University of Stuttgart
Institute of Industrial Automation and Software Systems
Prof. Dr.-Ing. M. Weyrich

20.07.2016

Master Thesis

Chuyu Wang 2782 MT

Development of a Software Prototype for a Remote Fault Handling and Reconfiguration System of the IAS Coffee Maker

Supervisor: Prof. Dr.-Ing. M. Weyrich
M.Sc. Huiqiang Wang

Table of Contents

Table of Figures

Table of Tables

Abbreviations

GUI	**G**raphical **U**ser **I**nterface
IAS	**I**nstitut für **A**utomatisierungstechnik und **S**oftwaresysteme
MySQL	**MY S**equential **Q**uery **L**anguage
PC	**P**ersonal **C**omputer
TCP	**T**ransmission **C**ontrol **P**rotocol
JSON	**J**avaScript **O**bject **N**otation
UML	**U**nified **M**odeling **L**anguage
ID	**Id**entification

Terminology

Browser	Computer program for displaying web pages.
Class	Generalization of an object that describes the attributes of the objects.
Diagnosis	Gives a result by analysing existing data.
GUI	Graphical user interface.
Automation System	Systems that can automatically carry out a process without additional operations.
IP	Numerical label assigned to each device participating in a computer network that uses the Internet Protocol for communication.
Java	Object oriented programming.
Java Swing	Swing is a GUI widget toolkit for Java.
Method	Defines the behaviour of objects.
Object	Instance of a class.
Port	Part of the network address, which causes the assignment of connections to programs by the operating system.
Socket	Bi-directional interface for network communication.
Thread	Sequential program segments which are executed in parallel.

Zusammenfassung

Das Konzept der ferngesteuerten Fehlerbehandlung und Systemneukonfiguration zielt auf eine Erhöhung der Verfügbarkeit von Automatisierungssystemen ab, indem nicht beeinträchtigte Funktionen im Fehlerfall aktiviert werden.

Im Rahmen dieser Arbeit wurde die IAS-Kaffeemaschine nach dem von Herrn Wang [WJW15] vorgeschlagenen Anforderung-Funktion-Komponenten-Modellierungsansatz analysiert. Basierend auf Untersuchungen früheren Arbeiten über diese Kaffeemaschine konnten deren Funktionen in drei Level eingeteilt werden: Die Hauptfunktionen, die Unterfunktionen und die Grundfunktionen. Ebenfalls lassen sich die Komponenten der Kaffeemaschine in drei Gruppen aufteilen: Das Gesamtsystem, das Untersystem und Komponenten. Bezüglich den Anforderungen werden in dieser Arbeit lediglich die nichtfunktionalen Anforderungen betrachtet. Danach werden alle logischen Zusammenhänge der IAS-Kaffeemaschine nach dem Anforderungs-Funktions-Komponenten-Modellierungsansatz identifiziert.

Ein Software-Prototype, welcher ferngesteuert alle Fehler der IAS-Kaffeemaschine behandelt, wurde entwickelt. Ein Algorithmus zur Funktionsanalyse des Software-Prototypen wurde vorgeschlagen. Dieser Algorithmus ist in der Lage die logischen Zusammenhänge der IAS-Kaffeemaschine zu vereinfachen und tabellarisch darzustellen. Des Weitern kann dieser Algorithmus leicht erweitert werden und an andere Automatisierungssysteme zur Funktionsanalyse angepasst werden. Der Software-Prototype wurde mit Java und MySQL entwickelt und anschließend in einen Kaffeemaschinensimulator integriert. Außerdem existiert eine Verbindung zu einem Simulator des Fehlerdiagnosesystems um die Fehlerbehandlung durchzuführen und Funktionen neu zu konfigurieren.

Testergebnisse zeigen, dass Fehlerinformationen in dem entwickelten Software-Prototypen korrekt behandelt werden. Geeignete Neukonfigurationen werden an den IAS-Kaffeemaschinensimulator zurückgeführt. Somit wurde verifiziert, dass die Verfügbarkeit von Automatisierungssystemen durch das ferngesteuerte Fehlerbehandlungs-und Neukonfigurierungssystem gesteigt werden kann.

Abstract

Automation systems vary from size and structures, but most of them are not intuitively understandable by non-professional persons due to the complex physical and logical structure behind them. With development of technology, complexity of automation systems is increasing, which leads to more frequent fault occurrences and more difficult fault management. When considering industrial automation systems that affect productive efficiency, economical cost could be huge if the fault removal period is long [Bord13]. Therefore, it is of great demand to improve availability of automation systems.

To meet this demand, a concept known as remote fault handling and reconfiguration system has been proposed by the Institute of Industrial Automation and Software Engineering (IAS) of the University Stuttgart. With the proposed fault handling and reconfiguration system, fault information is being handled in a remote server and function reconfiguration is carried out by a simulated automation system basing on fault handling results. Thus, instead of shutting down all functions, functions that are not affected by the occurred faults can be continueously used, thereby improving availability and efficiency of automation systems.

This concept starts from a requirement-function-component system modelling approach, which can be used to build abstract views of an automation system. Then, complex logical relations behind the automation system can be expressed by applying this model approach.

In this thesis, a software prototype for a remote fault handling and reconfiguration system was developed for the IAS coffee maker, which provides fast fault handling mechanism and effective function reconfiguration instructions. Two databases are involved in this remote system, one for storing fault information like fault id, fault description, corresponding reconfiguration solution etc.; one for storing the identified logical relations involved in the IAS coffee maker. Besides, in order to verify the remote system, a coffee maker simulator was developed to simulate technical processes of coffee producing, and a simulator of fault diagnosis system was developed to simulate fault occurrence.

1 Introduction

1.1 Motivation

As most automation systems undertake production tasks or are designed to provide services, occasional fault occurrence in those systems could lead to productivity loss and economical loss. Thus great attention has been focused on building fault management mechanism for automation systems. Fault diagnosis is one of the hot research topics, which has made great progress in fault detection, fault elimination and fault prevention basing on fault effect analysis [Bord13]. Current fault diagnosis approaches are effective to improve the speed of locating faults and reduce the repeated emergence frequency of similar faults, thereby improving availability of automation systems. This condition leads to a new possibility that to improve function availability of automation systems by enabling partial operations of the system despite some faults occur.

However, with the development of technology, automation systems are becoming more and more complex, which leads to huge difficulties in function analyzing. Take a coffee maker as an example, amount number of components are assembled together to act as a complete system. Behind the services that a coffee maker provides, a set of complex physical and logical structures are involved. So for an automation system without extra fault diagnosis or fault management mechanism, a fault occurrence could lead to a total shutdown of the system. And considering the limited number of professional repairmen, it is usually time consuming to wait for the complete overhaul, which leads to low service availability and costly economical loss.

The idea of remote fault handling and reconfiguration system is to connect automation systems to a remote system that can do fault information analyzing and function analyzing, via which automation systems can enable unaffected functions basing on the feedback from the remote system. Therefore, availability as well as productivity of automation systems is enhanced.

1.2 General Purpose

This master thesis focuses on developing a software prototype that deals with faults of the IAS coffee maker, which is an instance of automation systems. Theory parts of the thesis consider the increasing complexity of handling faults from automation systems and the great need of enhancing the availability of non-affected functions when problems occur. The main concept of fault handling and function reconfiguration system includes many different catalogs. This thesis aims to build a remote fault handling system to fulfill passive management for known faults, active management for unknown faults. To achieve this goal, a function-requirement-component modelling approach [WJW15] should be used to analyze physical and logical structure of the IAS coffee maker. Besides, an efficient function analysis algorithm should be used to realize main function of the software prototype.

1.3 Operation Area

1.3.1 Areas of Application

The application of this developed software prototype is in the research of fault handling and function reconfiguration system for the IAS coffee maker.

1.3.2 User Group

There are four user groups of the developed software prototype:

- Coffee maker users, who make coffee orders on the graphical user interface (GUI) of a coffee maker simulator.
- Fault diagnosis system users, who can simulate component defects by pressing certain buttons on the GUI of fault diagnosis system simulator.
- Remote system administrator, who can view and manipulate database information, processing information via the remote system GUI.
- Maintenance staff, who can add fault information via a GUI to the remote system fault handling and reconfiguration system.

1.3.3 Operating Conditions

No special operating conditions are required for the developed software prototype. Normal personal computers' operating conditions should satisfy our requirements.

1.4 Environment

1.4.1 Software

To develop the software prototype, a programming editor, related software design tools, database tools and document generating tools are needed. The details are listed below:

- Microsoft Office
- Eclipse
- XAMPP
- Microsoft Visio

1.4.2 Hardware

Average configuration of a personal computer should match the requirement for hardware.

1.4.3 System Interfaces

Three user graphical interfaces are required:

- The GUI of the remote fault handling and reconfiguration system
- The GUI of the fault diagnosis system
- The GUI of a coffee maker simulator and a fault generator

1.5 Requirements

1.5.1 Functional Requirements

The developed software should fulfill the following requirements:

The structure of the IAS coffee maker should be analyzed from three aspects: requirement, function and component. Either the classification or the examples should be introduced in the first part.

- Requirements for the IAS coffee maker are identified and classified into two catalogs: Functional Requirements and Nonfunctional Requirements, where nonfunctional requirements include Performance Requirements and Safety Requirements.

- The structure of the IAS coffee maker is investigated and functions of the coffee maker are identified and cataloged into Main Functions, Sub-functions and Basic Functions.

- Components of the IAS coffee maker are identified and should be listed in the interface GUI for simulating component defects. Hierarchical pattern of the identified components is as: the whole system, sub-systems and components.

The existing student works towards problem management of the IAS coffee maker should be investigated to define functionalities of the problem management system. Functionalities of the problem management system are of different parts:

- Passive management of known problems, whose process is as: fault id is assigned when a known problem occurs, then the fault diagnosis system will query the fault id in the local database, if the fault id is in the database, interface would tell the remote fault handling system to trigger known fault treatment. Solutions of treating known faults are stored in the remote server and will be give back to the interface to direct function reconfiguration.

- Active management of unknown problems, if the fault id is not in the local database, interface would tell the remote system to trigger unknown fault treatment, which includes processes of determining the impact scope from requirement view, function view and component view, determining the available functions and directing function reconfiguration.

The knowledge of system models should be extended with specific requirements. A method should be found to formulize the system models. The existing system model involves function view and component view, which should be integrated with requirement view.

A software prototype of remote fault handling and reconfiguration system for the IAS coffee maker should be developed. It includes three modules listed below:

- **The remote fault handling and reconfiguration system**, which implements the system model, stores fault knowledge, identifies incoming faults (known or unknown) and gives reconfiguration solutions. A GUI is designed to shown data structure and working scheme of the remote server. It includes two databases, one for fault knowledge and one for system models.

- A simulator of **fault diagnosis system** stores fault knowledge and current system states from the local machine and makes updates to the remote server via the interface. A GUI is required to show the database information.

- A **simulator** simulating functions of the IAS coffee maker should be developed. Functions includes simulating coffee order, updating historical data in the local database and executing function reconfiguration instructions received from the remote system. The simulator should aslo includes a **fault generator** that allows users to perform fault occurrence simulation.

1.5.2 Non-functional Requirements

Two non-functional requirements are:

- Visible for the user to use this demonstrator.
- A deliverable software prototype for the remote fault handling and reconfiguration system can be presented as a successful demonstrator.

2 Basics

This chapter introduces basic knowledge used in development of a remote fault handling and reconfiguration system.

2.1 Life Cycle of Faults in Automation Systems

A fault in an automation system is generally defined as a condition that can lead to failures of a part of a system or failures of the whole system [BJG14].

Basing on current study, typical life cycle of faults in automation systems involves stages shown in Figure 2.1 [GGBA15]. It starts with normal operation, then comes the fault development stage. After an accumulating period, faults occur, which leads to the fault removal stage. By getting rid of all faults, automation systems can get back to the normal operation stage.

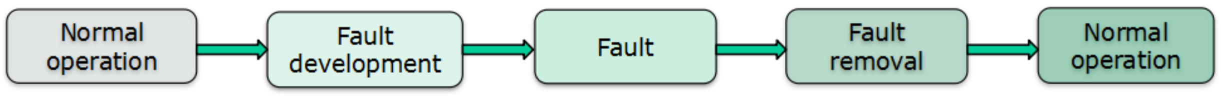


Figure 2.1: Lifte Cycle of Faults in Automation Systems without Fault Handling and Function Reconfiguration

Usually, the period between faults occurring and complete fault removal can be long because manual checking, manual repairing etc. are required. Considering possible economical loss causing by system failures during the system shutting down period, it is in great demand to build a fault management mechanism that can shorten this period or improve function availability during this period.

At present, fault diagnosis methods are able to speed up fault identification and fault removal, but it still needs a certain period of time to bring a defected system back to normal operation stage [LGW15]. Thus, some efforts should be dedicated to enable partial functions of a defected system before the overhaul is carried out and finished.

2.2 Fault Handling and Function Reconfiguration

Fault handling and reconfiguration system aims to handle fault information that comes from fault diagnosis process and give function reconfiguration solutions to enable the unaffected functions of an automation system before all faults are removed. In this way, availability of automation systems is improved during fault occurrence period. In this section, how does an automation work with a fault handling and reconfiguration system will first be introduced, then follows the software prototype of the proposed system.

2.2.1 Automation Systems with Fault Handling and Function Reconfiguration

Fault handling and function reconfiguration aims to determine available functions despite fault occurrence. It is a secondary analysis of the fault diagnosis, so it begins with the fault diagnosis results. Normally, fault handling and function reconfiguration processes are performed on a remote server that stores fault information and system rules that are used for function analysis and reconfiguration. In Figure 2.2, the life cycle of faults in automation systems with fault handling and function reconfiguration is depicted. By adding such a fault handling system, the period between fault occurrence and the partial operation stage becomes much shorter than the period between fault occurrence and the normal operation stage. The reasons are listed below:

- The added system runs automatically in a remote server after it is setting up successfully.
- Digital communication between automation systems and the remote system is very fast.
- Much less human activities are required to enable partial operation of automation systems.

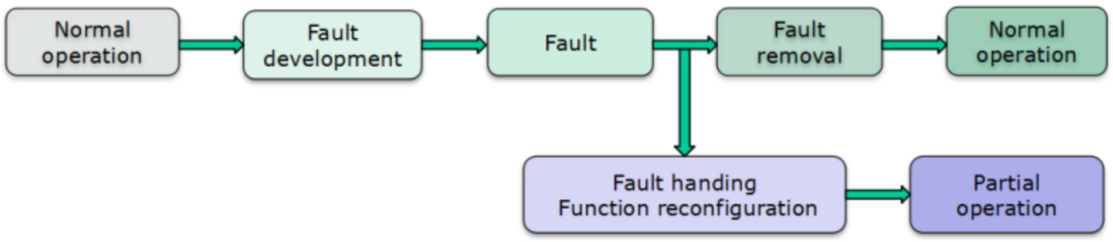


Figure 2.2: Life Cycle of Faults Automation Systems with Fault Handling and Function Reconfiguration

2.2.2 Problems of Remote Fault Handling and Function Reconfiguration

To insure partial operaiton of automation systems during fault occurrence period is meaningful, meanwhile there are some underlying problems and difficulties, among which, the algorithm to figure out available functions by known fault locaitons from analysis of fault diagnosis results ist he main concern. Since an automation system consists of a number of components which are organized in different ways in different technical processes, it involves lots of database data queries and desicions after each query. Generally, problems comes from two aspects: the ability to fully describe all involved relations of the algorithm and the system efficiency. To solve the problem of relaitons description and storing, it is importance of finding a systematically approach to dissect automation systems, which means physical structures of automaiton systems should be decipted in a hierarchical way that could be mapped to logical relations of components involved in each technical process. For the system efficiency, it is in great need to consider the database query efficiency and complexity of the function analysis algorithm.

2.2.3 Solutions

Regarding the aforesaid problems, current research focuses on representing function logical relations in a tree structure or a table structure.

The tree structure of relations forms all functions in a multi-level tree shown in Figure 2.3, where nodes (A, B, C, D, E, F) stand for functions of an automation system. Data formed in a tree structue is easy to be manipulated in programming language since it is one of the most typical data structure in computer science. Moreover, functios are intuitively depicted in a hierarchical way. The shortness of tree structure ist that OR relation between funcitons cannot be represented. One possible solution may be using one tree to represent function relations of one technical process, but for huge automation systems include tons of technical processes the tree structure requires a huge number of database queries and nodes checking.

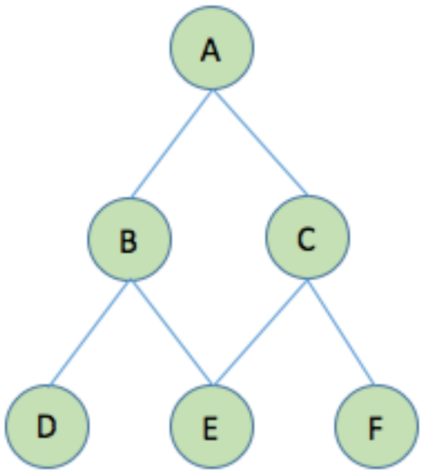


Figure 2.3: Tree structure of functions

Fort the table structure, if the same relations with the above tree structure are considered, a table like Table 2.1 shown can be constructed. It depicts relations by listing all related functions of a single function. The main benefit of this way is saving database space since all related functions are recorded in a cell of the table. Correspondingly, the shortness of this structure is obvious: first, the function hierarchy is not intuitive; second, the OR relation between functions is not included in the table, which burdens the function analysis program with defining explicit checking of each function queried.

Table 2.1: Related Functions

Function	**Related Functions**
A	B, C
B	D, E
C	F

2.3 Requirement-Function-Component Modeling Approach

Considering the increasing complexity of automation systems, in order to realize a remote fault handling and reconfiguration system, a proper modeling approach is first needed to abstract a complex automation system. My supervisor Mr. Wang [WJW15] of the IAS develop the concept of a requirement-function-component modeling approach. With this model, a complex automation system can be analyzed hierarchically from each aspect, and relations between objects from different aspects can be identified.

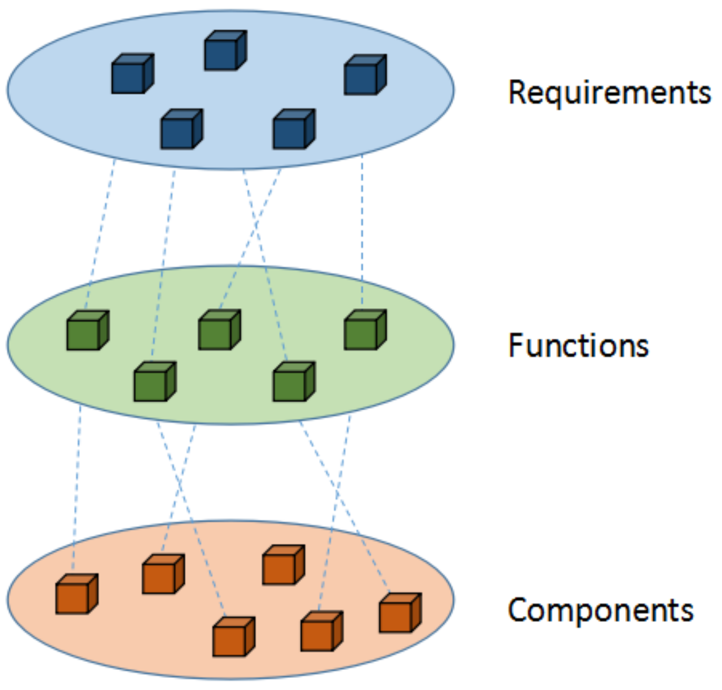


Figure 2.4: The Requirement-Funciton-Component Modeling Approach

Normally, these three models are provided by the manufacturer of an automation system. The development of an automation usually starts from requirement model, then function model and then component model. To identify which functions are available given fault locations, a fault propagation route could also be found using these three models and mappings between them [WJW15]. As shown in Figure 2.4, a whole automation system can be divided into several levels and signal components or elements are identified at the buttom level, where faults are identified. By mapping from component model to function model, defected basic functions will be first targeted. Then basing on relations between functions [PDA+08] a fault propagation route can be identified in the function model which also consists of a certain number of levels. Meanwhile, a fault propagation route is identified in the requirement model. After affected requiremtns are found, available functions achieved from function model can be further eliminated basing on results of requirement analysis. For instance, if a temperature sensor in the kettle of a coffee maker is defected, it has no effect on any functions of the coffee maker basing on function analysis, but it does affect on water heating function if safety requirement is considered. Thus, by combining requiremnt analysis result and function requirement result, a more accurate decision can be achieved.

2.4 Software Prototype Overview

In this section, a general introduction of the required software prototype [WJG15] is introduced. Figure 2.5 shows the overview this software prototype.

