Received 31 October 2021, accepted 12 November 2021, date of publication 16 November 2021, date of current version 20 July 2022.

Digital Object Identifier 10.1109/ACCESS.2021.3128431

FAT-AES: Systematic Methodology of Functional Testing for Automotive Embedded Software

KLEBER NOGUEIRA HODEL[1,2], JOSÉ REINALDO DA SILVA[1], LEOPOLDO RIDEKI YOSHIOKA[1], JOÃO FRANCISCO JUSTO[1], AND MAX MAURO DIAS SANTOS[3], (Senior Member, IEEE)

[1]Escola Politécnica, Universidade de São Paulo, São Paulo 05508-970, Brazil
[2]Mercedes-Benz do Brasil, São Bernardo do Campo 09680-900, Brazil
[3]Department of Electronics, Universidade Tecnológica Federal do Paraná–Ponta Grossa, Ponta Grossa 84016-210, Brazil

Corresponding author: Max Mauro Dias Santos (maxsantos@utfpr.edu.br)

This work was supported in part by Rota2030/FUNDEP - Fundação de Desenvolvimento da Pesquisa.

ABSTRACT Road vehicles have incorporated several functionalities over the last decade, with an increasing incorporation of electronic embedded systems. Most of those functionalities are controlled, managed, and supervised by distributed software, within many interconnected Electronic Control Units (ECUs). Within such context, new methodologies for tests of the distributed software functions must be developed to ensure proper performance at the integration level, complying with the requirement specifications. Many strategies have emerged to organize the multiple levels of software testing in automotive embedded systems, to reduce costs and improve its effectiveness and robustness, but there is a need for a methodology to structure, optimize, and plan the tests with real automotive criteria. This work aims to extend the multiple levels of testing concept and propose a method to analyze, design, and evaluate the application system upfront to derive a software testing plan. This method has been developed incorporating the automotive application characteristics, including functional safety requirements specified in the ISO 26262 standard to structure and plan the embedded software test in vehicle development. The proposed method combines an enhancement on plan testing strategy, matching requirement specifications, with adherence to the current methods used in practice by the automotive industry. Such a process will help the automotive industry to follow some concrete steps during validation, to optimize the test volume and facilitate documentation of developed activities, improving the safety and security of automotive systems in the early stages of automotive embedded software, when the details of each function are not yet implemented at the component level.

INDEX TERMS Automotive software, embedded system, testing, planning, quality, and V-model.

I. INTRODUCTION

The major transformations in road transportation systems have been associated with autonomous driving, electrification, car sharing, and connectivity. Particularly, embedded vehicular technologies have evolved substantially over the last decade. This has led to the increasing complexity of embedded hardware and software functions in vehicles.

With the increasing complexity of hardware and software, the topic of automotive software engineering has attracted much attention from scientists and engineers [1], [2]. Those developments must consider the incorporation of other technologies, such as artificial intelligence, computer vision, multi-domain sensors, and multi-agents [3], [4].

The associate editor coordinating the review of this manuscript and approving it for publication was Wasif Afzal.

The vehicular software has reached up to a few hundreds of millions of code lines, which is expected to continue increasing [5]–[8]. With the increasing complexity of intercorrelated systems that control and manage those vehicles, the carmakers require new and more sophisticated designing tools, particularly related to software development and testing [9]. Moreover, this leveraged the use of automatic code generation techniques, providing an easy and more functional way to develop and carry out maintenance.

Software functions are currently diverse and widespread in vehicular systems, ranging from low-level control software to advanced and critical safety features, such as advanced driver assistance systems (ADAS) and autonomous vehicles (AV) functions. All these functions require complex electronic architecture, distributed through the vehicle network in dozens of Electronic Control Units. Therefore, the development of such systems requires the effort and interaction of many teams, sharing responsibilities with the overall development of the complete vehicle. The standards and tools should also follow this trend to ensure the ability to analyze, design, and implement applications with high quality, safety, and performance.

Currently, connected cars allow original equipment manufacturers (OEMs) the opportunity to deepen their relationship with the drivers through a wide variety of infotainment, telematics, and vehicle diagnostics devices. Therefore, the next stage is to move beyond connected services, such as vehicle add-ons, to over-the-air (OTA) updates that define many aspects of the cars themselves, such as powertrain features, vehicle dynamics, and new on-board services. Software-defined cars (SDCs) could allow updating a vehicle's functionality through OTA updates, giving the vehicle the ability to continuously adapt to the needs of drivers and fleet operators. Consequently, software-based solutions could transform operational and ownership models.

The project management of interconnected embedded systems represents a major challenge, particularly in the automotive industry, which involves simultaneously several players in the process or workflow, from the OEMs to secondary and tertiary manufacturers. Additionally, the increase of the complex features must not compromise the vehicle's safety and should be developed in parallel with the introduction of safety standards, such as the ISO 26262 [10], [11], which provides a unified safety guideline for all automotive electronic systems. Therefore, all stakeholders should be involved to guarantee functional and timing-aware requirements.

Testing could represent as much as half of the overall costs of software development, and with the increasing complexity of software, the proportion of testing costs could rise unless more efficient testing methodologies are developed [12], [13]. The main focus of industrial research, to reduce cycle time and development costs, and simultaneously increasing software quality, is improving the software testing process [2], [14]. The whole electronic vehicle test cycle has several levels, and the main challenge is to identify a test strategy in which most of the failures could be detected in the beginning of the project.

The main contribution of this paper is a systematic methodology of automotive software testing: structures the complex vehicle software; organizing the test planning; creates labeling to specify functions, systems, and components; provides easy tracking of automotive tests with the labeling; provides easy reuse of automotive software; and defines the important factors for testing, reducing the number of required tests.

This paper is organized as follows: Section II presents the state-of-the-art of automotive software testing. Section III details all levels of tests to guarantee integrity. Section IV presents the proposed method. Section V presents the development of the structure and planning of the software test strategy derived from requirement specifications. Section VI shows how the proposed method can be applied to the structuring and planning of the multi-level software tests. Section VII presents final considerations of the proposed approach that motivates future work. Section VIII brings the conclusion and the perspective for future work.

II. THE STATE OF THE ART FOR SOFTWARE TESTING

Software testing and validation comprise a major part of development costs. Testing of embedded system software poses additional challenges due to the safety requirements when compared to the testing of regular software [9], [10]. Moreover, as the complexity of embedded software increases, the cost and effort for testing naturally follow. In this situation, the test strategy must be more efficient, reducing the cost and eliminating redundancies [15], [16].

Both Bringmann and Kramer [17] and Haghighatkhah at al. [1] have argued that a comprehensive development methodology along with seamless model-based development and integration tool support is required. Those authors also highlighted the scarce literature contribution to the advancement of research and support of future complex automotive systems. Similarly, Sundmark *et al.* [18] have discussed an automotive electrical system release process that would be very helpful for the industry, particularly for handling the testing of large configurations.

Research on test, verification, and validation in automotive domains revealed that contributions in this area have focused on methods developed for low-level model-based testing and verification [18], [19]. This indicated the challenges finding methodologies where the complete picture of testing in a full vehicle is considered.

Those authors have proposed an optimized verification strategy, to identify and mitigate gaps and redundancies in automotive software testing [20]. The strategy was based on the Use Case Tree (UCT) for each function, to better identify the use case and define the overall test strategy for each function to enhance the test coverage. The UCT method describes step-by-step interactions between systems and subsystems to map embedded system functions, which are written in a natural language for easy understanding. A model for applying UCT has also been presented [21]. This approach facilitates visualization, to identify each part of the system, helping to eliminate redundancies and gaps in the tests.

Another work has presented information on how an OEM organizes the general test activities [20]. Most OEMs use the V-model approach to system development and the test level is divided into several steps, as described in Fig. 1. Viewing the function scenarios to define the test strategy is only possible when the functionality requirements are available. There is a gap to be worked out when the testing strategy must be planned early in the project when most of the specifications are not available yet. It must be considered that the cost of testing activities represents a large part of the development and must be estimated to assess the overall picture of the project and the respective test costs.

Some works have introduced the multiple level of test strategy, mainly in the reuse of the test script [22], [23]. Those authors have also reported testing from different points

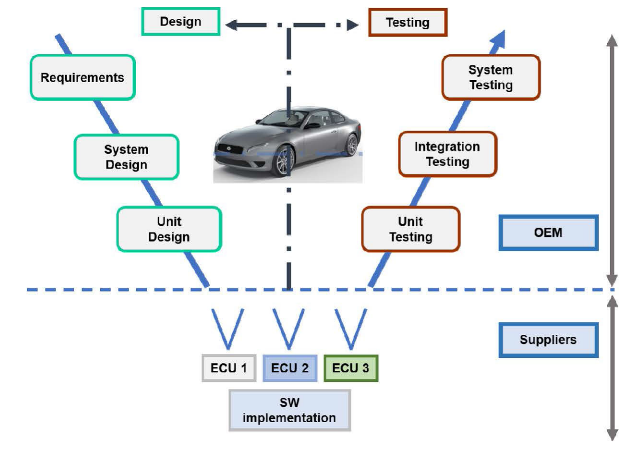

FIGURE 1. The V-model implementation on an OEM to distributed embedded systems.

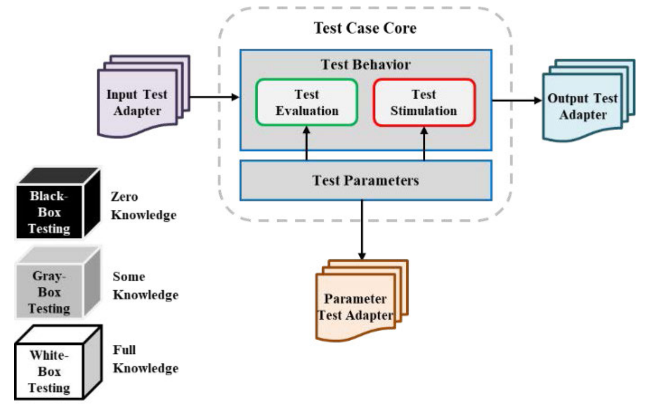

FIGURE 2. Structure of multi-level test case proposed by Pérez and Kaiser (2010) [23].

of view, such as integration, bottom-up, and top-down ones. The discussion goes beyond the scope of test-level details, proposing methods for defining a test strategy to reduce the effort, and sharing common test artifacts at all levels. It has also been discussed that the essence of the functional test is present at all levels, the test routine defined at the higher level can be reused for the lower levels, with the inclusion of adapters for inputs and outputs, and parameters that fit the route to other levels [23], as shown in Fig. 2.

