Automation of UDS

[1]Sunil Patil, [2] Veena H.S, [3] Manohar Mysore Srikant, [4] Megha Rammohan
[1] Student, [2] Associate Professor, [3] Technical Project Manager , [4] Software System Designer
Department of Electronics & Communication Engineering, Bangalore Institute of Technology , Bangalore, Karnataka, India

Abstract— **In today's world diagnosing the vehicle is not an easy task. In any vehicle, if there is found something unusual or showing unexpected behavior then, person has to test each and every ECU in the vehicle. This method is good if vehicle has one or two ECUs, but in real world high end cars contains nearly more than 100 electronic control units. Manual Testing of the ECU in the development phase is very time consuming, less accurate and low efficient. In this project, we are taking a step towards this vision by developing the test cases as per the requirements for the communication diagnostic services which are defined in ISO 14229 and automatic execution of test cases with the help of ECU TEST tool. This tool can also generate the test reports; ECU test engineer can get the test report faster. Using test automation, we can get faster test results, quicker and improvement in productivity of developers and testers. Also, this increases efficiency and quality of ECU software.**

***IndexTerms*— ECU TEST tool, ETAS Lab car, ISO-14229, UDS**

I. INTRODUCTION

Now a day's car is not only a belonging of a person but also it has become an integral part of our modern family. The user number of
the vehicles all over the worlds has drastically rises from last decade. It means demand of ECU also increasing drastically. To achieve this demand, we must adopt the modern and fastest method to perform testing of ECUs. ECU is the brain of the all the vehicle; it can control all activities inside the vehicle. So while developing and testing the ECU, it must be more reliable and accurate. The number of features with a modern power train system rises significantly. With rise for demand and high comfort and complex features, also increase of unwanted feature interactions between those features rises. This leads to new challenges with testing that can be achieved by transforming manual testing into automated testing with continuous integration. Here we are doing test automation of some the communication diagnostic services; those are defined in ISO standard 14229. This ISO 14229 describes about UDS, specifies requirements of diagnostic services of data link layer which is independent and allow a tester (client) to communicate and control diagnostic functions in an on-vehicle ECU and server, such as an electronic fuel injection, automatic gear box, anti-lock braking system, etc. connected to a serial data link embedded in a road vehicle.

II. RELATED WORK

According to Moore's Law, the count of transistors per square inch on Integrated Circuit's had doubled every year. This resulted in increasing amount of complexity of the embedded software in Industry. Manual testing of embedded code will not that much efficient and consistent. In safety-critical systems like automobile systems, bugs could result in loss of life, property damage, or damage to the environment. So care should be taking in testing process of ECU and some of the previous works
are here:

A. The past practices of different testing techniques are briefly described below[2]:

Model-In-Loop Testing- (MiL) – Functional developers create models based on the requirements using Model based development tools like MATLAB (MATrix LABoratory) or ASCET in a PC. Then the required test cases are written and tested for the model in Dynamic Testing tools like Time Partition Testing. Here no need of any Hardware for testing.

Software-In-Loop Testing- (SiL) – Functional developers would auto generate the embedded code from models or manually write the embedded code based on requirements. That code is compiled for a targeted ECU architecture. Then the required test cases are written and tested for the code in Dynamic Testing tools like Time Partition Testing. Also, this does not require any Hardware components for testing.

Processor-In-Loop Testing- (PiL) – Functional developers would use the "Executable Linkable File" which is generated for a targeted ECU, that is flashed in ECU. The test simulation are done in Dynamic Testing tools such as TPT (Time Partition Testing). Here, the object code is tested in a real target hardware component or an instruction set simulator.

Hardware-In-Loop Testing- (HiL) – It is real time simulation of the hardware where the final object code is tested in a Lab Car using test automation tools such as ECU-Test, Lab Car PT and INCA . Static Testing - The code is tested for code quality and complexity during development phase of embedded code.

B. The quality of software and correctness of software is challenging because of the various configurations. Exhaustive test process is impractical because of time and resource limitations.

C. Feature based interaction testing used to effectively detect faults that are difficult to and by another testing method [6]. In practice, the input configurations of software are applied to the constraints, especially in the case of highly configurable systems. Maintaining and handling constraints effectively and efficiently in interaction testing is a quit challenging problem.

D. Customizable software programs provide user selectable features to allow users program to an application environment scenario. In analyzing which feature yields the best performance is very difficult because a direct measurement of all possible feature combinations is infeasible [7]. In feature based interaction system, when features interact with each other and accurate predictions are challenging task [8].
E. In manual testing, once car came to the service center then he has to test the ECU which is reporting the problem. Here he has to perform manual calculations and calibration to fix the error in the vehicle. This needs man resource with good analytical and mathematical skill [10]. It is not that much reliable and accurate, it is also time consuming method.
Manually testing the embedded system will no longer be efficient and consistent. Also, manual testing consumes more time and effort of the developers. This would also result in delay in software product delivery. Manual testing is also error prone and could easily introduce bugs because of human intervention. In safety-critical systems like automobile systems, bugs could result in loss of life, property damage, or damage to the environment. To overcome above problem we are implementing the Test Automation in the development phase of ECU life cycle.