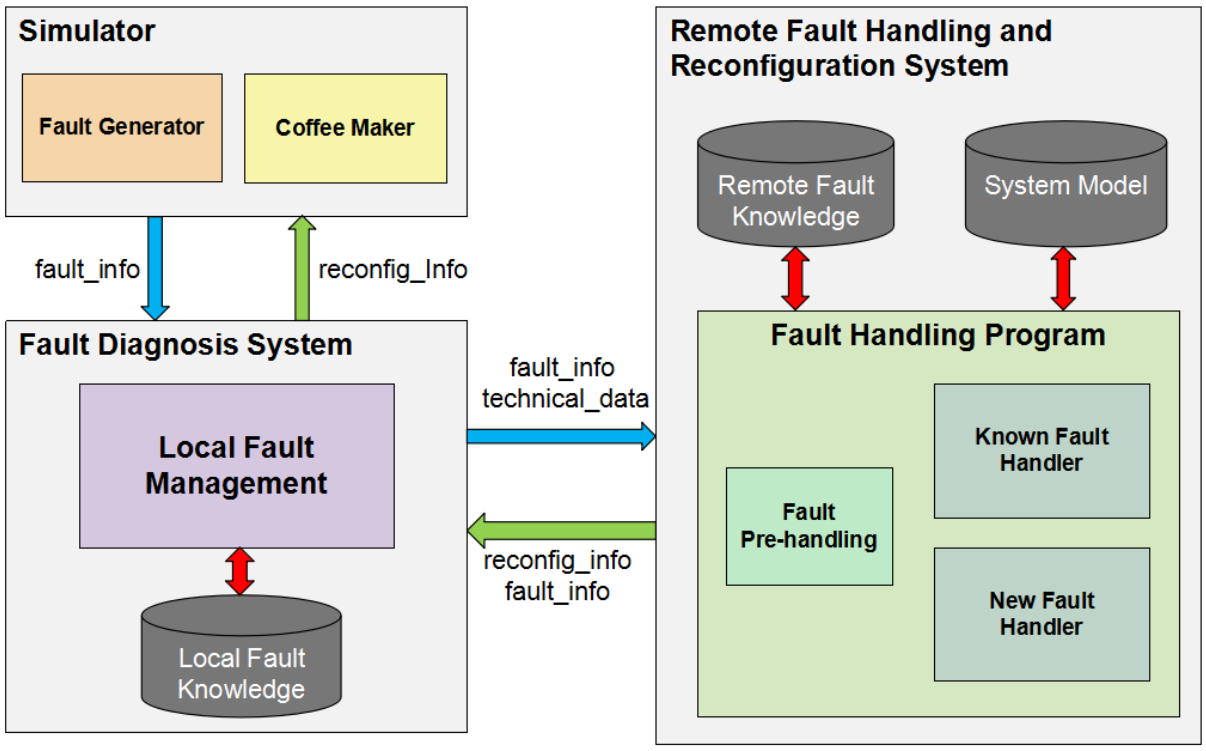


Figure 2.5: Software Prototype Overview

The remote fault handling and reconfiguration system involves two databases, one for storing fault information like fault id, fault reason, reconfiguration solution etc. The other one stores the rules that are used to handle faults and analyze functions of the IAS coffee maker i.e. the process of figuring out which functions are available and which are unavailable given the occurrence of a fault. Two key modules are included in this system, one to handle faults that are already recorded in the remote database, the other one to handle faults that are new to the system. The fault diagnosis system (FDS) and the simulator are used to verify the remote fault handling and reconfiguration system. Generally, fault information is generated from the simulator that simulates automation system activities, then the fault information is passed to a local fault diagnosis system that manages the automation system. The fault diagnosis system can make a decision basing on its own database knowledge that if the occurred fault ever occurred before. And the determined result is sent to the remote system which performs function analysis and function reconfiguration tasks. Finally, the reconfiguration solution indicating functions that still can be used is sent to the local FDS. And then the simulator executes the solution.

3 Conception

3.1 IAS Coffee Maker Analysis

In this section, requirement model, function model and component model of the IAS coffee maker will be introduced first and then follows a function analysis algorithm basing on these three models. In this chapter, all objects are assigned with cooresponding IDs.

3.1.1 Requirement Model of IAS Coffee Maker

In product development, a requirement is a physical or functional need that a product must be able to perform [Wiki16]. Requirements are classified into different types at different development stages. There are architecture requirements, user requirements, functional requirements and non-functional requirements etc. To make a precise decision of finding the affected and unaffected functions of an automation system when a fault occurs, it is in great need to consider requirement model during the system development phase. For the IAS coffee maker, this thesis only considers functional requirements and non-functions requirements. Functional requirements are capabilities of a system, while non-functional requirements can be used to judge the performance of a system, rather than behaviors or capabilities of a system. In this thesis, functional requirements are considered in the function model, so only nonfunctional requirements which include temperature safety requirements and pressure safety requirements of the IAS coffee maker are considered.

3.1.2 Function Model of IAS Coffee Maker

From aspect of function model, three levels are considered in this thesis: main function, sub-function and basic function. Taking the structure of IAS coffee maker into consideration, manufacturer materials and existing student work were investigated. Description of each function level is shown in the following [EGW15] [LGW15].

3.1.2.1 Main Functions

Table 3.1: IAS Coffee Maker Main functions' IDs and Names

ID	Name
MF1	Milk Coffee
MF2	Hot Water
MF3	Cup Coffee
MF4	Pot Coffee
MF5	Coffee Cream
MF6	Espresso
MF7	Latte Macchiato
MF8	Milk Foam
MF9	Cappuccino

The coffee types that the IAS coffee maker can provide are considered as main functions, whose IDs and names are described in Table 3.1.

3.1.2.2 Sub-functions

Basing on the investigation of manufacturer materials and student work of the IAScoffee maker, nine sub-functions are identified. Table 3.2 lists IDs and brief description of the identified sub-functions.

Table 3.2: IAS Coffee Maker Sub-functions' IDs and Description

ID	Description
F1	Espresso Grinding
F2	Coffee Grinding
F3	Coffee Granules Rolling
F4	Brewing
F5	Hot Water Producing
F6	Steam Producing
F7	Hot Milk Producing
F8	Trash Collecting
F9	Spouting

3.1.2.3 Basic Functions

In this thesis, each basic function is mapped to a component or an element, which means the number of basic functions depends on the number of components and elements. ID and brief description of each basic function is listed in Table 3.3 below.

Table 3.3: IAS Coffee Maker Basic functions' IDs and Description

ID	Description
f001	Sensing amount of resources
f002	Grinding coffee beans
f003	Grinding espresso beans
f004	Transporting coffee granules
f005	Providing hot water for brewing
f006	Measuring water flow speed
f007	Providing hot steam for espresso nozzle
f008	Ejecting liquid
f009	Sensing amount of coffee waste
f010	Transporting milk
f011	Releasing extra pressure

f012	Controlling steam
f013	In-drafting air
f014	Providing cold water
f015	Pumping water
f016	In-taking cold water
f017	Mixing hot steam and cold water
f018	Providing hot water
f019	Sensing temperature of milk
f020	Controlling automation processes
f021	Sensing brewer temperature
f022	Sensing brewer pressure
f023	Sensing kettle temperature
f024	Sensing kettle pressure
f025	Supporting sensing amount of resources
f026	Containing coffee beans
f027	Containing espresso beans
f028	Containing coffee granules
f029	Container for coffee brewing
f030	Contain coffee waste
f031	Container for water heating
f032	Ejecting hot water
f033	Ejecting coffee
f034	Switch for safely removing coffee waste
f035	Containing milk
f036	Connecting coffee maker to water tap

3.1.3 Component Model of IAS Coffee Maker

The IAS Coffee Maker is viewed as the whole system in this thesis. Component model of the coffee maker is also divided into three levels: the whole system, sub-system and component. Figure 3.1 shows the overall hierarchical structure of the IAS coffee maker from component view. In the following, details of sub-systems and components are described.

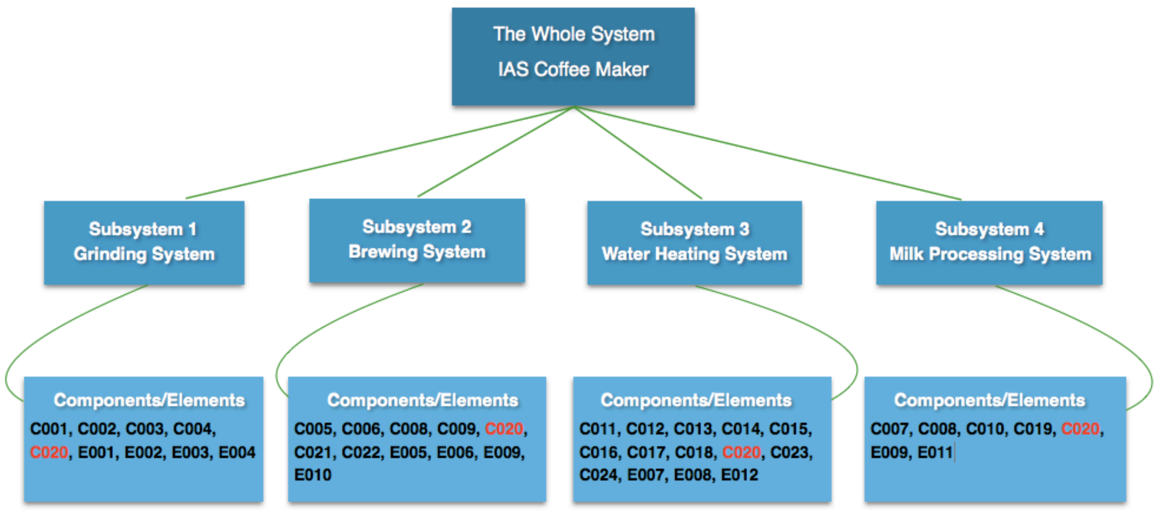

The Whole System
IAS Coffee Maker
Subsystem 1
Grinding System
Subsystem 2
Brewing System
Subsystem 3
Water Heating System
Subsystem 4
Milk Processing System
Components/Elements
C001, C002, C003, C004, C020, E001, E002, E003, E004
Components/Elements
C005, C006, C008, C009, C020, C021, C022, E005, E006, E009, E010
Components/Elements
C011, C012, C013, C014, C015, C016, C017, C018, C020, C023, C024, E007, E008, E012
Components/Elements
C007, C008, C010, C019, C020, E009, E011

Figure 3.1: Component Model of the IAS Coffee Maker

3.1.3.1 Sub-systems

Four sub-systems are identified for the IAS coffee Maker, which are: Grinding System, Brewing System, Water Heating System and Milk Processing System. An overview of the four sub-systems are shown in Figure 3.2.

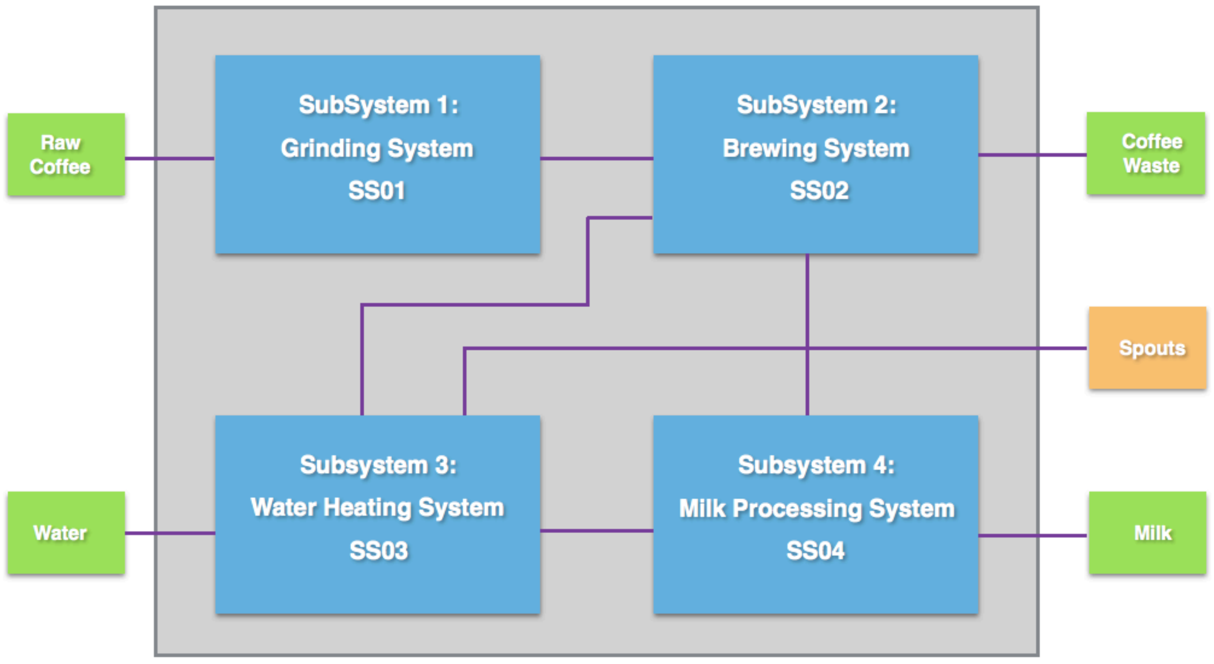


Figure 3.2: Sub-system Overview of the IAS Coffee Maker

Grinding System

This sub-system is identified by ID SS01. It deals with coffee grinding process. It concerns about turning raw coffee beans into granules that are ready for brewing.

Brewing System

The ID for brewing system is SS02. This sub-system mainly concerns the brewing temperature, brewing pressure, the amount of coffee and hot water and coffee waste collection.

Water Heating System

This sub-system is identified by ID SS03. All processes related to water heating, e.g pumping water from the water tap and heating water in the kettle, are handled in this sub-system.

Milk Processing System

The ID for the fourth sub-system is SS04. This is a sub-systems that involves all activities related to milk processing.

3.1.3.2 Components

The IAS coffee maker are assembled by 24 components and 12 Elements. Their corresponding IDs and names are described in Table 3.4.

Table 3.4: IAS Coffee Maker Components' IDs and Names

ID	Description
C001	HEW Reed Sensor
C002	Coffee Grinder Motor
C003	Espresso Grinder Motor
C004	Snail Scoop
C005	Hot Water Brewing Valve
C006	Flow Meter
C007	Jet-Valve
C008	Espresso Nozzle
C009	Proximity Switch
C010	Ball Valve
C011	Vent Valve
C012	Steam Valve
C013	Air Pump
C014	Cold Water Brewing Valve
C015	Pump
C016	Cold Water Intake Valve
C017	Mixing Valve
C018	Hot Water Valve
C019	Milk Temperature Sensor
C020	Micro Controller
C021	Brewing Temperature Sensor
C022	Brew Pressure Sensor

C023	Kettle Temperature sensor
C024	Kettle Pressure Sensor
E001	HEW Electromagnet
E002	Coffee Bean Hopper
E003	Espresso Hooper
E004	Granules Hopper
E005	Brewer
E006	Waste Container
E007	Kettle
E008	Hot Water Spout
E009	Coffee Spout
E010	Safety Switch
E011	Milk Container
E012	Water Tap Connector

3.2 Fault Handling and Function Reconfiguration Algorithm

3.2.1 Problem Description

In the above section, requirements, functions and components of the IAS coffee maker are identified hierarchically, but the relations between them have not been described. In human language, it is not hard to illustrate the relation between two objects, however, when a fault handling system for automation systems is required, one of most important preconditions is to describe those relations in a language that the computer can understand. In this thesis, the remote fault handling and reconfiguration system resorts to a database to store involved relations, which would lead to two critical considerations: the ability to fully describe all involved relations and the system efficiency. Taking these two considerations into account, a mathematical method has been found to represent relations involved in the IAS coffee maker.

3.2.2 Matrix Representation of Relations

Before introducing the algorithm, a few mathematical terms should be defined.

Relation Matrix

Relation matrix is of the form shown below, where $rel(x, y)$ indicates the relation between two variables x and y.

$$\begin{pmatrix} rel(x_1,y_1) & rel(x_1,y_2) & \cdots & rel(x_1,y_n) \\ rel(x_2,y_1) & rel(x_2,y_2) & \cdots & rel(x_2,y_n) \\ \vdots & \vdots & \ddots & \vdots \\ rel(x_n,y_1) & rel(x_n,y_2) & \cdots & rel(x_n,y_n) \end{pmatrix}$$

Here, the value 1 is used to represent that two variables are related while the value 0 is used to represent that two variables are not related. Thus relation matrices in this thesis only contain element 0 or 1. Besides, the term 'related' can refer to different relations between functions [PDA+08] depending on relation matrix types, i.e. AND relation matrix, OR relation matrix, dependency relation matrix considering the IAS coffee maker case.

Input Column Matrix

Elements of an input column matrix are used to indicate states of variables, which is in the form shown below:

$$\begin{pmatrix} I_{y_1} \\ I_{y_2} \\ \vdots \\ I_{y_n} \end{pmatrix}$$

Also, elements of the input matrix are 1 or 0. The digit 1 indicates that the state of a variable is negative, i.e. abnormal state while the digit 0 indicates a positive i.e. normal state of a variable. For instance, $I_{y_1} = 0$ represents that variable y_1 is at normal state.

Result Column Matrix

A result column matrix is also used to represent variable states. The differences between an input column matrix and a result column matrix are:

- Elements of an input column matrix are already known before matrix calculation while elements of a result matrix are determined after the calculation.
- Elements of a result matrix do not limited to 0 and 1. They can be any value depending on matrix calculation.
- Element 0 in a result column matrix represents normal state of a variable while a non-zero value represents abnormal state of a variable.

The form of a result column matrix is defined as:

$$\begin{pmatrix} R_{x_1} \\ R_{x_2} \\ \vdots \\ R_{x_n} \end{pmatrix}$$

Here, R_{x_n} represents the state of variable x_n.

3.2.3 State Determination Equation

Basing on the mathematical terms defined in the last section, an equation is proposed to determine states of variables by giving a relation matrix and an input matrix. Below follows the equation:

$$\begin{pmatrix} rel(x_1,y_1) & rel(x_1,y_2) & \cdots & rel(x_1,y_n) \\ rel(x_2,y_1) & rel(x_2,y_2) & \cdots & rel(x_2,y_n) \\ \vdots & \vdots & \ddots & \vdots \\ rel(x_n,y_1) & rel(x_n,y_2) & \cdots & rel(x_n,y_n) \end{pmatrix} \cdot \begin{pmatrix} I_{y_1} \\ I_{y_2} \\ \vdots \\ I_{y_n} \end{pmatrix} = \begin{pmatrix} R_{x_1} \\ R_{x_2} \\ \vdots \\ R_{x_n} \end{pmatrix}$$

The meaning of this equation is intuitive. Since digit 0 represents normal states in the input column matrix, the corresponding element in the result column matrix will definitely be 0. Therefore only abnormal variables have effects on the result column matrix and the abnormal variable only affects variables that are related to it. In the following, a simple example that consider relations of 5 variables is used to describe how states of a set of variables can be determined by giving states of another set of variables and the relation matrix of these two sets of variables. Below, variables x_1, x_2 are variables wholse states are required to be determined. Variable y_1 , y_2 and y_3 are variables whose states are already known by in the input column matrix. Supposing x_1 relates to y_1 and y_3 while x_2 only relates to y_2, then it gives the relation matrix as:

$$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

The first row of the relation matrix stands for x_1's relations with y_1, y_2 and y_3. The second row stands for x_2's relations with y_1, y_2 and y_3.

If we have known that y_3 is at an abnormal state while the other two variables are at normal state, according to the aforesaid state determination equation, we can achieve states of x_1 and x_2 by the following calculation:

$$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

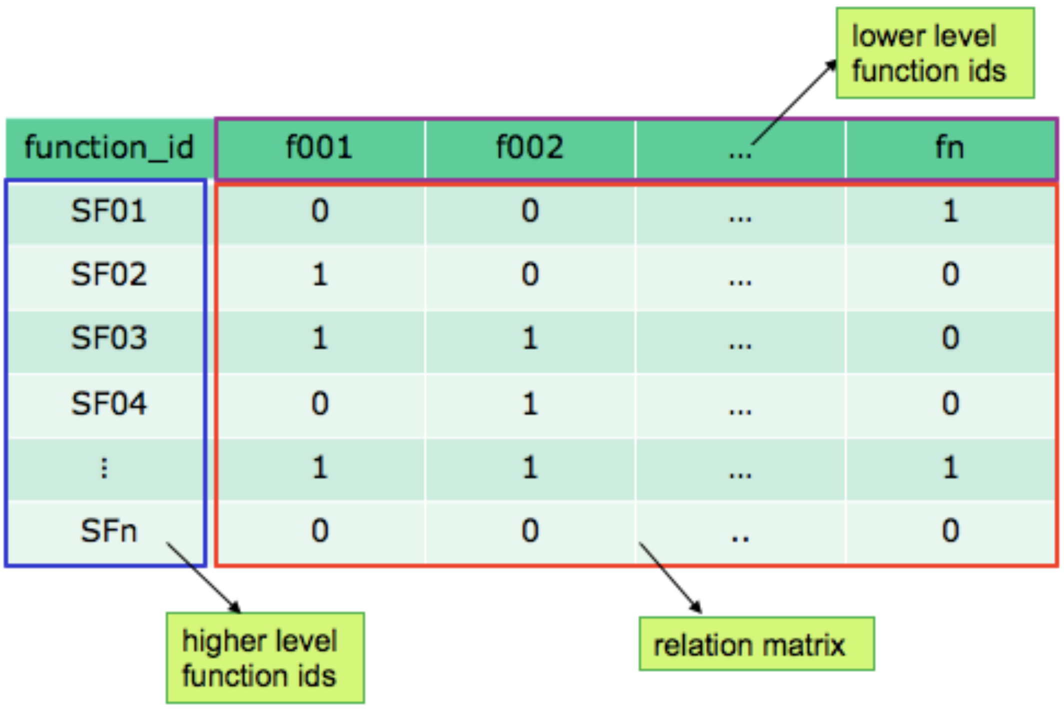
The value 1 in the result column matrix indicates that the state of x_1 is abnormal and the second element 0 means that x_2 is at normal state.

