Optimized Testing Process in Vehicles Using an Augmented Data Logger

Karsten Hünlich
Distributed Systems Engineering GmbH
Esslingen, Germany
karsten.huenlich@distributed-systems.de

Ulrich Bröckl
University of Applied Sciences Karlsruhe
Karlsruhe, Germany
ulrich.broeckl@hs-karlsruhe.de

Daniel Ulmer
IT-Designers GmbH
Esslingen, Germany
daniel.ulmer@it-designers.de

Steffen Wittel
Distributed Systems Engineering GmbH
Esslingen, Germany
steffen.wittel@distributed-systems.de

***Abstract*—The growing amount of electronic components in vehicles requires an increasing communication load between these components and hence an increasing load on the vehicles communication buses. Both aspects entail an increasing work load for the test engineer developing and executing test cases to verify the required system behaviour in the vehicle. This paper considers a way to automate and reduce the workload for in-vehicle testing by augmenting the functionality of current data loggers. The idea is to use the data logger for supporting the testing process for test drivers. The introduced implementation shows a way to verify the test cases' execution on the fly in order to avoid finding erroneously executed test cases at a later point in time. Additionally, the presented implementation seamlessly includes the test environment for in–vehicle testing into the tool chain which is already used on lower integration levels. This allows the test engineer to reuse test cases from the lower integration levels in vehicle tests and to compare the results from test runs on different integration levels. The paper ends with a summary of the feedback collected in a case study with a prototypical implementation.**

Keywords-automotive; data logger; intelligent data logger; test case development; test case monitoring

I. Introduction

Many data loggers in the automotive industry are designed to record the communication between Electronic Control Units (ECUs) [1]. In more advanced systems, the data content of the RAM (Random Access Memory) of the ECUs is additionally recorded [10]. These data loggers become more and more important to the test engineers because the number of the networked ECUs and hence the testing efforts in a vehicle is continuously increasing. From each requirement on vehicle level the test engineers have to derive test cases to ensure that the ECUs in a vehicle are performing the correct action within correct time constraints. To check this in an in-vehicle test it is necessary to record the bus traffic and the data content of the ECUs' RAM while executing a test case manoeuvre with a car. The result of the test is determined by evaluating the recorded data.

The amount of collected data can turn the evaluation of the recorded test case data into a time consuming challenge. The result of the evaluation can be classified as "valid", "failed" or "not valid". In case of a "passed" classification the recorded data show that the System under Test (SuT), e.g., an ECU, exhibited the expected behaviour described by the requirements. The classification "failed" shows a deviation of the measured data from the expected values and hence from the expected test result. The "error" case indicates a significant mistake during the test case execution that makes an evaluation of the recorded data impossible with respect to the test case's definition.

To minimize the error cases, and therefore the time for the test case execution and evaluation, the data logger can be augmented with additional functionality to monitor the correct execution of the test case. The necessary conditions are to be defined by the test engineer. The data logger can be extended with instructions supervising relevant signals. For these signals boundaries may be defined. A test case can, e.g., be successfully accomplished if the signal stays within these boundaries. However, the goal is not to test the driver's behaviour as mentioned in [2]. The goals are to give instructions to the test case executor, which may be a driver, a robot or a test automation tool, and to additionally supervise the execution's correspondence to the conditions predefined by the test engineer. Especially for a human driver being in the light of our experiences, the biggest error source in a vehicle during a test case execution the augmented data logger can help to avoid unnecessary work by immediately indicating erroneous test runs. For example, the augmented data logger can supervise an Antilock Braking System (ABS) manoeuvre where the driver speeds up to 60km/h and does a full brake without turning the steering angle.

Figure 1 shows an example of a system development process according to the V-Model as shown in [3]. In this example, the test on vehicle level is the last level of testing within the integration process. Before this stage many other tests have already taken place on lower integration levels. For efficiency reasons, it would be helpful if the test engineer could reuse test cases developed on lower integration levels, e.g., test cases from Hardware in the Loop (HiL) tests [4]. The reuse of these test cases minimizes the work for the test engineer to adopt the test cases to the desired test platform. The reuse also enables the comparability of the test results with the lower integration levels. For guaranteeing the reusability of the test cases it is essential to specify the test cases platform independently. This format is interpreted by certain testing platforms.

