A Decentralized Service Based Architecture for Fault Tolerant Control

Master Thesis

Supervisor at SCANIA: Jonas Biteus, Mattias Nyberg
Examiner at KTH: Prof. Ahmed Hemani
Master Student: Rui Li
Program: System-on-Chip Design

TRITA-ICT-EX-2012:202

Stockholm, Sweden
June, 2012
Acknowledgements

I really cherish the six-month master thesis experience at SCANIA Service Support Solutions Department, which is one of the best memories in my life. I want to express my sincere gratitude to the people involved in this master thesis project.

Special thanks are given to my supervisors at SCANIA, Jonas Biteus and Mattias Nyberg. They are both experienced engineers and they gave me patient instructions, advices and assistance when I confronted difficulties. I cannot go through smoothly without their help. Moreover, I appreciate the help of technical engineers Ulf, Jonnian and Kurre at SCANIA. They are very kind and willing to help me in terms of the system details, programming, and debugging, despite that they are very busy. Even the tedious problems we met, they can explain in a simple and humors way. Additionally, I want to thank the YSNS Department because of the internship opportunity, support and harmonious as well as interesting working environment.

Apart from the support at SCANIA, I would like to thank my examiners at KTH, Professor Ahmed Hemani and Doctor Ali, who continues care my thesis schedule, status and the presentation.

Last but not the least, without my beloved parents' encouragements, I cannot go so far in my master study. So I give my gratitude to them as well. Particularly, I want to thank my dear friend and partner Xia Zhou. The time we share during the master thesis and the constant assistance between each other are also the precious gifts that I got from the master thesis.

Rui Li

Stockholm, Sweden

June, 2012
Abstract

Fault Tolerant Control Systems (FTCSs) are control systems including fault tolerant control. These systems are famous for enabling reliability, maintainability and survival ability in safe vehicle design. In some SCANIA Electronic Control Units (ECUs), the ECUs FTCS is based on a centralized fault detector to detect faults and a centralized reconfigurator to reconfigure the system with degraded performance rather than, for example, completely shutting down the engine. However, with the size increasing in mechatronic system, the centralized architecture poses some problems in terms of performance, complexity and engineering facility.

This thesis will present a Decentralized Service Based Architecture for FTCS. It is a hierarchy architecture composed of a completely decentralized fault diagnoser and a completely decentralized reconfigurator. The decentralized implementation in this thesis is exemplified on part of the Exhaust Emission Control 3 (EEC3) system, one of the ECUs of SCANIA. There are two main parts, denoted a decentralized diagnostic manager (DIMA) and the service based communication framework for the interaction between DIMA and reconfiguration. Compared to the centralized architecture, a decentralized action handler has been built locally in each software module so that actions can be activated as soon as the fault is detected, through which a fast and guaranteed response can be obtained. The concept Service means that the dependency between modules which is solely based on fault propagation. Service communication framework reduces the complexity of the original FTCS. Each ECU can be regarded as a node in the entire communication network of the mechatronic system in SCANIA, and once all the nodes are implemented with the decentralized service based architecture, Bayesian Network can be constructed to model the FTCS with uncertainties.

Key Words: FTCS, Decentralized action handler, Reconfiguration, Service, Bayesian Network
List of Figures

Figure 1-1 General FTC structure ........................................................................................................... 10
Figure 1-2 General Structure of FTCS ..................................................................................................... 11
Figure 2-1 EEC3 system architecture based on module and layer .......................................................... 15
Figure 2-2 SCR System ............................................................................................................................. 15
Figure 2-3 Centralized DIMA Overview .................................................................................................. 16
Figure 2-4 Modeling of a diagnostic test .................................................................................................. 17
Figure 2-5 Typical ECU, Diagnostic Tests and connected physical Components Layout ...................... 18
Figure 2-6 Execution of a diagnostic test ................................................................................................. 19
Figure 4-1 Centralized calibration file ..................................................................................................... 22
Figure 4-2 Data structure of the Decentralized architecture ................................................................. 23
Figure 4-3 Decentralized calibration file .................................................................................................. 24
Figure 4-4 Centralized action handler and CAAM ................................................................................ 25
Figure 4-5 Decentralized action handler ................................................................................................ 26
Figure 4-6 The diagnostic test structure based on decentralized DIMA ............................................... 27
Figure 4-7 Debugging structure ............................................................................................................. 30
Figure 4-8 Hardware connection ............................................................................................................ 30
Figure 5-1 Service Overview .................................................................................................................. 33
Figure 5-2 Variant ..................................................................................................................................... 35
Figure 5-3 Open-loop Control ................................................................................................................. 36
Figure 5-4 Closed-loop Control ............................................................................................................... 36
Figure 5-5 SCR system overview .......................................................................................................... 39
Figure 5-6 I/O of PUMC (Appendix) .................................................................................................... 39
Figure 5-7 GeIne Tool modeled part of SCR system ............................................................................. 40
Figure 5-8 Abstracted modules .............................................................................................................. 41
Figure 5-9 Structure of the case study ................................................................................................... 43
Figure 5-10 Linear pressure sensor ........................................................................ 44
Figure 5-11 Pull-down resistor .............................................................................. 45
Figure 5-12 Pressure sensor voltage range ............................................................. 45
Figure 5-13 Pull-up resistor ................................................................................... 46
Figure 5-14 General structure of the sensor ............................................................ 46
Figure 5-15 Limp Home related modules and tests ................................................. 47
Figure 5-16 Working flow ...................................................................................... 49
Figure 5-17 Selector of PUMC .............................................................................. 51
List of Tables

Table 4-1 Typical initialization of data structure in application ........................................... 24
Table 5-1 Pseudo Diagnostic tests related to Limp Home mode ............................................. 43
Table 5-2 Look-up table of PUMC ......................................................................................... 50
Contents
Acknowledgements .................................................................................................................. 2
Abstract .................................................................................................................................. 3
List of Figures ............................................................................................................................ 4
List of Tables .............................................................................................................................. 6
Chapter 1 Introduction .............................................................................................................. 10
  1.1 Background ....................................................................................................................... 10
    1.1.1 Fault and Fault Tolerance ......................................................................................... 10
    1.1.2 Reconfigurable Fault Tolerant Control System ....................................................... 10
    1.1.3 Function and structure of FTCS ............................................................................... 11
  1.2 Motivation ......................................................................................................................... 12
  1.3 Scope and Goals ............................................................................................................... 12
  1.4 Related work .................................................................................................................... 13
Chapter 2 Prototype Introduction ............................................................................................. 14
  2.1 EEC3 System .................................................................................................................... 14
    2.1.1 Introduction .............................................................................................................. 14
    2.1.2 EEC3 Architecture .................................................................................................. 14
    2.1.3 SCR system ............................................................................................................ 15
  2.2 Module Introduction ......................................................................................................... 16
    2.2.1 KWP Server ............................................................................................................ 16
    2.2.2 Diagnostic Event Server ........................................................................................ 16
    2.2.3 Context of DIMA ................................................................................................... 16
    2.2.4 Diagnostic Architecture of DIMA ......................................................................... 17
    2.2.5 Working Flow of on-board diagnosis ...................................................................... 18
    2.2.6 Conclusion ............................................................................................................... 19
Chapter 3 Software Development Tools ................................................................................... 20
3.1 System Software Debugger ................................................................. 20
3.2 XCOM and SDP3 ............................................................................. 20
3.3 VISION .............................................................................................. 21
3.4 GeInE ................................................................................................. 21

Chapter 4 Decentralized Diagnostic Manager ............................................. 22
4.1 Decentralized calibration file ................................................................. 22
4.2 Event drive Test Filter ......................................................................... 25
4.3 Decentralized Action Handler ................................................................. 25
   4.3.1 Centralized Fault Accommodation Manager ................................. 25
   4.3.2 Decentralized Action Handler implementation ............................... 26
4.4 Implementation in EEC3 ....................................................................... 27
   4.4.1 Single mapping ............................................................................. 28
   4.4.2 Other types of mapping ................................................................. 29
   4.4.3 EEC3 Debugging and Verification ................................................. 29
   4.4.4 Discussion ...................................................................................... 30

Chapter 5 Service based FTC on SCR system ............................................. 32
5.1 Theory of Service based FTC ................................................................. 32
   5.1.1 Introduction .................................................................................. 32
   5.1.2 Basic Concept ............................................................................. 32
   5.1.3 General Principle of Service Provider ........................................... 34
   5.1.4 Diagnostic Modeling ..................................................................... 37
   5.1.5 Bayesian Network ......................................................................... 38
5.2 Implementation of SCR system .............................................................. 38
   5.2.1 SCR system introduction ............................................................... 38
   5.2.2 SCR system modeling ................................................................. 40
   5.2.3 Working Flow .............................................................................. 48
5.2.4 Discussion ................................................................................................. 51

Chapter 6 Conclusions and Future Work .......................................................... 53
  6.1 Conclusions .............................................................................................. 53
  6.2 Future Work ............................................................................................. 54

Appendix ........................................................................................................ 56

Bibliography ................................................................................................. 57
Chapter 1 Introduction

This Master thesis is the requirement of Master Program System-on-Chip Design in KTH Royal Institute of Technology. In general, the work has been conducted in Service Support Solutions Department of SCANIA CV AB. Moreover, this thesis project is in cooperation with Xia Zhou[1]. Both of us have different focus areas in this project and some parts of the master thesis are cross-referenced.

1.1 Background

It is acknowledged that Fault Detection and Isolation (FDI) makes contribution to reliability, maintainability and survivability of the vehicle system. Nowadays, Fault Tolerant Control (FTC) systems with efficiency are attracting more and more attentions in terms of higher system performance, product quality and cost [2]. In order to manufacture the safe vehicles to meet the current sophisticated requirements, the scale and the complexity of the system will surely beyond today’s situation. In this sense, it poses large burden for the controllers to develop with satisfied capability of fault accommodation and tolerance.

