RESEARCHERID
THOMSON REUTERS

[ICRTCET-2018]

ISSN 2348 – 8034
Impact Factor- 5.070

GLOBAL JOURNAL OF ENGINEERING SCIENCE AND RESEARCHES

FUNCTIONALITY SCHEDULING VERIFIER SUPPORTED AUTOMATED TESTING OF ECU

H. R. Sukhesh[*1], Sathiya R.[2], Archana Prasanthi R.[3] & Senthil Vadivu U.[4]

[*1]P.G. Student of Electrical and Electronics Engineering, People Education Society University, Bangalore, Karnataka, India

[2]Faculty of Electrionics and Communication Engineering, SRM Institute of Science and Technology, Chennai, Tamil Nadu, India

[3]Faculty of Electrionics and Communication Engineering, SRM Institute of Science and Technology, Chennai, Tamil Nadu, India

[4]Faculty of Electrical and Electronics Engineering, People Education Society University, Bangalore, Karnataka, India

ABSTRACT

The incorporation of automated testing methods are turning out to be the exemplary trusted elucidation in the discipline of automotive industry. This paper chronicles a novel approach of testing an Electronic Control Unit (ECU) instrumented in the modern automotive, by the effective utilization of the dynamic features of NI-Lab VIEW tool, aiming at superiorizing the level of automation embraced in the testing. The method reported in this paper featuring to its reduced computational effort and low complexity, proposes an expedient modus operandi for the automated testing of ECU. In addition to this, the Functionality Scheduling Verifier Tool has been developed, for upholding the accurate schedulability. Simulation results authenticate the effectiveness of the posited methodology.

Keywords: *Electronic Control Unit; Automated testing; Functionality Scheduling Verification; NI-LabVIEW*

I. INTRODUCTION

The requirement of vigilant and rigorous testing of ECU in every stages of development is at the peak, as the ECU is fabricated, targeting many crucial, complex and critical tasks. The detection of errors and processing its corrective actions at the early stages of development are very economical with less complexity than those detected in the post stages. Further, it is important for an ECU test environment to be closely resembling to a real vehicle. Addressing this purpose, NI -LabVIEW tool supported with few proprietary applications fulfils the needs optimally.

The ongoing explorations have fascinated on the repudiation of manual testing, without atrophying the efficacious dynamic performance of the ECU. The main advantages of automated testing of ECU are: faster test execution, appreciably high accuracy and reusability of tests [1].

Shruthi T. S. and K. H. Naz Mufeeda presented a work on a topic named “Using VT System for Automated Testing of ECU”, which clearly mentioned that the use and essentiality of the automated testing of ECUs applicable for headlamp, adapted in the modern cars. This work reported the use of Vector Tool System for the simulation of loads and sensors. The usage of VT System proposed is to attenuate the human factor influence, but with the desideratum of accrued memory space [1].

The work managed by J. Novák and P. Kocourek’s, suggests the level of complex standards involved with Controller Area Network (CAN) protocols. It also describes the necessity of meticulous testing of ECU response, at all the protocols layers of communication stack [2].

Each new generation of vehicles inaugurates more number of electronic and electrical equipment. To corroborate the targeted functionalities economically, T. Bardelang, in his work, proposes a novel idea of distributed system to

furnish the communication trajectory, amid of the electronic subsystems. An exemplar revealed in his research depicts the usage of 32 ECUs backed with 12 CAN networks in one truck [3].

Karl Koscher, A. Czeskis, Shwetak Patel, F. Roesner and T. Kohno from the Washington University worked on "Experimental Security Analysis of a Modern Automobile", which exhibits the evaluation methods of critical issues on a present automotive system, such as the safety and efficiency. They also briefs about the complex challenges involved in catering these crucial vulnerabilities [4].

Viacheslav Izosimov's work on the "Analysis and Standardization of Truck Architectures", aims at proposing a common electronic platform by analyzing diverse truck architectures of three leading truck pioneers, namely the Volvo Trucks, Daimler Chrysler, and Scania, targeting at significantly reducing the total net costs of electronic part and also to benefit the third-party vendors, by enlarge the market of components and services [5].

