A Novel Framework Design for Test Case Authoring and Auto Test Scripts Generation

Srinivas Perala*
Department of Electronics, Lovely Professional University, Phagwara, Punjab, India.

Dr. Ajay Roy
Department of Electronics, Lovely Professional University, Phagwara, Punjab, India.
* Corresponding author. E-mail: srinivaslabview@gmail.com

Article History: Received: 10 November 2020; Revised 12 January 2021; Accepted: 27 January 2021; Published online: 5 April 2021

Abstract: Every product has defects and identifying defects in the process of development and rectifying them before the launch of the product is very important. Embedded software testing process find the bugs in the software and report to the developer to fix issues. Sometimes to meet the product release deadlines, test engineers will not get much time to cover all test cases. That is why most software testing depends on test automation. In this paper, we focused on the area of automotive and home appliances embedded software test automation. Test automation is the only solution to improve the test phase and meet the timeline of the product launch. There are many test Automation tools like LabVIEW, test stand, and automation desk to automate testing embedded software. However, there is still manual efforts are required to use these tools. This paper deals automate those manual efforts. This Works shows how to generate test scripts from test cases to reduce the manual efforts, time, and cost.

Keywords: Machine Learning, Test Automation, Test Stand

---

1. Introduction

Embedded Software Development Life Cycle aims to deliver high-quality product software that satisfies all the requirements. Its starts with Planning and requirements analysis where senior people design product, based on market requirements and technical experts' suggestions. Once the product design has been done then the next stage is to document called SRS (Software Requirement Specification) the product requirements and get the approval from the client. The SRS document decides the product architecture and gets reviewed by the top management for final changes. Based on the SRS document, software developers develop the product software and develop test cases that cover all software functionalities. The software quality assurance team used the test cases and performed either manual Regression or do automation testing. SWQA team used test management software to automate the manual test scenarios. If there is any part of the software updated, the testing team must perform entire software testing to find any other aspects affected with part of the software update, and this is called the regression test.

Software developers follow one of the common software life cycle development models which are mainly waterfall model, Iterative model, Spiral model, V- model and Agile model. The waterfall model divides software development into different phases, and each phase depends on the previous phase. Moreover, it is not suitable for complex projects as it has high risk and uncertainty. The iterative model repeatedly improves the developing versions until the full system is ready as this suite for large projects, but more resources are required. In the spiral model, the development covers three phases like identification, design and build. Requirement changes can easily manage in the spiral model, but the process is complicated. In the Verification and Validation model, implementation of processes happens sequentially like a V-shape. Here each phase of development is directly linked with testing. The V -model is best suited for a small project where requirements are well defined, but it is not suitable for significant and ongoing projects. The agile model divides the product into minor incremental builds and a very realistic approach to software development. These builds follow the iteration method. Each iteration naturally takes from about one to two weeks. Every iteration contains sub functional teams working simultaneously on various areas.

Manual regression is essential for software validation and its time-consuming task. The solution is test automation where one needs to create script manual test cases using any test management software. Later, scripts can be executed as many as times to get results reports and can reduce manual effort and time. In embedded software industries, mostly using test management software are NI Test Stand and dSPACE Automation Desk. The Test Stand is test management software and allows test engineers to develop automated test scripts using code modules written in any programming language. One of the popular languages is LabVIEW, and it is a graphical programming language. At the same time, the automation desk is using mostly in the automotive industry for Hardware in loop test automation. Software testing is a crucial stage in the software development and maintenance cycle. It has been estimated that software testing involves between 30-50 percent of software development. With the wrong choice of testing, the software development process may become costly, inefficient and causes a high frequency of product release delay. We can script manual test cases using test automation tools, but this is costly, and maintenance required highly skilled test engineers. Our framework gives the solution to this problem.