1.1.1 Fault and Fault Tolerance

Each individual part in the system has its own functionality which serves for the entire system working in a harmony way. In a broad sense, fault is some abnormal behaviors of components or internal wrong event or even environmental condition abrupt change that causes the system behave out of control [3]. Hence, fault leads to change in the system in terms of structure or parameters and results in degraded performance or even loss of full functionality. In order to avoid or impair the deteriorations or damage to the system, it is very important to check the faults as soon as possible so that corresponding repair actions can be taken to stop faults propagation. The aim of fault tolerance is to make the system fault tolerant and not to lose the total performance. Thus, Fault Tolerant Control, which implements the control mechanism to tackle the fault detection and diagnosis, has been developed. Generally FTC consists of the system and the FTC controller shown in Figure 1-1.

1.1.2 Reconfigurable Fault Tolerant Control System

Fault Tolerant Control system (FTCS) is a control system that maintains some of the performances closed to normal working mode and keep the entire system under operation, not only in nominal case but also in the presence of faults, or at least can provides degraded
performances close to the desired ones [4]. It is very important for heavy truck to keep some of the function still running rather than stopping the engine despite of faults emerging in the system. Several publications about the new development in FTC methods applying in heavy vehicle have been introduced in [5]. The conventional way of building FTC is based on a centralized diagnoser and a centralized reconfigurator, which has been widely utilized in many fields of vehicle industry. The general structure can be seen from Figure 1-2 and it is a closed-loop control with feedback. To obtain a good performance, the entire system should have a fast reaction by reconfiguring control actions once there happens a fault inside and keep the uptime of the system as long as possible.

![Figure 1-2 General Structure of FTCS][5]

1.1.3 Function and structure of FTCS

This section is based on [5]. FTCS addresses the control system with the capability of maintaining the basic function throughout the working process and providing adaptive performance by reconfiguring itself in the event of fault failures, which is considered as the accommodation of inner failures. In this sense, typically, FTCS gives importance to the two working modes both in normal situation and in the presence of a fault. In normal working mode, it is the function quality and system action performance that counts a lot. However, in abnormal mode, the emphasis turns to the system survival with a degraded performance.

In general, there are four sections composing FTCS, denoted FDI scheme, reconfiguration mechanism, reconfigurable controller and command governor. FDI scheme is responsible for the real-time fault detection or diagnosis; reconfiguration mechanism is used to organize the reconfigurable controller so that the faults can be mended to some extent; command governor is needed to give useful information or suggestions to users or mechanics dealing with faults.
The key factor in designing FTCS is how to construct a reconfigurable control system and make FDI scheme that has guaranteed detection ability and fast response or action handler. However, the centralized FTC structure faces great challenges when dealing with the large-scaled mechanical system.

1.2 Motivation

It is known that the limited amount of reaction time to perform fault detection and control system reconfiguration is becoming more and more critical for FTCS. In this sense, how to construct a self-repairing and reconstructable control system and a sensitive FDI scheme is the point [6]. In SCANIA, the FTC development should be based on the goal to maximum the uptime of vehicle. It could be a waste of time and money to shut down the entire machine since there will be faults appearing in the system all the time.

The second critical element is that, with the increasing size as well as the complexity of the control system, it is very hard to maintain the centralized FTCS to meet the current requirement in terms of scaling. For SCANIA, it is requirement to develop safe control system as well as to maintain the software after development. Since there are many engineers working on the same system, and it would be difficult to keep the software functioning correctly all the time. In this sense, how to ensure the correctness and provide easiness and convenience for engineers to maintain and modify according to realistic industry requirement is also the point in designing FTCS.

1.3 Scope and Goals

This master thesis aims to develop a prototype of hierarchical architecture for fault tolerant control of large-scale mechatronic system of SCANIA, which can be regarded as the preparation of decentralized ECU nodes for Bayesian Network construction. This architecture pattern includes a completely decentralized diagnoser and a completely decentralized service based communication framework used for reconfiguration.

There are two general tasks to conduct in this thesis. One is to build a Decentralized Diagnostic Manager (DIMA) and the other is Service based FTCS. Both of the sub-tasks are exemplified on EEC3 (Engine Exhaust Control unit), one of the ECUs (Electronic Control Unit) of SCANIA.

For the first part, my work focus area is to construct a decentralized action handler with the target of notifying the system or mechanics when the faults are confirmed to be active. Moreover, decentralized calibration files have been built. As the final verification and demonstration of decentralized DIMA, use cases of two typical application have been implemented on EEC3.

For the second part, my work is concentrated on the service communication framework dealing with the interaction between decentralized DIMA and reconfiguration based on one of the sub-systems in EEC3.
There are three main objectives of the system listed below.

- Obtain good FTC performance.
- Reduce complexity of FTC.
- Facilitate efficient distributed engineering of large-scale system.

The first is about the control performance with respect of a fast and guaranteed response to diagnose. The second goal mainly deals with the complexity associated with the number of fault combinations and the number of reconfiguration possibilities. Since for large-scale mechatronic control systems, possibly with multiple function units distributed in the overall system, decentralized architecture can release the pressure of individual module. The last one contributes to the relation between modules and it minimizes the dependencies between subsystems.

1.4 Related work

The first part of the implementation is mainly based on the centralized Diagnostic Manager of EEC3 system inSCANIA. Further modification and construction in coding of EEC3 has been completed, regarding the application layer. Typical use cases of EEC3 have been conducted to verify the decentralized DIMA and the decentralized action handler. Additionally, relative simulation results by monitoring tools of SCANIA are presented.

The other part of this master thesis is about the theory analysis of Service based FTC and a sub-system in EEC3 has been modeled and implemented based on the service architecture.
Chapter 2 Prototype Introduction

This master thesis contains the general architecture construction as well as the case implementation based on EEC3 system. In this chapter, introduction about the use case EEC3 system will be illustrated.

2.1 EEC3 System

2.1.1 Introduction

There is growing demand and requirement to develop more efficient ways of taking care of the exhaust emissions of vehicles. The goal for SCANIA is to limit and control the amount of emissions that heavy trucks produce. A common way of solving this in the heavy truck industry is by after treatment of the exhausts. The Exhaust Emission Control 3 Electronic Control Unit (EEC3 ECU) is a system for exhaust after treatment. Among all the ECU’s of SCANIA, there is the Engine Management System (EMS), which have sensors and actuators linked to the engine. EEC3 will communicate with EMS by means of Communication Area Network (CAN-bus). The EEC3 ECU will be installed on chassis in various types of vehicles and used in all different climates and condition. It will be exposed to harsh electrical environments, wide temperature variations, high humidity and rough vibrations. EEC3 forwards exhaust sensor data to EMS. The EMS calculates dosing amounts based on engine, filter and catalyst operating point. EEC3 receives the requested dosing-amount and is responsible for supplying AdBlue. EMS holds all necessary knowledge about the engine and exhaust-system. As a consequence, exhaust-sensors are connected to EEC3 whilst their data is only used in EMS [7].

2.1.2 EEC3 Architecture

From the architecture point of view, EEC3 software system has been constructed based on the concept of modules and layers. Introduction of EEC3 is based on system description of EEC3 in SCANIA [8].

A module is a group of functional entity of source code that focuses on the solving a specified task by multiple functions embedded in this module. Each module is composed of a pure C file and the relative header file that contains the interface definition to the environment.

Each layer constitutes of several modules and there is also communication between different layers. The structure graph can be seen from Figure 2-1.
Layers that have been modified in our case study is High Level Application Layer (APPL). Details will be introduced in Section 2.2.

2.1.3 SCR system

Selective catalytic reduction (SCR) is one of the most powerful techniques for reduction of NO\textsubscript{x} in diesel engines. The SCR technology is based on a chemical reluctant that is applied in the exhaust flow upstream of the catalyst.

SCR-system is one of the sub-systems of EEC3, which injects AdBlue, a mixture of urea and deionized water, into a catalytic reactor in which the urea reduces NO\textsubscript{x} to nitrogen oxide and water. The main components of the SCR-system in SCANIA are the AdBlue tank, pumping unit, dosing unit, and a control unit (Figure 2-2). The SCR-system has sensors for temperature and pressure in the AdBlue tank. There is also a NO\textsubscript{x} sensor placed after the catalyst. Based on measurements of the temperature of the exhausts and in the catalyst as well as information about the current working point of the engine, the amount of produced NO\textsubscript{x} and the potential to reduce this can be calculated by the EEC3. EEC3 then calculates the amount of AdBlue that should be injected and sends this information to the dosing unit (injector). To further increase the performance the ECU can correct the amount of injected AdBlue by feedback of the difference between measured NO\textsubscript{x} after the catalyst and the setpoint of allowed NO\textsubscript{x} [9].

Figure 2-1 EEC3 system architecture based on module and layer

![Figure 2-1](image)

Figure 2-2 SCR System
2.2 Module Introduction
In this section, parts of the critical sub-modules related to diagnosis are introduced.

2.2.1 KWP Server
KWP (Keyword Protocol 2000) server can be configured to handle a number of KWP services coping with communication between modules or ECUs. Examples of services are reading Diagnostic Trouble Codes (DTC) and Security Access. This module is referenced from the Keyword Specification 2000 specification SSF 14230-3.

