploring HTTP/2 advantages and performance analysis using Java 9.* In: Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2017 40th International Convention on. IEEE, 2017, pp. 1522-1527.

[BMS07] **C. Bolchini, A. Miele and M. D. Santambrogio:** *TMR and Partial Dynamic Reconfiguration to mitigate SEU faults in FPGAs*. In Defect and Fault-Tolerance in VLSI Systems, 2007. DFT'07. 22nd IEEE International Symposium on, Publisher: IEEE, 2007, pp. 87-95.

[BoGö13] **M. Bordasch, P. Gohner:** *Fault prevention in industrial automation systems by means of a functional model and a hybrid abnormity identification concept*. In Industrial Electronics Society, IECON 2013-39th Annual Conference of the IEEE, IEEE, November 2013, pp. 2845-2850.

[Böhl10] **F. Böhle:** *Personenbezogene Dienstleistung als Interaktionsarbeit: Professionalisierung interaktiver Arbeit.* München, 2010.

[Bord16] **M. Bordasch:** *Abnormitäten-Management zur Fehlerprävention bei automatisierten Systemen im Betrieb*. Publisher: Shaker Verlag, 2016.

[BPK06] **M. Bruccoleri, Z. J. Pasek and Y. Koren:** *Operation management in reconfigurable manufacturing systems: Reconfiguration for error handling*. International Journal of Production Economics (2006). Publisher: Elsevier, vol. 100, issue 1, pp. 87-100.

[BRPN14] **J. Blesa, D. Rotondo, V. Puig and F. Nejjari:** *FDI and FTC of wind turbines using the interval observer approach and virtual actuators/sensors*. Control Engineering *Practice*, Jg. 24, 2014, pp. 138-155.

[BRU00] **J. Becker, M. Rosemann and C. von Uthmann:** *Guidelines of business process modeling*. In Business Process Management. Springer Berlin Heidelberg, 2000, pp. 30-49.

[BSP16+] **P. Bareiß, D. Schütz, R. Priego, M. Marcos and B. Vogel-Heuser:** *A model-based failure recovery approach for automated production systems combining SysML and industrial standards*. In Emerging Technologies and Factory Automation (ETFA), 2016 IEEE 21st International Conference on. IEEE, 2016, pp. 1-7.

[Bush14] **S. F. Bush:** *Smart grid: Communication-enabled intelligence for the electric power grid*. Publisher: John Wiley & Sons, 2014.

[CAA14] **S. Chitraganti, S. Aberkane and C. Aubrun:** *A novel mathematical setup for fault tolerant control systems with state-dependent failure process.* Journal of Physics: Conference Series (2014), IOP Publishing, vol. 570, No. 8, 2014.

[CCO12] **C.A.B. E Costa, M.C. Carnero and M.D. Oliveira:** *A multi-criteria model for auditing a Predictive Maintenance Programme*. European Journal of Operational Research (2012), Publisher: Elsevier, Jg. 217, Nr. 2, pp. 381-393.

[CGS14] **M. P. Cabasino, A. Giua and C. Seatzu:** *Diagnosability of discrete-event systems using labeled Petri nets*. IEEE Transactions on Automation Science and Engineering (2014), Publisher: IEEE, Jg. 11, Nr. 1, pp. 144-153.

[Cool03] **J. E. Cooling:** *Software engineering for real-time systems*. Dorchester: Pearson Education, 2003.

[CTT+17] **O. Cardin, D. Trentesaux, A. Thomas, P. Castagna, T. Berger and H. B. El-Haouzi:** *Coupling predictive scheduling and reactive control in manufacturing hybrid control architectures: state of the art and future challenges*. Journal of Intelligent Manufacturing, Jg. 28, Nr. 7, 2017, pp. 1503-1517.

[CZJW16] **F. Chen, K. Zhang, B. Jiang and C. Wen:** *Adaptive Sliding Mode Observer-Based Robust Fault Reconstruction for a Helicopter With Actuator Fault*. *Asian Journal of Control (2016)*. Publisher: John Wiley & Sons, Inc., vol. 18, Issue 4, pp. 1558-1565, 2016.

[DHJ17] **J. Dick, E. Hull and K. Jackson:** *Requirements engineering*. Publisher: Springer-Verlag, 2017, pp. 58-60.

[DIN10a] **DIN 62439-1:2010:** *Industrielle Kommunikationsnetze –Hochverfügbare Automatisierungsnetze – Teil 1: Grundlagen und Berechnungsmethoden*. Berlin: Beuth Verlag, 2010.

[DIN10b] **DIN EN 13306**:**2010-12**: *Instandhaltung - Begriffe der Instandhaltung*. Berlin: Beuth Verlag, Deutsches Institut für Normung eV, 2010.

[Donl07] **M. Donle:** *Strategien der Fehlerbehandlung: Umgang von Wirtschaftsprüfern, internen Revisoren und öffentlichen Prüfern mit den Fehlern der Geprüften*. Springer-Verlag, 2007.

[Dubr13] **E. Dubrova:** *Fault-tolerant design*. Berlin: Springer, 2013.

[DuSi16] **S. Duvvuri and B. Singhal:** *Spark for Data Science*. 1. Aufl., Birmingham: Packt Publishing Ltd., 2016.

[Eber14] **C. Ebert:** *Systematisches Requirements Engineering: Anforderungen ermitteln, dokumentieren, analysieren und verwalten*. Published by dpunkt. verlag, 5. Auflage, 2014.

