

**Fachhochschule Deggendorf
University of Applied Sciences
Fachbereich Maschinenbau**

Siemens VDO Automotive AG

**Testing of safety critical software for engine
management systems**

**by means of Matlab/Simulink function models, Hardware-in-the-loop
simulation, vehicle measurements and test automation**

Diploma thesis

by

Christian Miedl

Advisors:

Prof. Dr. rer. nat. Stefan Schulte, FH Deggendorf
Prof. Dr. Ing. Stefan Götze, FH Deggendorf

Dr. Ing. Johanna Schaffner, Siemens VDO Automotive AG

1 INTRODUCTION	1
2 SOFTWARE AND FUNCTION TESTING	2
2.1 DEFINITIONS	2
2.1.1 DEFINITION OF TESTING	2
2.1.2 DEFINITION OF VERIFICATION AND VALIDATION	2
2.1.3 WHEN TO STOP TESTING	3
2.2 BLACK BOX TESTS	4
2.3 WHITE BOX TESTS	6
2.4 FURTHER TESTING METHODS	7
2.5 HIGH ORDER TESTING – SYSTEM TEST	8
2.6 OPEN LOOP TEST AND CLOSED LOOP TEST	9
2.7 HARDWARE IN THE LOOP	11
2.8 MODEL QUALITY	16
2.9 DETERMINATION OF TEST CASES	16
3 ETC–MONITORING-CONCEPT – OVERVIEW	17
4 IDLE SPEED CONTROLLER AND IDLE SPEED CONTROLLER MONITORING	19
4.1 FUNCTION LEVEL: IDLE SPEED CONTROLLER	19
4.2 MONITORING LEVEL: IDLE SPEED CONTROLLER MONITORING	21
4.2.1 MONITORING OF IDLE SPEED SETPOINT AND ACTIVATION CONDITIONS	21
4.2.2 MONITORING OF THE PD-CONTROLLER	22
4.2.3 MONITORING OF THE I-CONTROLLER	24
5 TEST OF THE IDLE SPEED CONTROLLER MONITORING	26
5.1 BUILDING AND CHECKING OF THE ML/SL MODEL	26
5.2 VERIFICATION OF THE CODED MONITORING FUNCTION ON SYSTEM LEVEL	27

TABLE OF CONTENTS

5.2.1	BLACK BOX TESTS FOR THE IDLE SPEED CONTROLLER MONITORING	27
5.2.1.1	Test cases	27
5.2.1.2	Test oracle	28
5.2.1.3	Test execution	33
5.2.1.4	Black box test results	34
5.2.1.5	Summarized results of the black box tests	68
5.2.2	WHITE BOX TESTS FOR THE IDLE SPEED CONTROLLER MONITORING	69
5.2.2.1	Test cases	70
5.2.2.2	Test oracle	71
5.2.2.3	Test execution	74
5.2.2.4	White box test results	74
5.2.2.5	Summarized results of the white box tests	78
5.2.3	VEHICLE REACTION TESTS	79
5.2.3.1	Test cases	79
5.2.3.2	Test oracle	90
5.2.3.3	Test execution	97
5.2.3.4	Vehicle reaction test results	98
5.2.3.5	Summarized results of the vehicle reaction tests	121
5.3	CREATE AUTOMATED TESTS	122
5.3.1	PREPARATION OF THE AUTOMATED TESTS	122
5.3.2	TESTPLAN FOR THE AUTOMATED TEST EXECUTION	124
5.3.2.1	Activation condition for tests in trailing throttle and part load	125
5.3.3	RESULTS OF THE AUTOMATED TESTS	127
6	SUMMARY	133
7	ACRONYMS	134
6	REFERENCES	134

List of figures

Figure 1 Black box testing	4
Figure 2 Open loop system	9
Figure 3 Closed loop system	10
Figure 4 Software in the loop simulator (accordant to figure 4 [HUL02])	10
Figure 5 LabCar HiL Simulator (according to [ETAS])	11
Figure 6 LabCar ST software parts (according to [ETAS])	13
Figure 7 LabCar Developer (according to [ETAS])	14
Figure 8 LabCar Operator (according to [ETAS])	15
Figure 9 LabCar Teststand (according to [ETAS])	15
Figure 10 ECU-Design.....	17
Figure 11: ISC overview	19
Figure 12: PD-controller	22
Figure 13: I-controller	24
Figure 14 ML/SL model	26
Figure 15 Excel Sheet for creating the automation sequence	122
Figure 16 Teststand sequence file	123
Figure 17 Testplan for the automated test execution.....	124

1 Introduction

Nowadays the automotive exhaust emission regulations are intensified and the expectations of customers concerning ride comfort, driveability, performance and safety increase constantly. It is more and more a challenge for the automotive industry and the engineers to meet these demands.

