20XX-01-XXXX

Optimal test strategy for Clusters and CDCs in multi power train vehicle ecosystem

Author, co-author (Do NOT enter this information. It will be pulled from participant tab in MyTechZone)

Affiliation (Do NOT enter this information. It will be pulled from participant tab in MyTechZone)

Abstract

Innovation in energy storage and generation system will lead to multiple power train solutions across the vehicle categories in the Automative segment. With various options to the end consumer across different vehicle segments, the complexity associated with E/E Architecture and software engineering will be multi-fold both for the OEMs and Suppliers. Over the air updates shall become mandatory features to manage this complexity and to calibrate the vehicle features in line with changing trends and efficiency plus feature enhancements in post-market release scenarios. These upgrades are more common in digital clusters, in-vehicle entrainment and central digital cockpits. OEMs are introducing the vehicle platforms in multiple power train variants keeping comfort, instrument clusters and in-vehicle entertainment as core features across different power trains. A well-defined and managed comprehensive optimal test strategy and infrastructure will be critical to ensure seamless release of the software solutions to different power trains during the product development phase and post-launch software upgrades. In this novel work, the paper extensively explores the current practices of segregating features through common versus specific powertrains, managing the overall test strategy across varied test types, ~~test data~~ and test infrastructure; then address the advantages and challenges in current practices. The paper would summarize the optimal test strategy in approaching the software development pipeline for a multi-powertrain architecture and focus specifically on early test-driven interventions for robust software deployment on production for both parallel and staggered release pipeline

Introduction

Multi-powertrain architectures are becoming more a norm these days, than they were a decade earlier. This, though brings in various options to the end consumer across different market segments, the complexity associated with engineering and especially Software engineering and its associated tests is ever growing. The scale and complexity associated with defining and managing a comprehensive test strategy, for a vehicle architecture across multi-powertrains is more profound in recent years

The thought flow in this paper would be first to introduce the various powertrains in automotive, the complexity of feature variations due to those powertrains, followed by overview of Software development lifecycle and its impact on products like instrument panel cluster and cockpit domain controllers, the current challenges in defining test strategy for such products and finally introduce the audience towards optimal test strategy during early development to production pipeline.

Automotive Powertrains

The performance of a vehicle heavily relies on the efficiency and condition of its powertrain. [4]. This system is central to vehicle performance, efficiency, and reliability, making it a core area of engineering focus for both development and testing in the automotive sector [2]. An automotive powertrain [1, 2] is the collective system of components in a vehicle that generates propulsion and delivers it to the wheels.

**Components of powertrain**

The core components of powertrain are engine, transmission, driveshafts, differentials, final drive and control module. All these components can be selectively grouped into two major sets, namely engine [2] and drivetrain [2]. Engine is one crucial aspect, which converts the chemical energy from fuel / battery to mechanical energy to initiate motion and drivetrains, which has the remaining components like driveshaft, transmission differential and axles, is just a way to describe all parts of the powertrain not related to the engine [1]. Each of the components is important, however lets briefly see the variations introduced by engine in specific and drivetrain. The following section provides a high-level overview of engines and drivetrains.

**Engines**: Engines are of one of the six types [5,6,7]. They are

* Internal Combustion Engine (ICE) – Gasoline
* Internal Combustion Engine (ICE) – Diesel
* Hybrid Electric Vehicle (HEV)
* Plug-in Hybrid Electric Vehicle (PHEV)
* Battery Electric Vehicle (BEV)
* Fuel Cell Electric Vehicle (FCEV)

A good summary of their pros and cons is well described in multiple papers [5, 6]. However, for the context of this paper, it is important to understand that many of these types will remain part of vehicle architectures for several more years. As the industry evolves within these architectures to improve efficiency and meet regulatory requirements, this evolution will drive the development of numerous software-based features and functions.

**Drivetrains**:

Drivetrains are a sub-set of the whole powertrain, which helps the vehicle move. Drivetrains are one of the four following, which are

* front-wheel drive
* rear-wheel drive,
* all-wheel drive
* four-wheel drive.

Each of those are chosen for multitude of reasons by balancing some competing demands from, handling, fuel economy, traction control, road types, etc [1, 3]. Example: For a normal city drive, where fuel economy, handling and sufficient traction is more important, front-wheel drive system is chosen. This paper will not delve deeper into those aspects, except to note that these types introduce different features and functions at the software level to realize their respective benefits.