[ESA07] **J. M. Emmert, C. E Stroud and M. Abramovici:** *Online fault tolerance for FPGA logic blocks*. IEEE Transactions on Very Large-Scale Integration (VLSI) Systems (2007), vol. 15, No. 2, pp. 216-226.

[FA2722] **N. Laaber:** *Aufbau einer Schnittstelle zwischen einem automatisierten System und dem zugehörigen Problemmanagementsystem,* research thesis, Institute of Industrial Automation and Software Engineering, University of Stuttgart, 2015.

[FiRe14] **R. Fielding and J. Reschke:** *Hypertext transfer protocol (HTTP/1.1): Message syntax and routing*. 2014. https://tools.ietf.org/html/rfc7230

[FrGö15] **A. Friedrich and P. Göhner:** Fault diagnosis of automated systems using mobile devices. In Emerging Technologies & Factory Automation (ETFA), 2015 IEEE 20th Conference on, IEEE, 2015pp. 1-8.

[Frit05] **S. Fritz:** Customer Self Care Services im Internet: Aktueller Einsatz und Entwicklungsperspektiven der Selbstbedienungsangebote [M]. Diplomica Verlag, 2005, pp. 81-82.

[FSV13] **T. Frank, D. Schütz and B. Vogel-Heuser:** *Funktionaler Anwendungsentwurf für agentenbasierte, verteilte Automatisierungssysteme*. In Agentensysteme in der Automatisierungstechnik. Publisher: Springer Berlin Heidelberg, 2013, pp. 3-19.

[Gert15] **J. Gertler:** *Fault Detection and Diagnosis*. Springer Publishing, London, pp. 417-422, 2015.

[Glav06] **M. Glavic:** *Agents and multi-agent systems: a short introduction for power engineers*. 2006.

[GNYL15] **Z. Guo, E. Ngai, C. Yang and X. Liang:** *An RFID-based intelligent decision support system architecture for production monitoring and scheduling in a distributed manufacturing environment*. International journal of production economics, Jg. 159, 2015, pp.16-28.

[Goll12] **J. Goll:** *Methoden und Architekturen der Softwaretechnik*. Vieweg+ Teubner Verlag, 2011.

[GrHe13] **S. Gregor and A.R. Hevner:** *Positioning and presenting design science research for maximum impact*. MIS quarterly, Jg. 37, Nr. 2, 2013.

[GuGe04] **F. Guo and J. K. Gershenson:** *A comparison of modular product design methods based on improvement and iteration*. In Proceedings of the 2004 ASME Design Engineering Technical Conferences-16th International Conference on Design Theory and Methodology, Publisher: ASME, Sept. 2004, pp. 261-269.

[HAA+10] **K. Holmberg, A. Adgar, A. Arnaiz, E. Jantunen, J. Mascolo and S. Mekid:** *E-maintenance*. Springer Science & Business Media, 2010.

[Hane06] **A. Hanemann:** *A hybrid rule-based/case-based reasoning approach for service fault diagnosis*. In Advanced Information Networking and Applications, 2006. AINA 2006. 20th International Conference on. Publisher: IEEE, 2006, pp. 5.

[HeCh10] **A. Hevner and S. Chatterjee:** *Design science research in information systems*. In Design research in information systems. Springer, Boston, MA, 2010, pp. 9-22.

[HMDJ08] **Y. Huang, R. McMurran, G. Dhadyalla and R. P. Jones:** *Probability based vehicle fault diagnosis: Bayesian network method*. Journal of Intelligent Manufacturing (2008), Jg. 19, Nr. 3, pp. 301-311.

[HMWF17] **X. L. Hoang, P. Marks, M. Weyrich and A. Fay:** *Modeling of interdependencies between products, processes and resources to support the evolution of mechatronic systems*. IFAC-PapersOnLine, Jg. 50, Nr. 1, 2017, pp. 4348-4353.

[HSS05] **A. Hanemann, D. Schmitz and M. Sailer:** *A framework for failure impact analysis and recovery with respect to service level agreements*. In Services Computing *IEEE International Conference on*. Publisher: IEEE, 2005, pp. 49-56.

[IEC61508] **IEC 61508:** *Functional safety of electrical/ electronic/ programmable electronic safety-related systems***. 2010.**

[ImSa13] **A. Immonen and A. Saaksvuori:** *Product lifecycle management*. Springer Science & Business Media, 2013.

[IPW10] **P. G. Ipeirotis, F. Provost and J. Wang:** Quality management on Amazon Mechanical Turk. In *Proceedings of the ACM SIGKDD workshop on human computation*, ACM, 2010, pp. 64-67.

[Iser05] **R. Isermann:** *Model-based fault-detection and diagnosis–status and applications*. Annual Reviews in control (2005), Publisher: Elsevier, Jg. 29, Nr. 1, pp. 71-85.

[Iser06] **R. Isermann:** *Fault-diagnosis systems: an introduction from fault detection to fault tolerance*. Publisher: Springer Science & Business Media, 2006.

[ISO26262] **ISO 26262:** *Road vehicles - Functional safety.* ISO copyright office, 2011.

[JBM+17] **S. Jeschke, C. Brecher, T. Meisen, D. Özdemir and T. Eschert:** *Industrial internet of things and cyber manufacturing systems*. In Industrial Internet of Things. Springer, Cham, 2017, pp. 3-19.