When an automotive embedded system is created, in terms of software, it is crucial to make sure that the product is complete, secure, efficient, and of high quality. Therefore, there are three testing ways when considering software as a box. They are respectively:

a) **Black-box testing**: it consists of no knowledge about its internal structure or working, reviewing only the functionalities of an application;
b) **Gray-box testing**: it consists of some knowledge that compiles to test both the functionalities and functioning of an application;
c) **White-box testing**: it consists of full knowledge of an application and its internal structure, its processes, rather than just its functionalities.

The static and dynamic tests should also be considered. The test case should be defined according to the knowledge of the function architect or developer to avoid mistakes being made. It is advised to use tools and methods to build test cases automatically using, for example, model-based testing.

Any discussion about software testing of automotive electronic systems must include the ISO 26262 [10], [11]. It is a standard for the functional safety of electrical and electronic systems in production automobiles and has some specific requirements for software testing. For systems of functions where the Automotive Safety Integrity Level (ASIL) is A, B, C, or D, some defined test requirements must be carried, including the test level and the platform in which the test will take place. However, several works have not taken the ISO 26262 into consideration to define the test planning [20], [23]. This topic has been included in the methodology of this work.

We propose to use part of the methodology presented in Ref. [23], which has shown that the system-level specification is a common part validated in all tests. Additionally, the authors detailed a strategy at the low level to get effective verification, and each function contribution from each component was studied to define the test strategy methodology and give some traceability for the overall testing [20]. The combination of those strategies, the system vision [23] and the functional traceability [20], have been used as a reference, adapting to an upfront view, adding the evaluation of important elements of the automotive industry, including relevant safety issues of ISO 26262. This work proposes a methodology to design a systematic method to plan and structure software testing activities in automotive embedded systems.

III. DIMENSIONS OF AUTOMOTIVE EMBEDDED SYSTEMS

The automotive industry has used the standard V-model for the engineering processes of software-based system development and is highly dedicated to embedded systems [23]. The standard V-model is mainly used to describe the test process because it seamlessly links it to the development phases [24], [25]. The authors have stated that it constitutes the reference lifecycle model for the development and testing of embedded systems in the automotive domain [26]. Automotive system software testing generally takes place at several levels along the right side of a V-model. The exact test levels of the V-model differ slightly from project to project within an automotive organization and among different organizations [27].

In this paper, the V-model focuses more on the high levels from the component level up on the right-side testing activities. It consists of four test levels, starting at the bottom with component testing and ending at the top with acceptance testing, as shown in Fig. 3. The two intermediate test levels are system testing, representing the bundle of components related to a system, and vehicle system level, representing the bundle of vehicular systems.

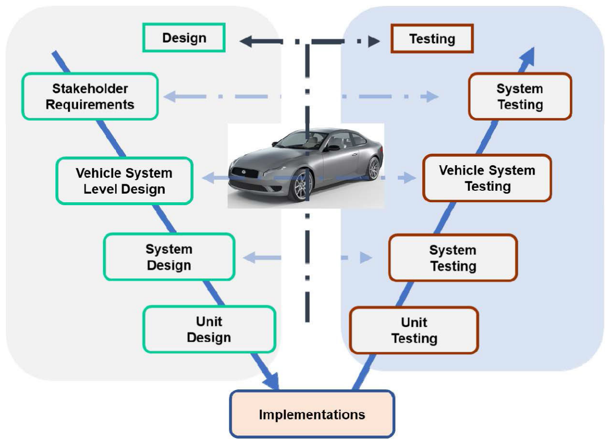

FIGURE 3. The V-model for the development of the automotive embedded system.

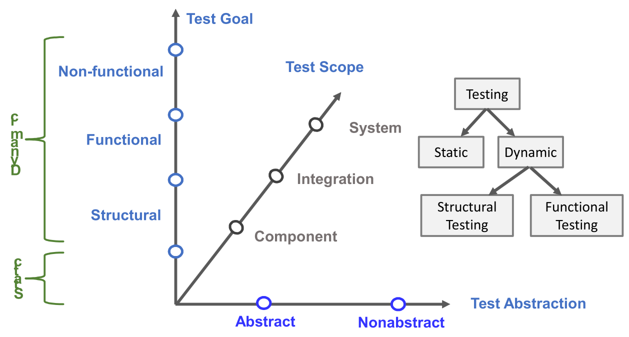

FIGURE 4. Selected test dimensions, as suggested by ZANDER ET AL., (2017) [28].

Due to the many different types of embedded system applications, it is difficult to have a single test classification that could capture all possible systems. However, the authors compiled a set of those tests, shown in Fig. 4 [28], considering the results of the authors [29]–[31].

A. TEST GOAL

The test phase is divided into two main critical activities when developing a product - verification and validation processes. The goal of the verification process is to assure that the product design was made correctly, while the validation process seeks to ensure that the customer's expectations have been considered. [32]. The validation checks whether the product addresses the customer's needs, while verification checks whether the product works as specified, either formally or not. Validation checks if the system, software, or hardware (or all together), meets the requirements, while verification checks the coupling with the user's needs and expectations - as recommended by the IEEE 1012 standard [33]. Verification is addressed using formal approaches within software engineering in academia but not widely used by practitioners, especially in the automotive industry.

The validation and verification software tests have different goals. They could be categorized as static testing, also called review, or a dynamic one, where the latter is based on test execution and further distinguishes structural, functional, or non-functional testing. After the review phase, the test goal is usually to check the functional behavior of the system, while non-functional tests show up in later stages [28].

- **Static Tests**- Testing is often defined as the process of finding errors, failures, and faults. Code errors could be identified without execution by just examining their source [34]. Some examples are revisions in the requirements, models, or test specification itself [28]. Dynamic testing, in contrast, is based on execution. In the automotive domain, these tests are executed at a level lower than the component level, and is therefore not within the scope of this work.
- **Structural Tests -**They cover the structure of the software under test (SUT), during test execution, for example, control or data flow. To achieve it, the internal structure of the system (e.g., code or model) must be known. Therefore, structural tests are also called white-box or glass-box tests [27], [34], [35].
- **Functional Tests -**They are related to the evaluation of the functional behavior of a SUT associated with the functional requirements. Unlike structural tests, they do not require any knowledge about the internal structure of the software system. They are, therefore, called black-box tests. In this category, functional safety tests are also included. Their goal is to determine the safety of a software product, and require a systematic, planned, executed, and documented procedure. Currently, functional safety testing is only a small part of software testing in the automotive field. With the introduction of security standards, such as IEC 61508 and ISO 26262, the significance of software security testing should increase significantly over the next few years.
- **Non-functional Tests -**They are similar to functional tests, but they are performed associated with the specification of system requirements [36]. In contrast to pure functional testing, they aim at assessing non-functional requirements, such as reliability, load, and performance, and are usually black-box tests. Nevertheless, for retrieving certain information, such as the internal clock, internal access during test execution is required. For example, during a robustness test, the system is tested with invalid input data that is out of range, to check whether the system is still safe and works properly. As a rule, robustness is ensured by dedicated plausibility checks integrated into the automotive software [32].

The contributions in this area have been mostly associated with methods developed for testing and verifications based on low-level models [18], [19]. This work focuses on testing the highest levels, starting from the component level (software and hardware integration) until reaching the system level.

B. TEST PLATFORMS

The test execution is managed by the so-called test platforms, which stimulate the test object with inputs and observe and analyze the outputs of the system. In the automotive domain, the test platform is a car with a test driver. The test driver determines the inputs of the SUT on driving scenarios and observes the reaction of the car, supported by special diagnosis and measurement hardware/software that records the data during the test drive and allows the behavior to be analyzed offline. An appropriate test platform must be chosen depending on the test object, purpose, or environment [28], [37].

- **Model-in-the-Loop (MiL)** - The first integration level, MiL, is based on the model of the system itself. In this platform, the SUT is a functional model or implementation model that is tested in an open-loop (i.e., without any physical model in the first place) or closed-loop with a physical model (i.e., without any physical hardware) [26], [38], [39]. This test checks the system in early development phases within a simulation environment.
- **Software-in-the-Loop (SiL)** - During SiL, the SUT is tested in a closed-loop or open-loop configuration. The software components under test are typically implemented in C or C++ programming languages. However, various high level programming languages, such Python and JavaScript, can be used to target microcontrollers and embedded systems. The codes are either handwritten or generated by automatic code generators based on implementation models. In this paper, our goal is mostly a functional test [38].
- **Processor-in-the-Loop (PiL)** - In PiL, the embedded controllers are integrated into the embedded devices with proprietary hardware (i.e., ECU). Testing on the PiL level is similar to the SiL level, but the embedded software runs onboard with the target processor or on a target processor emulator. Tests on the PiL level are important because they can reveal faults that are caused by the target compiler or by the processor architecture. It is the last integration level that allows debugging during tests in an inexpensive and manageable way [39].
- **Hardware-in-the-Loop (HiL)** - When testing the embedded system on the HiL level, the software runs on the final ECU. However, the environment around the ECU is still a virtual structure. The ECU and environment interact via the digital and analog electrical connectors of the ECU. This test on the HiL level aims to reveal faults in the low-level services of the ECU and the I/O services 21]. Additionally, final tests of components delivered by the supplier are executed on the HiL level, because the component itself is the integrated ECU [38]. HiL testing requires real-time behavior of the environment model to ensure that the communication with the ECU is the same as in the real application. The HiL could be for just a single ECU (sub-system) or with the interaction of many ECUs from one or many systems, or a complete HiL simulates the entire vehicle.
- **Vehicle** - Finally, the last integration level is the car itself. The final ECU is run on the real vehicle, which could be a prototype or pre-series unit of the production line. However, these tests, as performed only in late development phases, are expensive and do not allow configuration parameters to be varied arbitrarily [39]. Hardware faults are difficult to trigger and the reaction of the SUT is often difficult to observe because internal signals are no longer accessible [36].