Larses O. from Royal Institute of Technology presented a work on "Dependable Architectures for Automotive Electronics – Philosophy, Theory and Practice" for his Licentiate Thesis. This work clearly mentioned the capability of design based on a base platform, which focuses at filling the hitch of providing good engineering trade-off in between the production costs and the product originality. According to his ideology, the developer should systematize and consolidate the main system parts along with main functions of different types of automobile. It ultimately results in a unified platform, which could be used as a base platform for several products [6].

Further, a detailed study is done on the latest released truck architectures developed individually by the Volvo Group of Truck Technology [7], Daimler Chrysler [8], and Scania [9], focusing on the features of common properties of these architectures, their benefits and drawbacks along with donating a special concern on the dependability characteristics.

A new architecture for automotive ECUs has been proposed by B. Poudel and A. Muni from University of Nevada, which describes the incorporation of dependability and security primitives intended with imperceptible performance, energy, and resources overhead. The work utilizes Xilinx Automotive Spartan-6 FPGA, for implementing and demonstrating the proposed ECU architecture and it proves the effectiveness on steer-by-wire application over CAN with flexible data rate [10].

In addition to these, few conventional automation solutions with respect to ECU were still controversial due to inefficient user friendliness, over consumption of memory as well as less adaptability with the revisions over time. In order to cater this issue, few CAN communication protocol standards, AUTOSAR standards, proprietary automation standards and other literaturesare also referred[11]-[21].

In this proposed work, a Functionality Scheduling Verifier tool has been developed initially, which targets at conforming the schedulability of all the tasks associated for testing an ECU. Further, it clearly elaboratesan innovative automation methodologyfor testing an ECU, by the effective utilization of the NI-LabVIEW tool. The simulation results, evincing its effectiveness, are proffered, supporting the novel method.

II. FUNCTIONALITY SCHEDULING VERIFIER ALGORITHM

The tasks performed in any vehicular ECU can be categorized into two types:

i. Soft deadline periodic task:

This kind of tasks will not result in any fatal, even if it misses the deadline. It is preferable to use the imprecise computing methods for analyzing these tasks. The task is defined with the time constraint as (*l*, *L*).For *IJi* tasks in *Li* consequential jobs, if *li* jobs are categorized as mandatory and if all these *li* jobs accomplish the execution before the deadlines, then it can be inferred that *IJi* is satisfying the firm constraint (*li*, *Li*).

ISSN 2348 – 8034
Impact Factor- 5.070

For the detailed analysis, let us consider the following definitions:

IJi Periodic tasks
Li Consequential job set
Mi Mandatory job set
Oi Optional job set
ci Execution time
di Deadline
Ti Period
J(i,j) Job *j* of *IJi*

For *IJi* *Ti, ci, di* , considering to the equation (1), for a task *IJi*,

Li [(*J* – *1*) *li* / *Li*] / *li* + 1 (1)

- If *j* successfully satisfies the equation (1), then it is conformed that, *J(i,j) Mi*, i.e. *J(i,j)* must finish the execution within the deadline *di*. Hence *J(i,j)* is categorized under hard real time task.
- If *j*fails to satisfy equation (1), then *J(i,j) Oi*. Hence *J(i,j)* comes under the category of optional task.

ii. Soft deadline aperiodic task:

For this type of tasks, average response time is of at most interest.For the further analysis, let us consider the following additional definitions:

IJi Aperiodic tasks

Release time of job *J(i,j)* Finish time

Response time of *J(i,j) = f(i,j) - r(i,j)* Average response time Maximum average response time

Hence the Average response time of aperiodic task is evaluated by the equation (2)

Ai σ *(i,j)*] (2)

For aperiodic tasks, as long as *Ai di*, there is no threat of fatality.

III. IMPLEMENTATION OF FUNCTIONALITY SCHEDULING VERIFIER TOOL

Using the above specified equations, a Python GUI Application has been developed, which demands the human intervention, for selecting the mandatory tasks by checking the respective functions, using the check box widget. It has been designed and programmed such that, the order of checking the check box describes the priority of the mandatory tasks. Adding to this, the unchecked widgets are considered as the optional tasks among the selected functionality.

ISSN 2348 – 8034
Impact Factor- 5.070

Fig. 1. Successful Scheduling.