[Jela12] **M. Jelali:** *Control performance management in industrial automation: assessment, diagnosis and improvement of control loop performance*. Springer Science & Business Media, 2012.

[JiYu12] **J. Jiang, and X. Yu:** *Fault-tolerant control systems: A comparative study between active and passive approaches*. Annual Reviews in control, Jg. 36, Nr. 1, 2012, pp. 60-72.

[KAGB12] **E. Kamal, A. Aitouche, R. Ghorbani and M. Bayart:** *Robust fuzzy fault-tolerant control of wind energy conversion systems subject to sensor faults*. IEEE Transactions on Sustainable Energy, Jg. 3, Nr. 2, 2012, pp. 231-241.

[KeVo13] **K. Kernschmidt and B. Vogel-Heuser:** *An interdisciplinary SysML based modeling approach for analyzing change influences in production plants to support the engineering*. In Automation Science and Engineering (CASE), 2013 IEEE International Conference on. Publisher: IEEE, 2013, pp. 1113-1118.

[KhDe11] **K. Khazraei and J. Deuse:** *A strategic standpoint on maintenance taxonomy*. Journal of Facilities Management (2011), Jg. 9, Nr. 2, pp. 96-113.

[KhPo15] **A. H. Khan and I. Porres:** *Consistency of UML class, object and statechart diagrams using ontology reasoners*. Journal of Visual Languages & Computing (2015), Jg. *26*, pp. 42-65.

[KiCh13] **K. J. Kim and K. Y. Chung:** *IT Convergence and Security 2012*. Publisher: Springer, 2013.

[KJW17] **M. Klein, N. Jazdi and M. Weyrich:** *A Concept of Semantic Description for e-Production Systems in Manufacturing*. Procedia CIRP, Jg. 62, 2017, pp. 589-593.

[KoCa00] **F. Kon and R. H. Campbell:** *Automatic configuration of component-based distributed systems*. Dissertation, University of Illinois at Urbana-Champaign, 2000.

[KoKr07] **I. Koren and C. M. Krishna:** *Fault-tolerant systems*. Morgan Kaufmann Publishers. 2007.

[KuTu08] **T. Kurtoglu and I. Y. Tumer:** *A graph-based fault identification and propagation framework for functional design of complex systems*. Journal of Mechanical Design (2008), Jg. 130, Nr. 5.

[KuVa11] **B. Kuechler and V. Vaishnavi:** *Promoting relevance in IS research: An informing system for design science research*. Informing science: The international journal of an emerging transdiscipline, Jg. 14, Nr. 1, 2011, pp. 125-138.

[Lano09] **K. Lano:** *UML 2 semantics and applications*. Publisher: John Wiley & Sons, 2009.

[LaOv11] **P. A. Laplante and S. J. Ovaska:** *Real-time systems design and analysis: tools for the practitioner*. Publisher: John Wiley and Sons, 2011.

[LCR03] **F. Lima, L. Carro and R. Reis:** *Designing fault tolerant systems into SRAM-based FPGAs*. In *Proceedings of the 40th annual Design Automation Conference*, ACM, 2003, pp. 650-655.

[LeVo17] **C. Legat and B. Vogel-Heuser:** *A configurable partial-order planning approach for field level operation strategies of PLC-based industry 4.0 automated manufacturing systems*. Engineering Applications of Artificial Intelligence, Jg. 66, 2017, pp. 128-144.

[LGSZ14] **H. Li, H. Gao, P. Shi and X. Zhao:** *Fault-tolerant control of Markovian jump stochastic systems via the augmented sliding mode observer approach*. Automatica, Jg. 50, Nr. 7, 2014, pp.1825-1834.

**[**LiYa12] **X. J. Li and G. H. Yang:** *Robust adaptive fault-tolerant control for uncertain linear systems with actuator failures*. IET control theory & applications, Jg. 6, Nr. 10, 2012, pp. 1544-1551.

[LLWY09] **B. Lu, Y. Li, X. Wu and Z. Yang:** *A review of recent advances in wind turbine condition monitoring and fault diagnosis*. In Power Electronics and Machines in Wind Applications, 2009. PEMWA 2009, IEEE, 2009, pp. 1-7.

[LNH+04] **F. G. de Lima Kastensmidt, G. Neuberger, R. F. Hentschke, L. Carro and R. Reis:** *Designing fault-tolerant techniques for SRAM-based FPGAs*. IEEE Design & Test of Computers (2004), Jg. 21, Nr. 6, pp. 552-562.

[LWZ17] **L. Liu, Z. Wang and H. Zhang:** *Adaptive fault-tolerant tracking control for MIMO discrete-time systems via reinforcement learning algorithm with less learning parameters*. IEEE Transactions on Automation Science and Engineering, Jg. 14, Nr. 1, 2017, pp. 299-313.

[Lyu07] **M. R. Lyu:** *Software reliability engineering: A roadmap*. In 2007 Future of Software Engineering. IEEE Computer Society, Publisher: IEEE, 2007, pp. 153-170.

[LZZZ17] **H. Liu, W. Zhao, Z. Zuo and Y. Zhong:** *Robust control for quadrotors with multiple time-varying uncertainties and delays*. IEEE Transactions on Industrial Electronics, Jg. 64, Nr. 2, 2017, pp.1303-1312.