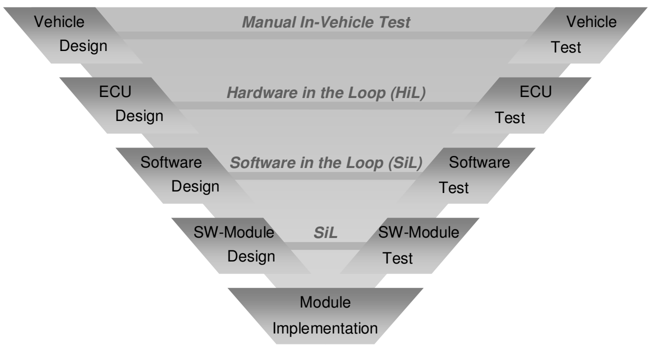

Figure 1. Commonly used application of the V-Model in the automotive industry

This paper describes a solution to reuse test cases from lower integration levels by adding information to guide the driver through the test case and to supervise the actions of the driver on the basis of the test case implementation of the test engineer. It starts with the description of the software and hardware components that extend the data logger to an augmented data logger. The paper ends with a summary of the feedback collected in a case study using a prototypical implementation.

II. State of the Art

Today in-vehicle tests are usually executed without the support of a software tool for giving feedback on the quality of the test execution or a tool that guides the driver through a test case. The test cases are often written in plain human readable text which describes what a tester has to do in the vehicle to fulfill the test case. These textual test cases are stored for example in a database. For taking a set of test cases to the car, they are either printed out or downloaded to a robust handheld computer. In both cases, they are read before or during a driving manoeuvre. The quality of the execution of the manoeuvre thus depends on the skills of the test driver. Details of the execution quality can be determined offline on a parking lot or by evaluating the information on the data logger. Especially if test driver and test engineer are not the same person, this process is error prone and time consuming. Since the test cases are in natural language there is enough room for misunderstandings between a test manager who writes the test cases and a test driver who has to execute the manoeuvre. This fact tends to result in multiple iterations of in vehicle tests of the same test case.

There are several solutions that have the aim to optimize in vehicle tests and to minimize the time overhead. A touch-display can be used in vehicles for check lists. A more advanced system is shown in [12] which comprises of a driver guidance system and a feature to immediately evaluate if the test is passed or failed.

For testing driver assistance functions, manoeuvres have to be executed very precisely by the test driver. That means in a significant number of tests the tests are failed not because the system is not working correctly but the test driver has made a mistake. To minimize this number of invalid tests this paper describes a way to detect deviations of the given test case during the execution. This avoids a usually time consuming evaluation of invalid test cases.

III. Augmented Data Logger

Current data loggers [10] are designed for recording data and neither for interpreting it nor for participating in the measurement process. This section describes a way of augmenting the functionality of the data logger in order to support the testing process and to seamlessly integrate the vehicle tests in the system integration and testing process.

A. Basic Data Logger System Design

A data logger to record digital information in vehicles might be designed in the way described in [5]: i.e., a host computer is connected via a network interface, e.g., Ethernet, to the data logger. Over this connection, the data logger can be controlled and configured. The configuration defines which signals are stored in the data storage and on which bus interface the signals can be received. The host computer is mainly used to start and stop the data logger and to visualize an excerpt of the recorded data on the fly. The data logger hardware is responsible for the real time processing of the data. A commonly found feature might be a trigger that starts a measurement when a predefined condition becomes true as it is described in [6].

For evaluating the trigger conditions the data logger needs information about the connected data buses and the data that is transferred over a particular data bus. Usually, this information is available in form of configuration and signal files that are interpreted by the host computer and transferred to the data logger.

In some parts of a data logger execution in real-time is mandatory. This is necessary because the test engineer needs to know exactly when some data have been transmitted on a particular bus. A common solution is that the communication on a bus system is recorded together with timestamps which indicate the time instance when a message is transferred over a bus [9]. Figure 2 shows the procedure of recording a message from a bus. If the data logger receives a message a timestamp is taken. For the evaluation of the recorded data it is possible to correlate in time the different recordings with the help of the timestamps, which means that the more precisely the timestamp is taken the more precisely the situation can be reproduced and evaluated.