**Feature variations for Cluster or CDC**

Each of the engine types, needs a list of supported features or functions to realize the end functionality, for that type on the vehicle. If the engine types are grouped into three segments—ICE, Hybrid, and Electric—it becomes evident that ICE, positioned at one extreme, requires a smaller battery, more in-car diagnostics for monitoring tailpipe emissions, and a heavier transmission. On the other extreme, a pure electric vehicle features a heavy battery pack, increased in-car diagnostics for monitoring battery health, and minimal transmission components. Most often everything that has to do with suspension, wheels, tires, steering, drive modes, lighting etc, is similar between ICE and electric cars. However, there are areas, like powertrain connected components, along with safety, convenience, braking, where powertrain specific selection effects the features and functions within the vehicle. At high level the choice of engine & transmission, along with various components in the overall vehicle would define the software aspects of it. However not everything is different across those as well. The following section summarizes those portions of features, which are different and their impact on Software aspects on Instrument panel cluster and related products and those which are more of less similar across. To name a few:

* Features, which are same across powertrains. (Example: Odometer, Speedometer, Indicators, Drive modes etc)
* Features, which seemingly look similar, however the architecture & systems behind it are very different. (Example: Distance to Empty, Average Fuel Economy, etc)
* Features, which are new or closely coupled to powertrains. (Example: Acoustic Vehicle Alerting System (AVAS), Fuel leak detection, Auto-Battery cut-off, gear Indicator, Tachometer, Low battery warning, Regenerative Gauge, etc)

**Advancements in Powertrain Technologies**

Though there are many variations in features and functions across powertrains, this is not a status-quo, as innovation in automotive powertrains continues to accelerate through integration of software, electrification, and materials science. These advancements allow manufacturers to meet stricter efficiency standards and deliver new customer value [2]. These can be broadly summarized into the following sections, as well explained in [2] and [10, 11, 12, 13]

* Rise of electrification & Hybrid systems.
* Lightweight materials and modular design.
* Advancing combustion technology (ICE, ex: variable value timing, Turbocharged, etc)
* Smart & Connected Powertrain.
* Advanced thermal management solutions. (Hybrid & Electric)
* Advanced EV Technology (ex: High efficiency motor, battery technology, two speed transmission, wireless charging, regenerative braking advancements)
* Integrated Control Systems: Use software to optimize torque delivery, shifting strategy, and energy recovery.

Each of these innovations improves the value proposition of modern vehicles, either through cost savings, performance enhancement, or reduced environmental impact. As observed above, many advancements have a direct impact towards software aspect of the overall vehicle architecture.

Vehicle Build phases

Vehicle build phases have been consistent across the last many decades, though there have been many changes in the underlying technological aspects of bringing efficiency in the overall vehicle build process. The various phases are concept phase, design & development phase, prototype phase, pre-production phase, production phase and lastly postproduction phase. These phases are common across the different vehicle architectures, including different powertrains. Though the phases are common, the procedures, tools, timelines when they occur and the stages within those phases are different across different powertrain. These variations are to be well defined and aligned as part of overall vehicle development process and as well to be cascaded as a crucial information during the overall software development, for products which are going to be common across those powertrains.

Software Development Life cycle

The software development life cycle (SDLC) [19], as depicted in Figure 1, is a structured process used to design, develop, and produce high quality, reliable, cost effective and within time software products in the software industry. This is also called software development process model [17,18]. Software Development Life Cycle (SDLC) models form the backbone of software engineering practices, guiding the systematic and structured approach to creating high-quality software products. The importance of SDLC models lies in their ability to provide a well-defined roadmap for software development projects, ensuring that each stage is carefully planned and executed.

Figure 1. Software Development Lifecycle phases [19]

There are various model types, right from linear, iterative, incremental, etc. In those, though there are variations in when a phase should occur and how interconnected is one phase to another, the phases as such are generic across those models. The focus is this paper is not to dwell in detail on the aspects of models, however, to allude to common aspect of software development lifecycle inspite of the model any organization might choose. The common phases are as described in the Figure 1 [19]. As described in this cited paper [18], the phases of software development life cycle are ‘planning’, ‘defining’, ‘designing’ , ‘building’, ‘testing’ and ‘deployment’. However as part of another research paper [19], the phases are ‘analysis & planning’, ‘definition of requirements’, ‘architecture design’, ‘implementation & development’, ‘product testing’, and lastly ‘operation and maintenance’. The phase names are not always the same or consistent across, as could be seen, there are subtle changes in the semantics. However, if one steps away from the semantic difference, the actual work / activity done in each of those phases are similar.

For the context of this paper, out of the six phases, the focus of this paper would be on testing and deployment phases. As part of next few sections, this paper explores those phases from a multi-powertrain product development context.