2.2.2 Diagnostic Event Server
A diagnostic framework has been developed, named Diagnostic Manager (DIMA) has been developed as well. In SCANIA’s heavy truck FTCS, precompiled diagnostic tests are responsible for the detection of the faults. These sets of diagnostic tests or single one correspond with a specified DTC number and a series of DTC calibration documentation. In essence, DTC is used for describing the non-normal behavior information in the system, including the detection, cause, symptom, system reaction, actions and the engineer comments of one specified fault. Based on the DTC information, a certain amount of or the most probable faults can be detected so that FTC can be realized. Fault isolation can also be built to shift the system into reasonable working modes and guide the mechanic to repair the machine in an efficient way. Diagnostic event server is constructed based on DTC calibration file. This part is based on [II].

2.2.3 Context of DIMA
DIMA generally has the responsibility of fault code storage handling, requesting warning lamps and degradation actions. The overview of DIMA can be seen in Figure 2-3. Hence, DIMA-BSW was developed to serve as a central manager to tackle all the diagnostic tests and the relative actions. In the application layer, there are multiple diagnostic tests scattered for different test purposes and each tests or a set of tests is connected to one specified DTC according to a certain logic rules, which is called the fault isolation mechanism.

![Figure 2-3 Centralized DIMA Overview](image-url)
2.2.4 Diagnostic Architecture of DIMA

The diagnostic architecture based on DIMA consists of several vital modules, which will be introduced in the following sections.

1. Diagnostic Test

One or several Diagnostic tests exist in modules of the application layer and each test will detect the presence of faults with different targets, such as electrical faults of sensors, battery and wires. The general principle of modeling diagnostic test can be seen from Figure 2-4.

![Figure 2-4 Modeling of a diagnostic test](image)

First, the characteristic signal calculation can be done through the data collected from the external sensors or actuators. Actually, the characteristic signal can in common lies in two aspects. One is the difference between a measured value and an estimated value of the same physical property (usually called a residual). The other can be the sensor value itself. Then, for this specified characteristic signal, it comes to the thresholded phase. When the characteristic signal goes through the threshold phase, the fault is considered to be pending. Under the condition that if the fault is pending for a certain amount of time, the fault becomes validated. The finished flag will be set to indicate if a fault is validated or not.

The calculation described above is controlled by a test condition flag. When condition is TRUE, it means that the test is running. If a test is not running, it does not necessarily mean that the execution of the test is stopped. Instead it means that the pending flag is set to FALSE, and that the validated flag and the finished flag are not updated.

The flags conditionsFulfilled and faultPending are used by the service workshops as help signals when repairing a vehicle. This means that when constructing the logic setting these two flags, one should have in mind, the usefulness for workshop staff. For example, the bit faultPending reflects the case that the fault is probably present but not necessarily validated.

2. DIMA on-board diagnosis
DIMA-BSW is responsible for diagnosis. DIMA is the centralized diagnostic module. DIMA-BSW in general has the following functionalities: Diagnostic test validation support; Handling of data produced by the tests, which includes storage of DTCs, erase of DTCs, enable of tests; DTC to degradations.

3. Fault Isolation

Fault isolation is used to, given a set of validated DTCs, find the most likely and less likely faults that can explain the validation of a specific DTC.

In SDP3, mechanics can be shown a set of DTCs or alarms. Moreover, mechanics can select the specified DTC and explain what is the actual cause.

2.2.5 Working Flow of on-board diagnosis

![Figure 2-5 Typical ECU, Diagnostic Tests and connected physical Components Layout](image)

**Phase 1**: Initialization

Initialization phase mainly deals with setting up, registering and enabling diagnostic server, setup an instance of DIMA.

**Phase 2**: Test filter

Diagnostic tests are located in the application layer, like in EEC3. The results of these tests have to communicate with DIMA via a test communication structure to each diagnostic test. This structure exchanged between a diagnostic test and DIMA-BSW is called structure type `testCom`. The purpose of the testCom is both to keep in memory a diagnostic test’s states between executions, and to report result to DIMA-BSW.
Figure 2-6 Execution of a diagnostic test

Phase 3: DIMA execution

2.2.6 Conclusion

The working flow of centralized DIMA is clear, however, this kind of architecture has problems. The first problem is that, with the development of large-scaled mechanical system, it scales poorly for the FTC to handle the entire system in an efficient way. Since every module in application layer will become larger and more complicated, so it increases the work load of centralized DIMA. Thus, the working efficiency of fault detection will decrease. But in truck system, if the system cannot react to the fault appearance and take the effective actions in time, there will be danger to both vehicles and drivers, but also decrease the uptime of the entire system. The second problem is that the centralized calibration file is very complex with C code and it needs multiple engineers to maintain. However, once there is one single or small modification in the centralized calibration file, the whole system should be changed accordingly and it is very hard for the engineers to remember all the data structures that are indexed in the system.
Chapter 3 Software Development Tools

During this thesis implementation, both the hardware and the software tools are used for validation in the code and model level. The software development tools include Eclipse, SDP3, XCOM, Vision and GelNe, which are used for the implementation in terms of modeling, debugging and monitoring the system. Moreover, some auxiliary hardware, such as VCI, USB keys, CAN bus and power supply are also used in this thesis.

3.1 System Software Debugger

The debugger consists of three parts: a front-end running on the user PC, a back-end that is running in EEC3 and a proxy that handles communication between the front-end and the back-end. The front-end is handled by the GDB and the graphical user interface to it is handled by Eclipse 3.4 with CDT 5.0. The proxy is a program called GDB to CAN can be received using VCI connection between the PC and EEC3 system running. The back-end of the debugger is called the GDB server (GDBS) [12].

3.2 XCOM and SDP3

XCOM and SDP3 are both Diagnostic tools to read out fault codes, program parameters, log variables and control IO signals. XCOM is more used at Research & Development (R&D) Department in SCANIA, while SDP3 is more frequently used outside R&D.

XCOM has been developed by Powertrain Control System Development NEVE Department of SCANIA and in the thesis, Version 2.16.0.0 was used. XCOM is connected to CAN with the required Bit rate at the start phase through VCI and it is feasible to choose the specified ECU available on CAN. In this sense, the application can be connected to CAN via KWP. The basic use cases that XCOM conducting are CAN connection, retrieving data from vehicle (DTC and DEC information view via KWP communication request), resetting ECU, clearing memory data and recovering defective ECU [13].

SDP3 has been developed by SCANIA and communicates with SCANIA vehicles and SCANIA industrial and marine engines. The program has been developed to support the electrical system with CAN communication. The program is used for troubleshooting, adjusting customer parameters, calibrations, conversions affecting the electrical system and updating software in control units. In this thesis, Version 2.10.0.4034 is adopted. Auxiliary devices, such as PC, VCI for connection and Black USB key for the access, are needed during the usage of SDP3. It can be used for the adjustment and check of the control system and the circuit and the components and the navigation of the electrical system. The main feature of SDP3 used in this thesis is the Fault code Monitoring. The registered fault codes of the entire vehicle or individual control part can be clearly displayed and can be cleared as well.

In SDP3, fault codes are divided into active and inactive ones. Active fault codes are fault codes which have been registered and where the fault persists. Inactive fault codes are fault codes which have been registered but where the fault has then disappeared. The fault codes
are also divided into primary and secondary fault codes. A primary fault code is an original fault code. A secondary fault code means a fault code which has been registered in a control unit because a primary fault code has occurred in another control unit.

Active and primary fault codes are always displayed, and you can then choose whether you also wish to view inactive and secondary fault codes in SDP3 [14].

3.3 VISION
VISION has been developed by ATI Company and it is an integrated development, calibration, and measurement system for ECU. ECU calibration comprises data acquisition of intermediate calculated variables, analysis of critical parameters, and the tuning of constants to classify the controls system. VISION can be used to systematically find optimal calibrations and rapid prototyping solutions for increasingly complex controls systems. Additionally, according to different requirement of the customers, VISION can be configured. In this thesis, Version 3.6.2 is used and the main functionality is monitoring the intermediate variable, inputs and the outputs inside the code for validation.

ECU design process consists of system analysis, system specification, system design, automatic code generation, integration of ECUs and the corresponding software in a real environment, and calibration. The VISION Strategy file stores the device description information, memory images, and system settings for a device. This device is usually the control module (ECU) under development. In order to monitor the intermediate results and the outputs as well we control some variables in the execution phase, a strategy file is necessary. In this thesis, the CB file is the link between the source code and the VISION. After all the system has been compiled and the related .vst file can be produced so that it can be downloaded or flashed into the real ECU hardware [15].

3.4 GeINe
GeINe is a development environment for building graphical decision-theoretic models. It has been developed at the Decision Systems Laboratory, University of Pittsburgh. In this thesis, the graphical decision-theoretic models are especially related to Bayesian networks (also called belief networks, Bayesian belief networks, causal probabilistic networks, or causal networks, which are acyclic directed graphs in which nodes represent random variables and arcs represent direct probabilistic dependences among them. This tool is mainly used corresponding to the Service based the FTC architecture [16].
Chapter 4 Decentralized Diagnostic Manager

Original DIMA is a centralized module coping with diagnosis. The diagnostic objectives of DIMA are the physical components connected to the application layer control unit and the internal signals of DIMA. However, with the increasing size and complexity of large-scale system, centralized DIMA will become heavy-loaded. Another fact is that in centralized DIMA, there is a centralized action handler to trigger the related actions after the fault has been detected. In order to get a fast and guaranteed response, a decentralized action handler has been constructed in each software module rather in DIMA. This section focuses on the Decentralized DIMA implementation on part of EEC3.

Decentralized DIMA is just the pre-development of the new architecture. So it is a simple implementation and has the basic fault control functionality. In this thesis, we discard the complicated parts like enable control, freeze frame and memory management. This new pattern, in general, there are three sections, denoted the decentralized calibration files, a new test filter module and a decentralized action handler.

4.1 Decentralized calibration file

The centralized calibration file can be seen from Figure 4-1.