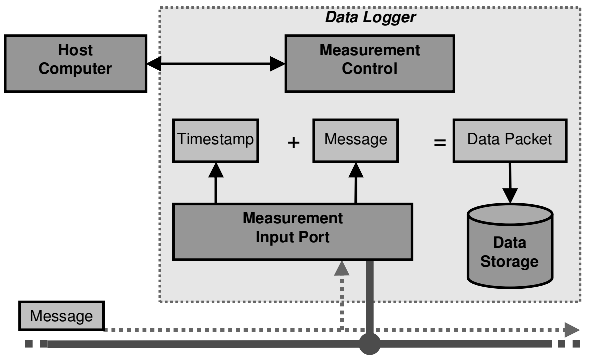

Figure 2. Schematic procedure of measuring a message on the bus

The example in Figure 2 shows a host computer which is connected over a communication interface with the measurement control unit within the data logger. The host computer is commonly a PC or a notebook with an operating system that does not support real time tasks. Via the host computer the engineer has access to and control over the data logger. Additionally, the host computer can access measurement data and visualize them to the user. Evaluating this data while conducting a manoeuvre is almost impossible since, in this case, the driver would have to fully concentrate on the monitor instead on his driving task.

B. Current Testing Process on Vehicle-Level

In the common testing process the test engineer starts looking at the requirements for the SuT. Based on these requirements the test engineer creates the corresponding test cases. How the test engineer writes down these test cases for in-vehicle tests is normally not defined. In some way, the test cases have to be readable by the driver while he is executing the manoeuvre in the vehicle. After finishing writing a test case, the test engineer has to hand over the test case to the driver who executes the manoeuvre specified in the test case in the vehicle. The role of the test engineer and of the driver might be taken by the same person or by different ones. If the test engineer and the driver are different persons who write and execute a test case the test case must be well defined to prevent misunderstanding. If the test case specification is not complete and therefore the driver does not execute the test case as intended by the test engineer, the following work might be unavailing.

After having recorded the data of the manoeuvre that is specified in the test case the driver hands over the recordings to the test engineer. Afterwards the test engineer evaluates the data. Usually this is done manually. The test engineer has to search through a database of signals with probably more than 10,000 entries. If the result of the test case is “passed”, the test case will be documented and closed. In case the result is “failed”, the test engineer has to find the exact reason. The SuT can either have a bug or the test case has not been executed accurately which means that the test is “not valid”. If the test case was executed within all defined constraints by the test engineer the test case is “valid”. Both cases generate lots of work of analyzing and documentation for the test engineer. Especially, the work for the second case can be minimized by finding out the validity of the test case in an earlier stage of the process.

Generally, the biggest drawback of finding invalid test runs late in the process is the time that the test engineer spends on one test case. It must be considered that the number of test cases that must be performed for each major release can be up to several hundred test cases. As a conclusion two main issues can be identified that can be possibly optimized:

- The time for evaluating the test results by avoiding invalid test cases
- The numbers of times moving from the office to the vehicle and to the test track for repeating invalid test cases

The introduced testing process on vehicle level is very different from the test processes on lower integration levels of the development process shown in Figure 1. In the lower levels, i.e., HiL or SiL, a test case is written in a defined way. The test case can be reused and usually returns a reproducible result. Another point is that the test result is directly available after the test has been finished. It can be said that the processes on different levels have mainly five important parts [7]:

- Environment simulation that simulates the environment of the SuT
- Test case execution system
- The System under Test itself
- Measurement and data logging system
- Evaluation system

The evaluation system compares the measured values with the ones that are specified in the test case for the SuT. The test case execution system reads the test case and controls the environment simulation that affects the SuT. In a vehicle, the parts for the test process are different. The test case execution system in a vehicle is the test driver. The test driver has control over the environment of the SuT. The evaluation system in a vehicle test is the test engineer who evaluates the measurements.

The measurement and data logging system might be the same as the one used in the vehicle. For the in-vehicle test, an environment simulation is not necessary because the vehicle is used in a real environment. Sometimes both environments are mixed for the vehicle tests, e.g., foot passengers are simulated with synthetic dolls or imaginary sensor information.