Due to the complexity of the automotive systems and the many different features of each function, there are many possibilities to test the functions of vehicles in different combinations of test platforms. The goal should be to choose the most efficient scenario, where the balance between cost, time, and quality is achieved.

Rapid Control Prototyping (RCP) is a very efficient method to develop, optimize, and test new control strategies in a real environment quickly, without manual programming. The control software is designed and embedded on the RCP hardware. The developed software can be tested in real-time mode, controlling or managing the input-output (I/O) devices connected to another real-time hardware that hosts a physical plant. A controller prototype developed using a real-time simulator is more flexible, faster to implement, and easier to debug. Then, it can be tuned on the fly or completely modified. Besides that, the rapid control prototyping model is the framework that allows developing and testing automatically functions and features in a run time environment.

C. TEST LEVEL OR TEST SCOPE

As described in the beginning of this section, test levels are divided according to the V-model, where the steps are correlated to the project specification steps. Each function can present a different type of failure at each level during testing [34]. In Fig. 3, the test levels are divided into components, system, vehicle, and acceptance, focusing on an automotive application. Fig. 4 shows that the tests were divided into components and system integration, which is generally applied to the embedded system.

- **Component-level** - The component test level represents the complete individual ECU test where the hardware is integrated with the software and the ECU tested as a black-box, focusing on their I/O interfaces and data communication buses, mainly the CAN network, one of the main protocols used in the automotive industry. Considering that most of the ECUs are developed by a supplier, rather than by an OEM, this validation would be likely executed on the supplier side. Instead of requiring that the OEM retests that component, the supplier must define a test case to work in a committed test case with the supplier and a standard test report must be set to receive the results and keep traceability.
- **System-level** - At this level, all ECUs related to that system are integrated. Until that point, each ECU is integrated step by step. All interfaces are tested ensuring the interoperability of the ECUs. In this way, it becomes possible to use system-level functional test cases to verify that the desired system functionality has been implemented correctly. The main goal of this level is to test the system as a whole, in which the particular

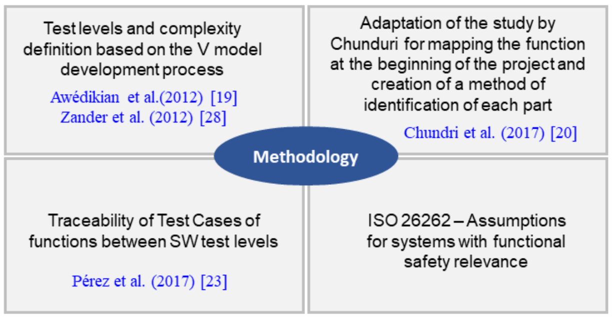
Test levels and complexity definition based on the V model development process
Awédikian et al.(2012) [19]
Zander et al. (2012) [28]
Adaptation of the study by Chunduri for mapping the function at the beginning of the project and creation of a method of identification of each part
Chundri et al. (2017) [20]
Methodology
Traceability of Test Cases of functions between SW test levels
Pérez et al. (2017) [23]
ISO 26262 – Assumptions for systems with functional safety relevance

FIGURE 5. **Base for the planning and structuring of multiple levels of software testing.**

focus is addressed on the ECUs interactions. At this point, many vehicle functions can be completely tested and approved, unless the ones in which tests with a real vehicle are necessary. On the OEM side, this level is very important for the integration of the supplier components. There are some situations where the vehicle test is mandatory and requires the same test procedure and contribution of the system test. In this case, the system test level could be eliminated and applied directly to the vehicle level test, reducing the overall effort.

- **Vehicle-level** - The vehicle test level must take into account some specific elements when the complete vehicle test homologation is the target. There are normally two ways to execute the vehicle level test: HiL in the laboratory or the real vehicle. The standard ISO 26262 recommends that the relevant safety functions be tested in the real vehicle, forcing the test in a more realistic condition.
- **Acceptance testing** - The acceptance test aims to receive approval from the customer and users. The tests mainly focus on user interface software, called the human-machine interface (HMI). The automotive literature of software testing generally does not focus on this level. However, with the increase of electronic functions in a vehicle, the acceptance test plays an important role in the final homologation.

IV. THE PROPOSED MODEL

This work proposes to use part of the contribution of Ref. [23], whose methodology shows that the system-level specification is a common part validated at all test levels. This helps the traceability of the test activities, based on a V-model development [20], which detailed a low-level strategy to get a view of each function contribution of each component.

The combination of the systemic view of Ref. [23], the functional traceability of Ref. [20], adapted to V-model, adding the assessment of the important points of the automotive industry, including relevant safety issues of ISO 26262 [10], comprises the contribution in this work, designing a systematic methodology to plan and structure the software testing in embedded automotive systems, as shown in Fig. 5.

V. STRUCTURE AND PLANNING OF SOFTWARE TESTS IN THE AUTOMOTIVE INDUSTRY

Due to the increasing complexity of electronic systems, the first proposal of this work is to bring a method that structures the interfaces of the functions, components, and systems, facilitating the definition of activities and responsibilities, based on the V-model approach.

The general electrical system of the vehicle can be defined as a set of systems, where each one is responsible for grouping a set of functions related to its theme, which has several components that are responsible for performing a function, as presented in Fig. 6.

Analyzing the V-model in Fig. 3, all test levels are directly linked to project specification steps. Based on that, a strategy is defined for the test execution that allows the identification of redundancies and gaps [20], [23]. The test strategy must be assertive and objective since the beginning of the project, considering that the test plan directly affects the project costs. Four layers are the basis for applying the method to organize and structure the complexity of the automotive system:

1. Function description;
2. System specification;
3. Systems, components, and functions mapping, and;
4. Definition of the application essential factors that influence the software tests in the automotive industry, as described in Fig. 7.

They are described in the next subsections. The strategy for defining the test levels for each function is then synthesized (Section V.E). Section V.F shows that the test activities are structured to enable the tracking, management, and control of testing activities.

The proposed method uses the minimum available information from the vehicle electrical/electronic system to define a strategy for testing software, looking from the OEM perspective. It is an ideal approach, allowing to test the project scope very early, keeping control of the complete project test plan, covering all functions, systems, and ECUs of the vehicle. A more detailed description of the function layers is presented in the following.

A. FUNCTION DESCRIPTION

The vehicle has several functions, which are written in an easily understandable way by the vehicle function users, especially in the initial phase of the project. The addressed user, in this case, can be a passenger, driver, operator, maintenance, or production workshop, among others.

In the initial phase of the project, the function is described independently of hardware or software. In industry, this is documented in a natural language, understandable by non-specialists. Its description does not consider how the function will be implemented and its interdependencies within the vehicle. Table 1 shows the proposed model to specify the functions. All functions of the vehicle should receive an identification (Fn), a name, and a basic description. As an example, the functions used in Refs. [20] and [23] are described.

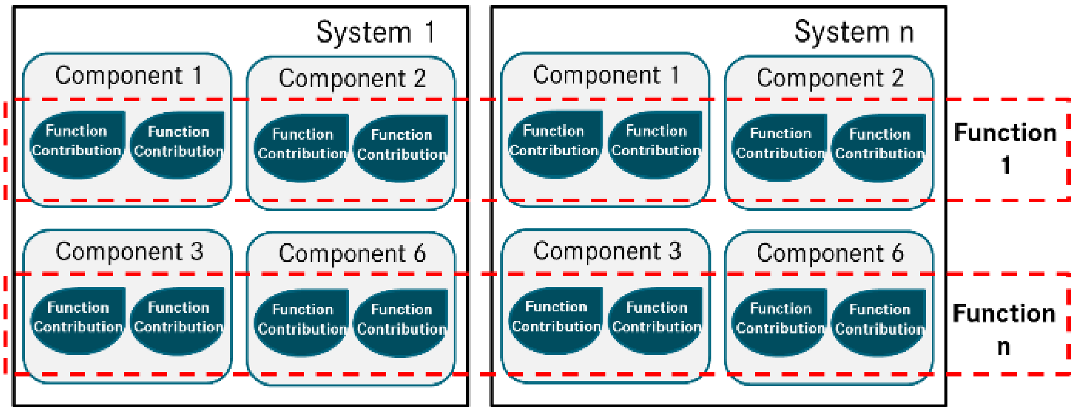

FIGURE 6. The entire vision of the system, component, function, and their integrations.

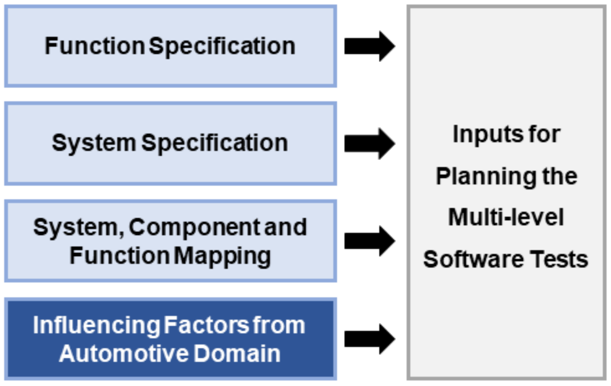
Function Specification
System Specification
System, Component and Function Mapping
Influencing Factors from Automotive Domain
Inputs for Planning the Multi-level Software Tests

FIGURE 7. The entire vision of the system, component, function and their integrations.

TABLE 1. Examples of vehicle function specification.

Function Identification (Fn)	Function Name	Short Description
F3	External Automatic Light	Turns on automatically external vehicle lights when the external illumination is dark.
F5	Fuel Level Display	Indicates the fuel level for the driver

B. SYSTEM SPECIFICATION

The vehicle contains several systems, and each system bundles a set of functions. A system is defined differently from one company to another, and it is usually a name correlated with a package of functions. Here, the proposal is to use a basic model to describe the system, according to Table 2, where the essential information is included, each system receives an identification, a name, a small description of the relationship of the system with the specific function, and in the last two columns the name of the functions that the system represents.