![Figure 4-1 Centralized calibration file](image)

To make the calibration file easier for the engineers to maintain, decentralized calibration file is built. It means that, in one specified application file, there may have multiple diagnostic tests and each test should have its own calibration file. The individual calibration file for each test is represented as the new type of data structure NEWDIMA_TEST_STR. In order to keep the original data unchanged, in this thesis, only the re-combination and some small modification of the data is done. The new data structure in decentralized DIMA can be seen in Figure 4-2. Throughout the decentralized DIMA, only a NEWDIMA_TEST_STR is enough, because it contains all the calibration information of the data used in the diagnosis system. Moreover, NEWDIMA_TEST_STR is global variable.
Figure 4-2 Data structure of the Decentralized architecture

In the thesis, we keep the testCom_str as the original one and used as part of the communication between diagnostic test and new DIMA. Compared to the old data structure, the modification lies in two aspects. The first one is the naming replacement and recombination. TSCOMDEC_STR is the new structure consists of the pointers to testCom_str and DEC_STR respectively.

DTC_STR has two parts, denoted DTCCAL_STR and DTCVAR_STR. DTCCAL_STR is the replacement of the original BMDEF_STR with the calibration information of DTC. While DTCVAR_STR is the renaming if the original BMVAR_STR. Some bits in the new data structure related to fault isolation are kept but unused.

The other change about the data structure is the modification. In the DTCCAL_STR, a pointer to the array, which contains pointers that are pointed to all the connected testCom_str:s, is added. The reason why we need this array is that there will be multiple DECs linked to one DTC. If any of the DECs is validated, the connected DTC will be active and the statuses will be changed accordingly.

At the beginning of each specified application, declaration and initialization of NEWDIMA_TEST_STR should be performed. The items that should be constructed and initialized are listed below in Table 4-1.

In the same application module, if there are more than one tests correlates to one DTC, only one DTC_STR should be declared and initialized. If the multiple tests relating to one DTC are not in the same application module, declaration of the DTC_STR once is enough and the other modules should include the header file of this module where the DTC_STR is declared. Additionally, the data structure that declared in application module should also declare the interface in the linked header file.
### Table 4-1 Typical initialization of data structure in application

<table>
<thead>
<tr>
<th>Item type</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>tNEWDIMA_TEST_STR</td>
<td>Declare the specified name and initialize it.</td>
</tr>
<tr>
<td>tDIMA_TESTCOM_STR</td>
<td>Declare a name and initialize all bits of it to 0 except enabled_1 to 1.</td>
</tr>
<tr>
<td>tNEWDIMA_TSCOMDEC_STR</td>
<td>Declare a name and initialize DEC_Nr and DECVAR_STR to 0.</td>
</tr>
<tr>
<td>tNEWDIMA_TSCOMDEC_STR*</td>
<td>Declare a name and initialize the array with all the pointers to the connected testcomdec_str.</td>
</tr>
<tr>
<td>tNEWDIMA_DTC_STR</td>
<td>Declare a name and initialize the DTCCAL_STR to the specified DTC information and all the bits in DTCVAR_STR to 0 except enabled_1 setting to 1.</td>
</tr>
</tbody>
</table>

The other important data structure is ACTIONHANDLERS_STR. Also in the application, the naming declaration and initialization should be performed.

All the new data structure are defined in the dima.h file, which includes all the interface and structure that are used in the DIMA. In decentralized DIMA architecture, the new data structures built are just added at the end of the original dima.h file so that once there is need to use the centralized DIMA interface in the future, it will be very convenient. On the other hand, both the centralized and the decentralized DIMA can run in parallel without disturbing each other.
Each application module may have several diagnostic tests and all these tests are represented by the data structure (Figure 4-3). It should be declared and initialized at the beginning of this software module. It is very easy for the responsible engineers to maintain the decentralized calibration file. Because it is located in the specified module and each engineer are responsible for a number of calibration files. Once there is need to modify his own calibration files, it can be performed directly without notifying the other engineers. Unlike the centralized calibration file, it has the connection between each other and it is very hard to find the index to trace what has been changed and what will happen to others linked. In other words, the decentralized calibration file facilities the efficient distributed engineering of large-scaled system.

4.2 Event drive Test Filter
For more information about event drive test filter, see Xia Zhou’s work [1].

4.3 Decentralized Action Handler

4.3.1 Centralized Fault Accommodation Manager
This part is based on the internal material on fault handling of SCANIA [17].

Fault Accommodation Actions
DIMA-BSW supports accommodation actions for each DTC.

Centralized Action Handler
In the original DIMA, Action Handler is centralized. The action handler structure in centralized DIMA.

According to different action states, software modules will react and perform the changes to the system or drive the hardware. Error! Reference source not found. displays how centralized Action Handler works throughout different layers. Action handlers are executed at the end of each run of DIMA.

Figure 4-4 Centralized action handler and CAAM

For example, in the SCR system, if the pressure sensor in the injector has been detected to be short circuit to ground and there is also something wrong with temperature sensor in pump.
Both the two electrical faults are found by the centralized DIMA. Each test is linked with one DTC and there is related actions as well. Once the DTC is active, centralized DIMA collects all the actions of the two DTC and triggers the functions in CAAM. Inside CAAM, there are several error action states used for controlling all the software modules of SCR. These action states are matched with the DTC actions. All the software modules (for example M1-M3 in Figure 4-) of SCR will work based on these action states. When the state changes, the working mode will alter accordingly.

4.3.2 Decentralized Action Handler implementation

RunActionHandlers() is the function handling decentralized actions in each specified diagnostic test. In total, there are three inputs which are all pointers, denoted the pointer to DTCCAL_STR, DTCVAR_STR and ACTIONHANDLER_STR. At the beginning of this function, it will check whether this specified DTC is active or not. If it is passive, there will be no actions. Once this DTC is active, this function will go on checking which function pointer is selected. However, in the new ACTIONHANDLER_STR, there are only two elements, denoted the function pointers to CAAM module for degradation. The following is the pseudo code.

New_ActionHandler_str_init
{
    function_pointer_action1;
    function_pointer_action3;

}

Warning lamp is discarded because in EEC3 system, the different color warning lamps are controlled by the other system. The other reason why it is discarded in this thesis is that warning lamp is better to be kept or considered as centralized. Since every DTC will trigger the lamps to notify mechanics once there is fault appearing. However in this thesis, the goal is to construct a completely decentralized action handler. The function pointers actually could be the functions located in the modules who will do the real actions (Figure 4-5). Consequently, according to different situations, the initialization of the new action handler structure will be different. In order to facilitate the design, we still take the example of CAAM as the action taker.
Unlike the centralized action handler, decentralized action handler will be executed in each diagnostic test. It will pose danger to the vehicle as well as the drivers if the actions are not taken in time. In this sense, it is the trend that decentralized action handler will improve the performance of FTC system. Actually the actual actions are executed by the related modules and the hardware can be driven to do the required actions.

The pattern is reserved so that it can be easily modified when needed in future. Another reason is that the service based FTC will be the enhancement of the completely decentralized action handler (Chapter 5).

**4.4 Implementation in EEC3**

In order to validate the decentralized DIMA, we implement part of the EEC3 system working based on decentralized DIMA. Use case is used to validate whether the decentralized DIMA works well or not, part of the diagnostic tests on different application layers are modified to be adaptive to the new FTC architecture. According to the mapping principle between diagnostic test, DEC and DTC, the use case on EEC3 system can also be divided into two types. The first one is the one-to-one relation and the other one is the multiple-to-one relation. The following section is the detailed content about how each test are connected with the new DIMA and how action handlers are distributed in each application module to take the actions to inform the mechanics or the engineers once the specified DTC is confirmed to be active.
Firstly, the basic principle or working flow in each software application module is illustrated here (Figure 4-6). At the beginning of each application C file, dima.h and caam.h should be included and there may be several diagnostic tests that should be declared and initialized. For each diagnostic test, the corresponding test_str as well as its inner structure, such as the testCom_str, DTCCAT_STR and the array of all connected DECs, etc, should be declared and initialized at the start of the c file. Moreover, the interface should be also declared in the linked header file so that once several DECs share the same DTC, only including the header file where the DTC is defined is enough. The next step is about the action handler structure definition and initialization. For each DTC_STR, it should have its own action handler structure. The declaration and the initialization are in C file, while the definition should be put in the linked header file. The reason is the same as the DTC_STR mentioned above.

Before the actual execution of the diagnostic test, the preparation about the condition to trigger decentralized DIMA should be performed, that is the judgment of the test condition. This phase will check the parameters that related to the diagnostic tests or the sensors that are connected to some electrical elements or components. More often, checking the test condition depends on the different diagnostic tests. In the figure below, it can be concluded that the part upper the dotted line is the data structure declaration and initialization phase.

Based on the test condition check, the Dima_EventDrive() function has the related inputs, such as the signs of whether condition is fulfilled or test is pending or not. This will trigger the decentralized DIMA to do the check validation function, which is the same as the original DIMA. When the test is checked, the status bits in the testCom_str will be updated. In this sense, the following two functions which are both used as the update functions, but for different structures. At the end of Dima_EventDrive(), all the status bits in test_str will be newly updated and the most important one is that the specified DTC is validated or not will be confirmed here. The next phase is about the work of action handler.
Since calibration file is located in each application module, the local action handler can have knowledge about the information of each related DTC. For a specified test, the corresponding action handler will first check the status of DTC, whether it is activated or not. If it is passive, there will be no actions executed. Otherwise, it will come to the two function pointers which handle the redetected action and the direct degradation respectively.

In the following section, use cases of one-to-one relation and multiple-to-one relation will be illustrated.