2. Related Work

Software developers seek test automation methods owing to the costs involved in manual testing. Reusable functionality is widely useful and can reduce engineering costs. Researchers worked on IEC 61131-3 programming languages and classified script generation methods. The study provides a fast method for generation for automation in different industries. The full validation of system functions by simulation at the sub-system and component level can significantly reduce system simulation time. This paper implemented a new framework for automation in the wood and the chemical industry. The framework talks about the requirements of key intelligent automation features and data characteristics, also presented the development of support decision systems into automation modules. A Middleware for Intelligent Automation has been proposed in the experiment using Big data and cloud infrastructure.

A robot operation and testing simulation framework have been proposed to detect logical and physical design flaws. The study uses High Level Architecture (HLA) as a middleware for real-time testing for any hardware device. Test automation approach reduces the software cost and improves the quality. An attempt has been made to measure the effect of automation of testing. A well-trained model can generate test scripts from the design document in the web application testing. For the experiment, a testing tool, TesMa was developed at the NTT Labs. Test automation method has been explored for various appliances employing embedded software. It introduced a novel method to select required test cases from large test suites to reduce time and cost.

3. Methodology

The proposed script generation framework is implemented in three phases. Phase-1 is the process of writing test cases as per the required format using the software requirements document. Phase -2 is the key part that generates test scripts using pre-defined library set based on intelligent logic. Phase -3 is about to execute scripts and to get test reports. Test script execution depends on the test management software. The three phases have been described in figure 1.

Test Case Authoring

Phase-1 aims to generate standard test cases and deals with Requirements documents, Natural Language Processing-based Test case Generation (NLP TCG) and Test case editor. Test case editor is an interface where we can write test cases manually using software requirement documents. Test case authoring is built on the structure of a pre-defined key with sections. The test case editor user interface is shown in figure 2.

Test steps are divided into three groups: setup, Main steps and Clean-up. Setup group covers pre-condition steps, Main steps group covers the actual test steps of the test scenario, and clean group covers the postconditions of the test scenario.

Five sections are provided to write the test step to cover most of the functionalities.

Action:

Action has the following key terms to define the action of the test step. We used four key terms write, read, set and check to cover the significant test actions. We can add new key terms into action depending on the requirement as shown in figure 3.