To observe current and future environmental laws, it is necessary to have modern engines, e.g. the valvetronic engines of BMW and the appropriate combination of all control units in an automotive. The number of control units sometimes goes up to 50 different units. Due to this the complexity of the systems and the control problems rise rapidly. To be able to master these complex systems linked with decreasing development cycles, new methods in software design and application have to be used. However, due to cost pressure the expenses for the use of new methods must not increase. Because of these latest trends, simulation is more and more used during the development of electronic control units (ECUs).

The increasing qualitative and quantitative guidelines (e.g. IEC61505) can be reached only with an optimal testing process in the functional development. The goal is to optimise the test process concerning deepness and expense of verification. This can be done with the automated test in the Hardware-in-the-Loop (HiL) simulation. In a HiL simulation an engine control unit (real hardware) is integrated in a hardware- and simulation-environment. The ECU controls and monitors the engine, which is simulated in the test environment through a software model.

In this thesis testing procedures for function development of ECUs, especially the monitoring of idle speed controller function will be described. The tests describe mainly functional tests. They are done for the function development department. The developer specifications are coded in a special software department. These software engineers have already done software tests to ensure the correct syntax of the coded software.

Next some definitions and basics about testing in general will be given. Moreover it will be described what to test and how to determine test cases. Finally automated HiL will be presented.

2 Software and function testing

2.1 Definitions

2.1.1 Definition of testing

Important to know is what testing is. Testing is a process, to execute a program with the aim to find faults. A good test case is thereby characterised that so far unknown faults can be found with a high probability. [MYE 01]

Therefore testing is not meant to prove that the software is working like it should. A test tries to find so many faults and bugs as possible. A successful test found a new mistake. It is not correct to mark a test as successful without finding a new fault. Although testing is a destructive work, the test-engineer has to be very creative to receive good results.

2.1.2 Definition of verification and validation

The difference between verification and validation is not always clear. Very plausible are the following short definitions. Verification gives an answer to the question: "Do we create the product right?" and validation answers the question: "Do we create the right product?"

Verification ensures the consistency of the results of a phase of the project with the preceding phase. For the verification of a product are only informations from the preceding project phases necessary. The end-user / customer is not involved in the verification process. Basis for a sensible verification is the traceability of all functional and non-functional needs. [ZUS 01]

Validation tests, whether the result of the project really complies with the needs of the customer. It is absolute necessary to integrate the end-user or customer in the validation process. Semi-finished goods are not considered in the validation. Typical

products which are validated are the system analysis, the system specification and the end product (the finished software). [ZUS 01]

2.1.3 When to stop testing

A difficult question in the field of software testing is the question of the end of software tests. In the technical literature no definite answer is given.

One possible ending condition could be that the testing ends after a certain period of time. The test ends on a fixed date. But this condition is rather senseless due to the human's purposeful behaviour. In common a tester will not do his best if he gets a goal like "test n-days". Especially if the ending condition is already defined before the tests are started. Then the aim could be to test the program for this certain period of time but not to find its faults.

Another definition of an ending condition could be "find n-faults". But this goal has also several disadvantages. It is really hard to determine n-faults. If "n-faults" is defined too low, some mistakes will not be found. If it is too high the tester is faced with an unsolvable problem. He will get frustrated and the quality of his tests will drop significantly.

Moreover, it can be assumed, especially if the program is very complex that there is always one more fault. A person who has to test until there are no more faults in the software will get frustrated again because of the unsolvable task. The tester will unintentionally run tests, which are probably not able to find faults.

A rule of thumb is that testing should not be stopped as long as faults are found rather regularly. On the other hand the company will sooner or later run out of business if the marketing of the product is never started. In fact the ending condition will be always a compromise between these definitions.