Software Testing

Software testing is defined as a process that consists of all lifecycle activities, both static and dynamic, concerned with planning, preparation, and evaluation of software products and related work products to determine that they satisfy specified requirements, to demonstrate that they are fit for purpose, and to detect defects [20] The core aspect of testing, which is very well needed is the test strategy, which means, A high-level description of the test levels to be performed and the testing within those levels for an organization or program (one or more projects).[21].

**Test Strategy**

A test strategy [21] in software testing is a high-level document that outlines the overall approach and guiding principles for testing a software application. It provides a framework for planning and executing testing activities throughout the software development lifecycle [21]. Essentially, it acts as a roadmap for the testing team, defining the objectives, scope, methods, and resources needed for effective testing. An effective strategy for automotive software testing encompasses a multifaceted approach that integrates various methodologies to ensure the safety, reliability, and performance of vehicles. As part of a standard [22, 23], the test strategies can be categorized into analytical, model-based, methodical, standard-compliant, reactive and consultative. Each of those has benefits and risks associated, however what fits best for the context of a project should be properly assessed and deployed. This often means that different test strategies apply to different test levels. Therefore, project teams should be able to evaluate strategies for suitability, determine the proper mix of strategies, and identify possible improvements where needed. The process of blending and adjusting test strategies should continue until alignment is achieved. This means alignment with the overall objective, the scope of testing, test types, and the test team.

**Test Type**

A software testing strategy outlines the approach to ensure a product meets quality standards. Test Approach [21] is the implementation of the test strategy for a specific project. It typically includes the decisions made that follow based on the (test) project’s goal and the risk assessment carried out, starting points regarding the test process, the test design techniques to be applied, exit criteria and test types to be performed. Of those, the important consideration as part of this paper is test types. Key test types in a strategy include functional, performance, security, and usability testing, along with levels like unit, integration, system, and acceptance testing. These tests are crucial for verifying functionality, performance, security, and user experience, ensuring a robust and reliable software product. From ISTQB [24], the different test types are categorized into ‘functional’, ‘non-functional’. Structural, and ‘re-testing / regression’.  In the context of this paper, the focus would be more on ‘functional’ and ‘re-testing / regression’ context

**Test Infrastructure**

Test infrastructure encompasses the essential hardware, software, and resources needed to support the software testing process effectively. This includes test environments, tools, equipment, and test data, all crucial for planning, executing, and evaluating tests [21]. A well-designed test infrastructure plays a vital role in ensuring efficient and effective testing. It is typically established before testing begins and is often managed by a specialized team or individual. By optimizing testing resources and processes, test infrastructure enables faster test execution, shorter release cycles, and quicker time to market.

The tester faces special challenges [25]. On one hand, he is expected to start testing as early as possible to find defects early in the development process. On the other hand, he needs a realistic environment to test the system and to find the defects that would appear in the completed product. The tester can solve this conflict by using suitable test environments that match the different development phases. In doing so, the tester can implement and execute his individual test tasks before the completely produced or developed electronic control unit (ECU) is available. By using different test environments, he can simulate situations and execute test cases that would be difficult to reproduce in the actual vehicle, for example, short circuits and open circuits in wiring harnesses or overload in network communications. Though there are various aspects of test environment categorized [25] into Model in the Loop (MiL), Software in the Loop (SiL), Processor in the Loop (PiL), Hardware in the Loop (HiL) and Vehicle in the Loop (ViL), for the scope of this paper, the focus would be on two types Software-in Loop (SiL) and Hardware-in Loop (HiL).

Software Deployment & Release

Though deployment & release sound similar, there are immensely different, not just in what activities are part of those, but also on how each of those activities are crucial for the overall success of delivery.

**Software Deployment**

This is one of the phases in the overall software development life cycle, and this phase [26] encompasses all the activities that make a software system available for use, including installation, configuration, running, testing, and making necessary adjustments. Software deployment is the technical process of moving code from one environment to another, typically from development to staging or from staging to production. It's primarily an operation focused on getting the code running correctly in its new environment. Different deployment strategies offer varying levels of safety, speed, and complexity. Choosing the right approach to software deployment depends on project specific needs and constraints. To name a few, nicely articulated by Steve Fenton and Tony Kelly [27] are, basic, rolling, blue/green, canary, multi-service and shadow deployments.

**Software Release**

A software release [26] is about making features available to users. It's a business decision that might involve marketing announcements, user training, or phased rollouts. Though its about providing the actual software to end user, the actual release content could be one sub-set of those already deployed. Infrastructure is the essence which would define the success of deployment and as well the release. Traditionally it was man-in the middle with lots of manual steps, however with technological improvements, today at lot of those steps are automated with many frameworks, which either support on-prime or cloud-based deployments. Two important concepts closely connected with release are ‘feature flags’ and ‘Over-the Air updates’.