Every function has a system as an owner, even though other systems could participate in the function, for example, the fuel level has the contribution of the powertrain system (S3) to measure and check the failure of the fuel level.

TABLE 2. System specification.

System Identification Sn	Function Name	Short Description	Function related to the system	Description of the Function
S1	Driver Cockpit	Indicate the level of the fuel for the driver	F005	Fuel Level Display
S2	External Lights	Turn on automaticall y external vehicle lights when the external illumination is dark.	F003	External Automatic Light

C. SYSTEM, COMPONENT, AND FUNCTION MAPPING

The mapping, in this case, uses the concept of the electronic system of the architecture, which defines how the components and systems are organized. This includes the mapping of the function within the system architecture, presenting each component within its system, and showing the correlation of each function between the components. The mapping focuses on the interaction of important data of the vehicle's electrical system, function, system, component, and function contribution. The lowest level of information in this method is the specification of the contribution of the function, which represents the part of the function that the component is responsible for.

In this step, the function is mapped within each component and system, represented by a line connecting each component that participates in the execution of the function. In the example shown in the Fig. 8, we used the same function reported in the literature [20], [23]. The function architecture contains three systems, the driver's cockpit, power train, and

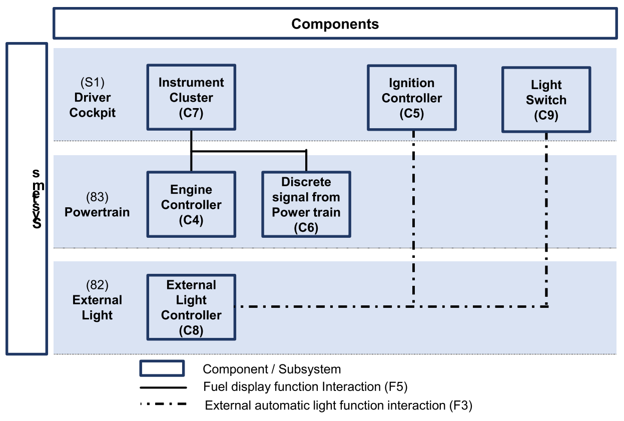

FIGURE 8. Example of the function architecture based on the system and component contributions.

TABLE 3. Function contribution and its relationship with the system, function, and component levels.

System Identification Sn	Main Function	Component involved (ECU)	Function Contribution
S1	F3	C5	Ignition switch identification
		C9	Light switch identification
	F5	C7	Display the fuel Level
S2	F3	C8	External luminosity identification
			Turn on/off the light
S3	F5	C4	Check the failures
		C6	Measure the fuel level

external lights. The driver's cockpit system represents the set of functions that are near the driver's position, with the components: instrument panel, ignition controller, and headlight switch. The power train system represents the functions related to the driving force of the vehicle, and the external lights system represents all the functions related to the lights that are outside the vehicle. The green line connects the components that participate in the external automatic light function and the blue line the function of the fuel level display.

This is a specification step of the system, where each component has its list of defined function contributions. Operationally, Table 3 is the basis defined to represent this system mapping. In the case of the powertrain system, it contributes to the function of the fuel level detection, but the system responsible for the function to show the fuel level is the cockpit system.

In addition to Table 3, a matrix is proposed in Table 4 for the mapping of the entire vehicle system, combining all information previously defined to point out how many components and systems participate in each function. One of

TABLE 4. ISO 26262 test recommendation.

	Methods	ASIL			
		A	B	C	D
1a	Hardware-in-the-loop	++	++	++	++
1b	ECU network environment (*)	++	++	++	++
1c	Vehicle test	+	+	++	++

the additional points in this table is the function contribution ID, FnSnCn, which represents a unique code for each function contribution and becomes part of the "DNA" function.

Here, the lowest level of function specification is the function contribution and the identification FnSnCn, such that this code allows identifying the component and the system responsible for that contribution. If the component has more than one contribution to the function, the Fn receives a point and the contribution number can be added after the point. For example, function F3 has three contributions of the components C5, C7, and C9. This method maps the function in the systems and components, where each contribution has a unique number. This number can be easily tracked, bringing a systematic and complete way to visualize the functions of the vehicle. This mapping approach could support any management activity, not only for testing, but for the complete development, production, and after-sale lifecycle. For example, if a vehicle in the field has some function failure, this can be easily tracked for specification history, tests, and reports. From left to right, we have the layers viewed as top-down from the user to component. Then, we might classify it in a system, function, and component. In order to ensure the integrity, it is suitable to define test modes at all levels.

This method also facilitates the development of a tool to manage and organize the software development process, considering that the identification has the "DNA" of the function. This represents a reference for the development of a framework to document each function, linked with components and the system, along with their interfaces.

D. AUTOMOTIVE CHARACTERISTICS

As described in Section II, the automotive industry has specific characteristics and challenges within the product development process that must be considered in the test strategy. Among them, one can highlight the characteristics of the application, security, and quality standard with some requirements for the test, relevance from the customer perspective, level of complexity of the system, and complexity of the test execution.

Safety-critical embedded systems have the factors that are proposed and might be assessed here: security levels, different levels of complexity, influences of the external environment or the operator on the functionality of the system, and, in some cases, the participation of customers. This is presented in Table 4.

1) FUNCTIONAL SAFETY

The main element for defining the test strategy concerning the functional safety factor is the risk analysis and assessment in the beginning of the project life cycle, where automotive safety integrity levels (ASILs) [10] are identified for each system. Based on this classification, which identifies systems with a risk of accidents, the requirements to avoid residual risk are defined and test methods are recommended to verify and ensure the system approval.

The software has no direct influence on the risks, but at the same time, it can be considered, in many cases, responsible for the security of the system level, since the software can pass faults through system elements. It is essential to understand the entire ecosystem, where the software function is applied. This information influences and contributes to the definition of the software testing strategy and is one of the factors used to define the test plan.

The ASIL identification of each function starts with a hazard analysis, in which all hazardous events are identified in the functional safety focus. There are several methods in the literature for risk analysis using the ASIL concept [23], [40]. Each hazardous event is classified according to the potential severity (S) of the lesions, the likelihood of injury risk is further classified according to a combination of exposure (E) (the expected relative frequency of operating conditions where the injury can occur), and control (C) (the relative probability that the controller can act to avoid injury).

The risk classification using the ASIL method recommended by ISO 26262 is used to evaluate each function of the vehicle. As described in Table 4, there are five possible results: QM, A, B, C, and D. When the level is A, B, C, or D, the system or component function has functional safety relevance. The QM level means that the risk associated with a hazardous event is not relevant to safety and does not require safety measures in line with the ISO 26262.

Once the specification of the function, system, and mapping is defined, the assessment of the risk level following the ASIL methodology should be applied individually for each function. This classification is one of the parameters defining the test strategy.

There are recommendations in the ISO 26262 [10] that define, according to Table 4, the tests that should be performed from each ASIL level. The ASIL levels are given by "++" indicates the method is highly recommended for the identified ASIL, "+" indicates that the method is recommended for the identified ASIL and (∗) means that the network environment can be a test bench partially or fully integrated with the vehicle system.

Following the ISO 26262 recommendation, it is mandatory to test the software in real conditions of the vehicle when ASIL is A, B, C, or D. However, in the A and B cases, the vehicle test could be eliminated if the component and the system-level test is performed in a test bench or HiL. In such cases, the goal is to provide evidence that the tests could be performed in a representative situation emulating the real vehicle conditions. However, this should be very well documented, since, in the event of a field accident, the OEM should be prepared to explain the details of the test. In our proposal, to avoid interpretation errors in such a relevant topic, if the function is ASIL, A, B, C, or D, the real test in the vehicle is considered mandatory. For this method, what is important is whether the function is relevant or not to functional safety no matter what the initial definition of the test strategy planning is.

2) CUSTOMER RELEVANCE

The customer must participate in the software testing strategy, introducing a co-design aspect to the proposed method. In practice, it should be pointed out that over the last few decades, the software has become part of embedded system functionality and innovation in a wide variety of domains, from telecommunications and medical devices to automotive systems [41]. Many products containing embedded software have been developed in several fields, increasing the relevance of stakeholder participation in product development, including software validation. These changes have led to a situation in which software engineering has become a vital subject in embedded systems with an important participant in innovation and market competition [41]. It has a direct impact on the usability of the vehicle functions. Within software testing, this factor must be considered.

Most methods in the literature do not consider customer validation in the test strategy [20], [23]. This validation is at one of the highest levels of the V-model, here called the acceptance test. For planning and definition of test strategy, the need for acceptance testing is evaluated with a question that must be answered in advance: Does the function require customer approval before being released for production?

3) SYSTEM COMPLEXITY

The embedded system in the automotive industry is an example to be explored when the subject is components and functions. Fig. 9 shows that several objects must be developed and tested, starting with the lowest to the highest level: models, software units, embedded software, hardware and software components, systems, and the vehicle itself. The tests can be performed at different levels and a systematic method must be developed to facilitate the understanding and definition of the tests.

This work focuses on the highest levels of software testing, such that the objects (modules, software units, and integrated software) are not part of the context, considering that in most cases these low-level tests have lower complexity and are usually managed by suppliers themselves, bringing the final report to the OEM.

Here, the complexity analysis is based on the number of interfaces that each function has, how many components and systems participate in the functionality of the function, and how many functions each component or system has. Some questions must be raised to structure the definition of tests at the component, system, and vehicle levels.