C. New Testing Process Supported by the Augmented Data Logger

To reduce the time for testing and evaluating of in-vehicle testing a new approach for the testing work flow should be considered. The first aspect is the form how the test case is written. A uniform platform independent language (see Section III C. for more detailed information) is used to define the test cases. With this uniform language, the test engineer can precisely describe the test case. The test case is now not only human readable but also machine-readable and can be interpreted by a programme. The additional code extends the abilities of the data logger. The system now knows about the manoeuvre which has to be executed for a particular test case. With the knowledge of how a test case must be performed, driving errors can be detected directly and time can be saved.

The new work flow has a strict separation between the office work and the work in the vehicle. Right after performing a test case, the driver gets a result if the test case was executed accurately. The feedback also includes the information why the test failed. This information depends on the test case description from the test engineer. If the test engineer describes the test case in many details more driving errors and failures can be detected without looking at the whole measured data back in the office. The advantage of this new approach is that the driver:

- Is guided through the test case execution process through a unified notification
- Gets a response directly after the manoeuvre if the test is executed correctly
- Gets the reason why a test case was classified as “not valid”

This reduces the evaluation work and the test case execution work. Since the data logger instructs and checks the manoeuvre it makes the execution more precise.

For this new approach parts of the evaluation system and the test case execution system are moved into the data logger. The schematic of a data logger shown in Figure 2 can be extended to execute additional code given by the test engineer that controls the data logger and guides the test driver through the manoeuvre. Figure 3 shows a simplified version of the extension of a measuring system. The CPU has to fetch the messages from the bus, add a timestamp to each message and extract the relevant signals with its values. The values of the signals are internally updated and provided for the test case code.

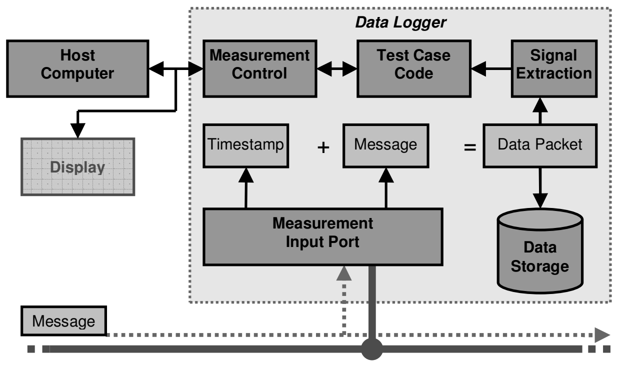

Figure 3. Schematic measuring system extended with the test case code

To control and configure the data logger the test case code needs a connection to the measurement control module. It starts or stops the test case and gets commands from the test case code to control, e.g., the data logging. All information that is known in the data logger can be used inside the test case code. The measurement control module has also the task to route the instructions for the test driver to the host computer or a connected display.

IV. Test Case Implementation and Execution in a Vehicle

In this section, the test case implementation and execution is shown using the following example: The driver starts the engine and accelerates the car to 60km/h. When 60km/h are reached, the driver performs a full braking. In this manoeuvre it is important that the driver keeps the steering wheel straight.

Such a manoeuvre is used, e.g., to measure data of an ABS and to evaluate if it has performed accurately during its intervention. A possible criterion for a "not valid" ABS-test execution is defined by looking at the steering angle. If the data show that the car did not drive straight, the test case has not been executed accurately. The manoeuvre can be described in a state chart manner represented by an XML file [8].