[MA2800] **L. Chen:** *Entwicklung eines Remotesystems zur Fehlerbehandlung und Rekonfiguration für die IAS-Abfüllanlage*, Masters thesis, Institute of Industrial Automation and Software Engineering, University of Stuttgart, 2016.

[MA2801] **M. Fei:** *Realization of fault handling and reconfiguration for the IAS-High bay ware house*, Master thesis, Institute of Industrial Automation and Software Engineering, University of Stuttgart, 2016.

[MA2913] **Y. Li:** *Extension and improvement of a user interface for the dynamic fault handling and reconfiguration system*, Master thesis, Institute of Industrial Automation and Software Engineering, University of Stuttgart, 2017.

[Mack13] **U. Mackenroth:** *Robust control systems: theory and case studies*. Springer Science & Business Media, 2013.

[Mate10] **J. Matevska:** *Rekonfiguration komponentenbasierter Softwaresysteme zur Laufzeit*. Publisher: Vieweg+ Teubner, 2010, pp. 73-81.

[McHa02] **I. McFarland and P. Harrison:** *Mastering Tomcat Development*. By Wiley Publishing, Inc. Indianapolis, Indiana. 2002

[MHR03] **J. Matevska-Meyer, W. Hasselbring and R. H. Reussner:** *Exploiting protocol information for speeding up runtime reconfiguration of component-based systems*. Universität Oldenburg, 2003.

[MHWF18] **P. Marks, X. L. Hoang, M. Weyrich and A. Fay:** *A systematic approach for supporting the adaptation process of discrete manufacturing machines. Research in Engineering Design*, Jg. 29, Nr. 4, 2018, pp.621-641.

[MJKJ14] **M. Mane, M. Joshi, A. Kadam and S.D. Joshi:** *Software Reliability and Quality Analyser with Quality Metric Analysis Along With Software Reliability Growth Model*. International Journal of Computer Science & Information Technologies (2014), AIRCC Publishing, vol. 5, pp. 364-369.

[Mobl02] **R.K. Mobley:** *An introduction to predictive maintenance*. Publisher: Butterworth-Heinemann, 2002.

[MoFu13] **M. Mori and M. Fujishima:** *Remote monitoring and maintenance system for CNC machine tools*. Procedia CIRP (2013), Publisher: Elsevier, Jg. 12, pp. 7-12.

[MRZ+13] **A. J. A. Majumder, F. Rahman, I. Zerin, W. Ebel Jr and S. I. Ahamed:** *iPrevention: Towards a novel real-time smartphone-based fall prevention system*. In Proceedings of the 28th Annual ACM Symposium on Applied Computing, ACM, March 2013, pp. 513-518.

[MSPB12] **R. Merzouki, A. K. Samantaray, P. M. Pathak and B. O Bouamama:** *Intelligent mechatronic systems: modeling, control and diagnosis*. Springer Science & Business Media, 2012.

[MT2782] **C. Wang:** *Development of a software prototype for a remote fault handling and reconfiguration system of the IAS coffee maker*, Masters thesis, Institute of Industrial Automation and Software Engineering, University of Stuttgart, 2016.

[MuGö10a] **H. Mubarak and P. Göhner:** *An agent-oriented approach for self-management of industrial automation systems*. In: Industrial Informatics (INDIN), 2010 8th IEEE International Conference on. IEEE, 2010. pp. 721-726.

[MuGö10b] **H. Mubarak and P. Göhner:** *Einsatz von Agenten für das Selbstmanagement von Automatisierungssystemen*. Multikonferenz Wirtschaftsinformatik *2010*, 2010, pp. 167.

[MUK00] **M. G. Mehrabi, A. G. Ulsoy and Y. Koren:** *Reconfigurable manufacturing systems: Key to future manufacturing*. Journal of Intelligent manufacturing (2000). Publisher: Springer, Jg. 11, Nr. 4, pp.403-419.

[NoJo09] **J. Noble and R. Johnson:** *Transactions on Pattern Languages of Programming I.* Berlin Heidelberg: Springer Verlag, 2009, pp. 67-68.

[OdSt12] **P. F. Odgaard and J. Stoustrup:** *Fault tolerant control of wind turbines using unknown input observers*. In: 8th IFAC symposium on fault detection, supervision and safety of technical processes. Elsevier Science, 2012, pp. 313-318.

[ÖFH15] **M. Öhman, M. Finne and J. Holmström:** *Measuring service outcomes for adaptive preventive maintenance*. International Journal of Production Economics (2015), Publisher: Elsevier, vol. 170, part b, pp. 457-467.

[PAEM15] **R. Priego, A. Armentia, E. Estévez and M. Marcos:** *On applying MDE for generating reconfigurable automation systems*. In Industrial Informatics (INDIN), 2015 IEEE 13th International Conference on. IEEE, 2015, pp. 1233-1238.

[PaHa07] **L. Paradis and Q. Han:** *A survey of fault management in wireless sensor networks*. Journal of Network and systems management (2007), Publisher: Spring, vol. 15, issue 2, pp.171-190.

[PAO+14] **R. Priego, A. Armentia, D. Orive, E. Estévez and M. Marcos:** ***A*** *Model-based Approach for Achieving Available Automation Systems*. IFAC Proceedings Volumes, Jg. 47, Nr. 3, 2014, pp. 3438-3443.

[Part12] **H. A. Partsch:** *Specification and transformation of programs: a formal approach to software development*. Publisher: Springer Science & Business Media, 2012.