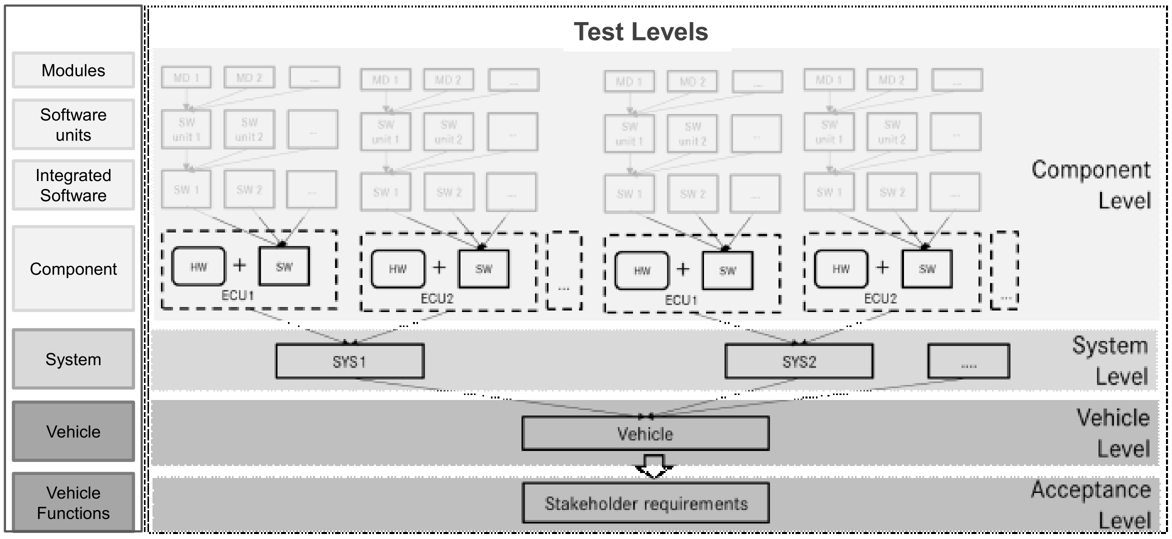

FIGURE 9. Electronic vehicle system structure.

Q1: How many components do the function interact? **A1:** This response is inserted in the context of the function mapping defined by the proposed model in Table V. If the function has only one component, the system-level test is unnecessary; when there is more than one component, a second question must be answered.
Q2: How many systems do the function interact? **A2:** In both cases, the response is contained in the mapping of the function defined by the model (Table V). There are two possible answers: • A single system: in this case, all the components are part of the same system and can be tested entirely at the system level, without considering the evaluation of interfaces between systems. • More than one system: in this case, a vehicle or HiL test is required.

4) DRIVER INFLUENCE IN THE SYSTEM

There are many function types in the vehicle, from very simple and passive functions to active ones that impact functional safety. Some of these functions have their behavior influenced by the driver. If the interference of the driver's action cannot be simulated in the laboratory, a vehicle test is necessary.

To facilitate the application of this factor in the proposed method, two questions are included to evaluate each function:

Q3: Is the function influenced by the driver? **A3:** If the answer is no, this factor does not influence the test planning, if yes, the next question must be answered.
Q4: Can the influence be simulated in the laboratory? **A4:** If the answer is yes, this factor does not influence the test planning, if not, the test in a real vehicle with the driver must be considered.

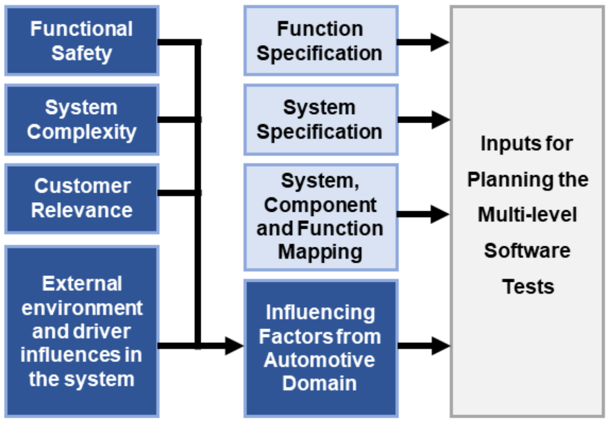
Functional Safety
System Complexity
Customer Relevance
External environment and driver influences in the system
Function Specification
System Specification
System, Component and Function Mapping
Influencing Factors from Automotive Domain
Inputs for Planning the Multi-level Software Tests

FIGURE 10. Parameters for the plan and strategy of multiple levels of tests in the automotive industry.

5) INFLUENCE OF THE EXTERNAL ENVIRONMENT AT THE SYSTEM

New technologies have more interactions with the external environment, which generate several variables within the vehicle system and are not always feasible for simulation in the laboratory.

It is possible to mention the elements of traffic, different types of user actions and reactions, as well as driving situations of a more dynamic nature, mainly towards autonomous vehicles [42]. Another example is related to connectivity or electromobility functions, where the city is part of the vehicle ecosystem. For instance, the structure of the Internet of Things (IoT), or system infrastructure of battery charging, which requires validations in real environments with the vehicle for final product approval and should be considered in test planning.

TABLE 5. Example of the function matrix and their distributions over components and system levels.

System		S1			S2	S3		...	A	B
Components		C5	C7	C9	C8	C4	C5	...		
Functions	F3	F3.S1.C5		F3.S1.C9	F3.S2.C8			...	3	2
	F5		F5.S1.C7			F5.S3.C4	F5.S3.C6		3	2
	...									
C		1	1	1	1	1	1	...		

A – How many components does the function interact with?
B – How many functions does the function interact with?
C – Number of functions contribution per component

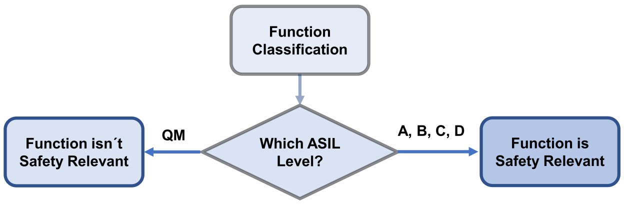

FIGURE 11. Workflow to check the safety relevance.

Most of the new vehicles have powerful networking, communication, and data processing capabilities, and can communicate with other vehicles or exchange information with the external environments in various protocols, such as Wi-Fi and Bluetooth. As a result, many innovative telematics services, such as remote security to disable the engine and remote diagnostics, have been developed to increase driver safety, efficiency, and comfort. One of the next steps in this evolution is the proliferation of solutions for autonomous vehicles [43].

In this scenario, the interfaces of the vehicle function, in many cases, go beyond the embedded systems and need the vehicle in a real environment to validate the software functions, since the simulation in the laboratory is not always possible. Similarly, to driver influence, some questions must be answered to define the test strategy:

Q5: Is the function influenced by the external environment?
A5: If not, this factor will not influence the test planning, if yes, the next question must be answered.

Q6: Can the influence be simulated in the laboratory?
A6: If the answer is yes, this factor does not influence the test planning, if not, the test in a vehicle in the real conditions of the application should be considered.

E. EVALUATION OF THE TEST REQUIRED FOR EACH FUNCTION

The previous sections presented important factors within the automotive industry that influence the definition of the test plan, a specification model of functions and systems, as well as a way of mapping the function within the system. This structuring of the methodology proposed here is summarized in Fig. 10.

This section presents the definition of the multilevel test strategy based on information from previous sections. The first flowchart, described in Fig. 11, evaluates factor 1,

TABLE 6. Function classification factors.

Symbol	Factors	Level
F1	Automotive Safety Integrity Level: risk analysis of a potential hazard by looking at the Severity, Exposure, and Controllability of the vehicle operating scenario	QM - No Safety relevant. A, B, C, or D - Yes-Safety relevant.
F2	Value to the customer / Does the function need customer validation before Production?	No (no perception) Yes
F3	Driver Influence / External Environment Influence in the functionality/ Can the function be easily simulated?	Yes(Could be easily simulated) No (Difficult to simulate)
F4	Interaction Level in the vehicle System	No (standalone - 1 ECU) Low (more than 1 ECU, same system) High (more than 1 system)

that is, the relevance of the function concerning functional safety.

This is one of the most important factors in automotive applications due to potential legal and monetary consequences when an accident results from a failure in the electronic system. The ISO 26262 [10] has specific requirements for functions that are relevant to functional safety and the test plan must follow those recommendations. The method proposed here divides the strategy and plan of multiple levels of tests into two parts: test strategy for relevant functional safety functions (ASIL A, B, C, and D) and test strategy for functions not relevant to functional safety (QM), as described in Table 6.

1) A PROCESS TO DEFINE THE TEST STRATEGY FOR SAFETY-RELEVANT FUNCTIONS

Functions relevant to functional safety should use the flowchart of Fig. 12. The first step of the flowchart evaluates factor 4 (system complexity). If the function interacts with more than one component of the same system, the system level must be applied. In the case where the function interaction is with different systems, the functionality is complete only at the vehicle test level, but there may be functions where the components have system-level interactions. In this case,

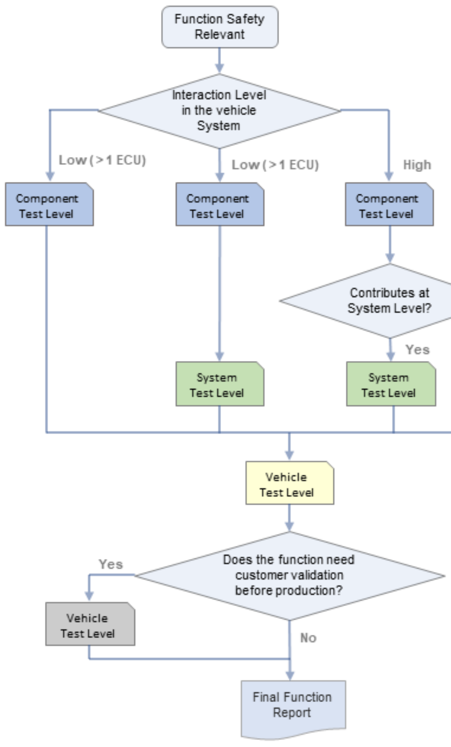

FIGURE 12. **Workflow for the definition of the test strategy for safety-relevant functions.**

TABLE 7. **Test level decision table.**

ID	Parameters				Test Levels Needed			
Function	A	B	C	D	E	F	G	H
Fn	Not	No	No	Not	X			
	Not	No	No	Yes	X			X
	Yes?	High?	High?	Not	X		X	
	Yes?	High?	High?	Yes	X		X	X
	Not	Low	No	Not	X	X		
	Not	Low	No	Yes	X	X		X
	Yes?	High?	High?	No	X	X	X	
	Yes?	High?	High?	Yes	X	X	X	X

X - Test needed
A – Safety relevancy
B – Interaction completely
C – Driver influence / External environment influence in the functionality
D – Customer validation needed
E – Component level
F – System level
G - Vehicle level
H – Acceptance level

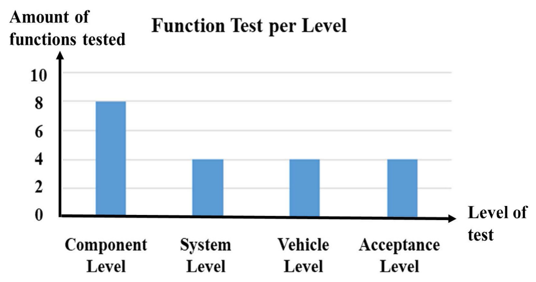

FIGURE 13. **Amount of function test per level.**

the system-level test is relevant and should be considered, otherwise the system level can be eliminated.