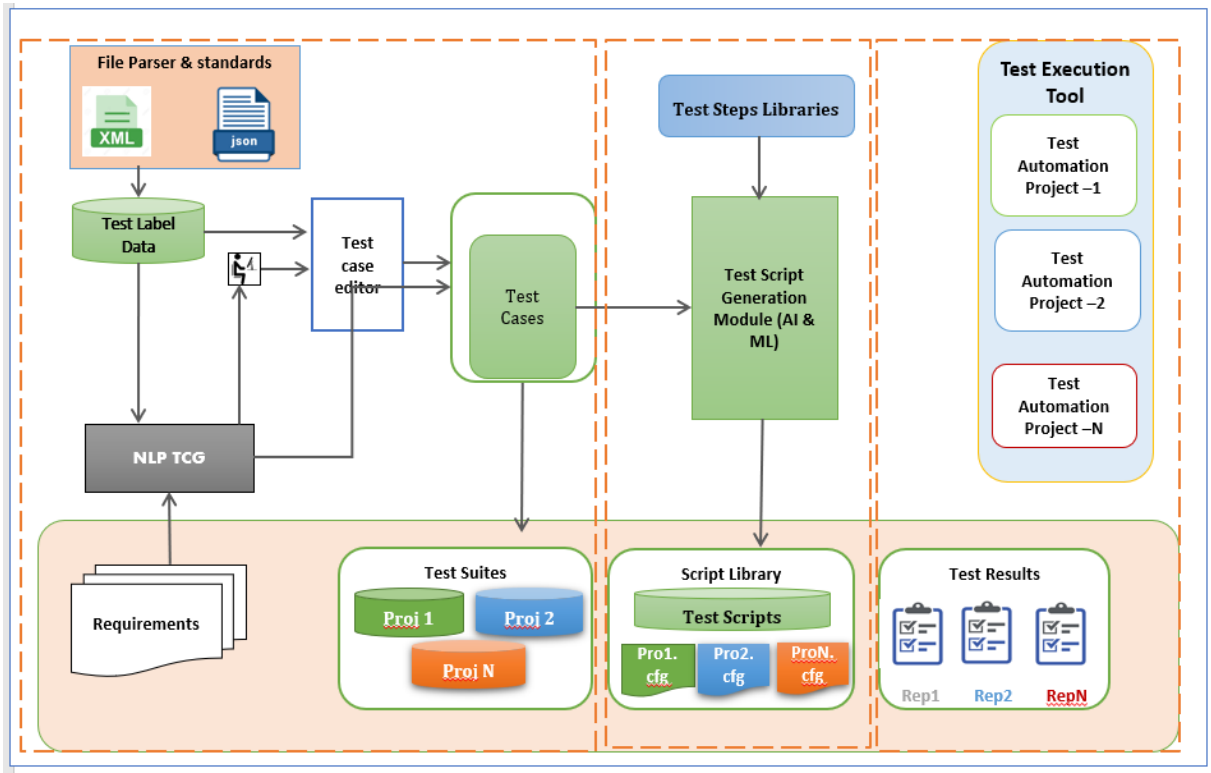

Figure 1 Proposed Framework block diagram

Action	Function/Service Type	Parameter	Value	Remarks
Setup				
Main Steps				
Clean-up				

Figure 2 Test case editor User Interface

Action
Write
Read
Set
Check

Figure 3 Sample Key set at Action Section

Function/Service Type:

The keys belong to this section to determine the service type of test step. For example, if we are using protocols like LIN, CAN or Ethernet in software development, writing into these protocol bus messages and reading from these protocol bus steps can be determined here as shown in figure 4**.**

Function/Service Type
ECU Signals
ERD Signals
CAN
LIN

Figure 4 Sample key set at Function/Service type section

Parameter:

Test parameterization is an efficient method and utilizing parameters rather than fixed values. So, one can execute the same test with different input values of test steps used as inputs or expected outputs that are stored separately from the test steps. Test Parameterization is a powerful practice to enhance the testing scenarios and reuse the same. During the execution of test steps, the parameters are filled in with their actual value, so a parameterized variation of the original test case is produced. The main benefits of parameterization are as follows:

- By externalizing the changing parts of a test case(s) as parameters, it is easier to manage complex test cases
- Users can automatically execute multiple variations of each test. Multiple Parameter values provided that will cause several executions of each test for each variation of the parameters.

Test Parameterization allows sharing information between multiple test cases.

Value:

In this section, we can define the value of the parameter and can define the expected output value. The remarks section can use to explain any additional notes, which is useful to consider complicated test steps.

Oven, a home appliance used for baking food items and given below is sample oven software requirement. The oven light should turn ON when the oven door is open, or the oven light switch is turned on. Also, the count of oven light ON and OFF, count of the door open and close should be stored. Test case writing using the proposed framework test case editor is simple and easy to understand as shown in figure 5.

Action	Function/Service Type	Parameter	Value	Remarks
Setup				
Set	Function	TestUnit	PowerON	
Main Steps				
Read	ECU Variable	OvenDoorOpenCloseCount	X_Count	Store value into X_Count
Read	ECU Variable	OvenLightOnOffCount	Y_Count	Store value into Y_Count
Set	ECU Variable	Oven Door	Open	
Check	ECU Variable	Oven Light	ON	
Set	ECU Variable	Oven Door	Close	
Check	ECU Variable	Oven Light	OFF	
Check	ECU Variable	OvenDoorOpenCloseCount	X_Count + 1	
Check	ECU Variable	OvenLightOnOffCount	Y_Count + 1	
Clean-up				
Set	Function	TestUnit	PowerOFF	

Figure 5 Sample Testcase

Embedded software development uses standard documents like JSON or XML files to maintain microcontroller variable data. Here we do parse these files and save them into test label data. Test label data useful write test cases easily. OvenDoorOpenCloseCount, Oven Door and oven light are the test label which parsed from the oven software JSON file.