[PDK15] **B. Pachauri, J. Dhar and A. Kumar:** *Incorporating inflection S-shaped fault reduction factor to enhance software reliability growth*. Applied Mathematical Modelling, 2015, Jg. *39 Nr.* 5-6, pp. 1463-1469.

[PIGM17] **R. Priego, N. Iriondo, U. Gangoiti and M. Marcos:** *Agent-based middleware architecture for reconfigurable manufacturing systems*. The International Journal of Advanced Manufacturing Technology, Jg. 92, Nr. 5-8, 2017, pp. 1579-1590.

[Pine16] **M. L. Pinedo:** *Scheduling: theory, algorithms, and systems*. Springer, 2016.

[PMD+17] **D. Pantförder, F. Mayer, C. Diedrich, P. Göhner, M. Weyrich and B. Vogel-Heuser:** *Agentenbasierte dynamische Rekonfiguration von vernetzten intelligenten Produktionsanlagen*. In Handbuch Industrie 4.0 Bd. 2, Springer Berlin Heidelberg in Press, 2017, pp. 31-44.

[Prie17] **R. Priego:** *A Model-based Approach for Supporting Flexible Automation Production Systems and an Agent-based Implementation*. Chair of Information Technology in Mechanical Engineering, Technische Universität München, 2017.

[PRK12] **K. Peffers, M. Rothenberger and B. Kuechler:** Design Science Research in Information System: Advances in Theory and Practice. In Proceedings of 7th International Conference, DESRIST 2012, USA, Springer-Verlag, Berlin Heidelberg, 2012.

[PSU13] **P. S. Pulat, S. C. Sarin and R. Uzsoy:** *Essays in Production, Project Planning and Scheduling: A Festschrift in Honor of Salah Elmaghraby.* Springer Science & Business Media, vol. 200, 2013.

[PTA10] **J. C. Ponsart, D. Theilliol and C. Aubrun:** *Virtual sensors design for active fault tolerant control system applied to a winding machine*. Control Engineering Practice, Jg. 18, Nr. 9, 2010, pp. 1037-1044.

[RaGo11] **B. S. M. P. S. Ramaiah and A. A. Gokhale:** *FMEA and fault tree based software safety analysis of a railroad crossing critical system***.** Global Journal of Computer Science and Technology, 2011.

[RFHG16] **M. Rakyta, M. Fusko, M. Haluska and P. Grznár:** *Maintenance support system for reconfigurable manufacturing systems*. In *Proceedings of 26th DAAAM International Symposium on Intelligent Manufacturing and Automation, DAAAM*, vol. 2015, 2016, pp. 1102-1108.

[RHWL11] **J. H. Richter, W. P. M. H. Heemels, N. van de Wouw and J. Lunze:** *Reconfigurable control of piecewise affine systems with actuator and sensor faults: stability and tracking*. Automatica, Jg. 47, Nr. 4, 2011, pp. 678-691.

[RNPB12] **D. Rotondo, F. Nejjari, V. Puig and J. Blesa:** *Fault tolerant control of the wind turbine benchmark using virtual sensors/actuators*. IFAC Proceedings Volumes, Jg. 45, Nr. 20, 2012, pp. 114-119.

[Roth10] M. Roth: *Identification and Fault Diagnosis of Industrial Closed-loop Discrete Event Systems: Identifikation und Fehlerdiagnose Industrieller Ereignisdiskreter Closed-Loop-Systeme*. Logos Verlag Berlin GmbH, 2010.

[RSS16] **S. Rastogi, S. Shrivastava and A. Sharma:** *A QoS based methodology for multiple fault handling in SOA.* In Electrical, Computer and Electronics Engineering (UPCON), 2016 IEEE Uttar Pradesh Section International Conference on. Publisher: IEEE, 2016, pp. 300-304.

[RuQu12] **C. Rupp and S. Queins:** *UML 2 glasklar: Praxiswissen für die UML-Modellierung*. Publisher: Carl Hanser Verlag GmbH Co KG, 2012

[SA2721] **S. Eichler:** *Entwicklung einer Smartphone-Anwendung für ein Problemmanagementsystem*, research thesis, Institute of Industrial Automation and Software Engineering, University of Stuttgart, 2015.

[SA2861] **Q. Wei:** *Portierung und Erweiterung des Remote-Systems zur Fehlerbehandlung und Rekonfiguration für den IAS-Kaffeeautomaten*, Masters thesis, Institute of Industrial Automation and Software Engineering, University of Stuttgart, 2016.

[ScHv18] **F. Schorr and L. Hvam:** *Design Science Research: A Suitable Approach to Scope and Research IT Service Catalogs*. In 2018 IEEE World Congress on Services (SERVICES). IEEE, 2018, pp. 25-26.

[ScVo13] **D. Schütz and B. Vogel-Heuser:** *Werkzeugunterstützung für die Entwicklung von SPS-basierten Softwareagenten zur Erhöhung der Verfügbarkeit*. In: Agentensysteme in der Automatisierungstechnik. Springer, Berlin, Heidelberg, 2013. pp.291-303.

[ShJh02] **L. Shang and N. K. Jha:** *Hardware-software co-synthesis of low power real-time distributed embedded systems with dynamically reconfigurable FPGAs*. In Design Automation Conference, 2002. Proceedings of ASP-DAC 2002. 7th Asia and South Pacific and the 15th International Conference on VLSI Design. 2002, IEEE, pp. 345-352.