The last parameter to be evaluated is factor 2 in the flowchart of Fig. 12, customer relevance in function functionality. If relevant, the acceptance test should be considered to plan the participation of the customer in the function validation.

2) A PROCESS TO DEFINE THE TEST STRATEGY FOR NOT SAFETY-RELEVANT FUNCTIONS

The combination of the parameters defined in Section V.D and using the workflow in Sections V.E.1 and V.E.2 results in eight possible test scenarios, as described in Table 7. The vehicle level could be driven by three different parameters: safety, complexity, or external and driver influence. The question mark in the table is to indicate if one of the parameters in line is set, it is enough to define the need for a vehicle test. In summary, the sequence of activities to have the test-level definition is:

- Specify the function;
- Specify the system;
- Define the system architecture and mapping of the function within the vehicle system;
- Fill in the four parameters of the application (Section V.D);
- Use the flowchart proposed in Sections I and II to define test levels;
- Fill in the template (Table 7) with the results.

When Table 7 is filled with all the parameters of the function and the test-level scenario defined, a view of the amount of test to level can be easily displayed per component, system vehicle and acceptance, as shown in Fig. 13. The sequence of tests starts with the component, system, vehicle, and acceptance. The target is to bring as many as possible tests in the beginning of the project to reduce the risk in the late stages. The proposed methodology is structured with a criterion defined by the classification factor (Table 7) that can bring a wide range of tests in the beginning stage of the project to improve quality and address the requirements and specifications. Relativizing in a general form the level of each bar can take a certain risk level per test phase. This is used in the management of the project, for example, to invest more in simulation technology to reduce risks and the timeline of the project.

This method brings a systematic way to organize and control the activities and could be easily used even by a non-specialist. This brings more flexibility and organization for the project with an embedded system.

F. TEST TRACEABILITY

The term "traceability" is used here specifically for the management of the test activities and controls the alignment of the test cases. It is divided into two parts:

a) Identification of testing activities and
b) alignment of the test case.

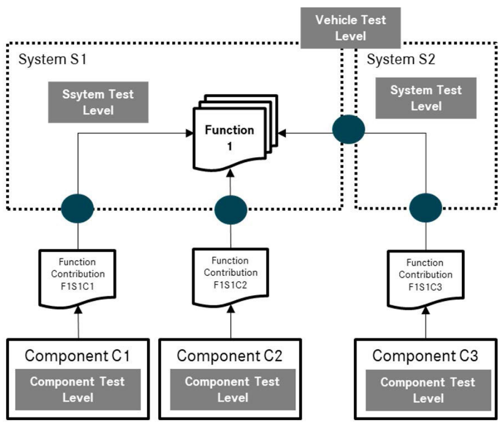

FIGURE 14. Example of interactions between systems and components from on function.

1) IDENTIFICATION OF TEST ACTIVITIES

To establish effective traceability, it is important to map the function, identifying each of the test steps and processes [44]. In the case of an automotive system, where there are usually requirements at multiple levels of abstraction [20], the component level only represents one part of the test activities at other levels that must have their scope and responsibility defined.

Once each function has defined the level of testing required to approve its functionality, a method must be defined to organize responsibilities and traceability between test levels, eliminating gaps, and avoiding test redundancies.

Each test level has a predefined sequence of steps to perform the functional tests, defined individually for each level, without interconnection with other levels or traceability of the results between them [23]. Fig. 14 presents a demonstration of the various interactions that a function can have between components and systems. In this case, two components C1 and C2 within the system S1 interconnect their contributions, the component C3 of the system S2 has another part that contributes to the function and the combination of S1 and S2 represents the function F1 entirely.

By analyzing the test plan for this example, especially factor 4 (complexity), the following phases for the function test should be presented:

- **First phase:** Component testing. Individual test for each component validates contributions of functions at the component level, component C1 to test the contribution of function F1S1C1, component C2 to test the contribution of function F1S1C2, and component C3 to test the contribution of function F1S1C3.
- **Second phase:** System test. Test of the combination of functions F1S1C1 and F1S1C2, to which the components are part of the same system S1. There are two possible scenarios at the system level: contributions of functions that do not interact at the system level, that is, nothing changes in the test scenario of the component level and system level; or the scenario having interaction. For example, it could be a simulated signal at the component level of C1 and the system level is received directly via the CAN Bus of C2.
- **Third phase:** Vehicle test, in this case involving only S1 and S2, which have the combination of all the function contributions that represent the F1 function.
- **Fourth Phase:** Acceptance test of the function by the client.

TABLE 8. Identification structure of the function at the test level.

Combination to identify the function at each level				
Component Test (CT)→	Function (Fn)	System (Sn)	Component (Cn)	
System Test (ST) →	Function (Fn)	System (Sn)	Component (Cn)	Component (Cn)
Vehicle Test (VT) →	Function (Fn)	System (Sn)	System (Sn)	
Acceptance Test (AT)→	Function (Fn)			

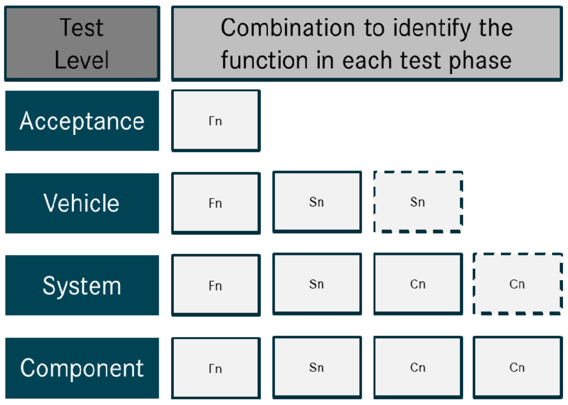

FIGURE 15. Illustration of the function identification at each level.

The mapping defined in Section III.C provides a method to identify each system, component, and function contribution, with a unique code. In the same way, we described in this work the unique identification for each test phase of the function, enabling the tracking of the activities by level. Considering that the documentation and history of each activity are necessary, mainly to structure the complexity of the software testing activities of the automotive embedded systems.

The test levels are identified by two letters: component test (CT); system test (ST), vehicle test (VT), and acceptance test (AT).

The identification of each function with the test levels must conform to the characteristics of the tests, as detailed in Table 8 and Fig. 15. At the component level, each activity only adds the letter CT to the function contribution code (CTFnSnCn). In the system test, the main identification on this stage is related to the components that are part of the function test. The identification of this step may have more than one component in the code (STFnSnCnCn). At the vehicle level, the highlight is on which systems represent the

TABLE 9. Code model of each test activity.

Component-Level Activities(CTFn)		System-Level Activities	Vehicle Level Activities	Accept Level Act.
Secondary	Main			
CTFn.1S1C1	CTFnS1C1	STFnS1C1C2	VTFn	ATFn
CTFn.2S1C1				
CTFn.nS1C1				
CTFn.1S1C2	CTFnS1C2			
CTFn.2S1C2				
CTFn.nS1C2				
CTFn.1S2C3	CTFnS2C3	STFnS2C3C4		
CTFn.2S2C3				
CTFn.nS2C3				
CTFn.1S2C4	CTFnS2C4			
CTFn.2S2C4				
CTFn.nS2C4				
CTFn.1SnC5	CTFnSnC5	STFnSnC5Cn		
CTFn.2SnC5				
CTFn.nSnC5				
CTFn.1SnCn	CTFnSnCn			
CTFn.2SnCn				
CTFn.nSnCn				

function, and the identification of this step may have more than one system in the code composition (VTFnSnSn).

Table 9 presents a model of the code and the interconnection between each activity in all test levels. At the component level, the second column of the function contribution CTFnSnCn is used, when there is more than one function contribution per component. The first column defines the secondary contribution that is the subdivision of the main part CTFn.SnCn. The index "n" after "Fn", refers to the sub-components.

2) ALIGNMENT OF THE TEST CASES

The test case is a sequence of specific test steps that examine all aspects, including inputs and outputs of a system or component, providing a detailed description of the steps that must be taken into consideration and the results that must be achieved. Normally, each test level of a function has a specific test case, which has no alignment with the content of the other levels.

In the automotive industry, most components are developed by suppliers. In that case, it would be ideal for the OEM to receive the fully tested component, as a black box, where only integration would be necessary. However, currently, these components are part of the combination of functions of the vehicle, and it is difficult to isolate the component, making the integration test necessary. To improve the efficiency and utilization of all levels, the test steps are needed to provide a model that the testing of a level is harnessed at a higher level, avoiding duplication of tests or even gaps in the tests.

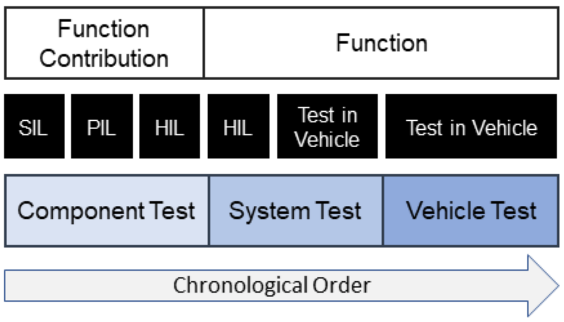

FIGURE 16. Type of tests for the components and systems of a vehicle in chronological order.