**Feature Flags**

Feature flags [28] change the traditional deployment workflow by decoupling deploy and release, allowing new code to exist in a production deploy but not be executed and, therefore, not released (aka a dark launch). This provides a safety net that enables teams to both lower risk and deploy more frequently.

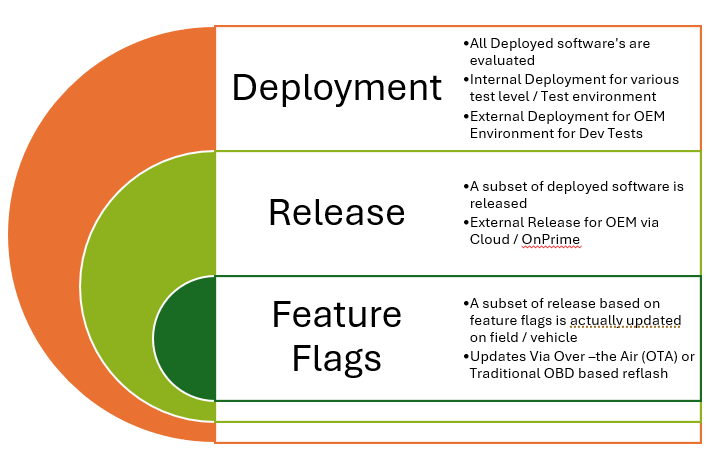


Figure 2. Software Deployment vs Release vs Feature flag view

As part of Figure 2, this paper captures the interconnected nature of deployment, release and feature flags. Example: Consider there are ten (10) software’s deployed. Of the ten deployed software’s, only two (2) software’s could be released (based on risk assessment and maturity, etc) and out of those 2 released software, based on feature flags a sub-set of features are actually installed on the targeted device / vehicle (based on priority, triage, etc). Feature flags

**Over-the Air updates (OTA)**

Over-the Air updates, are a crucial enabler for the whole Software release mechanism. The core steps involved within it are, Update Preparation, Distribution, Download, Verification, Installation, Post-Update Validation. Out of those listed steps, ‘Update preparation’, is part of phase where during the deployment itself Tier 1/OEM would validate the content.

**Software System Complexities**

Multi-powertrain vehicle architecture, require sophisticated software architecture to coordinate the intricate systems seamlessly. Developing software for these complex systems poses challenges, mainly due to ever growing changes in electric powertrain architecture and the efficiency improvements triggered in ICE architectures as well.  From few studies [29,30,31,32], the classification of such complexities can be grouped to the following:

* Hardware abstraction:Different powertrains (ICE, HEV, BEV) have a need for different sensors, actuators, and so, the requiring software are to be hardware-agnostic.
* Functional variability & Configuration: Each powertrain type has few functions which are core to that powertrain and few which are similar across powertrains, also the ability to configure those on vehicle pre-production and post-production as well.
* Software Deployment: To support across board of features and ability to have quick turnaround for feature enhancement, supporting OTA updates and diagnostics across diverse hardware and software configurations.
* Regulatory and compliance: Emission standards, safety regulations, and performance expectations vary by region and powertrain, and along with the complexity associated with compliance with standards like ISO 26262 and ISO/SAE 21434.

Though the complexities can be summarized into a few categories, as listed above, the paper focuses on drawing attention to ‘Functional Variability’ and ‘Software Deployment’. These two aspects are explored in terms of the challenges they present, along with a proposal on how they can be effectively mitigated, in the following sections.