**4.4.1 Single mapping**

Single mapping means that the relation among diagnostic tests, DEC and DTC is one-to-one. The case selected in this thesis comes from a module from APPL layer in EEC3 system. This module is responsible for the diagnostics to verify that SCR system is fully functioning. These diagnosis of this module are based on the Pulse Width Modulation (PWM) signal transmitted to the pump in the SCR system. PWM signal is determined by the regulator that regulates the SCR system’s AdBlue pressure and controls the speed of pump. The controller attempts to maintain a constant pressure when the system is in the dispensing mode i.e., when the predetermined operating pressure is reached. If the suction pipe for AdBlue become blocked or leak so the pump will have to work harder to maintain the working pressure, this results in an increased PWM signal from the controller. The same applies if there is a leak in the pressure line. A blockage in the return line or metering valve resulting however in a reduced PWM signal from the controller when a smaller AdBlue flow needs to pass through the pump to maintain pressure. By comparing the regulator from the PWM signal given to a model of what the PWM signal should be perfectly to the system can detect this failure mode. The test has two modes one who can put error codes, used in normal operation, and one which only report a status, used during the workshop and more.

For backflow clog test, Dima_EventDriveTime() is adopted and executed after the original DIMA check validation function so that both DIMA can run in parallel. When testCond_B is true and dosingActive_B equals false, the condition is fulfilled for this test. If the residual is lower than the limit of the amount, it will indicate there is a fault, which means that backflow clog test is pending. These are the two typical parameters that are used to trigger the decentralized DIMA. Then new DIMA will perform the update of the testCom_str, DEC_STR and DTC_STR accordingly. After DIMA’s work, action handler will come to power. It will check whether this DTC is active or not and then it decides what kind of action will be conducted via CAAM.

For the last test checking inlet or pressure leakage, it follows the same working flow as the first test. The only difference lies in the judgment of condition fulfilled and pending.

**4.4.2 Other types of mapping**

In this module, both the two diagnostic test are executed, which is used to calculates checksums of the different memory sections. Each test checking flow is the same, denoted
triggering decentralized DIMA by function `Dima_EventDrive()`, running action handler and updating the variables in VISION.

### 4.4.3 EEC3 Debugging and Verification

On the whole, the debugging and verification work (Figure 4-7) can be divided into several phases.

One step is to build up the EEC3 system and flash the program into EEC3 hardware. The last step is verifying the EEC3 system based on VISION, XCOM and SDP3 to monitor the outputs. If there is any error happening or the source code modification in the first step, all the steps have to be performed again from the beginning.

![Figure 4-7 Debugging structure](image)

The compilation work of EEC3 system is based on Eclipse CDT. After importing the whole programs into Eclipse, make sure that the `dima_newstu.c` file, the improved `dima.h` file with the decentralized architecture added and the modification of the configuration file with a new object added are ready. It should also include the header file where the DTC is defined. Then the structure should be declared and initialized. At last variables in VISION are updated. Another .bat file is run to compile all the files in EEC3 system. Figure 4-8 is the debug system in hardware view.
Figure 4-8 Hardware connection

By VISION, all the status variables in test_str can be seen so that these status bits can be checked in every step. By XCOM and SDP3, the EEC3 hardware can be recognized through VCI and the active and passive DTC can be seen through sending the KWP specific requests. Additionally, through SDP3, the detailed information about the DTC, such as the connected DECs, the components and the expressions about it, can be displayed.

4.4.4 Discussion

In general, in this thesis, from the design point of view, three main goals are realized for the decentralized architecture.

The first one is that in order to achieve the good FTC performance, a new event driven module is constructed, which can be executed locally in each module according to different requirements. Discarding the original centralized DIMA, the decentralized architecture is running locally in each application software module. The other aspect that contributing to the good FTC performance lies in the decentralized action handler that executed directly after the Test filter in each module. In this sense, a fast and guaranteed response and action can be obtained or performed as soon as the DTC is validated or a fault has been confirmed without waiting for collecting all the action information about the DTCs in the centralized DIMA.

The second goal is the reduced complexity. The specific relationship between diagnostic test and the DTC is established by us based on the previous centralized calibration file. In this sense, the work load or the burden of DIMA can be released and the working process dealing with fault can be more clear than before.

The third objective is that, in order to provide easiness and convenience for maintaining the calibration file, decentralized calibration files are scattered locally in each application module. Compared to the centralized calibration file, the decentralized calibration file is much shorter and clear. Only to meet the requirements of the new module, creation of a new data structure named TEST_STR, including all the information of the diagnostic test and the resulting DTC, has been built.

From the application point of view, despite the fact that in this thesis, only two typical cases are illustrated, the rest application modules are very easy to be implemented with this decentralized pattern. Moreover, the new DIMA can run in parallel with the centralized DIMA and they will not disturb each other. In the future, decentralized DIMA can be
extended to multi-functional and the comparison can also be conducted between the two architectures.
Chapter 5 Service based FTC on SCR system

In this chapter, a decentralized service based architecture and modeling of FTC system is presented. In FTC system, both the hardware and the software are considered as service providers and inside the system, the only communication and connection between the service components are based on fault propagation. The information about the fault from each service component is abstracted so that the entire framework is purely fault-oriented. Actually, the objective of the service based architecture is aiming to build a service communication framework used for reconfiguration. It means that there is a fault emerging in the system and the system needs to reconfigure itself with the degraded performance. The advantage of the service view architecture lies in the fact that it facilitates a design of the fault handling mechanism with noncyclic dependencies within the system. FTCS is the system that some faults will happen resulting in uncertain effects. In this sense, Bayesian Networks can be used for modeling since it is a very good method for the systems containing uncertainties.

This chapter mainly includes two parts, denoted the basic concepts introduction on service and the modeling as well as implementation prototype of SCR system based on the decentralized service view architecture.

5.1 Theory of Service based FTC

5.1.1 Introduction

For decentralized architecture, system has been divided into several hardware and software sub-modules. Each sub-module (component) can be considered as a service provider. Service provider only abstracts the information related to failure. So the dependency between service providers is only concerned with how fault propagates in the system. In this way, a purely failure-oriented view of the system can be obtained. Thus, this perspective is used as the foundation for modeling the FTC system.

5.1.2 Basic Concept

In general, the ECU for Fault Tolerant Control in this thesis is one of the nodes in Communication Network of mechatronic system. Inside the system, the SW and HW are taken as components or modules. These modules within the same ECU can communicate with each other in the scope of the communication network. Elementary concepts are introduced in this section. The entire idea of this new architecture is indexed from the paper of Mattias [18].

5.1.2.1 Service View

From the functionality point of view, each module has been developed to deliver one or more kinds of services, which is considered as the purpose or objective of the component. These modules are called service provider. The service from service provider is used by some other modules, which are called customer. Vice versa, the service provider also utilize the services
from other modules, which are called service supplier. In this sense, the basic service dependency can be concluded as the following figure.

![Figure 5-1 Service Overview](image)

From Figure above, the gray arrow represents the real service, while the black arrow is the real signal. It can be clearly found that the service flow does not have the same direction as the signal flow. Because the service supplier has to use the signal from its customer as the reference signal to control its own service, but the service flow direction depends mainly on the fault propagation in the system.

Additionally, there is no cycle in the service dependency graph, which is a merit of the service based architecture. With the size increasing of the scalable mechatronic system, it is very complicated to control and analyze with much dependencies or direct cycles between modules. Service based architecture simplifies the complexity of the entire FTC system.

In general, there are three main steps to build up the whole network based on the service view. The first one is to decide what kind of service that the module should provide concerning about the fault. This phase is mainly determined by the system engineers. The next step is to build up the dependency or connection between different modules, that is to decide what is customer and supplier. The last step is focused on the service communication between each other so that reconfiguration can be achieved based on the results of suppliers. If the node in the entire communication network is built up with service architecture, Bayesian network of FTC can be constructed for modeling system with uncertainties.

### 5.1.2.2 Service Status

The concept of service status is used to judge whether the service from the service provider is available or not. In this thesis, there are three types of service statuses, namely NOM, DIST and UNA. NOM represents that the service is always ready to be used. DIST means that this kind of service may be disturbed by some other modules. Under some circumstances, DIST
can be extended to several levels according to the different levels of disturbed. If the service is not available, UNA status are used in this case.

### 5.1.2.3 Scope

Each service provider should only use the information of its own scope. The scope should include the signals from customers, signals of the module itself, service statuses from its suppliers, and the knowledge of hardware and the electronics.

### 5.1.2.3 Service Status Estimation

The customer should have a clear view of its suppliers’ service statuses. In this sense, the service provider should not only be able to provide the service, but also has the responsibility for monitoring its own service status so that the customer can be notified in time if the input service is not available and perform the corresponding reconfiguration or adaption.

As mentioned above, the service provider should only be allowed to use the information of its own scope. So when the provider estimates its own service status, there can be less precision compared to using all the information throughout the whole system. We sacrifice this kind of accuracy in order to obtain less dependency among modules.

The estimation of service provider \( m \) is denoted as \( \hat{S}_{m|m} \). The true service status is denoted as \( S_{m} \). In real situation, \( \hat{S}_{m|m} \neq S_{m} \), that is because the limited scope of information to estimate is used. The two statuses have different domain.

As long as there is evidence indicating the service is not NOM, the estimated service status will change to DIST or UNA. Otherwise, the estimated status is considered as NOM.

Since the HW cannot communicate its service status with other modules, in our thesis, we assume that the HW module always has its service status as NOM.

### 5.1.3 General Principle of Service Provider

#### 5.1.3.1 Reconfiguration and Variant

Even though the service may be DIST or UNA, the customer tries to keep its own service performed to a degraded level. This kind of action is called **Reconfiguration**. Reconfiguration is performed based on **variant**. A service provider may exist in one or more variants, each of which individually can deliver the same service, but with different accuracy or service quality.