[Shyr12] **W. J. Shyr:** *Teaching mechatronics: An innovative group project-based approach*. Computer Applications in Engineering Education (2012), published by John Wiley & Sons, Inc. pp. 93-102.

[SiFe06] **M. L. Silva and J. C. Ferreira:** *Support for partial run-time reconfiguration of platform FPGAs*. Journal of Systems Architecture (2006). Publisher: Elsevier vol. 52, issue 12, pp. 709-726.

[SMBG02] **G. D. M. Serugendo, D. Mandrioli, D. Buchs and N. Guelfi:** *Real-time synchronised Petri nets*. In International Conference on Application and Theory of Petri Nets. Springer, Berlin, Heidelberg, 2002, pp. 142-162.

[Stap09] **R. F. Stapelberg:** *Handbook of reliability, availability, maintainability and safety in engineering design*. Published by Springer Science & Business Media, 2009.

[SWLV13] **D. Schütz, A. Wannagat, C. Legat and B. Vogel-Heuser:** *Development of PLC-Based Software for Increasing the Dependability of Production Automation Systems*. IEEE Trans. Industrial Informatics, Jg. 9, Nr. 4, 2013, pp.2397-2406.

[SZW16] **J. Schmidt, A. Zeller and M. Weyrich:** *Ansatz zur modellgetriebenen Entwicklung flexibler Automatisierungssysteme durch Serviceorientierung*. https://www.ias.uni-stuttgart.de/dokumente/publikationen/2016_Ansatz_zur_modellgetriebenen_Entwicklung_flexibler_Automatisierungssysteme_durch_Serviceorientierung.pdf

[SZW17] **J. P. Schmidt, A. Zeller and M. Weyrich:** *Modellgetriebene Entwicklung serviceorientierter Anlagensteuerungen*. at-Automatisierungstechnik 2017, Publisher: Berlin, Boston: Oldenbourg Wissenschaftsverlag. Jg. 65, Nr. 1, pp. 26-36.

[ThJa10] **B. T. Thumati and S. Jagannathan:** *A model-based fault-detection and prediction scheme for nonlinear multivariable discrete-time systems with asymptotic stability guarantees*. IEEE Transactions on Neural Networks (2010), Publisher: IEEE, Jg. 21, Nr. 3, pp. 404-423.

[Tolk12] **A. Tolk:** *Engineering principles of combat modeling and distributed simulation*. Publisher: John Wiley & Sons, 2012.

[TPB13+] **D. Trentesaux, C. Pach, A. Bekrar, Y. Sallez, T. Berger, T. Bonte and J. Barbosa:** *Benchmarking flexible job-shop scheduling and control systems*. Control Engineering Practice, Jg. 21, Nr. 9, 2013, pp. 1204-1225.

[Trep15] **T. Trepper:** *Forschungsmethodik*. In Fundierung der Konstruktion agiler Methoden. Springer Gabler, Wiesbaden, 2015, pp. 11-28.

[TYM10] **E. Tobin, H. Yin and K. Menzel:** *Methodology for Maintenance Management Utilising Performance Data*. *eWork and eBusiness in Architecture, Engineering and Construction*. Publisher: Taylor & Francis Group, London, pp. 331-338, 2010.

[VHBL15] **D. Vey, S. Hugging, S. Bodenburg and J. Lunze:** *Control reconfiguration of physically interconnected systems by decentralized virtual actuators*. *IFAC-PapersOnLine (2015)*, IFAC in Press, Jg. *48, Nr.* 21, pp. 360-367.

[VLL15] **B. Vogel-Heuser, J. LEE and P. LEITÃO:** *Agents enabling cyber-physical production systems*. at-Automatisierungstechnik, Jg. 63., Nr. 10, 2015, pp. 777-789.

[Voge17a] **B. Vogel-Heuser:** *Herausforderungen und Anforderungen aus Sicht der IT und der Automatisierungstechnik*. In Handbuch Industrie 4.0 Bd. 4. Springer Vieweg, Berlin, Heidelberg, 2017, pp. 33-44.

[Voge17b] **B. Vogel-Heuser and J. Prieler:** *Evaluation of selected metrics for flexibility of Cyber Physical Production Systems*. In: Automation Science and Engineering (CASE), 2017 13th IEEE Conference on. IEEE, 2017, pp. 701-708.

[VoNe17] **B. Vogel-Heuser and E. M. Neumann:** *Adapting the concept of technical debt to software of automated Production Systems focusing on fault handling, mode of operation and safety aspects*. IFAC-PapersOnLine, Jg. 50, Nr. 1, 2017, pp. 5887-5894.

[VRF+16] **B. Vogel-Heuser, S. Rösch, J. Fischer, T. Simon, S. Ulewicz and J. Folmer:** *Fault handling in PLC-based industry 4.0 automated production systems as a basis for restart and self-configuration and its evaluation*. Journal of software engineering and applications, Jg. 9, Nr. 1, 2016, pp. 1.

[Wagn14] **D. Wagner:** *The Graceful Degradation of the Knowledge Worker? : On getting back the attention of what used to be your most valuable resource.* International Conference: ReThinking Management, Karlshochschule, S. 1, 2014.

[Wang02] **H. Wang:** *A survey of maintenance policies of deteriorating systems*. European Journal of Operational Research (2002), Jg. 139, Nr. 3, pp. 469-489.