Challenges in managing test strategy in multi- Powertrains

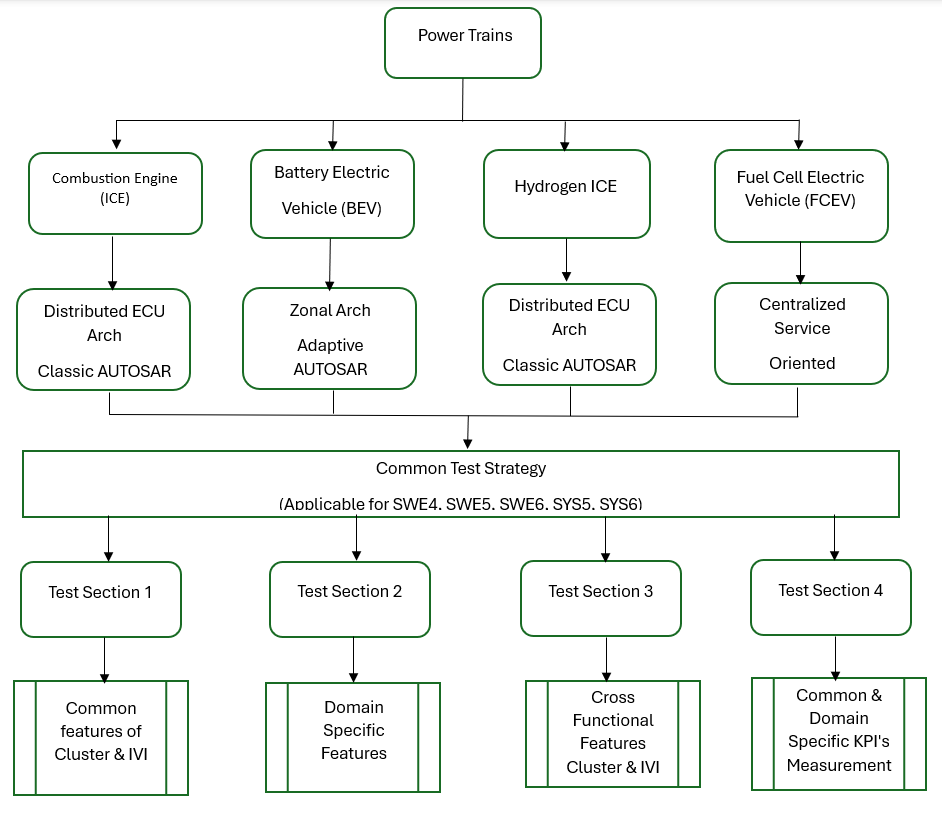
The project test strategy guides all testing activities within a project and details objectives, resources, schedules, and responsibilities. This strategy must be tailored to the unique requirements of the project. Key decisions include the selection of test levels, test types, and test techniques for static and dynamic testing and other test practices (e.g., scripted testing, manual testing, back-to-back testing).

Choosing the best approach for a test strategy can be a complex process that can be influenced by organizational test strategy, project context, and other aspects. Selecting and combining test levels, test types and test techniques is therefore critical to an effective project testing strategy as it significantly influences the efficiency and effectiveness of testing.

Challenges based on the Product:

1. Derive the common test approach for Individual Customer, Software and system specifications with respect to power Trian
2. Maintaining Requirements traceability
3. Revisiting the testcases based on the change requests
4. Upgradation of generic framework based on the test approach for individual projects or complex requirements

Challenges Based on the Project:

1. Common approach efforts should not impact individual project timelines
2. Strong Competent groups to be identified inside the project team members
3. Additional Efforts to have alignment with internal stakeholders and regular discussions

Proposed Optimal test strategy for multi- Powertrains

Aim to create the test strategy document which satisfies requirements of cockpit systems developed based on different types of powertrain vehicles, in this paper a well-defined test concepts and methodologies are mapped into traditional test strategy development stages and proposal of common test framework to achieve testability of multi powertrain architecture-based software implementations.

Key Elements to define test strategy

1. Master Test plan
2. Test strategy with Design
3. Software Architecture Differences
4. Test Approach
5. Test levels
6. Compliance Coverage

**Master Test Plan:**

1. Developing Master Test Plan:
   1. Identify the Scope of the testing of Clusters and CDC’s with respect Multi Power terrain architecture
   2. Differentiate the common and Domain Specific Features to be tested
   3. Identify the common test environment across domains (ICE, BEV, Hybrid, Hydrogen, FCEV)
   4. Identify competent resources and Responsibilities
2. Common Test strategy design for Multi Power Terrain Architecture

Multi Powertrain Architectures:

1. Internal Combustion Engine (ICE)
   1. Distributed ECU Arch Classic AUTOSAR
2. Battery Electric Vehicle (BEV)
   1. Zonal Arch Adaptive AUTOSAR
3. Hydrogen ICE
   1. Distributed ECU Arch Classic AUTOSAR
4. Fuel Cell Electric Vehicle (FCEV)
   1. Centralized Service Oriented architecture

Common Test Strategy:

1. Test Section1:
   1. Group Common features of Cluster & IVI
2. Test Section2:
   1. Identification Domain Specific Features
3. Test Section3:
   1. Cross Functional Features Cluster & IVI
4. Test Section4:
   1. Common & Domain Specific KPI's Measurement