The variants use different and possibly overlapping sets of suppliers and this is how the fault tolerance of the system is archived by selecting the most proper variant to be run. In this sense, the chosen variant \( m:i \) status \( \hat{S}_{m|i} \leftrightarrow S_{m} \).

Taken the following figure as an example to illustrate how variant is classified. If a service provider has two service suppliers, denoted Supplier 1 and Supplier 2. On the other side, it has one service customer. In total there are three variants inside the service provider. Variant 1
only uses the service given by Supplier 1, Variant 2 uses both suppliers’ services, while Variant 3 only uses the service of Supplier 2. In can be clearly concluded that Variant 2 can provide the best performance of this service provider because it possess the entire service of its suppliers and it can provide the complete functionality. However, the other two variants only adopt part of the whole service and the precision and the quality of the service is reduced compared to Variant 2.

![Service Customer](image)

**Figure 5-2 Variant**

### 5.1.3.2 Control theory

According to design objectives in engineering, variant can be classified with different methods. In this thesis, we divide the variants based on the open-loop and closed-loop control theory. Control system can be broadly classified into two types, denoted the open-loop control, whose input does not rely on the output, and the closed-loop control, whose input depends on its output. In control system, the control signal u(t) the output of another new additional component added to the system under control, which is called controller or regulator. On the other side, the controller is triggered by an external signal r(t) named reference or command. The reference signal specifies the normal performance of output. The ultimate objective of control system is to design a proper controller so that the output follows the reference signal as close as possible.

For open-loop control, the controller is only excited by reference signal and the output is the result of the input signal. Open-loop control system does not have the feedback to determine whether its input has reached the goal. In other word, the system does not observe the output of the control process. So open-loop control system cannot compensate for disturbances in the system.
However, in closed-loop control, the controller is influenced by both the reference signal and the output. It is also called feedback control system which uses a function of prescribed relationship between the output and the reference input to control the system. Often the difference between the output of the system under control and the reference signal is amplified and used to control the system so that the difference is continually reduced.

Based on the control theory above, variants can be divided as the open-loop variant using a default value and the closed-loop variant using feedback.

### 5.1.3.3 Selector

Selector is essentially the mechanism for choosing the variant to be used for reconfiguration. The selection is based on the mapping from service status of variant to the selection of variant which is realized with the preference relation among the variants defined by designers of the system. In order to keep the performance as good as possible, the selector will select the variant who has the best performance.

### 5.1.3.4 Principle of Service Status Estimation

Service status estimation can be seen as the mapping from each combination of estimated supplier service statuses and the relative Diagnostic test results to the service status of the variant. In general, there are two main approaches of mapping. One is the lookup table and the other one is the model based way in which the service status is considered as the combination of diagnostic test as well as the estimated service status from the service supplier.

**Without Diagnostic Test**

Each variant has a default service status which is considered as the most possible status to be delivered and in the thesis, it is NOM. However, situation will change with the status to be DIST or UNA.
The general principle are as following:

(1) If all suppliers deliver the estimated service status to be NOM, then the final estimated service status of the variant is the default service status;

(2) If any supplier communicates service status UNA, the estimated service status of the variant is UNA;

(3) If any supplier communicates the service status DIST, then the estimated service status of the variant may be NOM, DIST, UNA. The realistic result depends on the design of the SW engineers.

**With Diagnostic Test**

Service supplier can only communicate the service status estimated by themselves, and it will be better to use the diagnostic test to estimate. Moreover diagnostic test can only use the information within the service provider, which means the signals used by a test must be within the scope.

Even though the communicated service status is NOM, if the diagnostic test implies that it is DIST or UNA, the final estimation may change but also depends on the engineer’s decision according to different situations.

In conclusion, the service status estimation uses as inputs are the communicated service status from supplier and the diagnostic results.

**5.1.4 Diagnostic Modeling**

A diagnostic model is a model that includes all relevant faults and the symptom they cause in the system. By the diagnostic modeling, detailed analysis can be performed so that investigation such as what is the actual fault, what is the cause of this fault and what is the probability of the specific component, can be performed. Combination of the nodes or modules dealing with service and the detailed analysis about probability makes the actual Bayesian Networks.

**5.1.4.1 Algorithm**

1. For each diagnostic test, run the test within the module if the communicated service status from all its suppliers are not UNA.

2. With the diagnostic test results as inputs, run the diagnoser to obtain $\tilde{S}_{m|\text{tests}}$ for each supplier.

3. For each supplier, let $\tilde{S}_{n,m} = \min(\tilde{S}_{n|m}, \tilde{S}_{m|\text{tests}})$, the worst case will be used.
4. For each variant \(m : i\), use the estimated statuses \(\hat{S}_{n_{jm}}\) of the suppliers together with the principle before to obtain estimated service status of the variant \(\hat{S}_{m:jm}\).

5. If any variant has status NOM, run the most preferred one having status NOM. Else, if any variant has status DIST, run the most preferred one having status DIST.

6. If any variant \(m : i\) has been run, set the estimated service status of the service provider equal to the status of that variant, i.e. let \(\hat{S}_{n_{jm}} = \hat{S}_{m:jm}\). Else, set the estimated service status of the service provider equal to UNA, i.e. \(\hat{S}_{n_{jm}} = UNA\).

5.1.5 Bayesian Network

Bayesian Networks (BN) is one of the members of the probabilistic graphical model family. This kind of architecture is the abstraction of uncertain elements. Each node in the network is a random variable and the connection between nodes represents the probability dependency based on the specified random variable. There exists the conditional probability distribution for each node and the distribution will have effect on the other linked nodes [19].

As for FTCS, the nodes represent ECU and the connection between nodes is based on faults propagation. Each node in the network also has a probability table that shows the probability that fault happening in one or more of the suppliers will reduce the service quality of that customer, which can be seen as the measurement of how strong each of the service dependencies are. Apart from containing nodes that represent all of the service providers of the system, we also add nodes that represent all the diagnostic tests, estimated service statuses and selectors.

5.2 Implementation of SCR system

5.2.1 SCR system introduction

In SCANIA, SCR is an active emission control technique where Adblue is sprayed into the exhaust flow in order to reduce NOx emissions. In this thesis, we take the example of the SCR system of EEC3 to illustrate how service based FTC system can be constructed. An overview of the SCR system is shown in Figure 5-5.
Figure 5-5 SCR system overview. The green arrows are flow of Adblue and the dotted arrows are sensor signals.

The functionality of a Urea Dosing System (UDS) is to supply urea under a certain pressure to a dosing unit and inject the requested amount, which is important to be performed to get a good spray NOx reduction effect. In order to control the pressure, a sensor is fixed in the dosing unit and the pump PWM (Pulse Width Modulation), which is used to modify the pump speed, is controlled. Moreover a pump speed sensor is used to sense the actual speed. The related diagnostic tests detecting faults in electrical, functionality and other terms are also implemented in SCR system [20].

From figure above, it can be seen the critical unit is the pump. In SCR software, there is a module named PUMC, which aims to do pump control according to the sensors, the injector condition and the pump real-time working condition. In this sense, detail analysis on original PUMC module source code has been done and the I/Os of PUMC are displayed in Figure 5-6.

Figure 5-6 I/O of PUMC (Appendix)

The left side part of Figure 5-6 is mainly about the modules of central control state machine and based on these different states or conditions, PUMC will have several working modes, which leads to different working results or accuracy. The bottom part is a collection of the
diagnostic modules that are responsible for different detection for the components involved. Worthy of mention is that there are two inputs of PUMC itself, and they are responsible for the feedback and the reading data from the memory, which is used for the closed-loop control of PUMC. The top of the figure are the outputs of PUMC module. According to the information gathered from the original architecture, dependencies have been abstracted and the GeINe model in Figure 5-7 has been constructed. In each node, the service it provides is displayed in a sub-picture.

Figure 5-7 GeINe Tool modeled part of SCR system. The array can be seen as the service.

DOSC is the final customer and PUMC is considered as the service provider. PRES and PUMP serve as the service suppliers to PUMC. PRES and PUMP are both software components. CAAM and some other modules are all state control modules. It is worth noting that CAAM is final collection of the error information and all the diagnostic tests will take the actions through CAAM so that PUMC will turn to different working modes to give different levels of performances.

5.2.2 SCR system modeling

5.2.2.1 Module abstraction

Since decentralized service based FTC system aims to build a system handling the fault and the system reconfiguration according to diagnosis results, the modules relating to fault should be abstracted. From the fault propagation perspective, those control state machine modules should be discarded in use case, because they all do not contain the fault inside and the only thing they perform is to deliver control states. In addition, due to the exact information about different faults, states are created and classified. In this sense, some related modules are out of scope. Additionally, some modules are only used for off-board diagnostics, so they can be also out of our consideration. The real working conditions or the original states leading to different working modes should be only based on the service status of the service suppliers. The other point is that we assume that hardware components are always in normal condition and it will not pass faults to the system since hardware cannot communication like the
software. In terms of the diagnostic test modules, they should be re-located inside the related service modules according to their detection objectives so that decentralized architecture can be constructed. As for the CAAM module, it should also be eliminated since it is a centralized module aiming to collect the action information. However, the goal of this thesis is to build a decentralized framework. Based on the explanation above, the simplified architecture is constructed as in Figure 5-8.

![Figure 5-8 Abstracted modules](image)

5.2.2.2 Module analysis

1. **DOSC**

**Service Description:** This module provides the Adblue injection according to the requests in EMS if possible, otherwise inject as close to the request as possible. Actually it is the controller of the injector.

**Service supplier:** PUMC module

**Tests:** None

**Variant:** One

The only customer to DOSC is a module in the after treatment manager in the EMS. This module is the controller aiming to make decision of the amount of Adblue that should be injected with the help of the EEC3. It focuses on taking control of the Adblue injection to maintain the stoichiometric ratio of NH3 to NOx to reduce the final exhaust emissions. According to the source code of DOSC module, there is no diagnostic tests in it. Moreover, there is only one variant about it.