3.2.4 Function Analysis Algorithm

3.2.4.1 Basics of Function Analysis Algorithm

Basing on the above mathematical definition, a function analysis algorithm is found to determine unaffected functions of automation systems given occurrence of a specific fault. To apply this algorithm, functions' relations are variables of the relation matrix and functions' states are variables of input column matrix as well as result column matrix. In the following, higher level functions refer to functions whose states are required to be determined while lower level functions refer to functions whose states have already been identified in the fault diagnosis stage.

Before introducing the detailed work flow, a table form representation of the relation matrix and involved variables should be described. Figure 3.3 shows the structure of the depicted table, where the first column stores IDs for higher level functions, and the first row stores the IDs for lower level functions. The area marked by the red line is used to store elements of the relation matrix. Thus, the digit 1 in the second row second column means that SF02 requires f001 in the context of function analysis.



function_id	f001	f002	...	fn
SF01	0	0	...	1
SF02	1	0	...	0
SF03	1	1	...	0
SF04	0	1	...	0
⋮	1	1	...	1
SFn	0	0	..	0

lower level function ids
higher level function ids
relation matrix

Figure 3.3: A Table Containing Relation Matrix and Related Function IDs

By intergrating the relation matrix into a table, relations between functions can be intuitively depicted. But it has to be made sure that one relation matrix should only describes one kind of relations. If more than one kind of relaitons are considered, more relation matrices should be constructed. The rules of constructing tables containing relation matrices are introduced below.

One integrated table is used to define AND relation between higher level function and lower level function, as well as relations between higher level functions and extra functions. Here, extra functions stand for combination functions of some lower level functions. For instance, if three lower level functions' states (f1, f2, f3) are involved to determine the state of a higher level function SF. Considering a situation that SF requires f1 plus one of the other two functions. Thus an extra function standing for relation (f2 or f3) should be added, which leads to two tables to be constructed as Table 3.5 and Table 3.6 shown.

Table 3.5: A Table Containing a Relation Matrix of Extra Functions and Lower Level Functions

function_id	f1	f2	f3
f4	0	1	1

Table 3.6: A Table Containing a Relation Matrix of All Functions

function_id	f1	f2	f3	f4
SF	1	0	0	1

The reasons of constructing relation matrices in the above way are intutitive. To determine the state of the higher level function SF, f2 and f3 should not be considered lonely since only when both of them fail would lead to abnormality of SF. Thus an extra function f4 is added to represent a combination function of f2 and f3 such that the state of f4 can be determined before deciding the state of SF. The details of how to do function analysis resorting to tables of the above form is introduced in next section basing on logical function relations of the IAS coffee maker.

3.2.4.2 Function Analysis Algorithm for Analyzing IAS Coffee Maker Functions

Considering the IAS coffee maker as an analysis object, the function analysis algorithm is divided into 10 steps described below.

Step 1: Identify all involved relations and catalog them

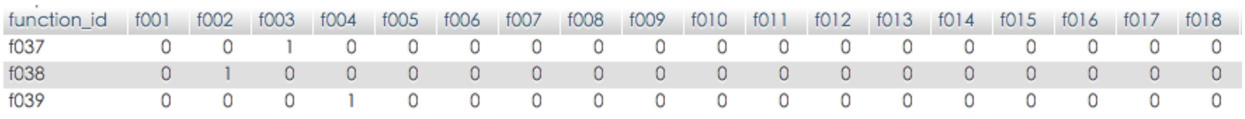
Relations involved in the IAS coffee maker were classified into the catalogs below:

- Dependency relations between basic functions
- OR relations between basic functions
- AND relations between basic functions and sub-functions
- AND relations between sub-functions and main functions
- AND relations between requirements and basic functions
- AND relations between requirements and main functions

Step 2: Create relation matrices in the database

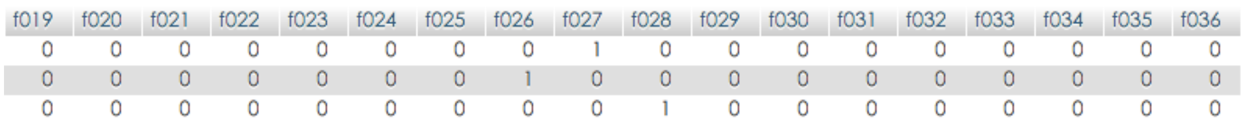
Basing on the above relation classification and function relations of the IAS coffee maker, six tables containing relation matrices were created in the remote database, which are shown from Table 3.7 to Table 3.8.

Table 3.7: The Table of Dependency Relations between Basic Functions (a)



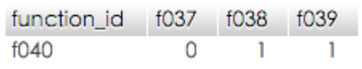
function_id	f001	f002	f003	f004	f005	f006	f007	f008	f009	f010	f011	f012	f013	f014	f015	f016	f017	f018
f037	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f038	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f039	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3.8: The Table of Dependency Relations between Basic Functions (b)



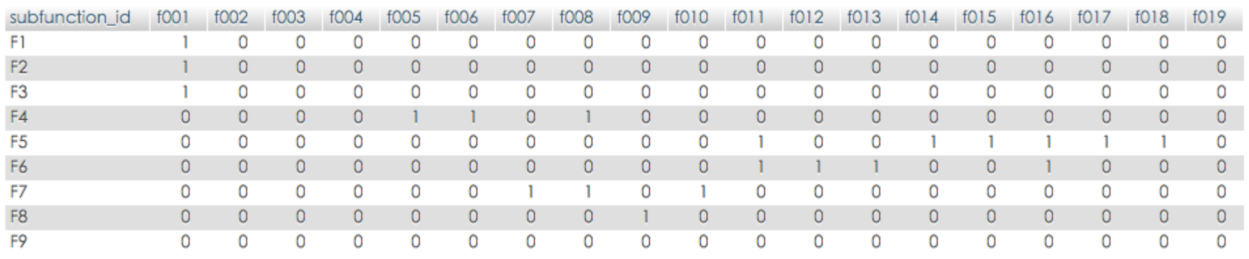
f019	f020	f021	f022	f023	f024	f025	f026	f027	f028	f029	f030	f031	f032	f033	f034	f035	f036
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Table 3.9: The Table of OR Relations between Basic Functions



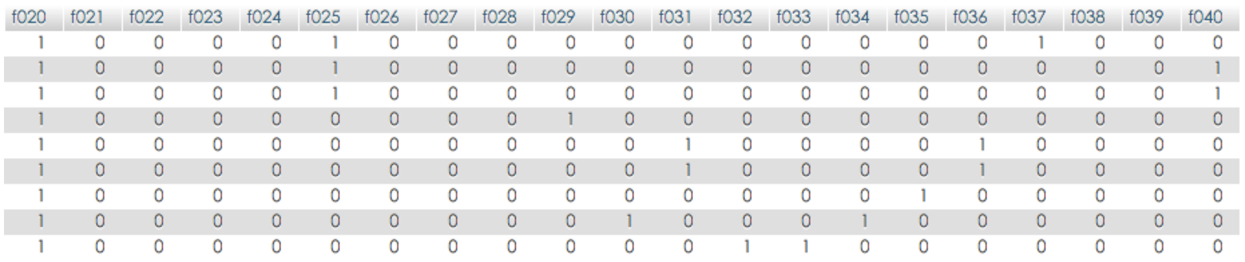
function_id	f037	f038	f039
f040	0	1	1

Table 3.10: The Table of AND Relations between Basic Functions and Sub-functions (a)



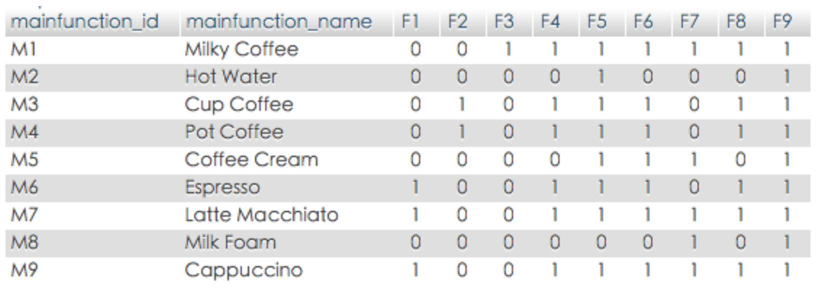
subfunction_id	f001	f002	f003	f004	f005	f006	f007	f008	f009	f010	f011	f012	f013	f014	f015	f016	f017	f018	f019
F1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F4	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
F5	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	0
F6	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	0	0
F7	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0
F8	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
F9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3.11: The Table of AND Relations between Basic Functions and Sub-functions (b)



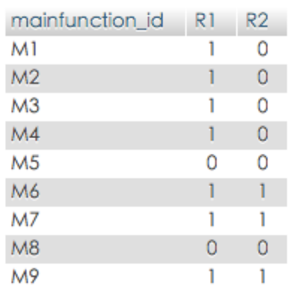
f020	f021	f022	f023	f024	f025	f026	f027	f028	f029	f030	f031	f032	f033	f034	f035	f036	f037	f038	f039	f040
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0

Table 3.12: The Table of AND Relations between Sub-functions and Main Functions



mainfunction_id	mainfunction_name	F1	F2	F3	F4	F5	F6	F7	F8	F9
M1	Milky Coffee	0	0	1	1	1	1	1	1	1
M2	Hot Water	0	0	0	0	1	0	0	0	1
M3	Cup Coffee	0	1	0	1	1	1	0	1	1
M4	Pot Coffee	0	1	0	1	1	1	0	1	1
M5	Coffee Cream	0	0	0	0	1	1	1	0	1
M6	Espresso	1	0	0	1	1	1	0	1	1
M7	Latte Macchiato	1	0	0	1	1	1	1	1	1
M8	Milk Foam	0	0	0	0	0	0	1	0	1
M9	Cappuccino	1	0	0	1	1	1	1	1	1

Table 3.13: The Table of AND relations between Requirements and Main Functions



mainfunction_id	R1	R2
M1	1	0
M2	1	0
M3	1	0
M4	1	0
M5	0	0
M6	1	1
M7	1	1
M8	0	0
M9	1	1

Table 3.14: The Table of AND relations between Requirements and Basic Functions (a)

requirement_id	requirement_desc	f001	f002	f003	f004	f005	f006	f007	f008	f009	f010	f011	f012	f013	f014	f015	f016	f017	f018
R1	Temperature Safety	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R2	Pressure Safety	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3.15: The Table of AND relations between Requirements and Basic Functions (b)

f019	f020	f021	f022	f023	f024	f025	f026	f027	f028	f029	f030	f031	f032	f033	f034	f035	f036
1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Step 3: Get information of defected components to form the input column matrix of basic functions

The input column matrix of this step is used to indicate states of basic functions, whose elements 0 stand for normal state and elements 1 stand for abnormal state. The proposed remote fault handling and reconfiguration system performs secondary analysis of results from the fault diagnosis process, which means defected components are detected after the fault diagnosis phase. Moreover, a one-to-one mapping from the component level to basic function level is applied in this thesis. Therefore, an input column matrix of basic functions can be constructed by setting elements on behalf of normal basic functions to 0 while setting elements on behalf of abnormal basic functions to 1. From here, the input column matrix is denoted as $input\_1$.

Step 4: Check dependency relations of the basic function level

In this thesis, three extra basic functions f037, f038 and f039 are added after identified 36 basic functions of the IAS coffee maker. Those three extra funcions stand for the dependency relation between f003 and f027, the dependency relation between f002 and f026 and the dependency relation between f004 and f028 respectively. Before moving on to the next step, states of three extra funcitons have to be determined by extracting relation matrix of Table 3.7 and Table 3.8 and multiplied it with $input\_1$, afterwards, states of f037, f038 and f039 can be determined by elements of the result column matrix whose 0 elements refer to normal state and non-zero elements stand for abnormal state. The result column matrix of this step is denoted as $result\_1$.

Step 5: Check OR relations of the basic function level

Only one OR relation is considered for the IAS coffee machine, which is the OR relation between f038 and f039, therefore an extra function named f040 is added to represent this OR relation shown in Table 3.9. To determine the state of f040, $result\_1$ from the last step should be transformed into a new column matrix which is further used to be multiplied by the OR relation matrix by setting non-zero elements to 0 and setting zero elements to 1. Elements of the result column matrix should take same transformation as column matrix $result\_1$ before moving to the next stop. And after transformation, 0 elements refer to normal state and non-zero elements refer to abnormal state. The transformed result column matrix of this step is denoted as $result\_2$.

Step 6: Get integrated input column matrix

An integrated input column matrix combines states of basic functions and states of extra functions is formed in this step. After dependency relations and OR relations are checked, the integrated input column matrix is achieved and is denoted as $input\_2$ here. The first 36 elements of $input\_2$ here are elements of $input\_1$ and then follows elements of $result\_1$ and $result\_2$.

Step 7: Determine available sub-functions

Here comes the step to determine available sub-functions via multiplying the relation matrix of AND relations between basic functions and sub-functions shown in Table 3.10 and Table 3.11 by the integrated input column matrix $result\_2$. The result column matrix of this step is denoted as $input\_3$. Similarly, the 0 elements of $input\_3$ indicate normal state and non-zero elements indicate abnormal state.

Step 8: Determine available main functions

This step is the same process with step 7. Since there is no dependency or OR relations between main functions and sub-functions regarding the IAS coffee maker, the available main functions can be directly figured out via multiplying the relation matrix between main functions and sub-funcitons by the column matix $input\_3$ achieved in step 7. The result column matrix of this step is denoted as $result\_3$.

Step 9: Analyze Requirements

For the IAS coffee maker, only nonfunctional requirements which includes the temperature safety requirement and the pressure safety requirement are considered. After available main functions are determined, the relation matrix representing AND relations between requirements and basic functions is multiplied by column matrix $input\_2$, which will output a result column matrix named $input\_4$ that indicates affected requirements using non-zero elements. Afterwards, the relation matrix of AND relations between requirements and main functions is multiplied by matrix $input\_4$, which gives a result matrix named $result\_4$ whose non-zero elements refer to unavailable main functions that are only under requirement analysis. By adding $result\_3$ and $result\_4$, a final result matrix is achieved, whose non-zero elements give the determined unavailable main funcitons.

Under the nine steps testing, function analysis and requirement analysis of the IAS coffee maker is finished. It breaks a complex function analysis process into a few steps that fully involves all logical relations of the IAS coffee maker. Moreover, the system efficiency is improved by avoiding frequent decision making when functions are queried from the remote database.

4 Software Development

4.1 Development Environment

The project requires the following software for development:

- Eclipse with java version 1.8 and JRE 1.8
- XAMPP that integrates MySQL database and Apache Web Server

4.2 System Use Case

4.2.1 Use Case Overview

The designed software involves three systems and thirteen use cases. All use cases and their corresponding systems are listed in the Table 4.1 below.

Table 4.1: Overview of Software Prototype Use Cases

Use Case Number	Use Case Name
Remote Fault Handling and Reconfiguration System	
1	Manage Server
2	Perform Routine Maintenance
3	Pre-handle Faults
4	Handle Known Faults
5	Handle New Faults
6	Reconfigure Functions
7	Update Remote Database
Fault Diagnosis System	
8	View FDS Information
9	Determine Fault Type
10	Update Local Fault Knowledge
Simulator	
11	Simulate Coffee Producing
12	Simulate Fault Occurrence
13	Execute Reconfiguration Solution

4.2.2 Use Case Diagrams

Figure 4.1 shows the use case diagram of the remote fault handling and reconfiguration system. The left side of the diagram shows actors i.e. primary actors that trigger use cases while the right side shows actors i.e. secondary actors triggered passively by use cases. As marked by different colors, three kinds of use cases can be identified. The one in green relates to actions taken by the system adminstrator, e.g. logging into the system, checking database information, checking system processing information etc. The use case in gray is about system maintenance that allows users to add known fault locations manually to the remote system. The remaining blue ones are fault handling and function analysis processes of the remote system.

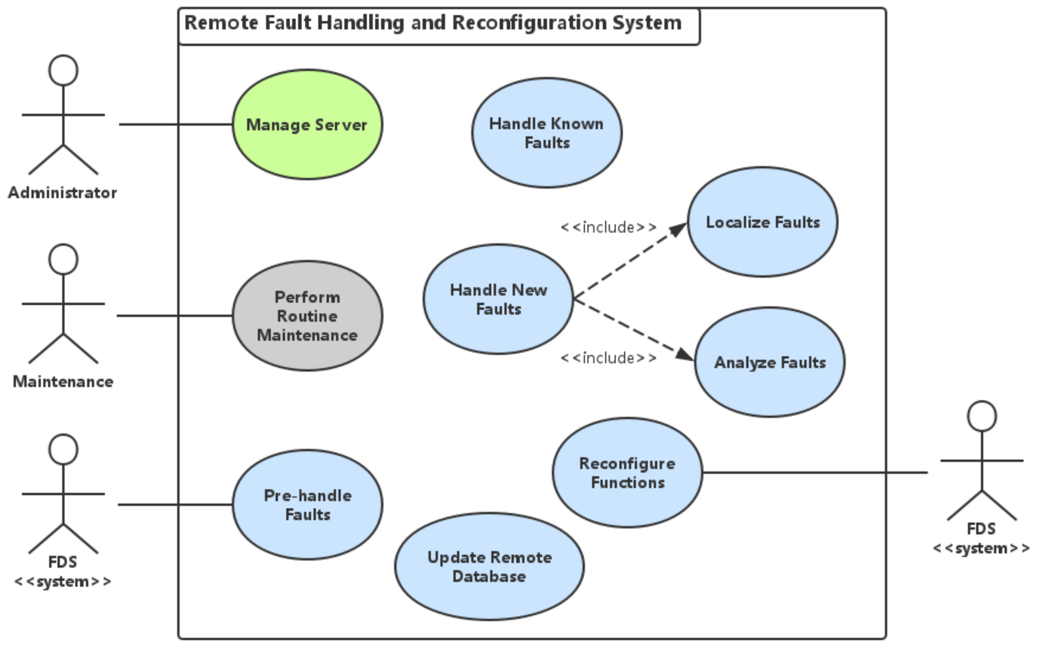


Figure 4.1: Use cases of Remote Fault Handling and Reconfiguration System

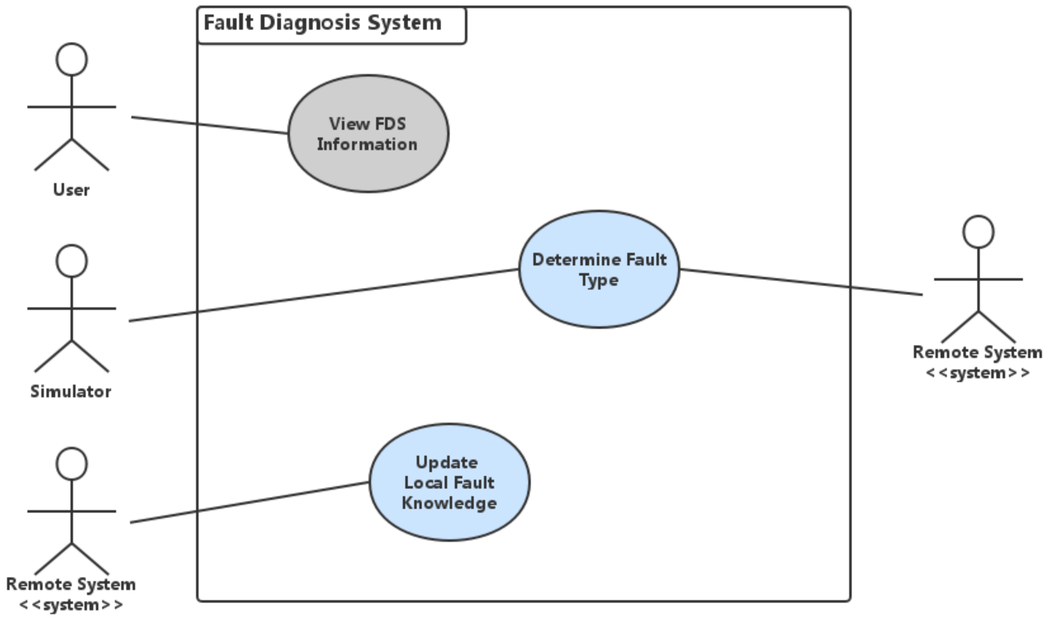


Figure 4.2: Use cases of Fault Diagnosis System

Figure 4.2 depicts uses cases of the fault diagnosis systems and involved primary and secondary actors. In this thesis, the fault diagnosis system is a simulator that simulates results of the IAS fault diagnosis system, whose processing outputs will be sent to the remote system for fault handling and function analysis. Therefore the key use case of this system is the *Determine Fault Type* use case which makes a decision if the occurred fault is known or new in the local system.