Software Architecture Changes:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Feature** | **ICE** | **BEV** | **H2 ICE** | **FCEV** |
| OTA Updates | Limited | Extensive | Emerging | Extensive |
| Middleware | Classic AUTOSAR | Adaptive AUTOSAR / POSIX | Classic AUTOSAR | Adaptive AUTOSAR / Linux |
| Cluster Type | Analog-Digital | Fully Digital | Hybrid | Fully Digital |
| Data Bus | CAN | CAN + Ethernet | CAN | Ethernet |
| Cloud Integration | Minimal | High | Low | High |

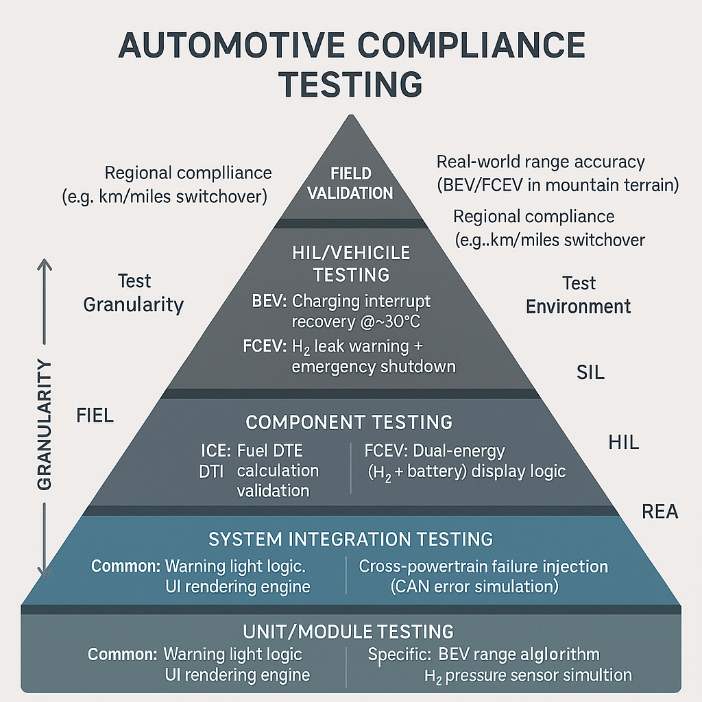
**Test Strategy Design**

Choosing a Test Approach Based on Test Levels

An experience-based testing approach depends on the domain, knowledge, and past experiences of the testing team and their understanding of the system. The team employs a wide array of heuristics, domain-specific data, and testing techniques to anticipate any possible pitfalls, user interactions, or edge cases that might lead to failures.

The Test levels and Proposals towards multi powertrain architectures proposals mentioned below.

**Test Levels:**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Level** | **Focus Area** | **Tools Used** | **Example** | **Multi Train Test proposal** |
| Unit | Individual functions | gTest, CppUTest, pytest | Speed logic | Grouping of common logic function testcases and Domain specific logics function testcases |
| Integration | Module interactions | CANoe, Debuggers | GPIO Register level verification, Fault injection tests for UDS, | Maintain Test suits for Different architectures and reuse the |
| Software Qualification | Simulated Software input and output to cluster | CANoe, SIL benches | Diagnostics, Software update  Gear info, State of charge, HMI | 1.Segregate AutoSAR and HMI Features  2. Customize the test scripts  With General functions based on the power trains |
| System | Full cluster behavior | HIL, test benches, Labcar | HVAC , Engine Status check, Communication between ECU’s | High level Generic Test specifications to be deployed based on type of Vehicle level and labcar |
| Acceptance | User & business requirements | OEM tools, manual scripts | UX validation | 1.FOTA, Software update over UDS and USB common use cases to be defined and tested  2. Feature and Variant specific checklists can be maintained to identify the use cases |
| Regression | Stability after changes | CI/CD, automated suites | Re-run tests | Common automated framework with multi test sections can be proposed to reuse test scripts across multi power terrain architecture |

Compliance Coverage

|  |  |  |  |
| --- | --- | --- | --- |
| **Compliance Standard** | **Test Level** | **Powertrain Coverage** | **Key Test Cases** |
| **ISO 26262** *(Functional Safety)* | Unit → HIL | ALL | - ASIL-B: Critical warnings |
| - Priority hierarchy |
| **FMVSS 101** *(Controls/Displays)* | HIL → Field | ALL | - Speedometer tolerance ±3% |
| - Units compliance (mph) |
| **UN R79** *(Assist Systems)* | Component → Field | ALL | - Steering input feedback |
| - Lane-keep alert timing |
| **GDPR/CCPA** *(Data Privacy)* | Unit → System | ALL | - Infotainment data anonymization |
| - Log validation |
| **SAE J2402** *(Hydrogen Safety)* | HIL → Field | H₂-ICE / FCEV | - H₂ leak detection < 3s |
| - Tank pressure warnings |
| **IEC 62196** *(EV Charging)* | Component → Field | BEV / FCEV | - Charging status accuracy |
| - Connector fault display |

**Configuration Example:**

   "powertrain\_profiles": {

       "BEV": {

         "features": ["regen\_display", "charging\_status"],

         "compliance": ["IEC 62196", "ISO 6469"]

       }

     }

feature toggles, variant management tools, and model-based development with parameterization

• Solution: Use robust bootloaders, rollback mechanisms, and UDS (Unified Diagnostic Services).