III. PROPOSED SYSTEM BLOCK DIAGRAM

The main object of the automation is to automatic execution of test cases and generation of reports. After the development of the software as per the requirements then we need to give build, it will generate executable files (A2L and HEX files). These are the basic files need to test any software.

Detailed step by step description of block diagram is given below.
1. In test system, it contains connecting the Engine Electronic Control Unit with real-time simulation model.
2. Start ECU TEST tool, which involves the initialization of all hardware and software required to execute the test cases.
3. Write test cases as per the requirements from the customers.
4. PC with INCA that helps in measurement, calibration.
5. Run test cases and generate reports.

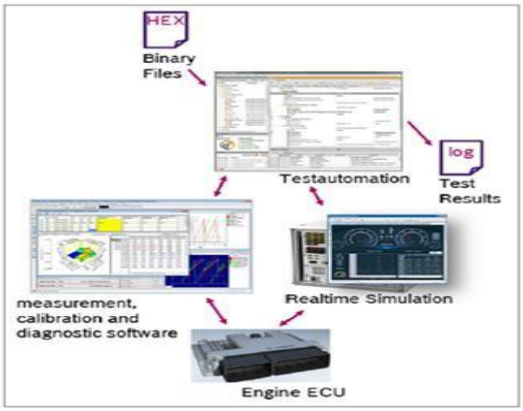

Fig 1: Test System

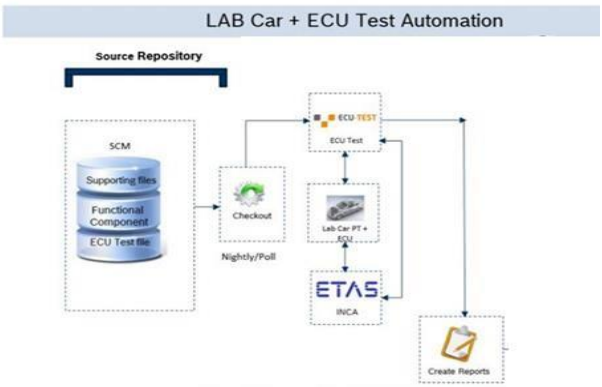

Fig 2: Automation of HiL

The software used for test automation was developed by Trace-Tronic. ECU-Test tool contains the test script that activates the downloading of the executable machine readable file onto Electronic Control Unit. HiL testing systems contribute to quality assurance before entering into development phase of ECU. That facilitates the testing of the functions or diagnostic behavior of ECUs in the Lab Car simulator system. SUV in HiL Environment can be achieved by automating the LAB Car PT, INCA and ECU- Test in Jenkins [9].

IV.PROPOSED METHODOLOGY

The Work flow of the test automation contains 3 main steps as given below;

1. Analysis of requirements from the customers(JLR).
2. Writting test cases as per the requirements.
- Precondition
- Action
- Post condition
3. Executing test cases and generating the reports.

Here we will discuss above steps in detail,

1. Analysis of requirements from the customers(JLR).

In this step we need to analyze the different requirements from the customer and how to achieve those requirements, which method
need to follow etc.

2. Writting test cases as per the requirements. Precondition

Each test case starts with a block called Precondition, It contains only preparations for the test step execution. If these preconditions are not satisfied, the further execution of the test case will be aborted. It mainly consists of configuring of 2 files namely TCF and TBC files. TCF is different for each project lines, it contain path of executable files, information about Diagnostic Data bases and simulation models and default directory to store the test reports. TBC contain tool dependent settings specific to test environment, measurement and calibration variables and vector hardware information.

Action

It contains the test steps and actual test cases as per the requirements from the customers.

Evaluation - steps to wait for result and then compare with predefined result record. Also Plots graph useful for report analysis and debugging.

Post condition

Every change made in Precondition or Action is reset in the Post condition block.

3. Executing test cases and generating the reports.

After configuration of precondition and we can run the test cases and generate the reports.

Tool Environments :

- ECU: It is tri-core processor that execute programs related to the Electronic Control Unit software.
- ETAS INCA: It is used for measurement, calibration and diagnosing the software and to download the binary files on the Electronic Control Unit, modifying the software variables.
- ETAS LABCAR: It is hardware-in-loop simulation system, which act as simulation model of the vehicle environment that including engine process and driver point of environment in vehicles.
- ECU TEST: Test automation software developed by Trace-Tronic GmbH, which is super ordinate software that execute test scripts of automation project and controls different soft wares. It is an automation tool for optimum support of Creation and management of test cases, Test Execution and Establishment of test documentation/Report.