In Figure 4.3, three use cases of the simulator are shown. They fulfill tasks of simulating technical processes of the IAS coffee maker, generating faults and executing function reconfiguration results achieved by the remote system.

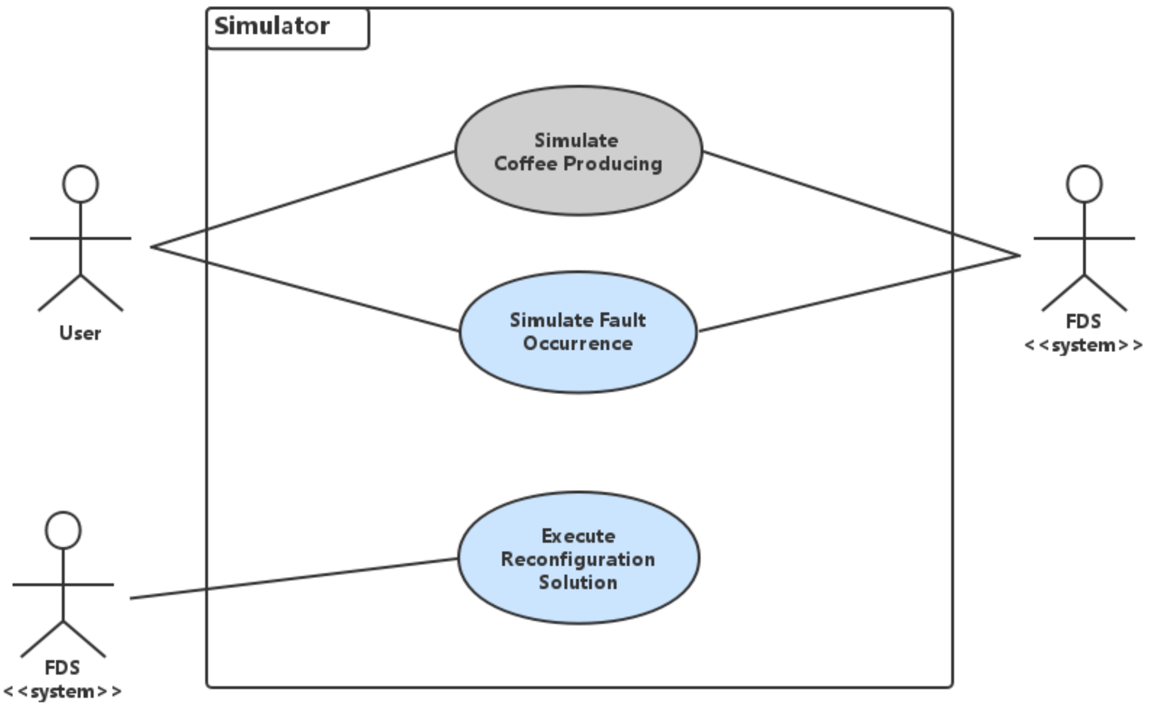

Simulator
Simulate Coffee Producing
Simulate Fault Occurrence
Execute Reconfiguration Solution
User
FDS <<system>>
FDS <<system>>

Figure 4.3: Use cases of Simulator

4.2.3 Description of Use Cases

This section gives detailed description of each use case by tables or by sequence diagrams.

4.2.3.1 Description of Use Case < Manage Server >

Table 4.2: Description of Use Case < Manage Server >

USE CASE	**Manage Server**
Goal in Context	Provide an authentication scheme and a GUI for remote system administrator to login the system, manage server information and log out in the end.

Category	primary	
External Actors	Administrator	
Trigger	User name and password are typed in and the login button is clicked.	
DESCRIPTION	**Step**	**Action**
	1	User name and password are typed in corresponding text fields.
	2	Login button is clicked.
	3	Verified the typed in user name and password.
	4	Login successfully or ask for valid input again.
	5	View server information.
	6	Log out.

4.2.3.2 Description of Use Case < Perform Routine Maintenance >

Table 4.3: Description of Use Case < Perform Routine Maintenance >

USE CASE	Perform Routine Maintenance	
Goal in Context	Maintenance staff can put fault information like defected component, symptom etc. into the remote database. A GUI should be provide to help the information typing.	
Category	primary	
Preconditions	None	
External Actors	Maintenance	
Trigger	An *update* button is clicked.	
DESCRIPTION	**Step**	**Action**
	1	Maintenance staff click the 'Maintenance' button on the remote fault handling and reconfiguration system GUI.
	2	A maintenance GUI pops up. Maintenance staff select the defected components on the GUI by clicking check boxes in front of component names.
	3	Fault processing information is shown on the remote system GUI.
	4	Maintenance staff click the 'Exit' button on the maintenance GUI if no further maintenance is required.

4.2.3.3 Sequence Diagram of Use Case < Pre-handle Faults >

The sequence diagram shown in Figure 4.4 describes interaction between four objects, which are *FDS*, *ServerPort_in*, *ServerFaultKnowledge* and *FaultUnderEvaluation*. From this diagram, we could see that the process starts from the fault diagnosis system who initiates a communication with the remote system by sending fault information after its own processing. Then the object *ServerPort_in* in the remote system accepts the connection and in coming information which is further passed to an object that manages all fault information in the remote database. After fault type (known or new) is determined by *ServerPort_in*, corresponding handling processes will be triggered by *FaultUnderEvaluation*.

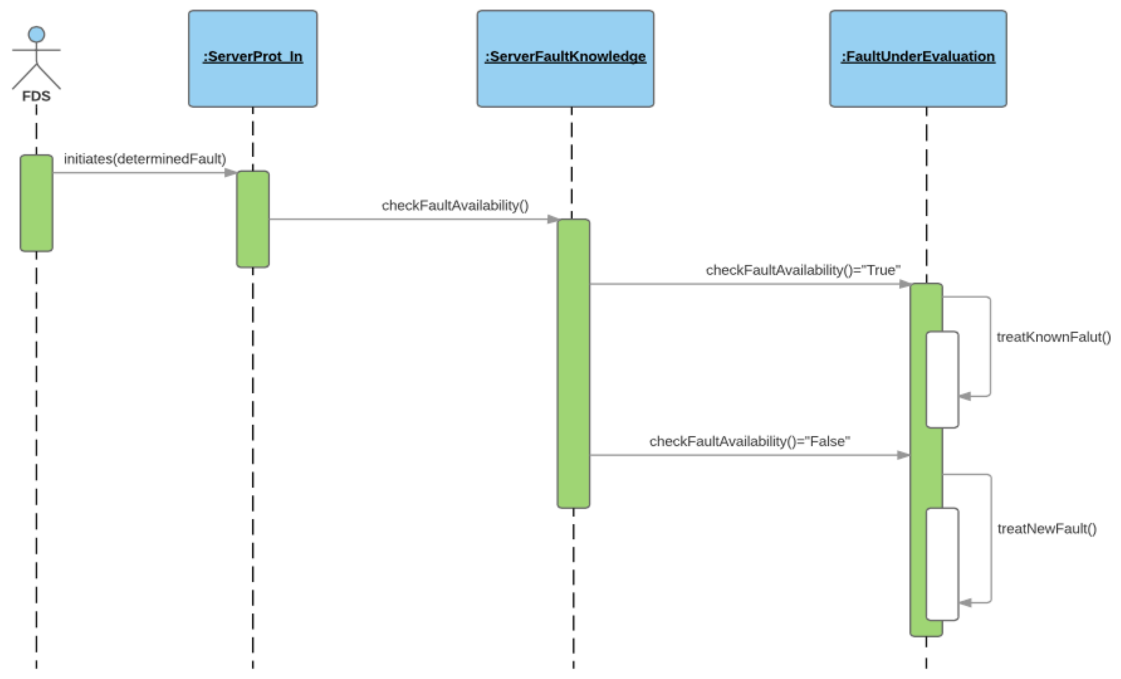


Figure 4.4: Sequence Diagram of Use Case < Pre-handle Faults >

4.2.3.4 Sequence Diagram of Use Case < Treat Known Faults >

The sequence diagram shown in Figure 4.5 describes the process of treating known fault in the remote system. Related object in this process are *PrehandleFaultInfo*, *RemoteFaultKnowledge* and *ReconfiguredFunctions*. From this diagram, we could see that the process starts from the *PrehandleFaultInfo* object who initiates a communication with the *RemoteFaultKnowledge* by sending fault information after its own processing. Then the object *ReconfiguredFunctions* receives handling results that contains available functions of the known fault in an array.

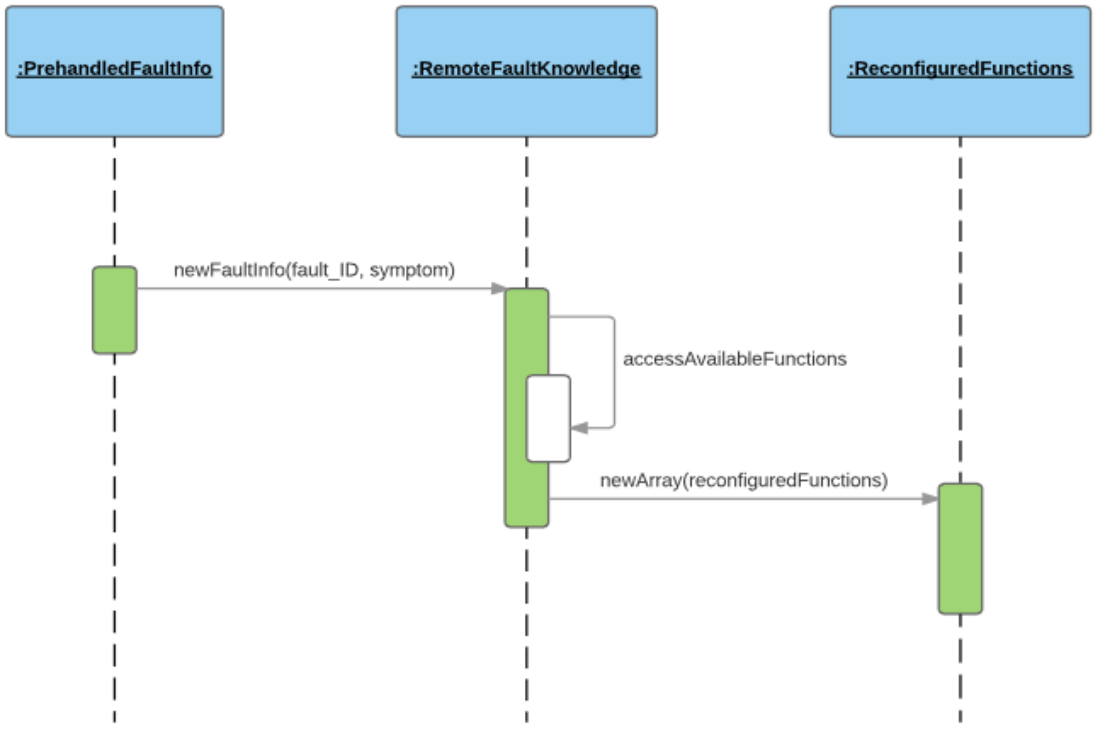


Figure 4.5: Sequence Diagram of Use Case < Treat Known Faults >

4.2.3.5 Sequence Diagram of Use Case < Treat New Faults >

The sequence diagram shown in Figure 4.6 describes the key process – treating new faults of the remote system. Five objects, which are Prehan*dleFaultInfo*, *FaultLocation*, *System_Functions, System_Requirements* and *ReconfiguredFunctions* are involved. The process starts from the object *ReconfiguredFunctions* who determines the occurred fault is new in the remote system. Then the object *FaultLocation* localizes the fault and sends the result with defected components' IDs to the object *System_Functions* who will do function analysis. After anlysis process is done, the available functions is checked in requirement analysis process, which will get rid of functions that should not be available because they do not meet some requirements. Afterwards, the final decided available functions can be processed in the *ReconfiguredFunctions* object.

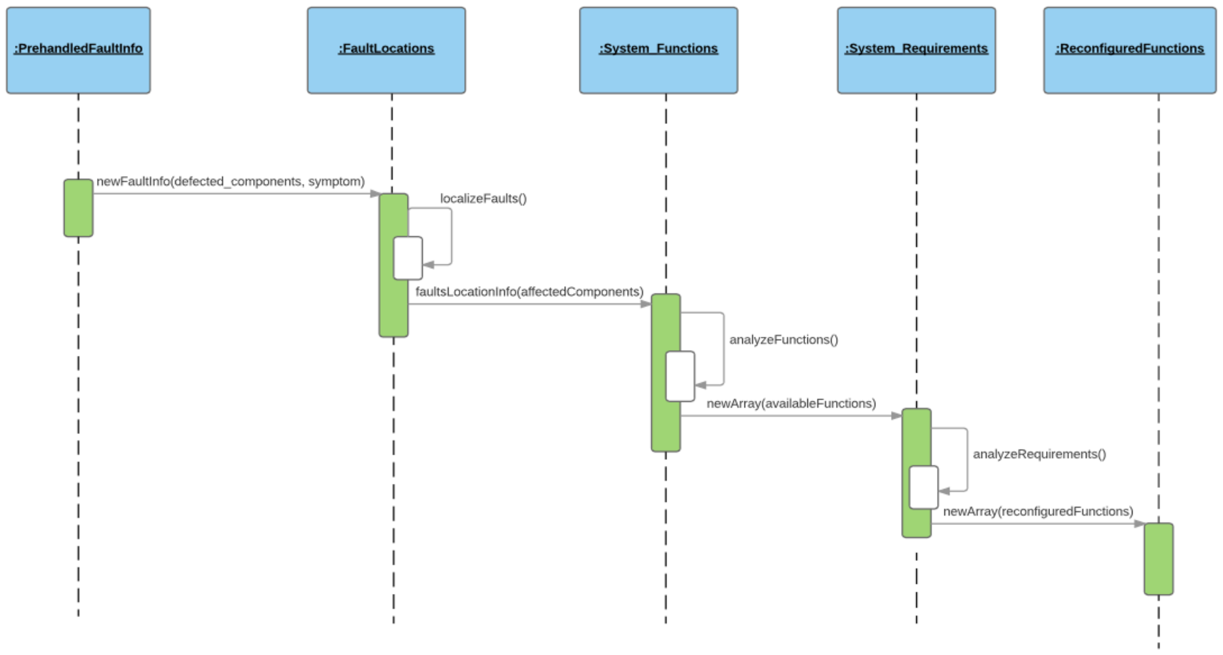


Figure 4.6: Sequence Diagram of Use Case < Treat New Faults >

4.2.3.6 Description of Use Case < Reconfigure Functions >

Table 4.4: Description of Use Case < Reconfigure Functions >

USE CASE	Reconfigure Functions	
Preconditions	• Fault signal is received • Fault signal is evaluated	
Success End Condition	Reconfiguration solution is given.	
External Actors	None	
DESCRIPTION	**Step**	**Action**
	1	If received fault is known, query the fault knowledge records
	2	If received fault is unknown, evaluate requirements vs. components relationship
	2.1	evaluate system vs. components relationship
	2.2	evaluate historical data vs. components relationship
	2.3	Get reconfiguration result

4.2.3.7 Description of Use Case < Update Remote Database >

Table 4.5: Description of Use Case < Update Remote Database >

USE CASE	Update Remote Database	
Preconditions	• The received fault signal is unknown remotely • Reconfiguration result has been given	
Success End Condition	Remote database are updated with new fault_ID, symptom, effected components and solutions	
External Actors	None	
DESCRIPTION	**Step**	**Action**
	1	Update fault_ID
	2	Update symptom
	3	Update effected components
	4	Update solutions

4.2.3.8 Description of Use Case < View FDS Information >

Table 4.6: Description of Use Case < View FDS Information >

USE CASE	View FDS Information	
Preconditions	• Data from ATS and remote sever are recorded in the FDS database	
Success End Condition	Information is shown correctly	
External Actors	User	
DESCRIPTION	**Step**	**Action**
	1	user clicks a specific field
	2	Corresponding information is shown on the GUI
	3	repeat this for every click query

4.2.3.9 Sequence Diagram of Use Case < Determine Fault Type >

The sequence diagram shown in Figure 4.7 shows the process of determining fault in the local fault diagnosis system. There are four related objects in this process, which are *FaultOccurrenceSignal*, *FDSFaultKnowledge*, *DeterminedFaultInfo* and *ServerPort_In*. The process starts from the *FaultOccurrenceSignal* object who triggers the *FDSFaultKnowledge* object to check the local database if the occurred fault has been recorded before. If the fault already recorded in the local system, a negative signal is sent to the next object, otherwise a positive signal is sent. The *DeterminedFaultInfo* objects mainly constructs fault information with known or null fault ID to the incoming port of the remote system.

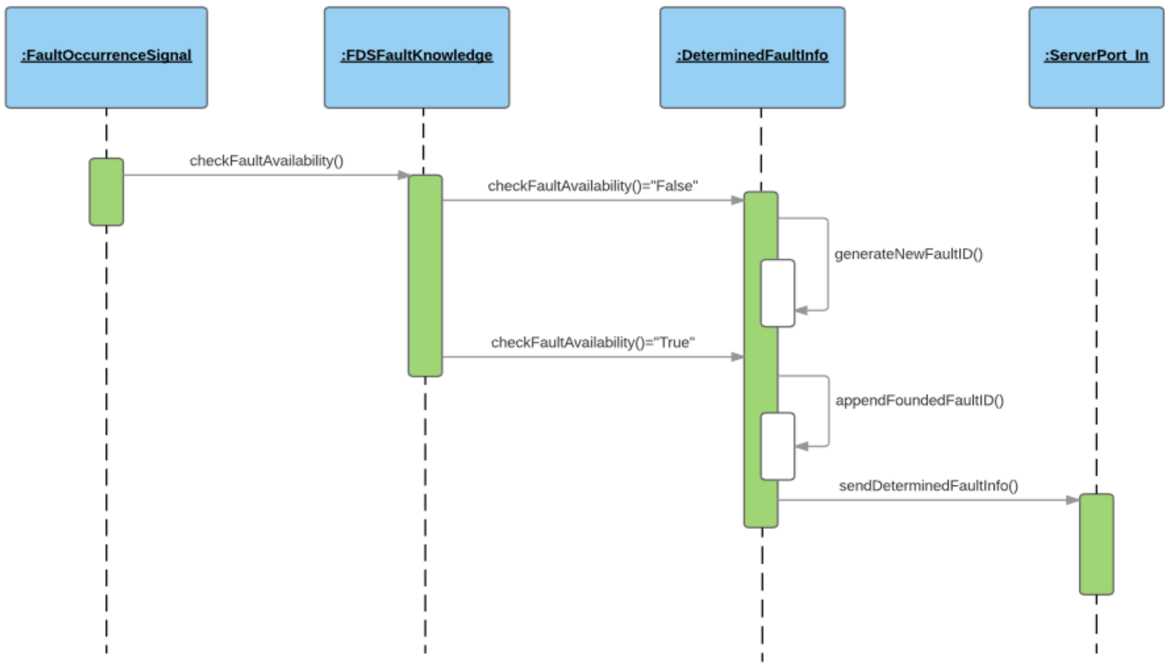


Figure 4.7: Sequence Diagram of Use Case < Determine Fault Type >

4.2.3.10 Description of Use Case < Simulate Coffee Producing >

Table 4.7:Description of Use Case < Simulate Coffee Producing >

USE CASE	Simulate Coffee Producing	
Goal in Context	Provide a GUI for coffee maker users make their orders by clicking specified buttons.	
Category	primary	
Preconditions	None	
Success End Condition	A specified button standing for one coffee type is single clicked and Prepare Coffee, Serve Coffee and Update Local Databases uses cases are triggered.	
Failed End Condition	No button is clicked or more than one button are clicked at the same time.	
External Actors	Coffee maker users	
Trigger	A specified button standing for one coffee type is single clicked.	
DESCRIPTION	**Step**	**Action**
	1	Press specified coffee button
	2	Generate corresponding technical data and insert them with the current date in the local database
	3	Show selected coffee information on the simulator GUI

4.2.3.11 Description of Use Case < Execute Reconfiguration Solution >

Table 4.8: Description of Use Case < Execute Reconfiguration Solution >

USE CASE	Get Reconfiguration Solution	
Preconditions	Remote server has finished function reconfiguration	
Success End Condition	A signal with reconfiguration solution is passed in Interface	
External Actors	Remote System	
DESCRIPTION	**Step**	**Action**
	1	Listen to income port
	2	Get the solution signal
	3	Disable coffee buttons that are unavailable and enable coffee buttons that are available according to the reconfiguration solution

4.3 Software Design

4.3.1 Design Decisions

The IAS coffee maker as an automation system could have faults occurred during its operation period. Due to complexity of handling faults, the embedded micro controller of automation systems is not suitable to bear the burden of analyzing fault information and reconfiguring system functions. Therefore, to fulfill the great need of enhancing the availability of non-affected functions when problems occur, a remote fault handlin and reconfiguration system should be designed to handle the complicated analyzing and reconfiguration tasks. Meanwhile local automation systems only need to collect fault-related information and communicate with the remote system to get function reconfiguration solutions. These all will not be possible without efficient, reliable and easily implementable system models that localize faults and figure out available and unavailable functions given the occurred faults. The system models applied in the software prototype development have been described in section 2.3.