• Solution: Maintain region-specific configuration profiles and compliance testing suites.

Summary/Conclusions

**Summary / Conclusion:**

Deriving common test strategy is the challenging task with the consideration of evolving technologies in the automotive domain with respect to Architecture, Protocols, Specification’s, Vehicle Variants etc ,Its achievable with the right test approach , design and mitigation of risks can be key elements to deliver a quality product on time. A more methodical & standards compliant approach towards strategizing the overall approach is documented to aid projects to achieve the ability to scale it for multi-powertrain vehicle architectures.

The complex nature of multi-powertrain vehicle architectures introduces lots of ambiguity as part of the whole software development and especially in the aspect of testing. In this paper, we navigated through the various aspects of it, right from defining the feature similarity or variations, across powertrains, the types of tests to be conducted during various vehicle build phases, the test infrastructure to be harnessed or provisioned during each of those vehicle build phase, and lastly and most importantly the overall software deployment pipeline assisted with feature flags and Over-the air updates.

In summary, approaching a test strategy for multi-powertrain vehicle architecture is though challenging, is still achievalbe with a methodical approach towards organising the crucial aspects in a way, where we could start to connect the dots across each aspect, and their effects on others. By narrowing down the factors which effect the test management to the crucial attributes on ‘Functional Variability’ and ‘Software Deployment’, the paper exemplifies the importance on upfront classification during design stage. As well,

References

[1] Powertrain Systems: What is a Powertrain & How Does it Work?, UTI, 12th Aug’2021

[2] What is Powertrain in Automotive? Opal RT, 10th April’2025

[3] Vinkesh Gulati, Exploring the Automotive Powertrain Evolution, Opportunities, and the Role of ASDC in Skill Development, ASDC, November 2024

[4] Comparing Types of Powertrain: Efficiency, Performance, and Sustainability, Scalvy, May 16, 2024,

[5] Powertrains & fuel types explained, CarMax

[6] Sebastian Grzesiak, Adam Sulich, Car Engines Comparative Analysis: Sustainable Approach, MDPI, *Energies* 2022, *15*(14), 5170

[7] Shri Sudhendu J. Sinha (Adviser), Joseph Teja, Gautam Sharma, Souvik Bhattacharjya, Trinayana Kaushik, Arpita Pandey, and Mandavi Singh, Toni Zhimomi, Comparative Analysis of Electric Vehicles and Internal Combustion Engine Vehicles from Resource Efficiency Perspective, TERI, 15th Jul, 2023

[8] Ameya Joshi , Review of Vehicle Engine Efficiency and Emissions, SAE International

[9] Stuart Chubbock & Ralph Clague, Comparative Analysis of Internal Combustion Engine and Fuel Cell Range Extender, SAE International Journal of Alternative Powertrains,

[10] MTW, Innovations In Powertrain For The Mobility Sector, December 26, 2024

[11] Prof. Dr. Yaojung Shiao , Dr. Mahendra Babu Kantipudi ,Advances in Hybrid Electric Powertrain and Vehicle, *Energies* (ISSN 1996-1073)

[12] S. Ali Qasem Mohammed, S. Saleh Hakami, M. Kassas and M. M. Almuhaini, "Critical Review on Powertrain Technologies for Electric Vehicles: Classification, Broadly Adopted Topologies, and Future Challenges," in *IEEE Access*, vol. 13, pp. 71755-71772, 2025, doi: 10.1109/ACCESS.2025.3561577

[13] Graham Conway, Ameya Joshi, Felix Leach, Antonio García, Peter Kelly Senecal. A review of current and future powertrain technologies and trends in 2020, Transportation Engineering, Volume 5, September 2021, 100080

[14] Ahmad Syed,Tara Kalyani Sandipamu, et al, Recent Advancements and Challenges of Powertrain Technologies in Electric Vehicles Applications, October 2023, DOI: 10.1201/9781003354901-4

[15] Anup KumarH S, Difference Between the Powertrain of an ICE Vehicle and an Electric Vehicle, Skill-Lync, 27 Sep 2022

[16] Loukas, EV vs ICE - differences and similarities, 24 April, 2022,Properties of Material on a Supporting Steel Bar,” SAE Standard J1637, Rev. Aug. 2007.