At the component test level, most of the work is performed by the supplier, where the complexity of the test environment is reduced due to focus on the component behaviors, failures of the source code, and integration with component hardware. Keeping most of the tests at the component level can identify the failures in advance, leaving to the other levels, like system and vehicle, only the integration and validation tests of the interface between components. Another important point is that costs to correct errors increase dramatically when they are discovered late in the development process. Fig. 16 shows a chronological order of tests in the field of the automotive industry, keeping most of the tests at the lowest level, which at the component level is quite beneficial.

One of the strategies to optimize the result of the test at each level is to create a test case link between the test levels and the proposal here is to use the multiple levels of test model [45], where a test sequence at the highest level of testing is defined as the common test core across all levels.

This proposal considers that the test case of a function consists of a common core part [45], called test case core (TCC), which is defined as being the test script at the highest level of the test function. For the tests of the lowest level, when needed, some adapters are added: output interface test adapter (OTA), input interface test adapter (ITA), and parameter test adapter (PTA), as shown in Fig. 2. The TCC is shared with all test levels, and the high-level test roadmap is reused. Only the adapters used at each test level are specific.

The TCC describes the test steps of a given function at the system level. The TCC is the basis of the test routine defined by the OEM. The lower levels would reuse this test routine by adding, when needed, the OTA, ITA, and PTA to tailor the interfaces and parameters. Both ITA and OTA must model the behavior of components that are not available at a given level of testing, to reach the functionally expected by the TCC. The PTA must map any test parameter that applies to the test object.

Here, the use of TCC is applied only between the test levels of the component, system, and vehicle. In the case of the acceptance test level, this concept is not used because this test does not have a standardized sequence that can be easily reused for other levels. When the acceptance test is required, it receives a specific test sequence.

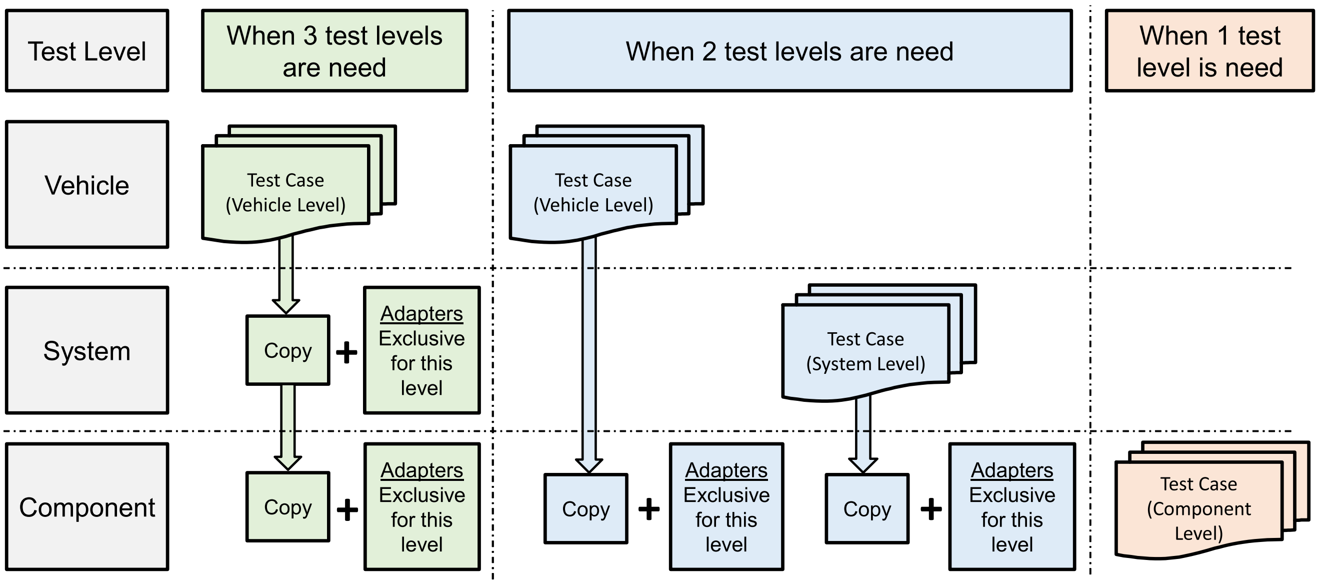
Test Level
When 3 test levels are need
When 2 test levels are need
When 1 test level is need
Vehicle
System
Component
Test Case (Vehicle Level)
Copy
Adapters Exclusive for this level
Test Case (System Level)
Test Case (Component Level)

FIGURE 17. Test case approach to reuse and adapt to other test levels.

As discussed earlier, most of the embedded systems in the automotive domain have their components developed by the suppliers, but the ultimate responsibility is the OEM. The biggest challenge lies in integrating the components with the general electronic vehicle system, where many functions interact with each other to obtain other functionalities in the vehicle. Adapting this approach [45], the functional test, which is the focus here, is controlled and aligned at all levels, as defined in Fig. 17. The design of the test sequence at the highest level becomes a reference for all other levels. This supports the goal of keeping all sequences of the core components from lower levels aligned and in the same direction as that defined at the highest level by the OEM.

The application of the model provides a methodology to align the functional test at all levels [45]. The code is added in the test activity that reuses the test case from another level, the component or systems that reuse the TCC receives at the end of its identification code RE, to identify the activity is aligned with the TCC of that function. The code of the component test activity is TCFnSnCnRE and in the case of the TSFnS2CnCnRE system

VI. THE APPLICATION OF THE PROPOSED MODEL

To validate the method of planning and structuring of multiple levels of tests, the application of the method is studied in a real function performing step by step the methodology proposed here. The selected function is the "automatic Light Control" that can show the entire example of our method, where there exists an interaction of systems and components. Any other function could be selected, knowing the number of systems and component interactions and the contribution of each component. Then, the workflow of the sequence follows as:

- Basic function specification;
- System specification;
- System architecture definition and function mapping within a system;
- Definition of function parameters (Section III.D);
- Definition of the test levels required for the function;
- Definition of the test sequence with the proposed traceability.

TABLE 10. Function description.

F003	Automatic Light Control (ALC)	Turn on automatically external vehicle lights when the external illumination is dark.

A. FUNCTION SPECIFICATION

The description must be at a high abstraction level, where no technical knowledge is required to understand what the function is. In practice, this type of description is performed by sales and marketing teams, which have direct contact with the customer. The model requires information on function ID, name, and description, as shown in Table 10.

To prevent repeated switching, the automatic light control function has a 10% hysteresis. When the external illumination is below the 60% limit, the darkness is detected, the headlamp is switched on and remains until the illumination increase above 70% (10% hysteresis), where the headlamp turns off and remain off until the illumination decreases to a 60% value. This is included in the function test case.

B. FUNCTION DESCRIPTION

In this step, the system that participates in the function is identified, in the ALC function, two systems are required: 1) S1 representing the system that makes the driver interface

TABLE 11. Description of the systems taking part in the ALC function.

Ident. Sn	System Name	System Description related to the Function	Functions related to system	Function Description
S1	Driver Cockpit	Control the activation of the external automatic light function	F003	External Automatic Light
S2	External Light	Identify the external light and activate the external light output	F003	External Automatic Light

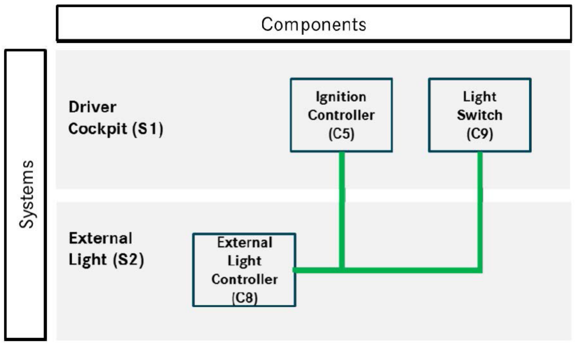

FIGURE 18. Function architecture.

of the function surroundings, ignition signal and also read the status of the light switch to detect when it is selected in automatic light mode; 2) S2 that identifies the external luminosity and also controls the drivers of external lights, as described in Table 11.

C. SYSTEM ARCHITECTURE AND FUNCTION MAPPING

There are two main points to be defined that are part of the system specification: the function architecture within the vehicle electronics system and the contribution of each component to the functionality of the function. Fig. 18 presents the architecture of the function and Table 12 presents the details of the contribution of each system and component. In this step, the information is still at an abstract level, but some elementary information about vehicle systems and components is required, which is usually the activity of the technical team in the product development department.

The S1 system has two components that participate in the function: the C5 ignition control component and the C9 control component of the light control knob. The S2 system has one component that participates in the function, the C8 component that is responsible for controlling the external light.

In Table 12, each part of the system is identified with a code, following the function's interaction logic and each part

TABLE 12. Function contribution.

System Identification Sn	Functions	Component involved (ECU)	Function Contribution	
S1	F3	C5	F3S1C5	Ignition switch signal
		C9	F3S1C9	Light switch signal
S2		C8	F3.1S2C8	External luminosity identification
			F3.2S2C8	Turn on/off the light

TABLE 13. Map of the ALC functions in the vehicle system.

System Identification			S1	S1	S2
Component Identification			C5	C9	C8
Function Identification	F3	F3S1C5	X		
		F3S1C9		X	
		F3.1S2C8			X
		F3.2S2C8			X

of the system receives a description of its participation, called a function contribution.

The function coding definition for each contribution is unique and can be easily traced and identified, it could be summarized the code has the "DNA" of function contribution. The mapping of the function within each system is shown in Table 13.

D. TEST LEVEL DEFINITION

With all the parameters defined in Table 13, considering that the function is relevant to functional safety, the workflow in Fig. 12 can be executed, resulting in the planning of the tests, where the need for each level of tests is evaluated for the function. The resulting flow to the ALC function is highlighted in red dotted line, shown in Fig. 19. The first point evaluated in the flowchart is the complexity of the function, which in this case is high, as the function is performed by more than one system (S1 and S2). In the second point, the contribution at the system level is evaluated. For the case of the ALC function, the combination among the components of each system has the same result as the vehicle level. Therefore, the assessment of the system level can be eliminated, evaluating only at the vehicle level, which will be mandatory, in any case, as we are dealing with the ASIL A classification. The last part of the flowchart evaluates the need for testing by the client. For the ALC function, there is no complexity or interference from the client that requires its validation, so the acceptance test is not planned.