Furthermore, four application situations, which have been specified in the system requirements specification, are considered in this software prototype. The four situations are:

1) Occurred fault is both known locally and remotely.

2) Occurred fault is both known locally and remotely.

3) Occurred fault is both new locally and remotely.

4) Occurred fault is known locally, but new remotely, which is about system maintenance.

To realize the aforesaid functionalities, software components are distinguished as below:

- **Remote Fault Knowledge** component
- **System Rules** component
- **Server Processing Program** component
- **Maintenance** component
- **Local Fault Knowledge** component
- **System State** component
- **FDS Processing Program** component
- **Fault Generator** component
- **ATS Simulator** component

4.3.2 System Architecture

In Figure 4.8, an overview of the system architecture is presented. The components listed above are also included in this diagram. They belong to three different systems: Remote Fault Handling and Reconfiguration System, Fault Diagnosis System and Simulator. As shown in the diagram, the remote fault handling and reconfiguration system consists of four components, three interfaces defining communication between components and two interfaces allowing interaction with external environment. The diagnosis system contains three components, two component interaction interfaces and three interfaces connecting outer systems. The simulator mainly includes two components simulating fault occurrence and coffee producing process respectively. Detailed description of each component will discussed in next section.

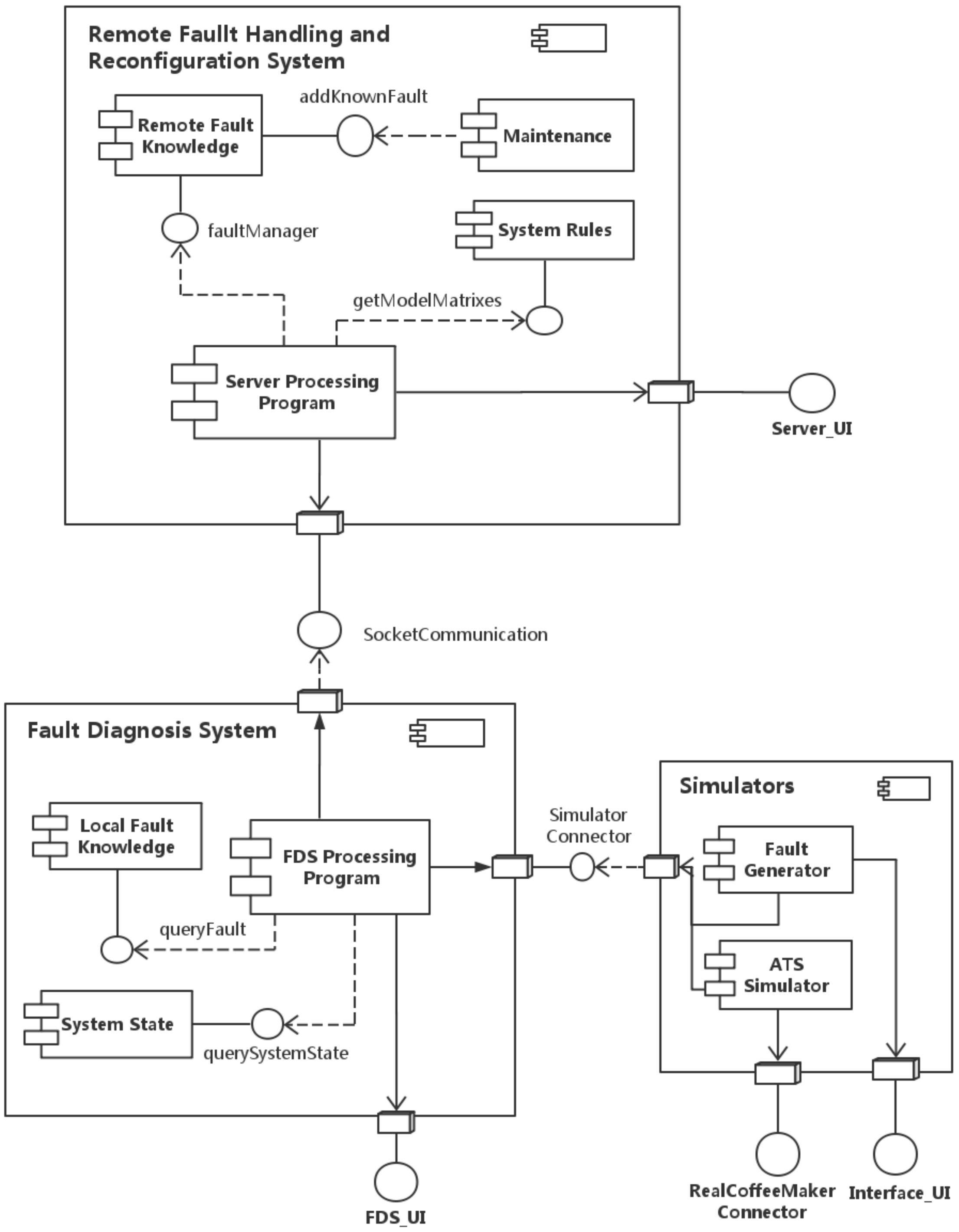


Figure 4.8: An Overview of The System Architecture

4.3.3 Software Components

4.3.3.1 Remote Fault Knowledge

The main function of this component is the implementation of operations towards the remote fault knowledge database, which stores faults information and automation system states in the remote system. The overview of remote fault knowledge is show in figure 4.9:

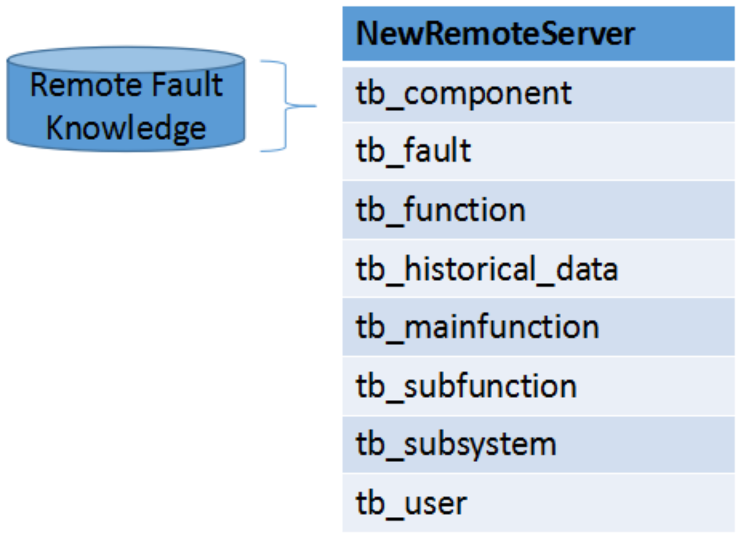


Figure 4.9: Remote Fault Knowledge Database Overview

Tables and their functions are listed in the following:

- Table tb_component is used to store system states extracted from the FDS database. It includes the following columns: component_id (unique identifier of each component), component_name, type (trivial or key), state, update_date.
- Table tb_fault is used to record fault information and their corresponding reconfiguration solutions. Columns of this table are: number (auto_increment int type, indicating the number of faults in the remote server), fault_id (unique identifier of each fault), symptom_id, symptom_desc, reconfig_command, available_functions, fault_location, fault_effect, fault_reason, device_location, occurred_date.
- Table tb_function stores the basic function information, which includes function_id, function_desc, function_status and update_date.
- Table tb_mainfunction stores the main function information, which includes mainfunction_id, mainfunction_desc, mainfunction_status and update_date.
- Table tb_subfunction records the information of sub functions. Its columns are: subfunction_id, subfunction_desc, subfunction_status and update_date.

- Table tb_subsystem stores the information of sub systems. Its columns are: subsystem_id, subsystem_name, state and update_date.
- Table tb_user is used to store the information of registered server users. Successful registered user information are recorded in this table and users can log into the server with registered username and password. The columns of this table are: id (auto_increment int type), username, password and email.

4.3.3.2 System Rules

This component implements the system models, which analyzes an occurred fault from three views: requirement view, function view and component view. It includes a database to help with function analysis and function reconfiguration applying the requirement-function-component modelling approach. The overview of the system model database is show in figure 4.10:

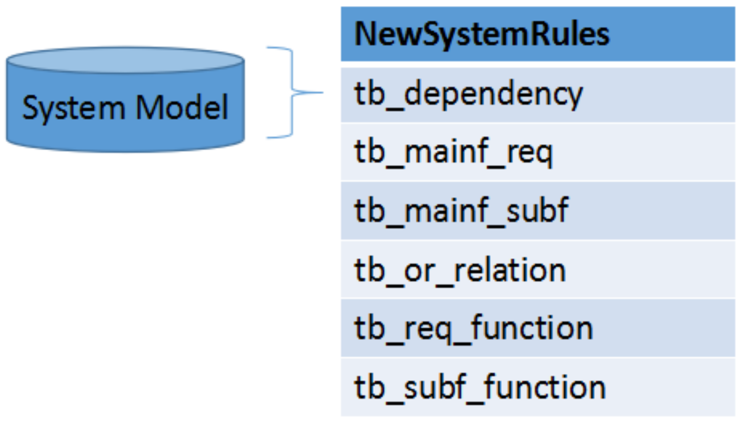


Figure 4.10: System Model Database Overview

The implementation is divided into two parts: faults localization and function analysis.

For fault localization, it involves three steps:

1) Analyze the received symptom. In our designed system, a symptom is identified by symptom id, symptom description and symptom array. The symptom array is an attribute that consists of ten values, which are Micro Controller Voltage, Micro Controller Temperature, Motor R.P.M, Motor Temperature, Flow Speed, Milk Temperature, Brewing Temperature, Brewing Pressure, Kettle Temperature, Kettle Pressure. These ten values are collected by sensors in the coffee maker. They indicate the symptom of a problem.
2) By checking if each value is in the setting scale, a rough fault localization would be determined, i.e. the defected subsystem would be locked.
3) Analyze the historical data trend to target the precise localization of the fault, i.e. the defected component id is determined. This is done by a java code to query recent 10 historical data from FDS database and plot them in the same line chart with 10 typical historical data stored in the server database.

For function analysis, it involves the following steps:

1) Check if the defected component is key component or not. If yes, disable the whole automation system, i.e. all simulated coffee buttons in our project. If not, follow the steps listed as below.
2) According to defected component and the following map of physical relationship, determine the affected scope of fault.
3) Mapping the component model to function model and perform function analysis algorithm described in section 3.2.4.
4) After the affected main functions are figured out, the available functions are determined. Then check requirements related to those available functions. If no other functions are affected because of requirements restrictions, we can move on to reconfiguration step.

4.3.3.3 Server Processing Program

The **server processing program** component coordinates tasks of **remote fault knowledge** component and **system rules** component, through which, it handles different kinds of faults and make reconfiguration solutions. Main function blocks of this component are listed as below:

- Front-end processing of the server, including fault information display, processing states display and client connection information display.
- Implementing server starting, accepting client connection and closing.
- Implementing different handling processes for different received faults, including calling corresponding methods in **remote fault knowledge** component and **system rules** component and cooperating those methods with its own methods.
- Implementing reconfiguration process and sending the result back to requested client, i.e. coffee maker in our project.

4.3.3.4 Maintenance

The **maintenance** component is a functional block of the remote server. Its main function is to allow system users add known fault to the server manually. In this context, coffee maker maintenance staff would find a defected component of the machine. This component provides a user interface for the user to input fault information in specified input fields, whose content will be analyzed by **system rules** component and **server processing program** component and finally the remote fault knowledge database will be updated with the input fault information and corresponding reconfiguration solution.

4.3.3.5 Local Fault Knowdege

The main function of this component is the implementation of operations towards the local fault knowledge stored in the local FDS database, whose overview is shown in Figure 4.11:

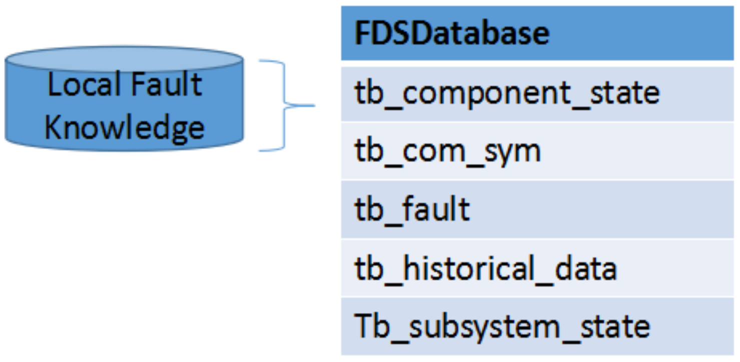


Figure 4.11: Local Fault Knowledge Database Overview

Tables and their functions are listed in the following:

- Table tb_com_sym is used to determine the symptom of defected components. In our designed system, fault occurrence is simulated by user clicking a specific number of components in the System Interface GUI, which will generate an array of defected component id. The defected component ids will then be queried in this table, and related symptoms and symptom information will de find out for further sending to the server. This table includes the following columns: symptom_id, symptom_desc, symptom_array and related_components.

- Table tb_fault is used to store fault information locally. It is part of the tb_fault in the remote database since the server handles more than one machine's problem. Columns of this table are: number (auto_increment int type, indicating the number of faults in the local FDS database), fault_id, symptom_id, fault_reason, device_id, device_location and occurred_date.

- Table tb_historical_data is used to store a certain amount of historical data of the connected coffee maker. Each row of data is updated into the database when a coffee button is clicked in the ATS simulator GUI. Historical data that are older than 5 days are deleted automatically, which will save space of the database. The recorded data i.e. the table columns are: device_id, mc_vol (micro controller voltage), mc_temp (micro controller temperature), motor_rpm, motor_temp, flow_speed, milk_temp, brew_temp, brew_pressure, kettle_temp, kettle_pressure and date_time.

4.3.3.6 System State

The main function of this component is the implementation of operations towards the system state database in the FDS, which stores automation system states. Tables and their functions are listed in the following:

- Table tb_component_state is used to record component state information. Component states can be queried by the server during the fault handling process. Columns of this table are: component_id, component_name, state and update_date.

- Table tb_subsystem_state is used to record states of subsystems, which can be queried by the server during the fault handling process. Columns of this table are: subsystem_id, subsystem_name, state and update_date.

4.3.3.7 FDS Processing Program

This component reports fault information to the remote server, as well as collects local automation system's state information and historical data. It receives fault occurrence information from the **Fault Generator** component and then determines the received fault type as known or new fault by knowledge of the local fault knowledge database. Afterwards, it actively connects to the remote server to send the fault information. When the server finishes fault handling process and makes reconfiguration solutions, the result will be sent back to the **fault diagnosis system**. Main function blocks of this component are listed as below:

- Front-end processing of the fault diagnosis system, including displaying local fault information, displaying component and subsystem states, and displaying connection information with the server.
- Implementing connection and communication with the server.
- Implementing communication with the simulator system.

4.3.3.8 Fault Generator

Due to the infeasibility of making real mechanical faults, a **fault generator** component is designed to simulate fault occurrence. Selection of faults in the Fault Generator GUI can be:

1) One or more than one component is defected;

2) One or more than one component plus one or more than one element are defected;

3) A subsystem is defected.

4.3.3.9 Coffee Maker Simulator

Main function of this component is to simulate coffee production, which will update technical process data into the table *tb_historical_data* of the local fault knowledge database. Also, this component shows reconfiguration result by disabling specific coffee buttons in the simulator GUI. As an optional function, this component implements interaction between the coffee maker simulator and the real IAS coffee maker. The implementation of this function depends on the project progress and lab conditions. The protocol CAN-BUS is used to fulfill the communication. Main purpose of this function is to allow users making coffee orders by pressing coffee buttons shown on the simulator GUI.

4.3.4 Test Specification

4.3.4.1 Test Requirements

For the software test of Remote Fault Handling and Reconfiguration System, requirements listed below must be followed:

- Two PCs connected to IAS LAN, one is installed with the Fault Diagnosis System and simulators, while the other is installed with the Remote Fault Handling and Reconfiguration System.
- The two PCs should be previously installed with Java VM and XAMPP Version 5.6.15-1, which includes MySQL Database, Apache Web Server and ProFTPD.
- Local and remote databases should be manually created first, which can be done easily by importing related sql files to corresponding PCs.

The whole test must be operated by more than one person, who is able to follow instructions shown in the GUI.

4.3.4.2 Test Criteria

The test will be considered successfully if the criteria below are met:

1) For coffee maker simulator, coffee producing information is shown in the user interface and technical historical data is update in the local database. In addition, this simulator disable specified coffee buttons according to reconfiguration instructions given by the remote server.
2) For the fault generator simulator, more than one components or elements can be selected to be as defected; more than one combination of components and elements can be selected as defected; one subsystem can be selected as defected.
3) For the Fault Diagnosis system, local database information should be shown on the GUI. FDS should determine the fault type (new or known basing on its own fault knowledge database) and construct related JSON data to send fault information and operation information to the remote system.
4) For the Remote Fault Handling and Reconfiguration System, four criteria should be fulfilled:

- Message from client i.e. local FDS is able to be reach the remote system successfully.
- Remote database information should be shown on the GUI.
- Fault handling process work flow should be shown on the GUI.
- Handling result i.e. reconfiguration solutions should be sent back to corresponding FDS.

4.3.4.3 Test Cases

Coffee Producing Simulation

This test case verifies the process of producing technical data of IAS coffee maker. It is used to produce reasonable technical data including micro controller voltage, micro controller temperature, motor rotation speed, motor temperature, flow speed, milk temperature, brewing temperature, brewing pressure, kettle temperature, kettle pressure.

Table 4.9: Description of Coffee Producing Simulation Test Case

Test case	Coffee Producing Simulation
To be proven	The simulation of coffee producing process is valid. Methods to be proven here: *updateATSHistoricalData(), updateComponentState(), updateSubsystemState().*
Initial condition	The FDS program starts correctly; XAMPP starts correctly and local database is built.
Necessary inputs	Specified coffee button pressing actions.
Expected results	Current task, remained resources and device_id is shown on the simulator GUI; *tb_historical_data, tb_component_state* and *tb_subsystem_state* in the local database are updated.

Fault Occurrence Simulation

This test case is used to generated a simulated fault of the coffee maker simulator. Cooresponding abnormal technical data occurs when some components or elements are selected in the check boxes. Detailed description is shown in Table 4.10.

Table 4.10: Description of Fault Occurrence Simulation Test Case

Test case	Fault Occurrence Simulation
To be proven	Fault occurrence information is created. Methods to be proven here: *createComponentCheckbox(), createElementCheckbox(), createSubsystemCheckbox().*
Initial condition	The FDS program starts correctly; XAMPP starts correctly and local database is built.
Necessary inputs	Specified check boxes are selected.
Expected results	Symptom data is generated and send to FDS.

Determining Fault Type

This test case is used to verify if type of the generated fault can be determined correctly in the fault diagnosis system resorting to a local database. Detailed description is shown in Table 4.11.

Table 4.11: Description of Determining Fault Type Test Case

Test case	Determining Fault Type
To be proven	Fault type is determined according to local fault knowledge and JSON data needed to be sent to remote system is created. Methods to be proven

	here: *queryFaultID(String), querySymptom(String), createToServerDataSet(String, String, String).*
Initial condition	The FDS program starts correctly; XAMPP starts correctly and local database is built.
Necessary inputs	Specified check boxes in the fault generator GUI are selected.
Expected results	Fault type is determined and shown in the FDS GUI; JSON data which includes *device_id, fault_id, symptom_id, current_task, symptom_array, resource_array* and *historical_data* is created.