For a particular functionality of interest, if the 'Vehicle Speed', 'Engine Speed', 'Gear Status' and 'Wheel Conditions' are selected as the mandatory tasks, in a specified order of priority, as shown in Fig. 1, and further processed by clicked 'OK', then the schedule verifying equations are executed internally and a message box widget will appear with a text 'SUCCESSFUL', if all the mandatory tasks can be finished before deadline.

Fig. 2. Failed Scheduling.

Similarly, for the same above functionality of interest, if the 'Temperature', 'Door status', 'Wheel Conditions', 'Audio' and 'Seat belt status' are selected as mandatory tasks, as shown in Fig. 2, then after process, a message box widget will appear with a text 'FAILED', if all the mandatory tasks, as selected by the user, cannot be finished within the deadline.

IV. NI-LABVIEW BASED ECU AUTOMATED TESTING RESULTS

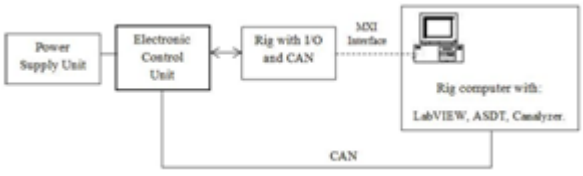

Fig. 3. Test rig setup.

ISSN 2348 – 8034
Impact Factor- 5.070

The Fig. 3 is pictorially describing the test rig setup required for the testing of ECU. The ECU is energized by a power supply unit with +24V/12V. It is further connected with the Rig, supporting the exchange of I/O signals. The Multisystem eXtension Interface (MXI), which is designed for high speed communication between the devices, enables the exchange of data between the Rig and Rig computer. In order to read/write the parameters associated with ECU from the Rig computer, CAN bus is used to serve the purpose.

The automation of test cases for any required functionality can be successfully simulated using a tool named as NI-LabVIEW [19]. The total simulation can be generally sub divided into Front panel design and back end design.

Fig. 4. Front Setup Panel.

The front panel serves as a user interface. It involves the controls, indicators which enables the dynamic interactions with the user, as depicted in the Fig. 4.The setup is so innovatively programed that, once the user successfully proves his authentication, based on the functionality name being tested, all the necessary files automatically get loaded in the respective entry widgets. The setup panel is supported with Trace and Log panels, which help in keeping the track of the execution and the status of testing respectively. Later the user can start the testing by using the necessary buttons.

The back end design involves the following stages, about which a user is unknown of, is pictorially described using the pseudo code and are further discussed:

- Pre initialization
- Load Program
- Load Parameter
- Preparations and Steps

Pre initialization

ISSN 2348 – 8034
Impact Factor- 5.070

```
Start ( )
// PRE INITIALIZATION
Setup in ( )
if (User name & Password matched)?
{
        Initialize Trace and Log Panels
        Initialize the Ignition
        if (Actions getting  traced)?
        {
                Ignition on ( )
                Display the action in Trace Panel
                Select the CAN Frame
        }
        else
        {
                exit (ERROR)
        }
}
else
{
        exit (ERROR)
}
// LOAD PROGRAM
```

Fig. 5. Pre Initialization.

The pseudo code represented in the Fig. 5 describes the Pre Initialization process, where the execution flows through the following stages:

i. The processor checks for the authentication by matching User name and Password.
ii. If the authentication is proved positively, the Trace and Log Panels and the Ignition is initialized.
iii. Further, the processor makes sure that all the actions are getting traced in the Trace Panel.
iv. Set the ignition ON for 1000ms and check for its trace in the Trace Panel.
v. Based on the signals of interest, the CAN frame has to be selected.
vi. Next to this, the processor gives the control to Load Program stage.

Load Program:

ISSN 2348 – 8034
Impact Factor- 5.070

```
// LOAD PROGRAM
Setup in ( )
if (Software download enabled)?
{
        Turn off the CAN communication
        Load the program to ECU
        Ignition on ( )
        Display the action in Trace Panel
        Ignition off ( )
        Display the action in Trace Panel
}
else
{
        exit (ERROR)
}
// LOAD PARAMETERS
```

Fig. 6. Load Program.