The example in Figure 4 shows the ABS-manoeuvre in XML code. The definition of the XML code is described by Ruf [11] for Hardware in the Loop tests. The test case is composed of states, actions, events and one rule. The rule checks the steering wheel angle for the whole test case. The states are following in chronological order. Each state has one or more actions that have to be performed by the test driver. If the condition of an event is fulfilled the state machine enters the next state.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Testcase xmlns="http://www.it-designers.de/adl"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.it-designers.de/adl
http://www.it-designers.de/adl">

  <Rule SteeringAngle_deg_eqal="0" Tolerance_deg="5"/>

  <State num="1">
    <Action text="Get ready to start the manoeuvre"/>
    <Event wait_seconds="5"/>
  </State>

  <State num="2">
    <Action text="Start the engine"/>
    <Event wait_seconds="5"/>
  </State>

  <State num="3">
    <Action text="Accelerate to 60km/h"/>
    <Event velocity_kmh_equal ="60"/>
  </State>

  <State num="4">
    <Action text="Full braking"/>
    <Event velocity_kmh_equal ="0"/>
  </State>

  <State num="5">
    <Action text="Turn-off engine"/>
    <Event wait_seconds="5"/>
  </State>

  <State num="6">
    <Action text="Manoeuvre finished"/>
    <Event wait_seconds="3"/>
  </State>