Handling Fault Known Locally and Remotely

This test case aims to test the process of handling a locally and remotely known fault. Detailed description is shown in Table 4.12.

Table 4.12: Description of Handling Fault Known Locally and Remotely Test Case

Test case	Handling Fault Known Locally and Remotely
To be proven	The remote system is able to handle fault that is recorded locally and remotely. Methods to be proven are: *receiveFaultInfo(JSONObject), getSymptomValues(JSONObject), prehandleFault(JSONObject), reconfigureFunctions(String, JSONObject, int), updateRelatedDB().*
Initial condition	The Remote Fault Handling and Reconfiguration program starts correctly; XAMPP starts correctly and local database is built.
Necessary inputs	JSON Data is received successfully.
Expected results	Correct reconfiguration solutions are given and sent back to corresponding FDS.

Handling Fault New Locally but Known Remotely

This test case is used to verify the process of handling faults that are new in the local system but known in the remote system. Detailed description is shown in Table 4.13.

Table 4.13: Description of Handling Fault New Locally but Known Remotely Test Case

Test case	Handling Fault New Locally and Known Remotely
To be proven	The remote system is able to handle fault that is new locally but known remotely. Methods to be proven are: *receiveFaultInfo(JSONObject), getSymptomValues(JSONObject), prehandleFault(JSONObject), reconfigureFunctions(String, JSONObject, int), updateRelatedDB().*

Initial condition	The Remote Fault Handling and Reconfiguration program starts correctly; XAMPP starts correctly and local database is built.
Necessary inputs	JSON Data is received successfully.
Expected results	Correct reconfiguration solutions are given and sent back to corresponding FDS.

Handling Fault New Locally and Remotely

Handling faults that are new both in local system and remote system is the key process in the remote system. This test case gives the detailed process information on the remote system GUI. Detailed description is shown in Table 4.14.

Table 4.14: Description of Handling Fault New Locally and Remotely Test Case

Test case	Handling Fault New Locally and Remotely
To be proven	The remote system is able to handle fault that is new locally and remotely. Methods to be proven are: *receiveFaultInfo(JSONObject), getSymptomValues(JSONObject), prehandleFault(JSONObject), findDefectedComponents(double[]), findDefectedSubsystem(double[]), treatNewFault(JSONObject), reconfigureFunctions(String, JSONObject, int), updateRelatedDB(),getHistoricalData(JSONObject), sendResults().*
Initial condition	The Remote Fault Handling and Reconfiguration program starts correctly; XAMPP starts correctly and local database is built.
Necessary inputs	JSON Data is received successfully.
Expected results	Correct reconfiguration solutions are given and sent back to corresponding FDS.

Maintenance

Maintenance test case is used verify if known fault locations can be added manually to the remote system. Afterwards, function analysis process is triggered basing on the given fault locations.

Table 4.15: Description of Maintenance Test Case

Test case	Maintenance
To be proven	The remote system is able to allow system users adding fault information and is able to handle those faults. Methods to be proven are: *handleMaintenance(), reconfigureFunctions(String, JSONObject, int), updateRelatedDB(),getHistoricalData(JSONObject), sendResults().*

I0nitial condition	The Remote Fault Handling and Reconfiguration program starts correctly; XAMPP starts correctly and local database is built.
Necessary inputs	Defected components information and symptom data are input in the maintenance GUI.
Expected results	Correct reconfiguration solutions are given and sent back to corresponding FDS.

4.4 Software Implementation

4.4.1 Implemented Classes

The whole software prototype consists of three parts: the remote fault handling and reconfiguration system, the fault diagnosis system and a simulator of coffee maker and fault generator. So classes are implemented in three packages. In the following, class diagram of each package are shown from Figure 4.12 to Figure 4.14.

Implemented classes of the remote fault handling and reconfiguration system can be divided into three types: GUI-related classes, process-related classes and database-related classes. The GUI-related classes include:

- RSMainUI
- RSConnectionUI
- RSTabUI
- MaintenanceUI

The process-related classes are:

- CAPTCHALabel
- Maintenance
- RelationMatrix
- FaultHandler
- Register
- Login

The database-related class is DBHelper.

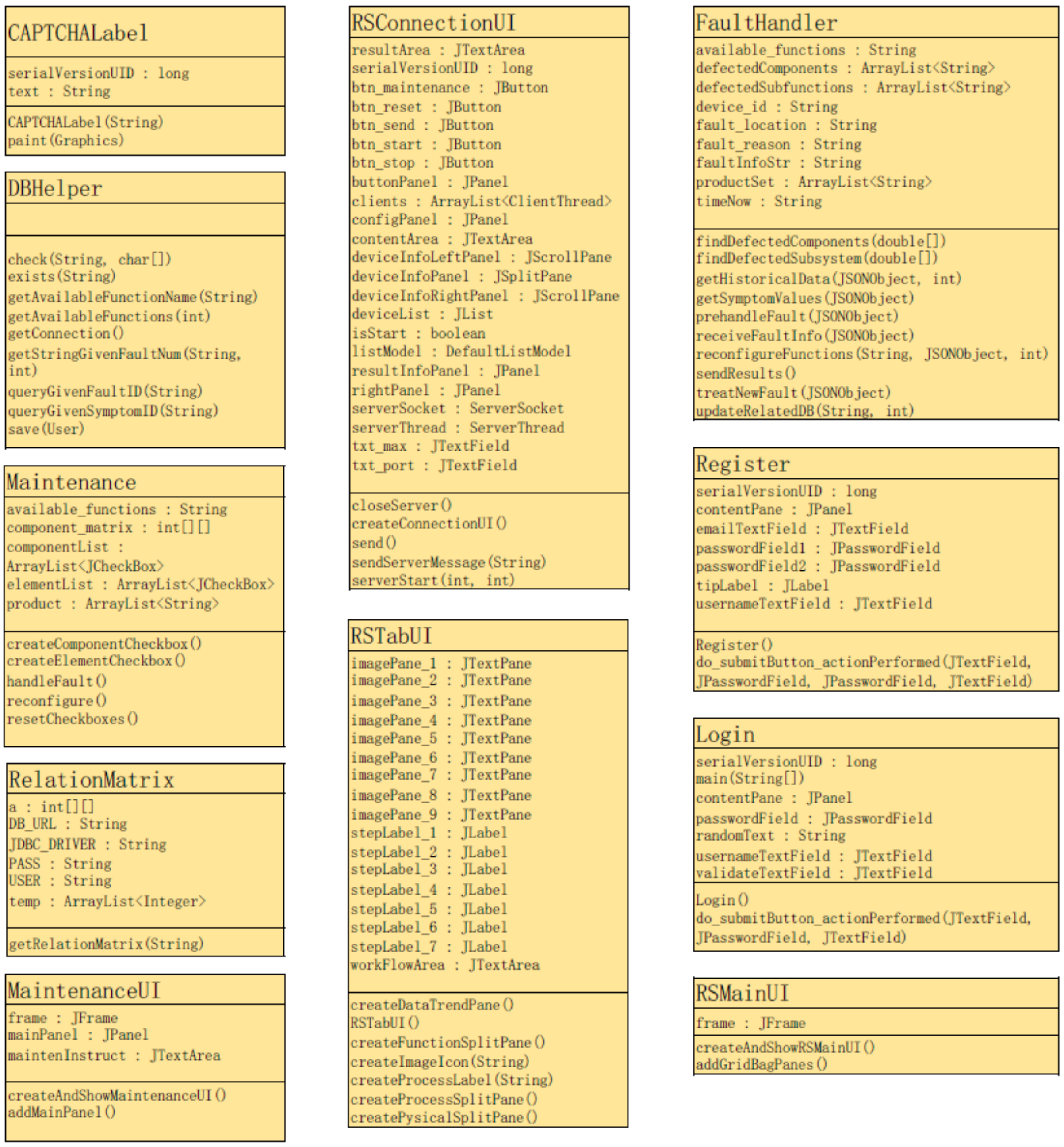


Figure 4.12: Class Diagram of the Remote Fault Handling and Reconfiguration System

Implemented classes of the fault diagnosis system can be divided into three types as well: GUI-related classes, process-related classes and database-related classes.

The GUI-related classes include:

- FDSMainUI
- FDSTabUI
- FDSConnectionUI

The process-related classes are:

- ATS
- JSONDataSet
- FDSTabUIActions
- MessageObservable

The database-related classes are FDSDBConfig and FDSDBHelper.

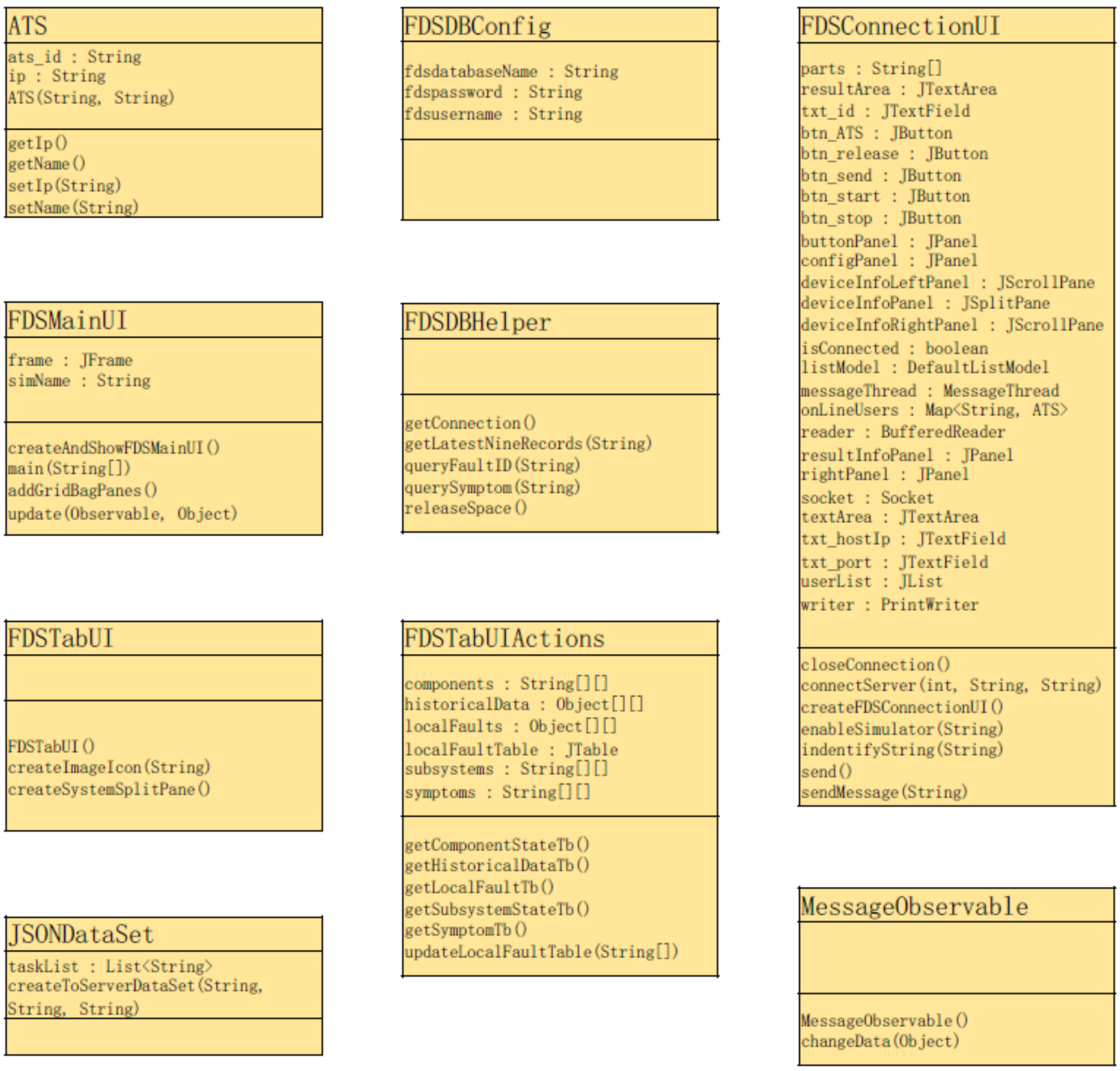


Figure 4.13: Class Diagram of the Fault Diagnosis System

Figure 4.14 shows the implemented classes of the simulator whose classes can be divided into three types as well: GUI-related classes, process-related classes and database-related classes.

The GUI-related classes include:

- CoffeeMaker
- FaultGenerator
- SimulatorUI

The process-related classes are:

- ButtonActionListener
- CoffeeMaker
- FaultGenerator

The database-related class is UpdateDatabase.

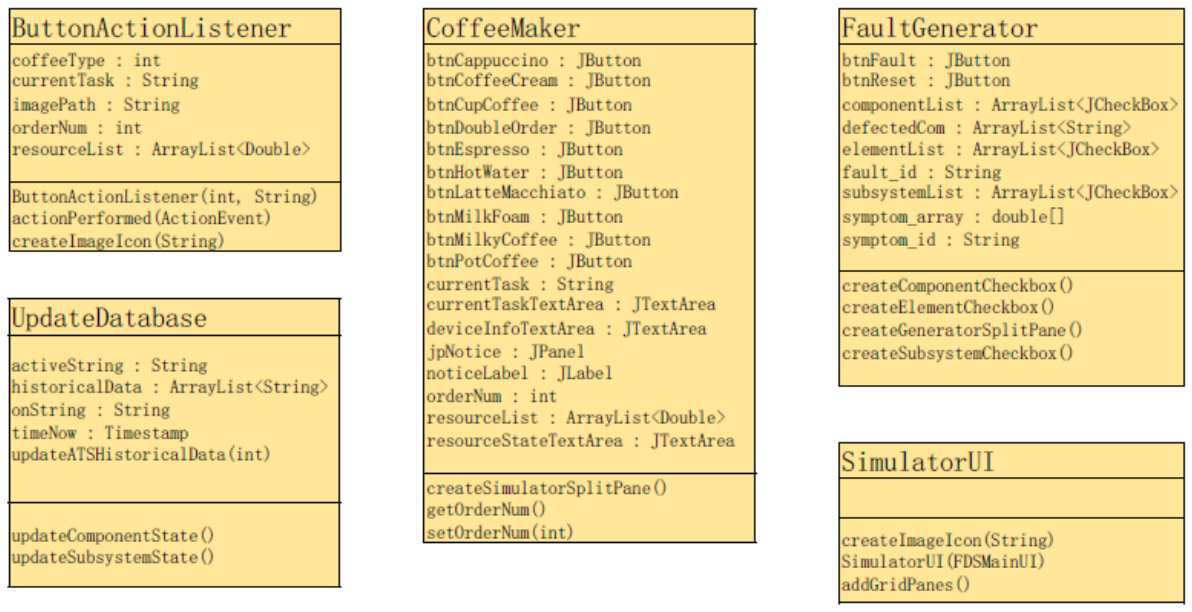


Figure 4.14: Class Diagram of the Simulator

4.4.2 Test Procedures and Results

4.4.2.1 Coffee Producing Simulation

Procedure of Testing:

Steps for the test case *coffee producing simulation* are:

1) Start the Fault Diagnosis System (FDS) software;

2) Click each coffee button in the ATS simulator GUI one by one;

3) Check if the information shown on the GUI is reasonable;

4) Check if corresponding technical data has been updated in the local database.

Test Results:

The test is successful basing on the results listed below:

1) Device information including device id and device location is shown in the Coffee Maker Simulator GUI;
2) Resource state information including remained expresso weight, remained coffee beans weight and remained coffee granules weight are shown the in GUI;
3) Current Task is shown by the button clicked and by the text in the text field;
4) Once a coffee button is clicked, after two seconds, image field of the GUI notices users that the ordered coffee is ready;
5) Technical data including micro controller voltage, micro controller temperature, motor rotating speed, motor temperature, flow speed, milk temperature, brewer temperature, brewer pressure, kettle temperature and kettle pressure are updated in the local database.

4.4.2.2 Fault Occurrence Simulation

Procedure of Testing:

Steps for testing are:

1) Start the Fault Diagnosis System (FDS) software;
2) Select components, elements or subsystems that are assumed to be defected in the simulator GUI. It should be noticed that one or more than one component can be combined with one or more than one element in the selection, but subsystems should be selected alone;
3) Click the ‘Send Fault Info to FDS’ button to confirm the fault occurrence;
4) Click the ‘Reset Checkboxes’ button to reset all check boxes to the original state.

Test Results:

The test is successful: check boxes with coffee maker components, elements and subsystems are available for selection; resetting check boxes are successful; all coffee buttons are disabled once the fault occurrence is confirmed.

4.4.2.3 Determining Fault Type

Procedure of Testing:

Steps for testing are:

1) Start the Fault Diagnosis System (FDS) software;
2) Perform fault occurrence simulation;
3) Check the text area named ‘Determined Fault Type’ in the FDS GUI.

Test Results:

The test is successful: the type of the simulated fault is determined. If it is already recorded in the local database, the fault id is given the text area. Otherwise, the fault id is set to null.

4.4.2.4 Handling Fault Known Locally and Remotely

Procedure of Testing:

1) Start the Remote Fault Handling and Reconfiguration System software;
2) Register or log into the system with user name and password;
3) Start the server by clicking the 'Start' button in the system GUI;
4) Connect the Fault Diagnosis System to the remote system by clicking the 'Connect' button in the FDS GUI;
5) According to the local database information, if defected components are selected from C001, C002, C003, C004, C020, E001, E002, E004, the fault is known in the local database. Choose the aforesaid component or element ids in any combination then click the 'Send Fault Info to FDS' button in the simulator GUI;
6) After the determined fault id is shown on the FDS GUI, click the 'Send Fault Info to Server' button;
7) Wait until the remote system finishes steps shown the remote system GUI and once the result information is attached to the 'Fault Handling Results and Reconfig Info' area, click the 'Send Result to Client' button;
8) Check if the FDS GUI received the message from the remote system;
9) Check simulator GUI if the available coffee buttons are enabled.

Test Results:

The test is successful basing on the results listed below:

1) Handling process information is shown as expected;
2) Handling results and reconfiguration solutions are given in corresponding text area and is sent to the FDS successfully;
3) Corresponding coffee buttons are enabled.

4.4.2.5 Handling Fault New Locally but Known Remotely

Procedure of Testing:

Steps for testing are:

1) Start the Remote Fault Handling and Reconfiguration System software;

2) Register or login with username and password;

3) Start the server by clicking the ‘Start’ button in the system GUI;

4) Connect the Fault Diagnosis System to the remote system by clicking the ‘Connect’ button in the FDS GUI;

5) According to the local database information, if defected components are selected from C006, C014, C015, C016, E007, E012, SS01, SS02 or SS03 the fault is known in the remote database. Choose the aforesaid component or element ids in any combination, or choose one aforesaid subsystem, then click the ‘Send Fault Info to FDS’ button in the simulator GUI;

6) After the determined fault id is shown on the FDS GUI, click the ‘Send Fault Info to Server’ button;

7) Wait until the remote system finishes steps shown the remote system GUI and once the result information is attached to the ‘Fault Handling Results and Reconfig Info’ area, click the ‘Send Result to Client’ button;

8) Check if the FDS GUI received the message from the remote system;

9) Check if the local database is updated with the new fault information.

10) Check simulator GUI if the available coffee buttons are enabled.

Test Results:

The test is successful basing on the results listed below:

1) Handling process information is shown as expected;

2) Handling results and reconfiguration solutions are given in corresponding text area and is sent to the FDS successfully;

3) Local database is updated with new fault information;

4) Corresponding coffee buttons are enabled.

4.4.2.6 Handling Fault New Locally and Remotely

Procedure of Testing:

Steps for testing are:

1) Start the Remote Fault Handling and Reconfiguration System software;

2) Register or login with username and password;

3) Start the server by clicking the 'Start' button in the system GUI;
4) Connect the Fault Diagnosis System to the remote system by clicking the 'Connect' button in the FDS GUI;
5) According to the local database information, if defected components, elements or subsystems are selected from sets that not mentions in Section 4 and Section 5, it is a fault new locally and remotely. Choose corresponding item ids then click the 'Send Fault Info to FDS' button in the simulator GUI;
6) After the determined fault id is shown on the FDS GUI, click the 'Send Fault Info to Server' button;
7) Wait until the remote system finishes steps shown the remote system GUI and once the result information is attached to the 'Fault Handling Results and Reconfig Info' area, click the 'Send Result to Client' button;
8) Check if the component states, subsystem states and function states are updated in the remote system GUI;
9) Check if the remote fault knowledge is updated;
10) Check if the FDS GUI received the message from the remote system;
11) Check if the local database is updated with the new fault information.
12) Check simulator GUI if the available coffee buttons are enabled.