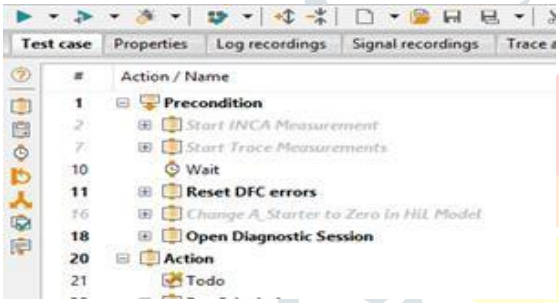

Fig 3: Showing different steps in precondition

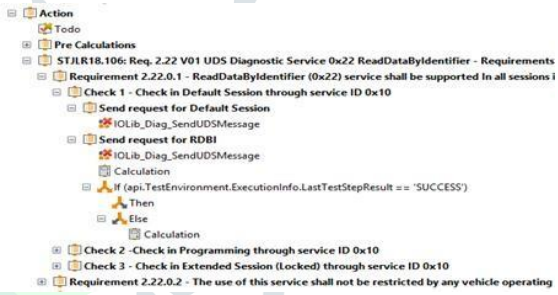

Fig 4: Sample test cases along with requirements

V. RESULT AND DISCUSSION

Once executed the test automation scripts, It will runs the all test cases and generates the test report. That reports contains the

evaluation of test cases means after getting the response from the ECU it compares with the predefine expected test result, if both results are same then it will predict as SUCCESS in the test report documentation. Else it may gets fail because of unexpected results or gets error because of some inconsistency in the tools and connections. At that time again we need to verify the connection and test cases then execute to get the positive results.

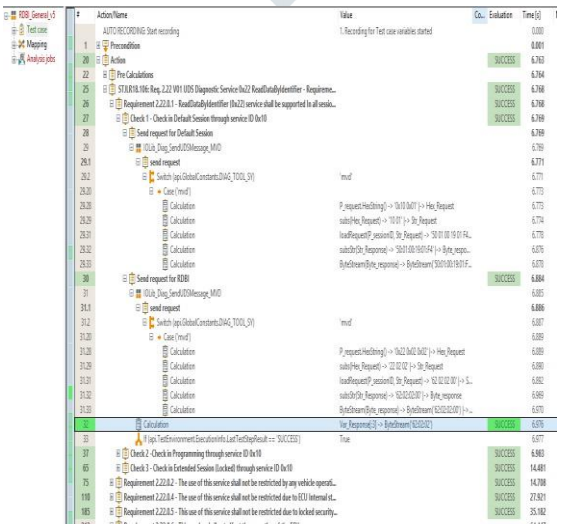

*Fig 5: Showing c*omplete test report

VI. CONCLUSION

There are numerous advantages when we automate testing process, by automating we can get faster test results, quicker and
improvement in productivity of developers and testers. For every delivery of an ECU functional component, automated tests are triggered for all test cases for all variants as per the configuration files. By taking minimum required inputs from user, we automated the testing process which takes very less time as compare to manual testing and maximize the productivity of the users. Totally this increases the efficiency of testing process and quality of ECU software. In future work by using parallel execution in CI server (Jenkins), test could run in parallel for different functional component and system components. This also gives more transparency in the build and test process and helps to achieve continuous delivery, which increases the further efficiency in testing process.

VII. REFERENCES

[1] P. C. Raut, S. L. Tade and R. S. Jadhav, "Test automation for Engin Warmup feature testing," 2016 International Conference on
Computing Communication Control and automation (ICCUBEA), Pune, India, 2016.
[2] Vijayaraghavan, Jagadeeswara Vijayaraghavan Angamuthu, Saravanakumar C Shanmugam Vinodhini “Automated Continuous
Verification & Validation for Automobile Software”, Robert Bosch Engineering and Business Solutions Private Limited. [3] D. Wagner, H. Kaindl, S. Dominka, M. Dübner. “Optimization of
[4] feature interactions for automotive combustion engines”. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing (SAC '16). ACM, New York, NY, USA, 2016, pp. 1401-1406.
[5] P. Machado and A. Sampaio, “Automatic Test- Case Generation,” in Testing Techniques in Software Engineering.
[6] S. Apel.“Feature-Oriented Software Product Lines”. 1st ed. [S.l.]: Springer-Verlag Berlin An, 2016. Print.
[7] N. Siegmund et al.,"Predicting performance via automated feature interaction detection," 2012 34th International Conference on
Software Engineering (ICSE), Zurich, 2012, pp. 167-177.
[8] N. Nilsson, The Quest for Artificial Intelligence: A History of Ideas and Achievements. New York, NY, USA: Cambridge Univ. Press, 2009.
[9] P. Jalote, “Testing,” in An Integrated Approach to Software Engineering, P. Jalote, Ed. New York, NY: Springer New York, 1997,pp. 403– 479.
[10] https://www.guru99.com/differenceautomated- vs-manual-testing.html.