Phase -2: Test Script Generation

Test steps library based on test management software required to prepare before script generation. Test steps library is part test automation process. Test steps library need to fill with all basic functionality with respective test management software's code modules. In NI test stand mostly, LabVIEW code module used, and for Dspsce automation desk python code used. The test configuration is part of the proposed framework which configure test management software. Script Generation Engine is the business logic which generates test script taking test cases as input and identify the functions based on key terms then replace parameters and values. For example, Step1 action key is Set, the Service type is Function, Parameter is Test Unit and value is Power ON. Based on this combination, it identifies the test unit function and set the write value as Power ON.

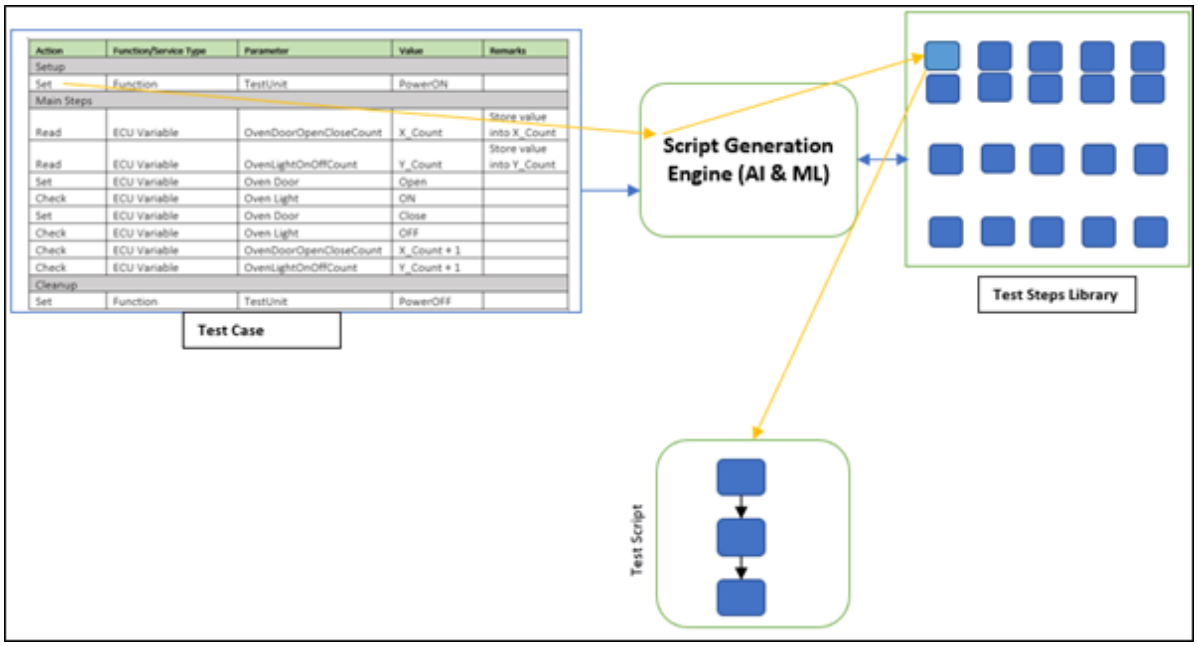

Figure 6 Test Script generation block diagram

Step	Description	Settings
Setup (1)		
Set Function TestUnit PowerON	Action, PowerCycle.vi	Additional Results
<End Group>		
Main (7)		
Read ECU Variable OvenDoorOpenCloseCount X_Count	Action, ERD Read.vi	Additional Results
Read ECU Variable OvenLightOnOffCount Y_Count	Action, ERD Read.vi	
Set ECU Variable Oven Door Open	Action, DAQ Controls Write.vi	Additional Results
Check ECU Variable Oven Light ON	Pass/Fail Test, DAQ Controls ...	Additional Results
Set ECU Variable Oven Door Close	Action, DAQ Controls Write.vi	Additional Results
Check ECU Variable OvenDoorOpenCloseCount X_Count + 1	Numeric Limit Test, x == Local...	Additional Results
Check ECU Variable OvenLightOnOffCount Y_Count + 1	Numeric Limit Test, x == Local...	Additional Results
<End Group>		
Cleanup (1)		
Set Function TestUnit PowerOFF	Action, PowerCycle.vi	Additional Results
<End Group>		