Test Results:

The test is successful basing on the results listed below:

1) Handling process information is shown as expected;
2) the component states, subsystem states and function states are updated in the remote system GUI;
3) The remote fault knowledge is updated;
4) Handling results and reconfiguration solutions are given in corresponding text area and is sent to the FDS successfully;
5) Local database is updated with new fault information;
6) Corresponding coffee buttons are enabled.

4.4.2.7 Maintenance

Procedure of Testing:

Steps for testing are:

1) Start the Remote Fault Handling and Reconfiguration System software;
2) Register or login with username and password;
3) Start the server by clicking the ‘Start’ button in the remote system GUI;
4) Click the ‘Maintenance’ button in the remote system GUI;
5) Select any combination of components and elements in the pop up window;
6) Click ‘Information Confirmed’ button to trigger fault handling process, or click ‘Exit Maintenance’ button to exit the maintenance process;
7) Check process information in the remote system GUI.

Test Results:

The test is successful basing on the results listed below:

1) Handling process information is shown as expected;
2) Handling results and reconfiguration solutions are given in corresponding text area.

5 Installation and User Manual

5.1 Installation

This chapter describes installation instructions for the developed software prototype. To install the software prototype in a personal computer, the following software requirements must be met:

- Java Runtime Environment 8
- MySQL server (XAMPP)

5.1.1 Installation of Java Runtime Environment

To run the developed software prototype, Java Runtime Environment 7 or above is required to be installed in the software prototype running computer. The Java Runtime Environment can be download from http://www.java.com/de/download/, where also includes the installation instruction for setting up the environment. Besides, if further development of this software prototype is required, Eclipse the integrated development environment (IDE) for Java should be installed and used.

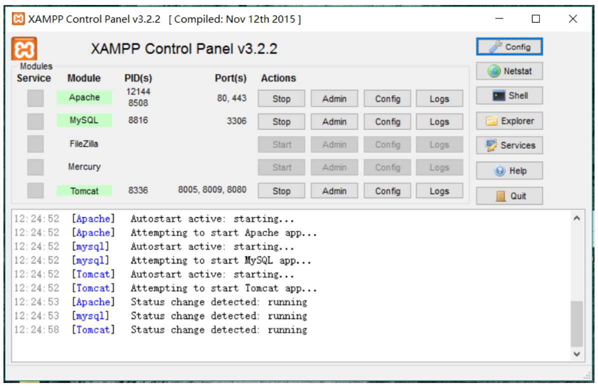
5.1.2 Installation of XAMPP

To access and maintain databases for the software prototype, the tool named XAMPP is resorted. XAMPP integrates MariaDB, PHP and Perl. The version used in this software prototype is XAMPP v3.2.2 and only MySQL Database has been used in this project. Since it is not a web based application, no web server is required here. To get download source and installation instructions for XAMPP, the link https://www.apachefriends.org/index.html is recommended.

5.2 Start-up

5.2.1 Launching of Databases

First, start XAMPP, then the XAMPP Control Panel will pop up. Click the 'start' button follows MySQL to start the database server, which is shown in Figure 5.1 below:

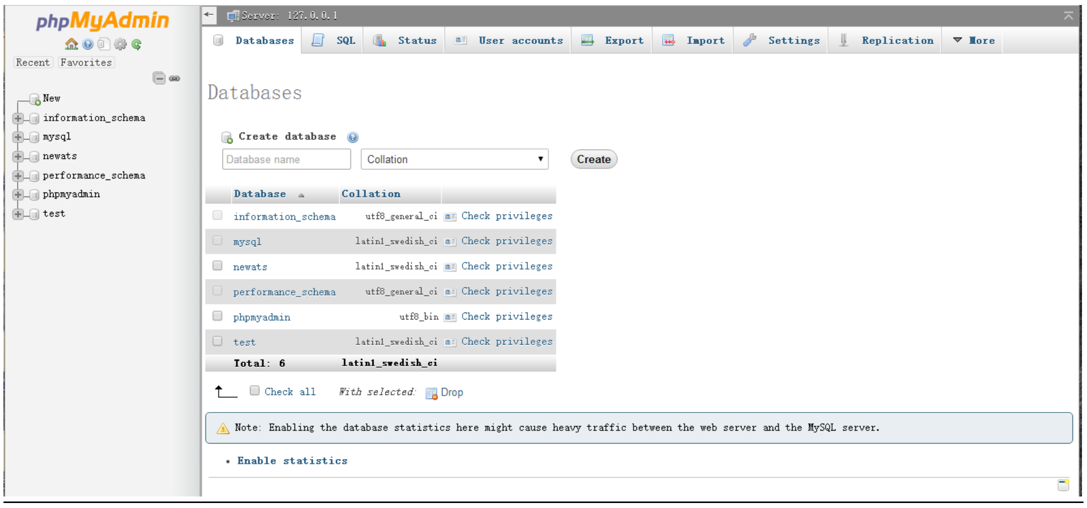

XAMPP Control Panel v3.2.2 [Compiled: Nov 12th 2015]
XAMPP Control Panel v3.2.2
Modules
Config
Netstat
Shell
Explorer
Services
Help
Quit

Service	Module	PID(s)	Port(s)	Actions			
	Apache	12144 8508	80, 443	Stop	Admin	Config	Logs
	MySQL	8816	3306	Stop	Admin	Config	Logs
	FileZilla			Start	Admin	Config	Logs
	Mercury			Start	Admin	Config	Logs
	Tomcat	8336	8005, 8009, 8080	Stop	Admin	Config	Logs

```
12:24:52  [Apache]   Autostart active: starting...
12:24:52  [Apache]   Attempting to start Apache app...
12:24:52  [mysql]    Autostart active: starting...
12:24:52  [mysql]    Attempting to start MySQL app...
12:24:52  [Tomcat]   Autostart active: starting...
12:24:52  [Tomcat]   Attempting to start Tomcat app...
12:24:53  [Apache]   Status change detected: running
12:24:53  [mysql]    Status change detected: running
12:24:58  [Tomcat]   Status change detected: running
```

Figure 5.1: XAMPP Control Panel

After starting MySQL successfully, click the 'Admin' button to open the web administration page of MySQL database, which is shown in Figure 5.2:



Databases

Create database

Database name | Collation | Create

Database	Collation	
information_schema	utf8_general_ci	Check privileges
mysql	latin1_swedish_ci	Check privileges
newats	latin1_swedish_ci	Check privileges
performance_schema	utf8_general_ci	Check privileges
phpmyadmin	utf8_bin	Check privileges
test	latin1_swedish_ci	Check privileges
Total: 6	latin1_swedish_ci	

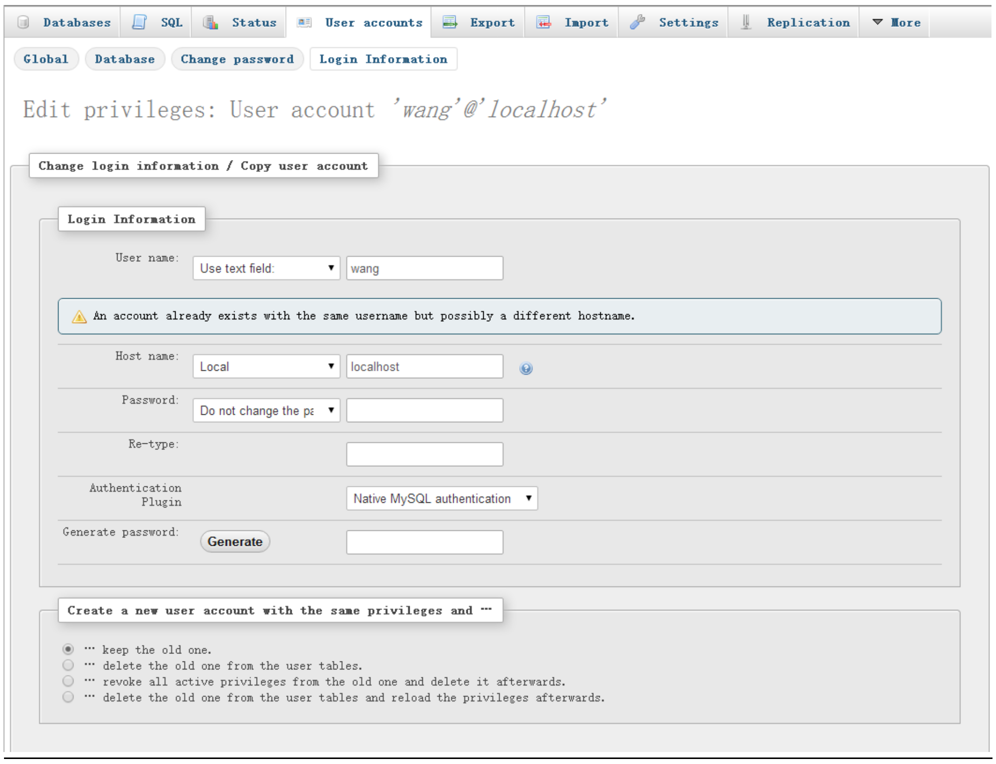
Check all *With selected:* Drop

Note: Enabling the database statistics here might cause heavy traffic between the web server and the MySQL server.

- Enable statistics

Figure 5.2: Administration Web Page of MySQL

Before launching the required databases, a new user must be created in the MySQL server. It can be done by clicking 'User accounts' in the main menu of the administration web page of MySQL shown above. Then click 'new' button to create a new user. The creation settings are shown as Figure 5.3.



Edit privileges: User account *'wang'@'localhost'*

Change login information / Copy user account

Login Information

User name: Use text field: wang

An account already exists with the same username but possibly a different hostname.

Host name: Local localhost

Password: Do not change the pa

Re-type:

Authentication Plugin: Native MySQL authentication

Generate password: Generate

Create a new user account with the same privileges and …

- ● … keep the old one.
- ○ … delete the old one from the user tables.
- ○ … revoke all active privileges from the old one and delete it afterwards.
- ○ … delete the old one from the user tables and reload the privileges afterwards.

Figure 5.3: Database New User Settings

After the new user is created, three databases must be imported into the MySQL server and authorized full access and modification privileges to the new created user. All needed SQL files can be extracted from "sourcecode-extract" from the folder "mt-2782Wang" of the sadapro drive on the computer. In folder "sourcecode-extract", there is a sub-folder named "sqlFiles", under which three files are presented:

- FDSDatabase.sql
- NewRemoteServer.sql
- NewSystemRules.sql

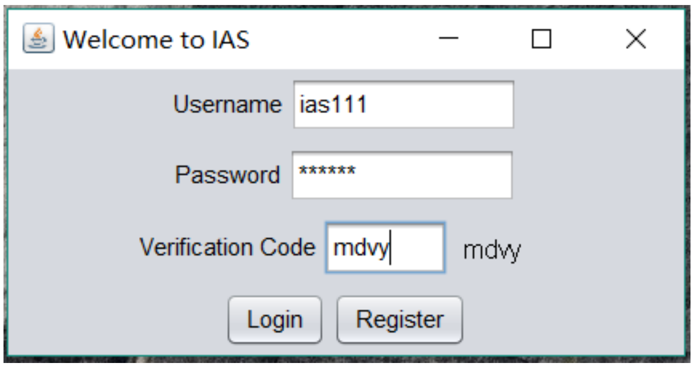
These three SQL files can be imported into the database server by clicking 'Import' button in the main menu of the web page. And corresponding tables of each databases will be generated in the server after successful importing. It must be marked here that before importing, three databases with same name of the SQL files must be created in the database server first, after which, corresponding SQL files should be imported into databases with the same name respectively.

5.2.2 Launching the Runnable JAR files of the Software Prototype

Two runnable jar files extracted from the Eclipse, which are under the direction "/mt-2782Wang/upmt-data/sw-doc/implementation/sourcecode-extract". These two runnable jar files are:

- RemoteSystem.jar
- LocalSystem.jar

To start the software of remote fault handling and reconfiguration system, double clicking the RemoteSystem.jar file. A login dialog shown in Figure 5.4 will pop up and user name 'ias111', password 'ias111' can be used to login to the system. Or new users can be registered by clicking the 'Register' button.


Welcome to IAS

Username: ias111

Password: ******

Verification Code: mdvy mdvy

Figure 5.4: Login Dialog

To start the simulator of IAS coffee maker and the simulator of the fault diagnosis system, the LocalSystem.jar file should be double clicked. And two user interfaces will pop up.

Three interfaces of the software prototype will be introduce in the next section.

5.3 Control Elements

Control elements of the software prototype are three graphical user interfaces programmed in Java Swing.

5.3.1 Remote System Graphical User Interface

The remote fault handling and reconfiguration system is the core system of this thesis. Figure 5.5 shows the graphical user interface of this system.

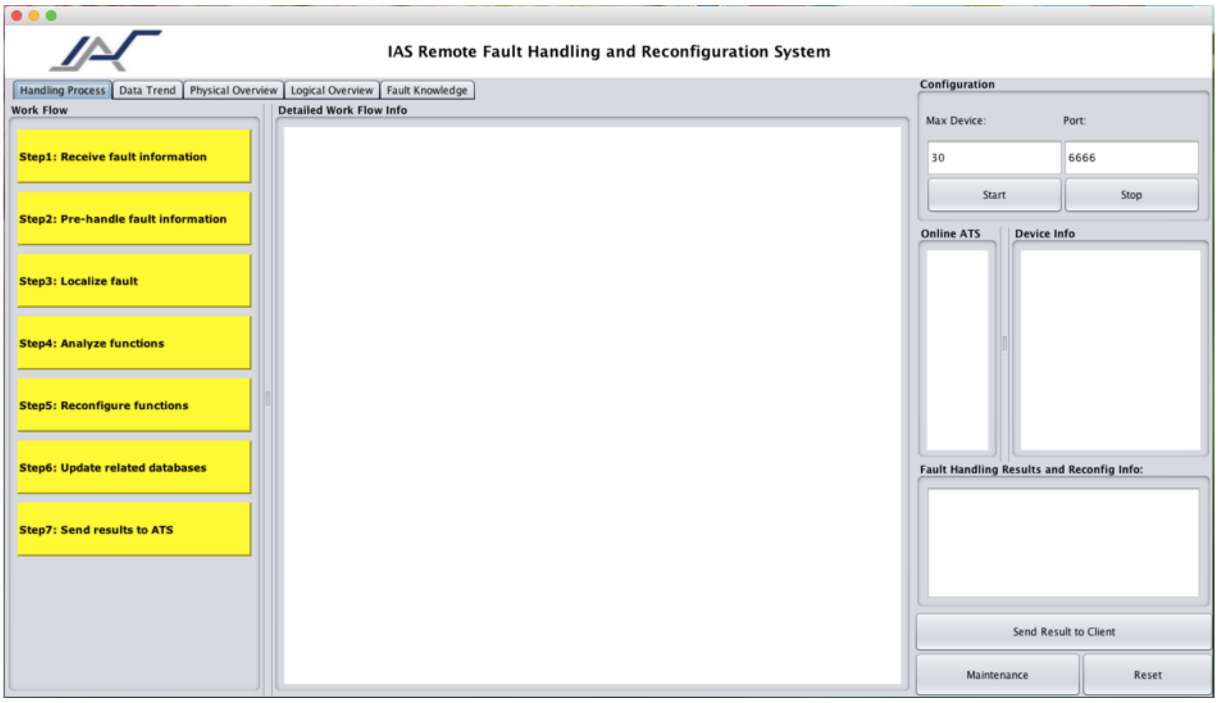


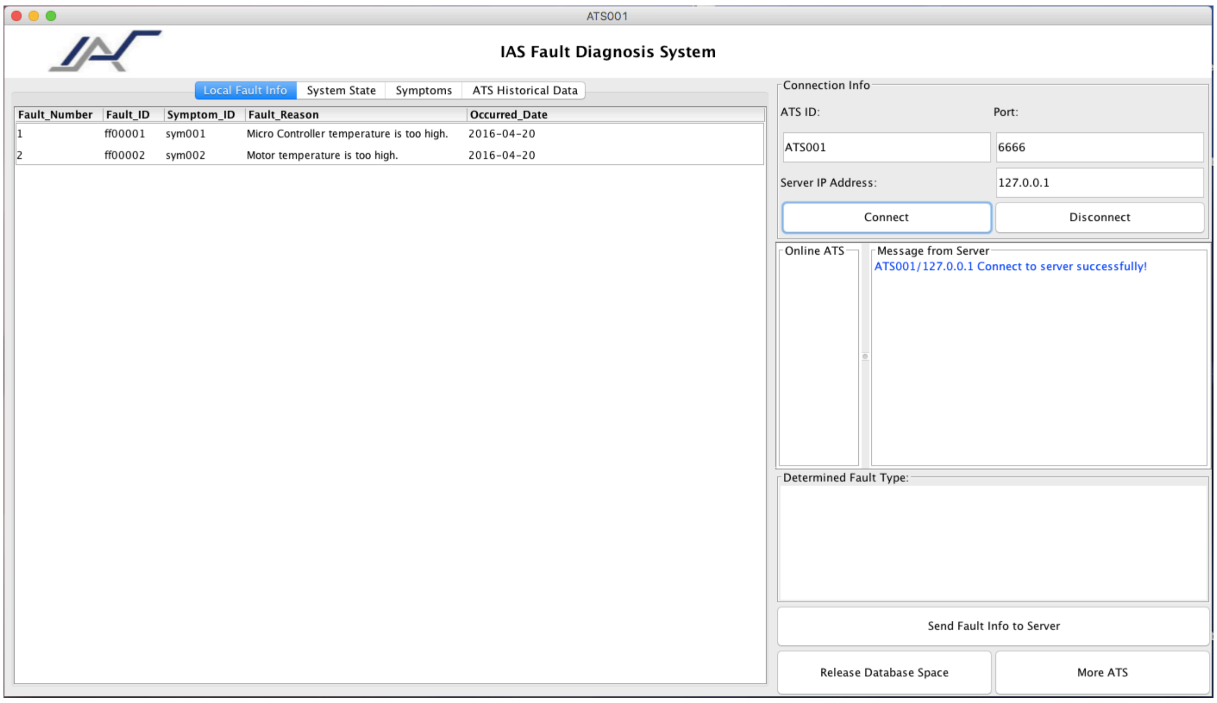
Figure 5.5: GUI of the Remote Fault Handling and Reconfiguration System

The tab area on the left side contain five tabs. The 'Handling Process' tab shows seven key steps of fault handling and reconfiguration. The yellow step labels will turn green when they are triggered and detailed process information of each step will be presented on the 'Detailed Work Flow Info' area. The 'Data Trend' tab area shows data comparison line charts of nine kinds of technical data: micro controller temperature, motor rotating rate, motor temperature, flow speed, milk temperature, brewing temperature, brewing pressure, kettle temperature and kettle pressure. Those line charts compare ten recent data with ten typical historical data. The 'Physical Overview' tab area presents components' state information and sub-system's state information while the 'Logic Overview' tab area shows states of main functions, sub-functions and basic functions. The 'Fault Knowledge' tab area extracts data from the table named tb_fault in the 'NewRemoteServer' database. It shows fault information such as fault ID, fault reason, fault location, available functions etc.

This area on the right side shows the communication information between the remote system and local systems, like the maximum number of the connecting devices, connecting port, which devices are connecting, what messages have been received. And the ‘Remote Handling Results and ‘Reconfig Info’ area shows the handling results of each case.

5.3.2 Fault Diagnosis System Graphical User Interface

The fault diagnosis system developed in this project were a simulator used to manage local fault information and determine if the occurred faults are new or known according to its own database. The graphical user interface is shown in Figure 5.6.