[17] ISO/IEC/IEEE International Standard - Systems and software engineering -- Software life cycle processes," in *ISO/IEC/IEEE 12207:2017(E) First edition 2017-11* , vol., no., pp.1-157, 15 Nov. 2017, doi: 10.1109/IEEESTD.2017.8100771

[18] S, Shylesh, A Study of Software Development Life Cycle Process Models (June 10, 2017). Available at SSRN: https://ssrn.com/abstract=2988291 or http://dx.doi.org/10.2139/ssrn.2988291

[19] Shravan Pargaonkar, A Comprehensive Research Analysis of Software Development Life Cycle (SDLC) Agile & Waterfall Model Advantages, Disadvantages, and Application Suitability in Software Quality Engineering, International Journal of Scientific and Research Publications, 13(8):120-124, DOI: 10.29322/IJSRP.13.08.2023.p14015

[20] ISO/IEC/IEEE 29119-1:2022Software and systems engineering — Software testing Part 1: General concepts, Published (Edition 2, 2022)

[21] Standard Glossary of Terms used in, Software Testing, International Software Testing Qualifications Board, Version 2.4

[22] Improving the Testing Process, International Software Testing Qualifications Board, V2011 Release

[23] Test Management (Managing Testing, Testers, and Test Stakeholders), International Software Testing Qualifications Board, V2011 Release

[24] foundation Level Syllabus, International Software Testing Qualifications Board, V2011 Release

[25] Foundation Level Specialist , CTFL® Automotive Software Tester (CTFL®-AuT), International Software Testing Qualifications Board, 2.0.2 (English Edition) 04.07.2018

[26] What is software deployment?, ATLASSIAN,

[27] Steve Fenton, Tony Kelly Software deployment: 7 strategies, 5 steps, and why you should automate, Octopus Deploy, Monday, August 12, 2024

[28] Brian Rinaldi, What Are Feature Flags?, LaunchDarkly, NOV 17 2023

[29] Kenji Nishikawa, Kenji Hontani, Challenges towards New Software Platform for Automated Driving and High Computational ECU’s, Toyota, 20th-21st Nov. 2018

[30] David Obando, Hamzeh Alzu'bi, Erwin Carreón Vásquez, Qusay Alrousan, Mohammad Sami Alnajdawi, Thomas Tasky,High Performance Computer Based Software Defined Vehicle Hardware in the Loop Testing, SAE World Congress Experience, 2025-01-8137

[31] Aditya Pathak, Manufacturers must take the wheel to scale SDV success, SAE News, 2025-02-04

[32] Robert N. Charette, How Software Is Eating the Car, IEEE Spectrum, 07 Jun 2021

Contact Information

Contact details for the main author should be included here. Details may include mailing address, email address, and/or telephone number (whichever is deemed appropriate).

* Name: Rajaram, Saravanan,
* Mailing address: Visteon Technical & Services Centre, Ground Floor, A4, Shriram Gateway Office Parks SEZ, New Perungalathur, Perungalathur, Chennai, 600048
* eMail Address: [srajara3@visteon.com](mailto:srajara3@visteon.com)

Acknowledgments

If the Acknowledgments section is not wanted, delete this heading and text.

Definitions/Abbreviations

|  |  |
| --- | --- |
| OEM | Original Equipment Manufacturer |
| E/E | Electrical & Electronic |
| ICE | Internal Combustion Engine |
| HEV | Hybrid Electric Vehicle |
| BEV | Battery Electric Vehicle |
| FCEV | Fuel Cell Electric Vehicle |
| CDC | Cockpit Domain Controller |
| PHEV | Plug-in Hybrid Electric Vehicle |
| AVAS |  |
| EV | Electric Vehicle |
| SDLC | Software Development Life Cycle |
| ISTQB | International Software Testing Qualifications Board |
| MiL | Model in the loop |
| PiL | Processor in the loop |
| SiL | Software in the loop |
| HiL | Hardware in the loop |
| ViL | Vehicle in the loop |
| Tier 1 | a company that supplies components, systems, or modules directly to the Original Equipment Manufacturer (OEM) |
|  |  |

Appendix

The Appendix is one-column. If you have an appendix in your document, you will need to insert a continuous page break and set the columns to one. If you do not have an appendix in your document, this paragraph can be ignored and the heading and section break deleted.