The pseudo code depicted in the Fig. 6 explains the Load Program process, where the execution flows is as follows:

i. The processor checks the positive result for software download enablement.
ii. With the successful response, the CAN communication bus is turned OFF to avoid unnecessary data exchange which may be ambiguous.
iii. Further, load the necessary programs into ECU.
iv. In order to store the program in the ECU memory, the ignition is turned ON and OFF with a gap of 1000ms in between.
v. The processor makes sure that these actions are traced in the Trace Panel.
vi. Next to this, the processor gives the control to Load Parameter stage.

Load Parameter:

```
// LOAD PARAMETERS
Register all the Parameters
for (Count = Parameter Count; Count ≤ Total Parameters; Count++)
{
        Set initial values to all the parameters
        Store them in the memory
}
// PREPARATIONS
```

Fig. 7. Load Parameter.

The pseudo code shown in the Fig. 7 details the Load Parameter process, where the execution drives through the following stages:

ISSN 2348 – 8034
Impact Factor- 5.070

i. Initially, all the necessary parameters are registered and the count of total number of parameters available is stored into a variable named Total_Par.
ii. In order to make sure that all the registered parameters are initialized, the index count of each parameter is monitored using a variable Par_Count.
iii. As long as the Par_Count is less than or equal to Total_Par, the initial values for each corresponding parameters are set sequentially.
iv. Once the condition specified in (iii) fails, then the execution moves into the next stage called the Preparations process.

Preparations and Steps

```
// PREPARATIONS
Initialize CAN receive
Select the Brand of concern ( )
{
        Hardware preparations, based on the brand
        Delay (1000)
}
Display the action in Trace Panel
Clear the faults, if any
Delay (1000)

    // Step 1
    Display the action in Trace Panel
    // Step 2
    Display the action in Trace Panel
    .
    .
    .
    // Step N
    Display the action in Trace Panel

    Ignition off ( )
    Stop ( )
```

Fig. 8. Preparations and Steps.

The pseudo code illustrated in the Fig. 8 outlines the Preparations process, with the execution flows of:
i. The processor enables the CAN initialization.
ii. Based on the Brand selected from the user, the hardware preparations are performed. A delay of 1000ms is given to make sure that the processor gets sufficient amount time for setting the mechanical parts as per the requirement.

iii. Further, the processor checks for the trace of above actions, in the Trace Panel.
iv. Later the actions are programmed towards clearing the faults, if any, and the execution control is transferred towards the next process named Check for Fault Codes.

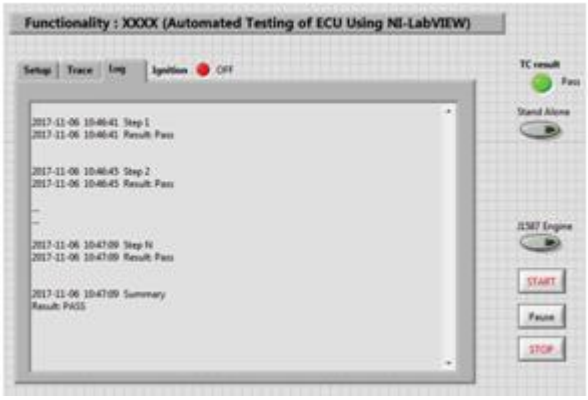

Fig. 9. Log panel.

After the successful execution of all the steps, based on the performance of the ECU with respect to a functionality of interest, the LED indicator, labeled as 'TC result' will glow,as shown in the Fig. 9. If the ECU passes the test, then the LED glows green with a message 'Pass', else the LED shows red with a message 'Fail'.

Further, the detailed test report can be obtained in the Trace panel, which facilitates taking the printout for documentation purpose. Adding to this, the brief summary of each step can also be witnessed in the Log panel as represented in the Fig.9.