Figure 7 Auto-Generated Sample script for Test Stand

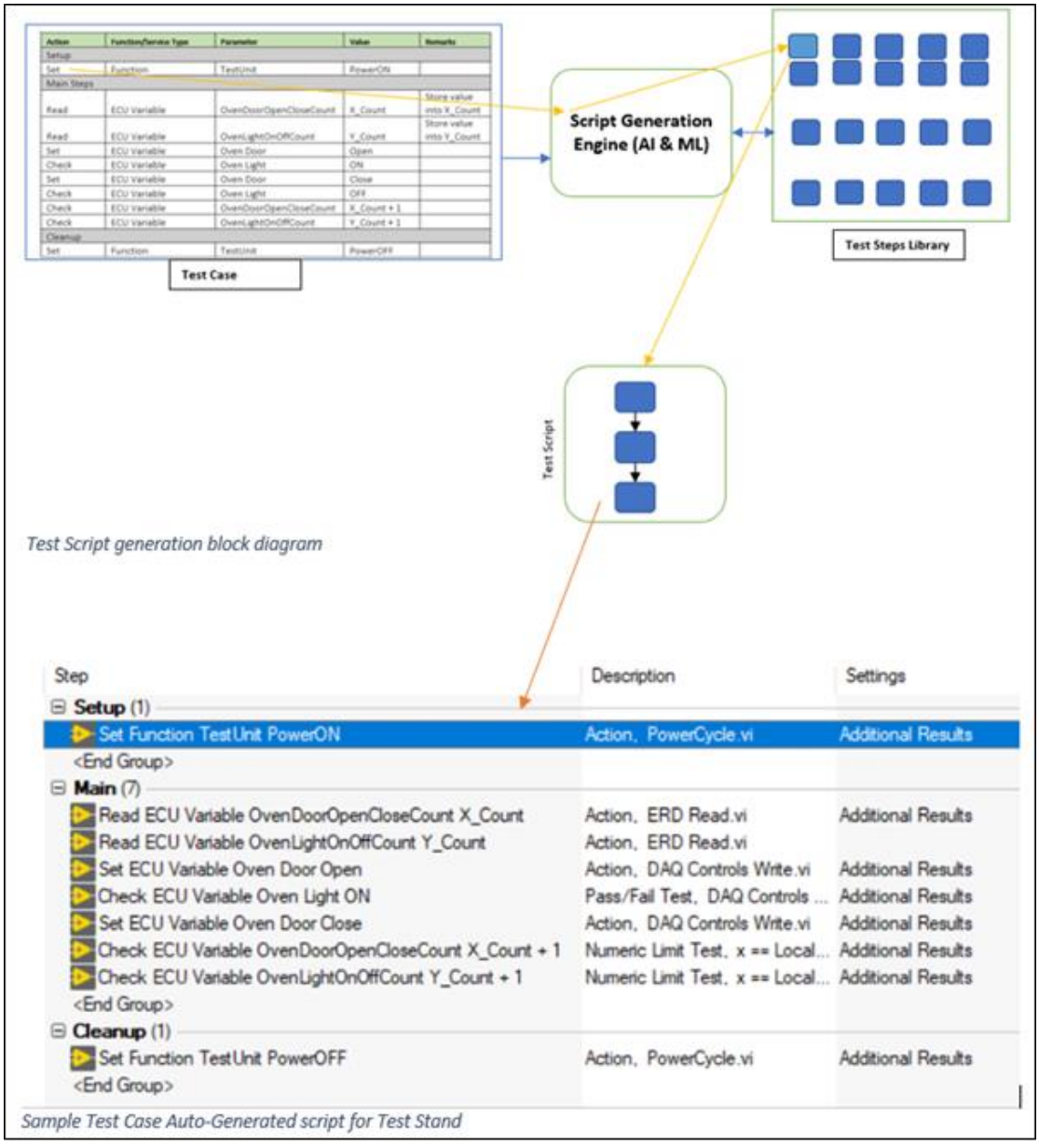

Figure 8 Auto Script generation Block Diagram

The general process takes a minimum of 6 hours to write test cases and for scripting 8 hours. So total time approximately 14 hours to automate this requirement testing. Our proposed framework takes 4 hours to write test cases and script generation and validation take a maximum of 15 minutes.