[Wang04] **M. H. Wang:** *Grey-extension method for incipient fault forecasting of oil-immersed power transformer*. Electric Power Components and Systems (2004), Publisher: Taylor and Francis Ltd, Jg. 32, Nr. 10, pp. 959-975.

[Wann10] **A. Wannagat:** *Entwicklung und Evaluation agentenorientierter Automatisierungssysteme zur Erhöhung der Flexibilität und Zuverlässigkeit von Produktionsanlagen*. 2010. Doktorarbeit. Technische Universität München.

[WaVo08a] **A. Wannagat and B. Vogel-Heuser**: *Increasing flexibility and availability of manufacturing systems-dynamic reconfiguration of automation software at runtime on sensor faults*. IFAC Proceedings Volumes, Jg. 41, Nr. 3, 2008, pp. 278-283.

[WaVo08b] **A. Wannagat and B. Vogel-Heuser**: *Agent oriented software-development for networked embedded systems with real time and dependability requirements in the domain of automation*. *I*FAC Proceedings Volumes, Jg. 41, Nr. 2, 2008, pp. 4144-4149.

[WaWe16] **Y. Wang and M. Weyrich:** *Towards a novel learning assistant for networked automation systems*. In Machine Learning for Cyber Physical Systems, Springer Berlin Heidelberg, 2016, pp. 51-57.

[Weyr18] **M. Weyrich:** *Towards future Automation Systems – Cyber physical, intelligent, flexible and efficient*. In International Conference SIMULATION 2018, 12.-14. September 2018, Kiev, Ukraine, 2018.

[WiPa16] **M. Witczak, M. Pazera:** *Marcin. Fault Tolerant-Control: Solutions and Challenges*. Pomiary Automatyka Robotyka, Jg. 20, Nr. 1, 2016, pp. 5-16.

[WJG15] **H. Wang, N. Jazdi and P. Göhner:** *Higher availability of an Industrial Automation System based on a Remote Problem Management System*. In Proceeding of ICICM 2015 - International Conference on Information Communication and Management, Paris, 2015.

[WJW15] **H. Wang, N. Jazdi and M. Weyrich:** *Fault Effect Analysis based on a modelling Approach for Requirements, Functions and Components*. In Proceeding of ACEC 2015 - 3rd International Conference on Advances in Computing, Electronics and Communication, Zürich, 2015.

[WKS+17] **M. Weyrich, M. Klein, J. P. Schmidt, N. Jazdi, K. D. Bettenhausen, F. Buschmann, C. Rubner, M. Pirker and K. Wurm:** *Evaluation Model for Assessment of Cyber-Physical Production Systems*. In: Industrial Internet of Things. Springer, Cham, 2017, pp. 169-199.

[WMSP17] **X. Wang, S. McArthur, S. Strachan and B. Paisley:** *Decision support for distribution automation: data analytics for automated fault diagnosis and prognosis*. 4th International Conference on Electricity Distribution, Stevenage in Press, 2017.

[WSV13] **A. Wannagat, D. Schütz and B. Vogel-Heuser**: *Einsatz von Softwareagenten am Beispiel einer kontinuierlichen, hydraulischen Heizpresse*. In: Agentensysteme in der Automatisierungstechnik. Springer, Berlin, Heidelberg, 2013. pp. 169-185.

[WWBF14] **M. Weyrich, L. Wior, D. Bchennati and A. Fay:** *Flexibilisierung von Automatisierungssystemen*. Systematisierung der Flexibilitätsanforderungen von Industrie, Jg. 4, 2014, pp. 106-111.

[WWW08] **Q. Wang, B. Wen and X. Wang:** *Multi-agent based intelligent video monitoring for unattended substation*. Intelligent Networks and Intelligent Systems, 2008. ICINIS'08. First International Conference on. Publisher: IEEE, 2008, pp. 515-518.

[WXH+10] **B. Wang, Y. Xiong, Z. Hu, H. Zhao, W. Zhang and H. Mei:** *A Dynamic-Priority based Approach to Fixing Inconsistent Feature Model*. Model Driven Engineering Languages and Systems Lecture Notes in Computer Science, 2010, pp. 181-195.

[WZS+16] **M. Weyrich, A. Zeller, J. P. Schmidt, A. Faul and P. Marks:** *Engineering und Betrieb Smarter Komponenten in IoT-Netzwerken für die Automatisierung der Produktion.* in VDE-Kongress 2016 Internet der Dinge, 2016.

[Yang06] **C. Yang:** *Service-Oriented Architecture*. The International Encyclopedia of Geography. 2006.

[YJSZ15] **H. Yang, B. Jiang, M. Staroswiecki and Y. Zhang:** *Fault recoverability and fault tolerant control for a class of interconnected nonlinear systems*. Automatica *(2015),* Publisher: Elsevier, Jg. 54, pp. 49-55.

[ZhLy10] **Z. Zheng and M. R. Lyu:** *An adaptive QoS-aware fault tolerance strategy for web services.* Empirical Software Engineering, vol. 15, Issue 4, pp. 323-345, 2010.

[ZhLy15] **Z. Zhenga and M. R. Lyu***: Selecting an optimal fault tolerance strategy for reliable service-oriented systems with local and global constraints*. IEEE Transactions on Computers (2014). Publisher: IEEE, vol. *64, Nr. 1*, pp. 219-232.