V. CONCLUSION

The methodology illustrated in this research is an innovative methodology for upholding the level automation in testing the ECU using NI-LabVIEW tool. The performance of this arrangement proves its superiority with respect to the accuracy in testing as well as with the effortless documentation. The development of Functionality Scheduling Verifier tooladds to the proposed work, which ensures, all the necessary tasks execute within the deadline

REFERENCES

1. *Shruthi T S and K H Naz Mufeeda, "Using VT System for Automated Testing of ECU", International Organization of Scientific Research Journal of Computer Engineering, Volume 18, Issue 3, May June 2016.*
2. *J. Novák and P. Kocourek "Automated Testing of Electronic Control Units Compatibility in Vehicle CAN Network", IEEE International Symposium on Industrial Electronics, Croatia, June 20-23, 2005.*
3. *T. Bardelang, "CAN in the Hardware in the Loop (HiL) Simulator - Experiences with Networked Vehicle Functions and Their Testing in the Application of a HiL Simulator for Heavy Trucks", Vector Symposium CAN in Commercial Vehicles, Stuttgart, May 12, 2004.*
4. *Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, and Tadayoshi Kohno "Experimental Security Analysis of a Modern Automobile", IEEE Symposium on Security and Privacy, USA, May 16-19, 2010.*
5. *Viacheslav Izosimov "Analysis and Standardization of Truck Architectures", IEEE International Control Conference, Anchorage, AK, May 8-10, 2013.*
6. *Larses, O., "Dependable Architectures for Automotive Electronics – Philosophy, Theory and Practice", Licentiate Thesis, Dept. of Machine Design, Royal Institute of Technology, 2003.*
7. *"Presentation ofBuilders Future at Volvo" at Mälardalen Högskola, Volvo 3P, Sweden, May 26, 2004.*

8. *Appel, W., and T. Dunke, "Electrical/ Electronic architecture of the new Mercedes-Benz Actros", 7th Int. Conference on Trucks and Buses, Austria, 2003.*
9. *Johansson, K., M. Törngren, L. Nielsen, "Vehicle Applications of Controller Area Network, in the Handbook of Networked and Embedded Control Systems", IEEE Conference on Automotive Communications, 2005.*
10. *Bikash Poudel and Arslan Muni, "Design and Evaluation of a Novel ECU Architecture for Secure and Dependable Automotive CPS", 14th IEEE Annual Consumer Communications & Networking Conference, 2017.*
11. *F. Gustafsson, "Automotive safety systems, replacing costly sensors with software algorithms," IEEE Conference on Signal Processing, vol. 26, pp. 32-47, 2009.*
12. *Jelena Kociü and Harald Axmann, "A novel solution for an ECU simulator as a key component for automated testing and verification of a vehicle diagnostic device", IEEE 25th Telecommunication Forum, India, Aug 10-11, 2017.*
13. *Krisztian Enisz, Denes Fodor, Istvan Szalay, and Laszlo Kovacs, "Reconfigurable Real-Time Hardware in-the-Loop Environment for Automotive Electronic Control Unit Testing and Verification", IEEE Instrumentation & Measurement Magazine, 2014.*
14. *Yutaka Onuma, Yoshiaki Terashima and Ryozo Kiyohara, "ECU Software Updating in Future Vehicle Networks", IEEE 31st International Conference on Advanced Information Networking and Applications Workshops, Taiwan, March 27-29, 2017.*
15. *Florin Prutianu, Viorel Popescu, Pop Calimanu Ioana Monica, "Validation System For Power Supply Module Part Of Automotive ECUs", IEEE 10th International Symposium on Electronics and Telecommunications, Romania, 2012.*
16. *Xiaofeng Yin, Jingxing Tan, and Lei Li, "Development of a Real-time Monitoring System for ECU based on CAN Bus", IEEE 2nd International Conference on Industrial and Information Systems,China, July 10-11, 2010.*
17. *W. Feng and J. W. S. Liu, Algorithms for scheduling realtimetasks with input error and end-to-end deadlines, IEEETransactions on Software Engineering, 23(2):93–106, 2007.*
18. *J. Y. Chung, J. W. S. Liu, and K. J. Lin, Scheduling periodicjobs that allows imprecise results, IEEE Transactions ofComputers, 19(9):1156–1173, 2014.*
19. *P. Ramanathan, Overload management in real-time controlapplication using (m,k)-firm guarantee,IEEE Transactions onParallel and Distributed Systems, 10(6), June 2016.*
20. *Wilfried Voss, "A Comprehensible Guide to Controller Area Network", Published by Copperhill Technologies Corporation - 2005.*
21. *National Instruments, "LabVIEW User Manual", July 2016 Edition.*