The Script generation Engine has almost had all combinations of logic to generate script generation. Moreover, it is practically not possible to write all combinations of logic. Artificial intelligence and machine learning-based algorithm to predict the unknown combination have been introduced.

Machine learning is part of the Artificial Intelligence technique which will minimize the code to cover unknown cases. Supervised Machine learning algorithms are used to train the model using labelled data, such as an input where the desired output unknown. We trained the script generation model using the k-nearest neighbours (KNN) algorithm with pre-defined key section combinations to predict the test function from the test library for the unknown combination of input keys. However, we have not tested much prediction accuracy and precision. We will discuss the ML algorithm related information in future work.

Phase -3: Test Script execution

The Scripts generated in phase-II is loaded into test management software which is used to execute the test scripts and will get the automation report. We can see the sample automation report in figure 9.

As per the above table, we can save 5 hours for one test case. Similarly, we can have 200 to 300 test cases for one software development depend on the project. In this way, we can save 1000 hours to 1500 hours per software test in the case of 200 to 300 test cases with the proposed new test automation framework. However, this script generation has a limitation that cannot generate complicated test steps. In the future, we are going to validate the Machine learning algorithm to get more accurate script generation.

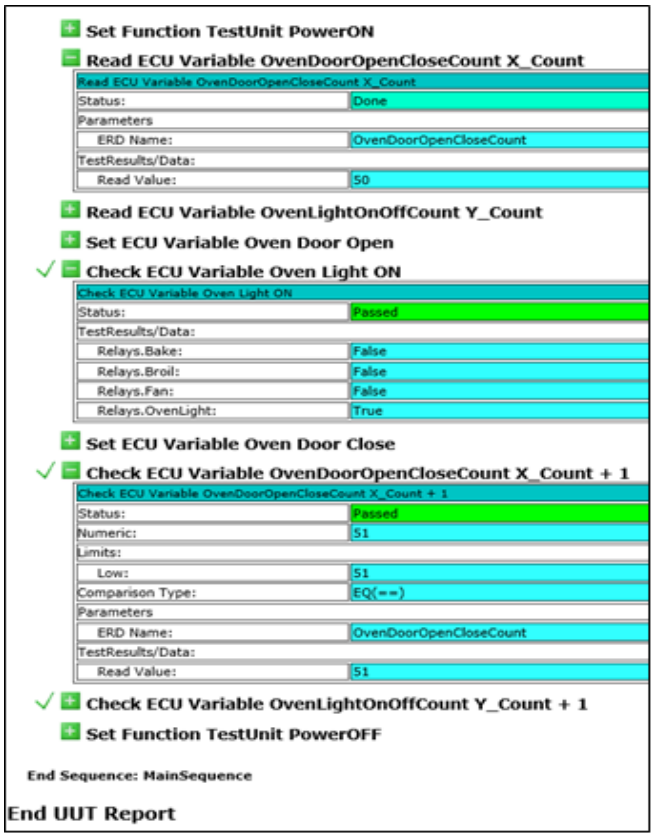

Figure 9 Test Automation Report generated by Test Stand

4. Results

Table 1 shows the time required for traditional agile test automation and new approached test automation.

Table 1 Time Comparison

S.No	Task	Time is taken by "Traditional Test Automation."	Time is taken by "new approach Test automation."
1	Test Case authoring	4 hours	3 hours
2	Test case validation by expert	1 hr	1 hr
3	Test Script development	4 hours	< 1 minute
4	Test Script validation	30 minutes	30 minutes
Total Time		9.5 Hours	4.5 hours

5. Conclusion

The proposed method is very useful in the field of test automation to reduce the time and cost. This paper need improvements in the selection of right library for key combination. In future, this library selection can replace with trained model with machine learning algorithms.