</Testcase>
```

Figure 4. Listing o f a test case in XML

The “*wait_seconds*” events are needed to give the test driver time to perform the actions. The data logger is able to interpret the XML code and guide the test driver through the described manoeuvre. To start the manoeuvre in the vehicle the test driver has to start the data logger.

In summary, the test cases that are written for lower integration levels using this XML language can be reused for in vehicle testing.

V. CASE STUDY

A case study has been performed to determine the benefits of the augmented measurement system for test drivers. The case study was conducted with a group of eleven candidates. The group consisted of team leaders, developers and testers. In the first step the content of the executed XML test case was explained to the candidates. With this knowledge the candidates were guided by the augmented measurement system to execute the test case in the role of a test driver.

The vehicle for the case study was equipped with an extra display that is attached to the windscreen. The setup in the vehicle looks similar to an external navigation system attached to the windscreen. In this setup the display shows the instructions and the current state of the running test case. The execution of the test case was done on a locked test track. This assures a save environment and that the candidates are not disturbed by surrounding vehicles.

After executing the test cases multiple times the candidate was interviewed about the experience with the augmented measurement system. The collected feedback is summarized in Figure 5.

Most candidates are confused and distracted by the display driving the test case for the first time. The reason might be that the candidates do not yet intuitively follow the instructions on the display. As soon as the instructions are known to the candidate he can concentrate less on the display and more on his driving task. After a short learning curve the confidence and sureness working with the augmented data logger raised. In summary, 7 candidates are seeing a benefit of such a system to speed up and assist them in their daily work.

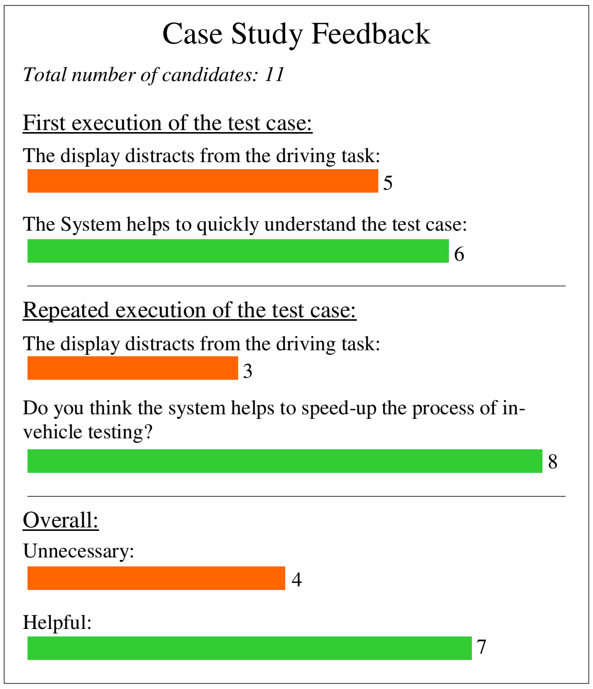

Figure 5. Case Study feedback results

Furthermore, the feedback also includes suggestions for improvements. The four mostly mentioned suggestions were:

- Additional speech output for instructions
- Direct connection to quality and lifecycle management tools
- More detailed information in case of a “not valid” result
- Using LCD glasses instead of a display attached to the windscreen

The feedback of the case study indicates that the augmented data logger helps to speedup the testing process for in-vehicle testing.

VI. CONCLUSION AND FUTURE WORKS

This work shows an approach how the process of in-vehicle testing can be improved. The introduced approach shows a way to reduce the costs for the testing process by reusing test cases from other testing platforms and by optimizing the workflow of in-vehicle testing. A major part in the optimized workflow is the possibility for declaring a test case “valid”.

The extended classification of a test case enables an early feedback about the quality of the executed test case and hence makes sure that only valid test cases are evaluated. In the introduced approach a test case can be classified as "passed", "failed", "valid" and "not valid". The first two classifications are based on the requirements for the SuT while the other two classifications reveal if the test case was executed within defined constraints that are based on additional testing requirements. The test engineer has only to look at the measurements of the test cases that are classified as "valid". This helps to reduce the evaluation time especially if the test case manoeuvre is very complex or time critical.

A tool chain supports the test engineer during the test case development process. He can use test case descriptions from lower integration levels and use them as a basis for the in-vehicle test. The test engineer needs no knowledge in programming languages for implementing and running a test case on the introduced augmented data logger. Several test cases can be downloaded to the data logger and are automatically executed.

While driving a test case the test driver has precise instructions on his current tasks and is guided through the test case manoeuvre. The test driver has immediate feedback if the constraints of the test case added by the test engineer are fulfilled. The augmented data logger observes the execution and the driver gets a response if the manoeuvre is “valid” or if the test driver has made a mistake during the execution. It is then up to the test driver to decide if he wants to immediately repeat the manoeuvre or continue with the next test case.

To further improve the system it is planned to work on the interface between the augmented data logger and the test driver. The visualization of the manoeuvre can be optimized by using intuitive icons, by using speech output or even by an augmented reality display.

Another benefit of the introduced augmented measurement system is to being able to reuse identical test case descriptions in XML on several test platforms. It is left for future work to investigate on comparing the results of e.g. a Hardware-in-the-Loop platform with the results from an in-vehicle test which is guided by the augmented measurement system.

REFERENCES

[1] K. Athanasas. Fast Prototyping Methodology for the Verification of Complex Vehicle Systems. Dissertation, Brunel University, London, UK, March 2005

[2] Petersson, L., Fletcher, L., and Zelinsky, A. 2005, ‘A framework for driver-in-the-loop driver assistance systems’, Intelligent Transportation System Conference 2005: Proceeding of an IEEE International conference, 13. – 15 September 2005, Vienna, Austria, pp. 771 – 776.

[3] Meier, E., February 2008, ‘V-Modelle in Automotive-Projekten’, AUTOMOBIL-ELEKTRONIK Journal, pp. 36 – 37

[4] Schlager, M. 2008, ‘Hardware-in-the-Loop Simulation’, VDM Verlag Dr. Mueller e.K., ISBN-13: 978-3836462167.

[5] Park, J. and Mackay, S. 2003, ‘Practical Data Acquisition for Instrumentation and Control Systems’, An imprint of Elvester, ISBN-10: 075-0657-960.

[6] Koch, M., Theissler, A., September 2007, ‘Mit Tedradis dem Fehler auf der Spur’, Automotive Journal, Carl Hanser Verlag, pp. 28 – 30.

[7] S. Dangel, H. Keller, and D. Ulmer. Wie sag’ ich’s meinem Prüfstand? RD Inside, April/Mai:7, 2010.

[8] B. Ruf, H. Keller, D. Ulmer, and M. Dausmann. Ereignisbasierte Testfallbedatung - ein MINT-Projekt der Daimler AG und der Fakultät Informationstechnik. spektrum 33/2011, pp. 68–70,.

[9] D. Ulmer, A. Theissler, K. Hünlich. PC-Based Measuring and Test System for High-Precision Recording and In-The-Loop-Simulation of Driver Assistance Functions. Embedded World 2010.

[10] Simon McBeath (2002). Competition Car Data Logging: A Practical Handbook. J. H. Haynes & Co.. ISBN 1-85960-653-9.

[11] B. Ruf, H. Keller, D. Ulmer, M. Dausmann. Ereignisbasierte Testfallbedatung, Spektrum 33/2011 pp. 67 – 68.

[12] mm-lab, Driver guidance system, Automotive Testing Technology International, September 2009, page 89.