[ZNM18] **A. Zeller, N. Jazdi and M. Weyrich:** *Verifikation verteilter Automatisierungssysteme auf Basis einer Modellkomposition*. at-Automatisierungstechnik, Jg. 66, Nr. 6, 2018, pp. 456-470.

[ZXL07] **J. Zhao, L. Xu and L. Liu:** *Equipment fault forecasting based on ARMA model*. In Mechatronics and Automation 2007, ICMA 2007. International Conference on, IEEE, 2007, pp. 3514-3518.

# Appendix A

**Table of the category of student works**

| | |
|---|---|
| **Concerning investigation works** | |
| Automatic recognition and performance of the per-classified reconfiguration possibilities in automation systems | FA2887 |
| Investigation of the possibility of Reconfiguration and Building a 3D-Simulator for the IAS High-bay warehouse | FA2799 |
| Conception and realization of an application to handle new faults in automation systems | FA2887 |
| Survey and Analysis of Problem-Management-Technologies | SA2732 |
| **Concerning conception test works** | |
| Conception and realization of a software for identification and classification of unknown faults of automated systems | FA2828 |
| Development of a concept based on ontology for the fault-handling and reconfiguration | FA2796 |
| Development of a concept to formalize the knowledge for the reconfiguration of automated systems | SA2783 |
| Development of a concept to analyze problems in a problem management system | FA2740 |
| Development for a software for determination of fault effects in industrial automation systems | MA2735 |
| Development of a concept of fault diagnosis based on error detection and localization for industrial automation systems | MA2720 |
| Concept of diagnosing problems from existing automation systems | FA2695 |
| **Concerning system development works** | |
| Development of a remote system for fault handling and reconfiguration for the IAS-Bottling plan | MA2800 |
| Realization of fault handling and reconfiguration for the IAS high-bay warehouse | MA2801 |

| | |
|---|---|
| Development of a software prototype for a remote fault handling and reconfiguration system of the IAS coffee maker | MT2782 |
| Porting and extension of the remote fault handling and reconfiguration system for the IAS coffee maker | SA2861 |
| Design and realization of a web-based maintenance software and the corresponding user-interface | MA2834 |
| Development of a universal problem management system for variant automated system | BA2737 |
| Development of an interface of an automated system and a problem management system | FA2722 |
| Development of a smartphone application for a problem management system | SA 2721 |
| Development of a simulator of the IAS bottling plant | BA2719 |
| Development of a problem-management system for the IAS bottling plant | FA2717 |
| Development of a function-oriented concept for the agent-based problem management system of a high-bay warehouse | MA2665 |
| Development of an agent-based concept to identify unaffected functions in case of a module failure | BA2664 |
| Development of a simulation to check the appropriate agent knowledge in the agent-based problem management system | BA2629 |
| Extension and improvement of an user interface for the dynamic fault handling and reconfiguration system | MA2913 |

**Concerning further improvement and application works**

| | |
|---|---|
| Application possibilities of the dynamic fault handling and reconfiguration of cyber-physical systems | MA2912 |
| Design and development of a tool for formalizing the system knowledge | MA2914 |
| Conception and design of fault localization software to estimate the possible fault locations | MA2930 |

# Appendix B

**Table of the content of the fault handling database**

| Table | KType | Description |
|---|---|---|
| Symptom_ES | SMK | Listing all symptoms for the entire system |
| Symptom_SS | SMK | Listing all symptoms for each subsystem (the number of this type table is equal to the number of subsystems) |
| Fault | FK | Listing all occurred faults |
| Fault_Statistic | FK | Listing all processed faults with the info from defective industrial automation system |
| Reconfiguration | FK | Listing all reconfiguration commands for all processed faults |
| Matrix_CF | SSK | Depicting the relationship between components and functions |
| Matrix_CS | SSK | Depicting the relationship within a component model |
| Matrix_FF | SSK | Depicting the relationship between functions |
| Matrix_FR | SSK | Depicting the relationship between functions and requirements |
| Matrix_RR | SSK | Depicting the relationship between requirements |
| Matrix_CR | SSK | Depicting the relationship between components and requirements |
| Matrix_Redund | SSK | Depicting the relationship between redundant entities, e.g. functions |
| Component | SSK | Listing all attributes for each component |
| Function | SSK | Listing all attributes for each function |
| Requirement | SSK | Listing all attributes for each requirement |
| Contact | SSK | Listing contact information regarding various maintenance staff |

# Lebenslauf

## Persönliche Daten

| | |
|---|---|
| 20.03.1987 | geboren in Qingdao (China) |

## Schulbildung

| | |
|---|---|
| 1993 – 1999 | Longquanwangjia Xiaoxue (Grundschule), Qingdao, China |
| 1993 – 2005 | Hongshiya Zhongxue und Jiaonanshiyan Zhongxue (Mittelschule), Qingdao, China, Abschluss Abitur |

## Studium

| | |
|---|---|
| 2005 – 2009 | Studium der Elektrotechnik an der Qingdao Universität<br>Studienschwerpunkt: Automatisierung<br>Abschluss: Bachelor of Science |
| 2009 – 2012 | Studium der Elektrotechnik an der Tongji Universität<br>Studienschwerpunkt: Kontrolltheorie und Steuerungstechnik<br>Abschluss: Master of Science |

## Berufstätigkeit

| | |
|---|---|
| Seit 2012 | Wissenschaftlicher Mitarbeiter am Institut für Automatisierungstechnik und Softwaresysteme der Universität Stuttgart |