References

[1]. D. Iqbal, A. Abbas, M. Ali, M. U. S. Khan, and R. Nawaz, "Requirement Validation for Embedded Systems in Automotive Industry through Modeling," *IEEE Access*, vol. 8, pp. 8697–8719, 2020, doi: 10.1109/ACCESS.2019.2963774.

[2]. D. Kumar and K. K. Mishra, "The Impacts of Test Automation on Software's Cost, Quality and Time to Market," in *Procedia Computer Science*, 2016, vol. 79, pp. 8–15, doi: 10.1016/j.procs.2016.03.003.

[3]. H. J. Vishnukumar, B. Butting, C. Muller, and E. Sax, "Machine learning and deep neural network - Artificial intelligence core for lab and real-world test and validation for ADAS and autonomous vehicles: AI for efficient and quality test and validation," in *2017 Intelligent Systems Conference, IntelliSys 2017*, 2018, pp. 714–721, doi: 10.1109/IntelliSys.2017.8324372.

[4]. S. Thummalapenta, S. Sinha, N. Singhania, and S. Chandra, "Automating test automation," in *Proceedings - International Conference on Software Engineering*, 2012, pp. 881–891, doi: 10.1109/ICSE.2012.6227131.

[5]. Kumar, K. Verma, S. Perala, and P. R. Chadha, "Automatic Attendance Visual by Programming Language Lab VIEW," in *IEEE international Conference on Power E;lectronics, intelegent Control and Energy Systems (ICPEICES-2016)*, 2016, pp. 1–5.

[6]. Jantsch, J. Notbauer, and T. Albrecht, "Functional validation of mixed hardware/software systems based on specification, partitioning, and simulation of test cases," *Des. Autom. Embed. Syst.*, vol. 5, no. 1, pp. 83–113, 2000, doi: 10.1023/A:1008943617450.

[7]. Lawanna, "HTTCS: Hybridization technique for test case selection," *Walailak J. Sci. Technol.*, vol. 16, no. 2, pp. 95–105, 2019, doi: 10.48048/wjst.2019.1301.

[8]. V. Vyatkin, "Software engineering in industrial automation: State-of-the-art review," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 3. pp. 1234–1249, 2013, doi: 10.1109/TII.2013.2258165.

[9]. H. Koziolek, A. Burger, M. Platenius-Mohr, and R. Jetley, "A classification framework for automated control code generation in industrial automation," *J. Syst. Softw.*, vol. 166, p. 110575, 2020, doi: 10.1016/j.jss.2020.110575.

[10]. S. M. Shah and M. Irfan, "Embedded hardware/software verification and validation using hardware-in-the-loop simulation," in *Proceedings - IEEE 2005 International Conference on Emerging Technologies, ICET 2005*, 2005, vol. 2005, doi: 10.1109/ICET.2005.1558931.

[11]. T. Coito *et al.*, "A novel framework for intelligent automation," *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 1825–1830, 2019, doi: 10.1016/j.ifacol.2019.11.501.

[12]. J. C. V. S. Junior, A. V. Brito, L. F. S. Costa, T. P. Nascimento, and E. U. K. Melcher, "Testing real-time

embedded systems using high level architecture," *Des. Autom. Embed. Syst.*, vol. 20, no. 4, pp. 289–309, 2016, doi: 10.1007/s10617-016-9178-0.

[13]. H. Tanno and X. Zhang, "Test script generation based on design documents for web application testing," in *Proceedings - International Computer Software and Applications Conference*, 2015, vol. 3, pp. 672–673, doi: 10.1109/COMPSAC.2015.74.

[14]. Roy, S. Perala, and K. Barman, "A novel method of Test Automation for Testing Embedded Software," *Think India J.*, vol. 22, no. 37, p. 146, 2020.