2. **PUMC**

**Service Description:** PUMC is the controller for the Adblue pump by the PWM signal. The objective of the controller is to keep the pump pressure constant at the level required by the current dosage. To do this it uses feedback in the form of the current Adblue pressure
delivered from a pressure sensor built into the Adblue injector of DOSC. It also gets information about the current dosage amounts from the DOSC module.

**Service:** Providing the pressure of pump at constant level according to the dosage unit request.

**Service suppliers:** PRES, PUMP

**Tests:** Two

**Variant:** Three

There are three variants for PUMC module, denoted Normal, Limp Home and Stop pump. The variant classification is based on the control theory. In Normal variant, it adopts the closed-loop control, while the limp home variant and the stop pump variant both use the open-loop control using the default value. However, limp home and stop pump depends on different service suppliers.

1. **Variant 1 – Normal**

Normal means that PUMC works under the nominal working mode and it uses the closed-loop control, via a feedback signal, to control the pump unit with a constant pressure level. Actually, there are no diagnostic tests related to this variant since the operation has no faults involved.

2. **Variant 2 - Limp Home**

**What is Limp Home Mode?**

Even though some faults will emerge in the EEC3 system leading some normal functionality to failure, however, it is not reasonable and unrealistic to shut down the entire engine and leave the vehicle off road. Because it is not the final objective of fault tolerant control system. EEC3 has been equipped with the feature of fault tolerance, that is the degradation of the system when a certain amount of faults happening in part of the components. From the fault tolerance perspective, degradation of the system means keeping part of the functionality or the degraded work still performing rather than the normal full functionality so that the whole truck can still work partly instead of completely shutting down. In this sense, in SCR system, there is the main state machine, responsible for the controlling the system in different working conditions, among which Limp Home mode is the one handling the degradation state of the system when some faults appear. In conclusion, the condition that triggers the main state machine stepping into the limp home mode is the faults appearing in the system. Moreover, in the specific SCR system, these faults are mainly tend to be the electrical faults.

When there is a fault, there will be the diagnostic test designed responsible for the related checking for this specified fault. Consequently, limp home mode is tightly linked to the diagnostic tests in the SCR system. As long as there is a fault happening in the SCR system or in other word, any DTC is validated or active, it will inform the main state machine so that the working condition turns to the limp home mode. On the other hand, this working mode means
that the system need some special handling and the mechanics should pay attention to some of the parts or components in the system.

**Limp Home mode related diagnostic tests**

In total, there are two service suppliers of PUMC, denoted PUMP and PRES. PUMP provides two kind of services, denoted the delivery of the actual pump speed and making the pump rotates with the given speed. PRES can supply the service of actual Adblue pressure in Pa. Based on the input services, PUMC can provides the service that the Adblue pressure at the dosing unit according to the constant bar to DOSC. In the following section, all the diagnostic tests related to limp home mode will be illustrated.

![Diagram](Figure 5-9 Structure of the case study. Purple color is the module related to Limp Home mode.)

Actually there are four DTCs corresponding to the faults to PRES module and they are located in different modules of different layers. From the table (Table 5-1) below, it can be found the detailed information of these diagnostic tests.

<table>
<thead>
<tr>
<th>DEC Number</th>
<th>Diagnostic Test name</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>008B</td>
<td>Pres_a</td>
<td>PRES_A</td>
</tr>
<tr>
<td>008C</td>
<td>Pres_b</td>
<td>PRES_B</td>
</tr>
<tr>
<td>007A</td>
<td>C</td>
<td>Module_C</td>
</tr>
<tr>
<td>0049</td>
<td>D</td>
<td>Module_D</td>
</tr>
<tr>
<td>006E</td>
<td>E</td>
<td>Module_E</td>
</tr>
<tr>
<td>0045</td>
<td>F</td>
<td>Module_F</td>
</tr>
</tbody>
</table>

Table 5-1 Pseudo Diagnostic tests related to Limp Home mode. The bolded DECs are considered in this thesis.

- 66DEC=008B
This diagnostic test is executed in PRES module to check whether the pressure of Injector is too low by using a pressure sensor. PRES module is responsible for calculating the pressure obtained from the sensors. Once it gets the voltage samples from the injector pressure sensor, it will perform some processing to these samples, such as letting them go into the low pass filter to make the samples more easy to handle in the following procedure. Then the specific diagnosis to the pressure sensor will be performed and it mainly focuses on the electrical faults of the pressure sensor. Finally the conversion from the sensor voltage to the physical signal, the pressure in Pa, will be calculated and then to be delivered. The pressure sensor used is [21] and the relation between voltage and pressure can be seen from Figure 5-10.

These diagnosis includes the check for the pressure sensor is SCG, SCB (OL) and the battery of the pressure sensor behaves within the valid voltage range. Diagnostic test 008B is mainly responsible for the diagnosis of the pressure sensor of the injector and whether the voltage of it is too low or not, under the regulated voltage limitation. Because from the pressure sensor part, the pressure is actually measured by the voltage in the real circuit. So the voltage will tell what is the practical situation of the pressure. Since the voltage of the sensor is lower than this value, it can be considered to be connected to ground, that is the SCG fault. In real electrical circuit, the sensor is not directly connected to the battery and the ground. Usually the pull-down resistor (Figure 5-11) will be used. Pull-down resistor is used to ensure that the input to the logic circuits is reasonable if the connected devices are disconnected or with high-impedance. Actually, pull-down resistor will weakly pull down the voltage of the pressure sensor to 0 volt if all the other components in the circuit are inactive. In this sense, the SCG fault can be explained according to Figure 3, if the switch is closed and one of the ends of the pressure sensor will be connected directly to the ground. So the output of pressure will be very low. In this diagnostic test, the limitation of the low voltage is set to some value, which is considered as the lowest pressure.
For the *condition fulfilled* flag, it depends on the battery condition of the pressure sensor. If the battery voltage is within some range (Figure 5-12), the battery is taken as in normal condition. Otherwise, the battery is out of functionality and behaves abnormally, for example, time stamp like T1-T2 and T3-T4 in Figure 2 are indicating that the battery is out of range. Once the battery condition is bad, this diagnostic test is condition fulfilled so that the check validation can be performed.

This diagnostic test is executed in PRES module to check whether the pressure of Injector is too high by using a pressure sensor. It goes the same way as the diagnostic test 008B in the PRES module and it executes exactly after 008B. However, this diagnostic test will check on the opposite to 008B, 008C will focuses on whether the pressure sensor is SCB or OL.

The Similar to the diagnostic test 008B, 008C depends on the condition of the pressure sensor battery. As for *pending* flag, if the filtered voltage value is higher than some value, then the *pending* will be set to true. The same working principle will be used in this circuit, in the realistic circuit, the pressure sensor will use the pull-up resistor (Figure 5-13) to protect the electronic circuit and provides easiness to the implementation of the electrical diagnostic tests.
This diagnostic test is executed in one module to check whether the temperature of Injector is too low by using a temperature sensor. It adopts the same limit as the pressure sensor.

This diagnostic test is executed in one module, responsible for performing the Adblue pressure level diagnostics. 0049 is the diagnostic test that check whether the pressure of Adblue is too high. Compared to the 008C test also checking the pressure is too high in PRES module, the only difference is diagnostic test focus. Another important one is the PRES module. If it is higher than some level of pressure, this fault will be considered as true. Otherwise, the test is not pending.

This diagnostic test is also located in one module like 0049. However, the test is handling the different field. 006E is mainly dealing with the unstable pressure check. Though Adblue itself is not damaged by the freezing and so does the SCR-system, but the crystallization of the liquid makes it impossible to inject the frozen Adblue into the exhausts. To prevent freezing and for thawing of a frozen system, heating is required. That is the module and linked tests are the diagnostic tests that are responsible for the checking for the related components.
This diagnostic test is executed in the one module which is responsible for the diagnostics. This test will be out of the thesis scope and consideration for some technical reasons.

**Limp Home related Diagnostic tests Re-location**

All the related tests are illustrated and since we want to build a decentralized service view architecture, the next step is to figure out how to organize and where to put these tests. Since we want to build a decentralized service view architecture. According to the paper of Mattias, diagnostic tests dealing with each variant should be located in each service provider. Based on this idea, the following figure (Figure 5-15) for Limp Home variant is constructed.

The principle used for the new structure for the limp home mode is listed below:

1) The scope of this thesis about the service based SCR case is limited to DOSC, PUMC, PRES and PUMP module. DOSC is considered as the top customer. PUMC is the service provider to DOSC. PRES and PUMP serve as the service suppliers to PUMC.

2) Any diagnostic test belonging to any service provider will be kept in the original module, such as 008B and 008C, the pressure sensor electrical tests. It worth noting that 007A is the temperature diagnosis. Both the two tests are tightly connected and we put it together within the diagnostic test group with PRES module to form a new module named PRES_mix module.
3) The rest of the diagnostic tests related to limp home mode will be put in the diagnostic test group with PUMC module to form a new module named PUMC_mix.

4) Moving the diagnostic test does not mean that just putting the DIMA check validation function into the new module because each test has its context and the parameters it uses may be come from its original module. The other point is that test are also executed with a certain sequence or loop and broken this kind of time sequence will make the entire system completely mass. Actually moving tests means that keep the original location and context are remained as well. In the new module, including the header file of the original module and it will check the DTC status of these diagnostic tests periodically.

(3) Variant 3 – Stop pump

This variant has the relationship with the PUMP module and there are also diagnostic tests inside this module. This part is related to my partner Xia Zhou’s work.

3. PRES

Service Description: Provide the injector pressure with sensor from voltage to Pa.