Fault_Number	Fault_ID	Symptom_ID	Fault_Reason	Occurred_Date
1	ff00001	sym001	Micro Controller temperature is too high.	2016-04-20
2	ff00002	sym002	Motor temperature is too high.	2016-04-20

ATS ID: ATS001

Port: 6666

Server IP Address: 127.0.0.1

Message from Server

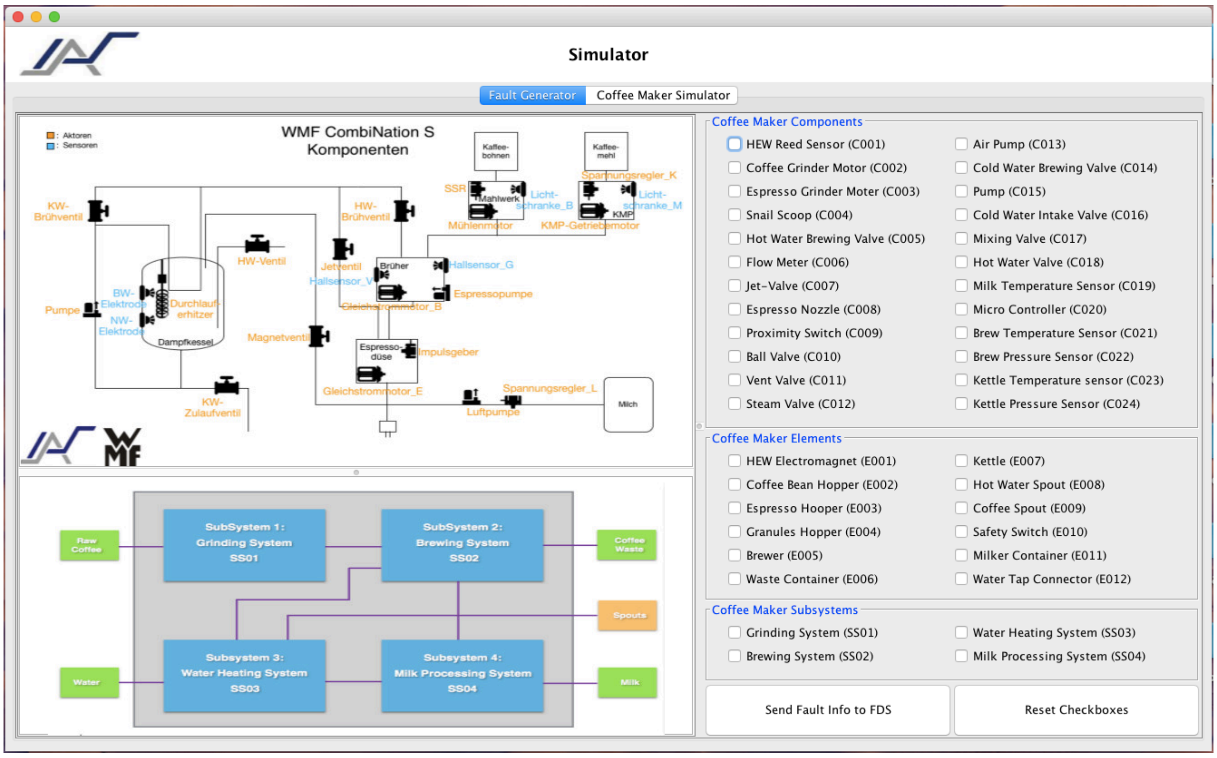
ATS001/127.0.0.1 Connect to server successfully!

Figure 5.6: GUI of Fault Diagnosis System

The tab area of this GUI on the left side shows similar with the tab area of the remote system GUI. The difference is that FDS GUI shows information that comes from the automation system assigned to its management while remote system GUI shows information from many devices that are connected to it.

5.3.3 Simulator Graphical User Interface

The simulator GUI contains two tab areas, one is used to generate faults while the other is used to simulate coffee producing. The layout of each tab is shown in Figure 5.7 and Figure 5.8.



Coffee Maker Components

☐ HEW Reed Sensor (C001)	☐ Air Pump (C013)
☐ Coffee Grinder Motor (C002)	☐ Cold Water Brewing Valve (C014)
☐ Espresso Grinder Moter (C003)	☐ Pump (C015)
☐ Snail Scoop (C004)	☐ Cold Water Intake Valve (C016)
☐ Hot Water Brewing Valve (C005)	☐ Mixing Valve (C017)
☐ Flow Meter (C006)	☐ Hot Water Valve (C018)
☐ Jet-Valve (C007)	☐ Milk Temperature Sensor (C019)
☐ Espresso Nozzle (C008)	☐ Micro Controller (C020)
☐ Proximity Switch (C009)	☐ Brew Temperature Sensor (C021)
☐ Ball Valve (C010)	☐ Brew Pressure Sensor (C022)
☐ Vent Valve (C011)	☐ Kettle Temperature sensor (C023)
☐ Steam Valve (C012)	☐ Kettle Pressure Sensor (C024)

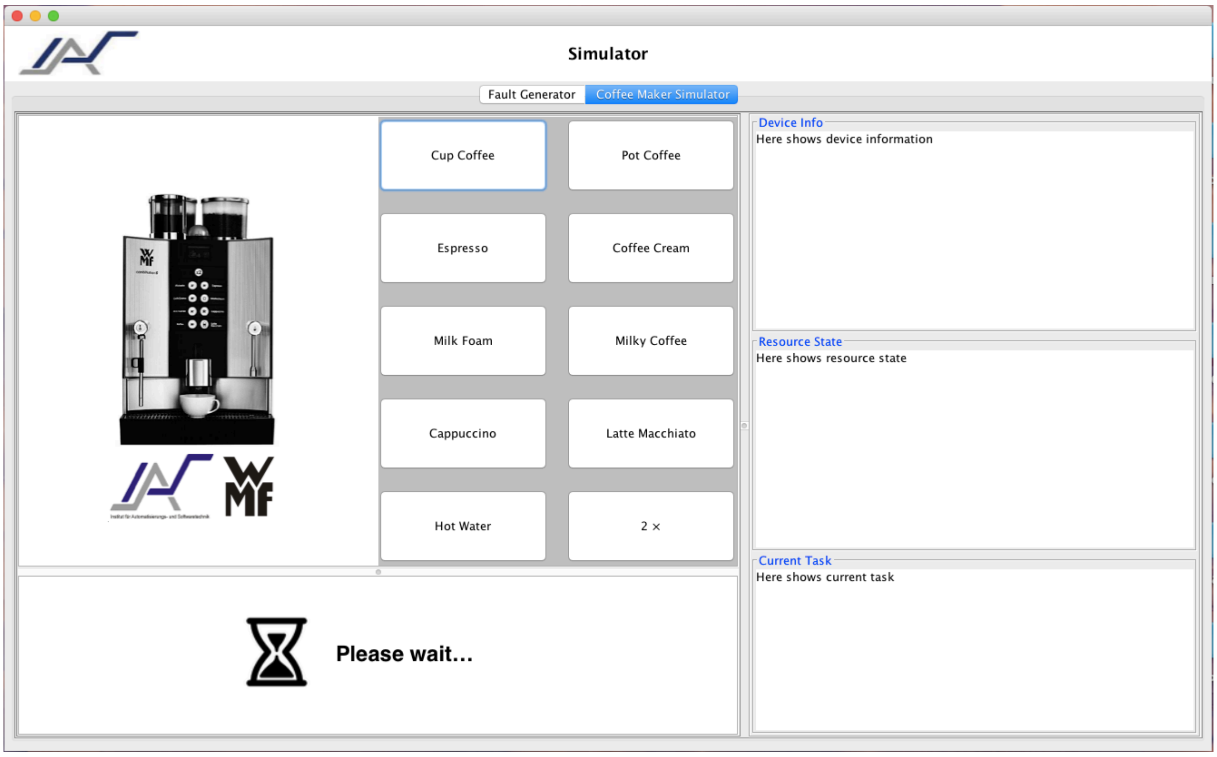
Coffee Maker Elements

☐ HEW Electromagnet (E001)	☐ Kettle (E007)
☐ Coffee Bean Hopper (E002)	☐ Hot Water Spout (E008)
☐ Espresso Hooper (E003)	☐ Coffee Spout (E009)
☐ Granules Hopper (E004)	☐ Safety Switch (E010)
☐ Brewer (E005)	☐ Milker Container (E011)
☐ Waste Container (E006)	☐ Water Tap Connector (E012)

Coffee Maker Subsystems

☐ Grinding System (SS01)	☐ Water Heating System (SS03)
☐ Brewing System (SS02)	☐ Milk Processing System (SS04)

Figure 5.7: GUI of Fault Generator



Device Info
Here shows device information

Resource State
Here shows resource state

Current Task
Here shows current task

Figure 5.8: GUI of Coffee Maker Simulator

5.4 Functions

In this section, functions of three parts: the remote fault handling and reconfiguration system, the fault diagnosis system and the simulator will be introduced.

5.4.1 Functions of the Remote Fault Handling and Reconfiguration System

The remote fault handling and reconfiguration system is designed to handle received fault information and give function reconfiguration solutions. Functions of this system can be illustrated by the work flow shown in Figure 5.9.

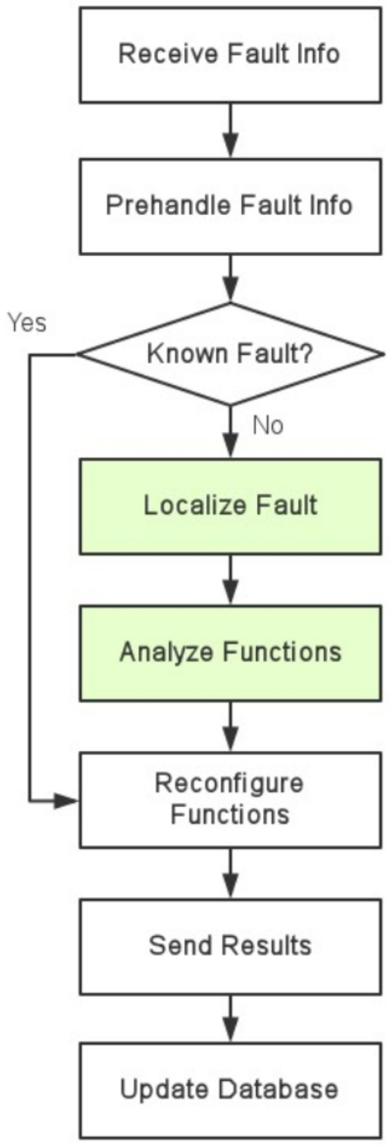


Figure 5.9: Work Flow of the Remote System

As highlighted in light green in the work flow diagram, key functions of the remote system are fault localization and function analysis.

First, fault information comes from an assigned port, the remote system will notice the user if the data is received successfully. Then the fault pre-handling module will determine if the received fault has been recorded in the remote database before.

Main function of the fault localization module is to figure out what components are defected basing on technical data analysis and then output the defected components' ids. With those ids, the system is able to determine which basic functions are affected by the occurred fault since a one-to-one mapping from components to basic functions are applied in this software prototype.

After defected basic functions are found in the fault localization step, the function analysing module extracts relation matrices from the system rules database and do matrix calculation to determine the affected upper level functions. Then this process continues until main functions at the top level are identified.

Reconfiguring functions is as simple as to translate the mathematical result of the function analysing module into text indicating the available function names.

Finally, fault handling and function reconfiguration results are formatted in JSON and can be sent to the message sender. Updating remote databases will take place right after result sending. The updated contents are listed below:

- The latest number of faults
- The new fault id
- Location of the handled fault
- Reasons for the handled fault
- Effects of the handled fault
- The symptom id
- A brief symptom description
- Available functions given the fault occurrence
- The current date

5.4.2 Functions of the Fault Diagnosis System and the Simulator

The fault diagnosis system and the simulator both run on the local computer and they cooperate together. Their functions are introduced together in this section. As shown in Figure 5.10, blocks highlighted in light green are functions of the simulator and the remaining blue blocks are functions of the FDS.

Functions of the simulator are intuitive. The first function is to simulate coffee producing. After a certain button with a coffee name is clicked, four actions take place:

- Information including the current task, remaining resources, device id are shown on the simulator GUI.
- Notice of ordered coffee status is shown the simulator GUI
- Technical data including are updated in the local database

- System states are updated in the local database

Simulating fault occurrence is another main function of the simulator. Faults are simulated to occur with some check boxes in front of component names and sub-system names are clicked. Afterwards, the generated fault information can be sent to the fault diagnosis system by click a button in the simulator GUI. Meanwhile all buttons with coffee names in the simulator will be disabled. In the end, the simulator will enable coffee buttons that are unaffected by the simulated fault after a reconfiguration solution is fed back by the remote system.

The fault diagnosis system is also a simulator that manages operation information and fault information of the coffee maker simulator. Key functions of it are:

- Show the local information on the GUI
- Determine the fault type (new or known) according to local fault knowledge
- Communicate with the remote system

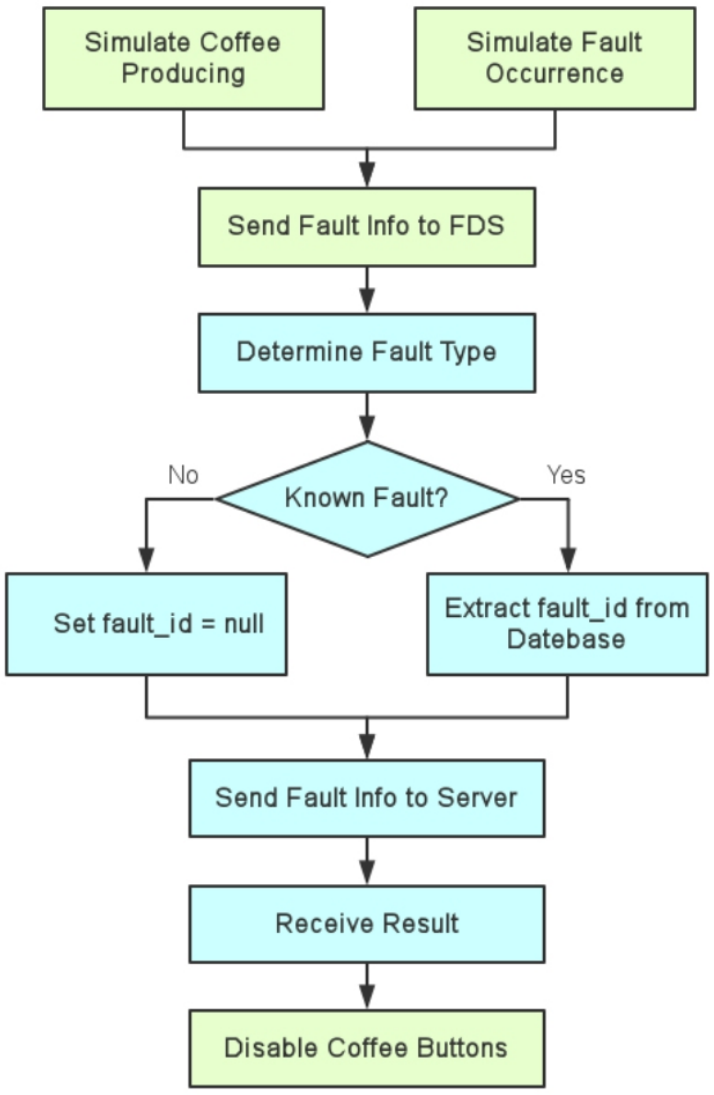


Figure 5.10: Work Flow of the Local System

5.5 Error Removal

5.5.1 General Errors

Errors that occur during starting of the local system and the remote system may due to the following reasons:

- XAMPP MySQL server has not been started successfully.
- Accessing and modifying privileges are not authorized to the new created account in the MySQL server.
- The allocated port for the remote system has already been occupied by other application.

5.5.2 Errors in the Remote System

In the remote system, one possible error can happen because of delayed manual operation between two handling cases. In order to have better demonstration effect, the remote system requires a manual action i.e. clicking the ‘Send Results to Clients’ button after each single fault handling case. After the button is clicked, results are sent to the corresponding connected FDS and the text area is set to blank waiting for the next handling result being printed there. This feature leads to a possible error that, if the previous result text is not sent by clicking the sent button, the new handling result text will be appended to the old one, which leads to a wrong reconfiguration action in the simulator. Solutions to remove this error is simple, no extra operations are required in the remote system. The system will get back to the right track after that error automatically as long as all operations are correct for the next case.

5.5.3 Errors in the Local System

For the local system which consists of the fault diagnosis system and the simulator, possible errors come from improper database management. There is a ‘Release Database Space’ button on the FDS GUI. It is used to drop technical data inserted by the coffee maker simulator that are older than five days when the database storage reaches its limit. But if the coffee maker simulator has not been used more than ten times before this dropping action, number of the technical data records in the local database will be less than ten, which leads to an error when FDS sends fault information to the remote system. The reason is that the remote system requires ten technical data records from the FDS for analysis and plotting data trend line charts.

To remove the aforesaid error, steps are described below:

1) Restart the local system;

2) Make at least ten times of coffee producing simulation via the coffee maker simulator;

3) Connect the local system to the remote system again;

4) Perform normal operation on both sides.

6 Conclusion and Future Work

6.1 Conclusion

To conclude, first, the requirement-function-component modeling approach is effective for analyzing the IAS coffee maker. This modeling approach reduces complexity of automation systems by dividing the whole system into proper hierarchical levels. In this thesis, a three-level component model is mapped to a three-level function model, which helps to find the propogation route of a fault.

Second, the proposed function analysis algorithm has two main benefits: one is that it is able to represent all involved function relations of the IAS coffee maker in a few number of tables, the other is that it is easy to be implemented and to be extended to other automation systems.

Third, according to testing results, it is proved that the software prototype for the remote fault handling and reconfiguration system can improve function availability of the IAS coffee maker. For instance, if a component merely related to milk processing is defected, the developed software prototype is able to give a reconfiguration solution which indicates the coffee functions that requires no milk can be continueously used.

6.2 Experience

Experience I gained during these six months are beneficial for my coming career life as a software engineer. Working with my supervisor Mr. Wang and two other teammates who worked on similar topics improved my teamwork ability. All the meetings and discussions we had inspired me and gave me a new view of software development.

The most important experience that I acquired should be programming skills in Java. I chose java to realize the designed software prototype because it is powerful and flexible. At the beginning of this project, I was impeded by implementing GUIs for the system since I was unfamiliar with Java Swing. I read lots of tutorials of Swing in order to find an effective way to realize my designed layout of each GUI. Fortunately, after a period of self-study, I mastered Swing well, which gave me more confidence to overcome other difficulties that I encountered in the later development stage.

Another important skill that I improved through this thesis was the ability to manipulate data via MySQL technology. I had some experience with IBM DB2 before, which made it easier for me to catch up in MySQL. They share many similarities, but MySQL is easier to manipulate and more applicable for small-scale databases. Thanks to this project, I am able to be familiar with MySQL and I could imagine that it will helps me a lot in my personal mini projects in the further.

Also, the comfortable working environment at IAS motivated me a lot and gave me a glimpse of my further working environment. Prof. Weyrich is always kind and gives proper instructions to help me within the right track. My supervisor Mr. Wang had great trust in me and provided much assistance during the whole thesis period.

6.3 Problems and Future Work

The main problem for my thesis was the difficulty to understand the physical and logical structure of the IAS coffee maker, because I do major in mechanical engineering or automation. Though my key work was to design and realize a software prototype, lack of knowledge in automation systems slowed down the development speed. In the future, it would be much better if students or specialists from automation field and computer science field work together to improve this software prototype.

Another problem was communications between the software prototype and the real IAS coffee maker. Currently, the implemented system is able to handle faults simulated in a coffee maker simulator. If a fault signal comes from a real machine, there are still some unforeseen errors that could happen.

For the future work, it is hopefully that the developed software prototype can be extended to handle faults of different kinds of automation systems, thus an integrated remote fault handling and reconfiguration system can be built.

Literature

[BJG14]	**Bordasch, M.; Jazdi, N.; Göhner, P.:** *Abnormality management for fault prevention in industrial automation systems,* In: International Conference on Control, Mechatronics and Automation, 2nd ICCMA Dubai 2014, presentation, p. 4
[Wiki16]	**Wikipedia:** http://en.m.wikipedia.org/wiki/Requirement, 2016
[GGBA15]	**Garcinuño, R.; Göhner, P.; Bordasch, M.; Abele, S.:** *Optimization of an abnormality management system in cooperation with a bottling plant*, Master Thesis, IAS, 2015
[WJW15]	**Wang, H.; Jazdi, N.; Weyrich, M.:** *Fault Effect Analysis based on a modelling Approach for Requirements, Functions and Components*, ACEC 2015 - 3rd International Conference on Advances in Computing, Electronics and Communication, 2015
[WJG15]	**Wang, H.; Jazdi, N.; Göhner, P.:** *Higher availability of an Industrial Automation System based on a Remote Problem Management System*, ICICM 2015 - International Conference on Information Communication and Management, 2015
[EGW15]	**Eichler, S.; Göhner, P.; Wang, H.:** *Entwicklung einer Smartphone-Anwendung für ein Problemmanagementsystem*, Studienarbeit, IAS, 2015
[LGW15]	**Laaber, N.; Göhner, P.; Wang, H.:** *Aufbau einer Schnittstelle zwischen einem automatisierten System und dem zugehörigen Problemmanagementsystem*, Studienarbeit, 2015
[BBG15]	**Bordasch, M.; Brand, C.; Göhner, P.:** *Fault-based identification and inspection of fault developments to enhance availability in industrial automation systems*, 20th IEEE International Conference on Emerging Technologies and Factory Automation, 2015

[PDA+08] **P. Trinidad, D. Benavides, A. Duran, A. Ruiz-Cortes, and M. Toro**: *Automated error analysis for the agilization of feature modeling,* Journal of Systems and Software, vol. 81, Nr. 6, pp. 883–896, 2008.

[Bord13] **Bordasch, M.; Göhner P.:** *Fault Prevention in Industrial Automation Systems by means of a functional model and a hybrid abnormity identification concept*, IECON 39th Annual Conference of the IEEE Industrial Electornics Society, p. 2848, 2013.

Declaration