Service suppliers: Other software modules and some hardware related to the analog urea pressure unit.

Tests: Three

Variants: Two (One is the closed-loop control based on feedback signals and the other one is the open-loop control using the default value).

4. PUMP

Service Description: Provide the actual speed of pump and make pump rotate under the given speed.

Service suppliers: Other software modules and some hardware related to the pump motor driver.

Tests: Five

Variants: One

Since the focus of this thesis is on the PUMC module, so the details about who are the service suppliers of PRES and PUMP is out of consideration.

5.2.3 Working Flow

Based on the service view FTC architecture from Mattias paper and the practical situations of SCR system of EEC3, the following work flow (Figure 5-16) is abstracted and constructed.
Applying the service concept to the SCR system, PRES and PUMP serve as the service supplier to PUMC. PUMC will respond to the services, denoted the actual pump speed and the actual pressure of injector. According to the service status, on one hand, PUMC will select the variant who has the best service status, that is the reconfiguration part. On the other hand, PUMC will give its own service status passed to the customer DOSC.

Service Supplier

As mentioned above, PRES and PUMP are the two service suppliers of the case study. PRES tries to offer the service of converting the analog sensor voltage to pressure unit Pa of the injector. PUMP aims to give the services of delivering the real pump speed and trying to make the pump rotate under the given speed. There are different diagnostic tests in each supplier.
responsible for different variants. There are also the corresponding look-up table for each supplier. The DTC status of the tests will be the input of the table and according to different focus of these tests, the service status classification will be different. For example, tests in PRES module are concentrated on pressure leading to the limp home variant, and limp home itself is indicated by DIST. Once there is a DTC validated, the estimated service status of PRES will be DIST. Otherwise, PRES is under normal working condition and the estimated service status will be NOM. The same principle goes for PUMP module. Since it is more related to stop pump variant, so there are two statuses of it, denoted UNA and NOM. If any of the tests in PUMP is validated, it means that it will trigger the pump to be stopped. So the estimated service status of PUMP will be UNA. On the other side, it indicates that the pump operates in good condition and the status will be set to NOM. Both the service suppliers are working independently and there is no dependency between each other. After going through the local look-up table of each supplier, the estimated input service status will be sent to the service provider PUMC.

Service Provider

The inputs to PUMC are the estimated service status of suppliers PRES and PUMP. Inside PUMC, it will firstly check whether all of the inputs are NOM or not. If it is true, the diagnostic tests will be executed. Otherwise, the diagnostic tests will be skipped and the inputs will go directly to the look-up table of PUMC (Table 5-2). Actually, the final service status of service supplier is decided by both the estimated status and the results of diagnostic tests in PUMC. Moreover, the worst case will be selected to serve as the input of the look-up table.

<table>
<thead>
<tr>
<th>Supplier name</th>
<th>Supplier status</th>
<th>PUMC’s variant status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Normal (NOM)</td>
</tr>
<tr>
<td>PRES</td>
<td>NOM</td>
<td>√</td>
</tr>
<tr>
<td></td>
<td>DIST</td>
<td>-</td>
</tr>
<tr>
<td>PUMP</td>
<td>NOM</td>
<td>√</td>
</tr>
<tr>
<td></td>
<td>UNA</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 5-2 Look-up table of PUMC

Via the look-up table above, the status of each variant can be fixed. Then the selector (Figure 5-17) will come to power to decide what is the final variant that PUMC should adopt and what the status of this variant exactly is to be passed to the customer DOSC. The general principle that the selector uses is that best case. It means that the variant who has the best service status will be chosen as the final variant to be run as the reconfiguration. From the PUMC look-up table’s perspective, the right side preference will be used. It can be clearly
seen that there are two outputs from selector. One is for the PUMC itself to decide which variant should be run. The other is the variant status that should be the output of PUMC, serving as the input to customer DOSC.

![Selector of PUMC](image)

**Service Customer**

In this case study, DOSC is the final customer that will work based on the service from PUMC. Actually, in real situation, PUMC is not the only service supplier of DOSC. However, what we focus in the thesis is service provider part and the service supplier part. As long as the pattern can be implemented successfully in this case study, it can be extended to all the modules in SCR system so that the Bayesian Networks can be constructed then.

### 5.2.4 Discussion

The ultimate objective of this paper is to build a decentralized architecture for diagnosis and FTC. Based on the decentralized DIMA as the new diagnostic event server, the concept as well as mechanism of service view can be realized, which not only provides the interface for the interaction between diagnoser and reconfiguration, but also constructs the service communication framework among nodes in B.N as well. In the decentralized service based architecture, both the software and the hardware component are viewed as service providers and the dependency between each other is purely relied on the way or direction of service failure or fault passing within the system. In a word, service-oriented architecture is equivalent with the fault-view. If each service node is implemented with the service mechanism for relationship, the B.N. can be used as the modeling method to conduct inference in the entire system with uncertainties.

According to the theory mentioned above, case study on part of the SCR system has been done. The objective modules of service view are abstracted as DOSC, PUMC, PRES and PUMP with only the fault related information abstracted and left. In conclusion, service communication framework results in three main aspects of improvement compared to the centralized architecture. The first one is the FTC performance has been enhanced by the diagnostic tests that have been re-located into the responsible modules so that decentralized
action handler are improved. Since as long as the fault has been proved active, the action can be activated immediately rather than via CAAM, the centralized action manager. The second point is that this pattern reduces complexity of each module. The original PUMC module has a number of I/Os, while the modified PUMC with only fault information abstracted may only has 4 I/Os for input service status and output service status. No matter from the code lines view or from the depth of nested if-else judgment for variant selection, the complexity has been decreased to a large extent. In addition, the modules have lose weight with merely dependency on fault, which makes them much more easier to understand and analyze. The last advantage of this new architecture lies in the facility for the large-scale system. Since the pattern can be adopted by all the other modules in application layer and the reusability in the context of component based software makes great contribution to the software development.
Chapter 6 Conclusions and Future Work

6.1 Conclusions

The purpose of this master thesis is to build a service based completely decentralized Diagnosis and Fault Tolerant Control System. In the new FTC pattern, there is a decentralized diagnostic event server and a service view communication framework. In comparison with the conventional centralized architecture, the new structure is outstanding in terms of performance, simplicity and facility of engineering.

Specifically, our system have achieved a good FTC performance by using an event driven DIMA to eliminate the central processing used in the old architecture, and a decentralized action handler based on service communication framework to get fast and guaranteed response locally.

- Achieve good FTC performance
  1. Event driven DIMA to eliminate the central processing in the old architecture, see [1].
  2. Decentralized action handler based on service communication framework to get fast and guaranteed response locally. Since as long as the fault has been detected, the actions can be triggered rather than the centralized action handler, see section 5.2.3.

- Reduce complexity
  1. Decentralized calibration file scattered in each software module and in addition, the relationship between diagnostic test, DEC and DTC are fixed, see section 4.1.
  2. Dependency between modules is solely based on the fault propagation and service FTCS makes it easier to understand and analyze the entire system. Service based architecture is the actual fault-oriented FTC, which discards all the centralized state machines.
  3. Code size as well as the I/O of each module are reduced to some extent. For example, the original PUMC module has several I/Os, however, the new PUMC with service based architecture has only 4 I/Os related to fault. It provides more easiness for the engineers to understand and analyze the code.
  4. For each module, the condition of selecting different variants for reconfiguration is simplified rather than the original deep nested if-else statements. Since the service based architecture only use the input service status to select the variant rather than several centralized state machines. Details can be seen in section 5.2.3.
Facilitate large-scale engineering

1. Decentralized calibration file needed for accommodation actions facilities the engineers to modify or maintain in the long run. Because the calibration files are in separate modules and instead of the entire centralized calibration file, the responsible engineers only have to take care of several small files. The distribution of responsibility is much clear than before. See section 4.1.

2. Decentralized pattern can be extended to all the other modules and the reusability in the context of component based software.

6.2 Future Work

In order to construct a simple system to validate the decentralized idea, some modules or functionalities unrelated to fault have been intentionally disregarded in this master thesis, however, they are not valuable to be investigated or implemented in the future work. Another point is that, due to the limited time, only some modules of the system have been implemented with the decentralized architecture rather than the entire system. In this sense, some work has been left out for the future work. But this thesis can be a solid base material or reference for the further investigation in future.

Future work lies in three aspects.

- Functionality improvement
  
  Decentralized Dima lacks of the memory area management module, the freeze frame module, enable control module etc.

- Validation and test
  
  The implementation of the master thesis is just based on the software debugging (Eclipse) rather than the test benches or user cases. Further validation and test for each module of decentralized Dima can be conducted in the future work to ensure the correct functionality, which is also the necessary procedure in the software development.

- Architecture enhancement
  
  The goal of the master thesis is to construct a completely decentralized FTC system. However, from the realistic point of view, some part can be reserved as the centralized, such as the main state machine, the communication node to get all the information of the system since it is distributed inside the entire system. Even for the action handler, if two modules share the same action, for example, warning lamp, it would be reasonable to keep it as the centralized part. Based on the analysis above, partly decentralized architecture fits the practical requirement more. But which part should be decentralized or centralized are determined by the situations in the future development.
Comparison

Since our master thesis is the pre-development of the decentralized architecture for FTC, there is not enough statistics to prove the advantages. Moreover, the pattern have been implemented in part of the modules in the system rather than all. So the performance comparison between the centralized architecture and the decentralized one cannot be directly obtained. Another point is that the whole B.N has not been established yet, so the most outstanding feature of this architecture cannot be displayed. However, in the future, if the decentralized pattern has been improved to a certain level, and it can be applied in all the modules in the system, detailed analysis about the performance can be conducted.
Appendix
Bibliography