The result of the ALC workshop defined the component level and vehicle level is required for approval of the function, which is summarized in Table 14.

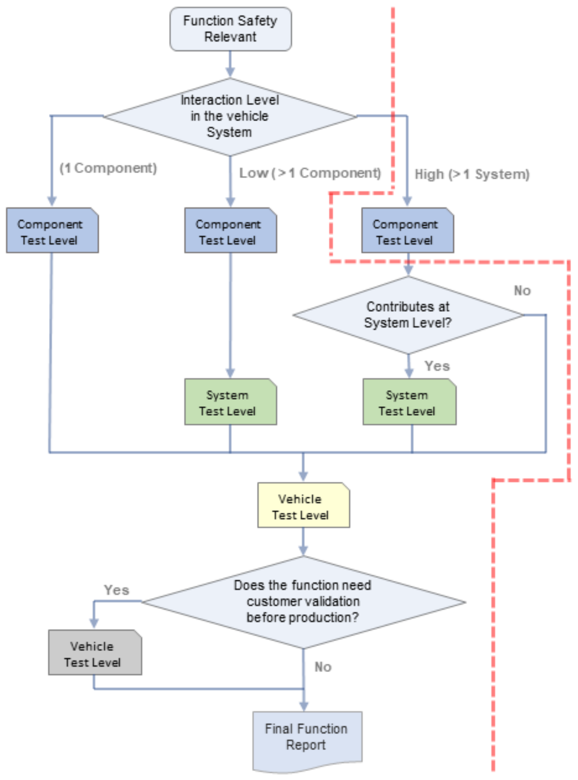

FIGURE 19. Test level workflow for ALC function.

TABLE 14. Test level required for ALC function.

	Test Levels Needed			
Function	Component Level	System Level	Vehicle Level	Acceptance Level
F03	X		X	

E. TEST TRACEABILITY

The traceability of the tests is divided into two parts:

a) identification of testing activities; and
b) alignment of the test case between the test levels of the V-model.

1) IDENTIFICATION OF THE AL FUNCTION TEST ACTIVITIES

In the methodology proposed, each test activity must receive a unique identification, according to the result of the evaluation performed in Section VI.D. The ALC function needs two test levels: component and vehicle ones.

For component testing activities, the three components involved in the function are considered. Three main activities must be carried out, according to the methodology presented in Section VF1. Each activity at the level of the component test receives a code with the initial letters CT, added to the function contribution code, which generates the CTFnSnCn code. Using information defined in Table 11, the test activities of the ALC function component are:

- In the C5 - CTF3S2C5 component;
- In the C9- CTF3S1C9 component;

TABLE 15. Test levels required for the ALC function before applying the reuse of the TCC.

Component Activities Level		System Activities Level	Vehicle Activities Level	Acceptance Activities Level
Secondary	Main			
-	CTF3S1C5	nn	VTF3S1S2	nn
-	CTF3S1C9			
CTF3.1S2C8	CTF3S2C8	nn		
CTF3.2S2C8				

TABLE 16. Test case of the ALC function used as TCC at the vehicle level.

Step	Action	Pass Condition
1	Set ignition switch to ON, light switch to AUTO, and illumination to LIGHT	Headlights = OFF
1	Set ignition switch to ON, light switch to AUTO, and illumination to DARK	Headlights = ON
1	SW-TC-3 Hysteresis 1 Set ignition to ON, light switch to AUTO, and ill	Headlights = OFF
2	Ramp the illumination toward PAR ILL THRESHOLD-PAR ILL HYSTERESIS	Headlights = OFF
3	Continue the ramp further reducing the illumination by 5%	Headlights = ON
4	Keep illumination constant for PAR MIN ON TIME	Headlights = ON
5	Ramp the illumination toward THRESHOLD + HYSTERESIS	Headlights = ON
6	Continue the ramp until the illumination reaches LIGHT	Headlights = OFF

- In the case of the C8 component, there is more than one contribution to function, F3.1S2C8, and F3.2S2C8, in this case, the main activity is defined representing the total participation of the component CTF3S2C8.

At the vehicle level, two systems participate in function S1 and S2. The identification of the definition of Section VI.D starts with the VT code added to the number of the function and the systems involved, which generates the Code VTFnSnSn. Using information defined in Table 11, the test activity of the ALC function component is VTF3S1S2.

Table 15 shows the codes for all activities of the ALC (F3) function. The ALC function has four test activities, three at the component level and one at the vehicle level.

2) ALIGNMENT OF TEST CASE FOR ALL TEST LEVEL

As proposed in Section V.F.2, the highest-level test case is used as the basis for all other levels. In the case of the ALC function, the test sequence of the highest level is the testing of the vehicle. In Table 16, one test case of the ALC function [23] is used to be the TCC at the vehicle level of this application.

At the vehicle level, the test is normally performed using all the real sensors and actuators. Only the external environment, which in this case is the light, can be simulated in a test bench

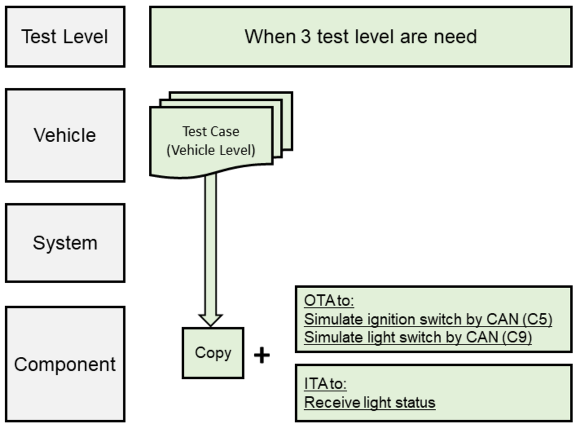
Test Level
When 3 test level are need
Vehicle
Test Case (Vehicle Level)
System
Component
Copy
+
OTA to:
Simulate ignition switch by CAN (C5)
Simulate light switch by CAN (C9)
ITA to:
Receive light status

FIGURE 20. Reuse of test case in the ALC function test.

TABLE 17. Test levels required for the ALC function considering the reuse of the TCC.

Component Activities Level		System Activities Level	Vehicle Activities Level	Acceptance Activities Level
Secondary	Main			
-	CTF3S1C5	nn	VTF3S1S2	nn
-	CTF3S1C9			
CTF3.1S2C8	CTF3S2C8RE	nn		
CTF3.2S2C8				

or the real environment application. The component-level test can be divided into three parts, the core part of the test is from the C8 component, the other ones are called auxiliary components (AC) C9 and C5. The C8 component is selected as the base for the TCC, adding an ITA and an OTA [45].

ITA and OTA adapt the TCC interface to the test interface by configuring the test case for the testing object, which in this case is for the C8 component. In the case of the ALC function, the illustration of the reuse of the test case at the component level with the description of the ITA and OTA is shown in Fig. 20.

An OTA is required to simulate the C5 auxiliary component, to stimulate the CAN-bus ignition and the C9 component signals, light switch signal by CAN. The adapter must follow the auxiliary component specification to stimulate the correct values, for example, when to send a CAN message on the CAN bus. After the definition of the ITA and OTA, the test of the component can be performed, following the test case proposed in Table 15, without modification in its sequence.

Additionally, a specific test must be performed on C5 and C9 to validate the full component-level test of this function. The vehicle test should only be performed when all components are approved at the component level to avoid identifying basic component failures at the higher test levels. In this application, the goal was to show the methodology of traceability and not to define and discuss the contents of the test case.

Table 17 presents the final code of all ALC test activities, considering the reuse of the test case in the C8 component, that is, only in the C8 component test was added the letters RE, registering that this test case is aligned with the TCC.

Applying the proposal in the literature in the structuring of the traceability [45], the identification of the reuse of the test case is explained in the code CTF3S2C8RE, defining an alignment with the tests at higher levels, and establishing a procedural mode and systematic connection of the tests between the levels, as defined in Section V.F.2.

F. ANALYSIS OF THE METHOD APPLICATION

The application of the proposed method in the case of the ALC function allows the specification to initiate the planning and structuring activities of the tests fairly fast, not needing a technical deepening. All stages can be carried out without a detailed specification of each component level, which shows clearly how to describe the interaction of the systems and components, what could be deployed in any other vehicle function, taking the concept of the modularity and portability, considering that any function has two levels of interaction with well-defined interfaces for system and component level, regardless of its complexity.

The methodology used to identify each part of the function is structured in a systematic way to map functions in the embedded system, facilitating the visualization of all parts in the vehicle systems.

The automotive factors add a qualifier in the definition of the test levels of each function, including a procedural, structured, and reasoned form to define the test strategy. This helps to discard the system-level test that is redundant in the ALC function.

The traceability applied in the ALC function transformed the complexity of the interconnections between the structured and mapped components and systems, providing control, management, test execution, and documentation of the results. In the first step, the ALC function was divided into three main contributions of the function, where each part receives a unique code: F3S1C5; F3S1C9; and F3S1C8. The composition of the code determines exactly which function, component, and system the role contribution belongs. In the second step, the use of the method of Ref. [45] for reuse of the test case, along with the method of coding the activities proposed here, brings a systematic alignment of the test activities.

VII. FINAL CONSIDERATIONS

This work presents a methodology for structuring and planning the automotive embedded software tests. We defined the relevant factors for testing and designing a systematic approach. This methodology aimed to define a functional test plan for the development of a complete vehicle, and alignment of the test activities. We proposed the methodology using the characteristics of the automotive domain. However, it could be extended to other applications, following the same reasoning, detailing each application and environment requirement. In the proposed methodology, two main points should be highlighted: 1) structuring and planning of the tests; and 2) traceability of the tests.

A. STRUCTURING AND PLANNING OF SOFTWARE TESTING

We have developed a method to identify all electrical and electronic function interfaces of the vehicular system and components. Starting from a high-level function, the method allows the mapping and structuring of all the vehicular functions. Each function is divided into their contributions, creating identification for each part of the function. It is assigned the proper role for each component, showing the correlation between components and systems. The proposed structure organized the embedded system, facilitating the visualization and understanding of a system, even if the system is complex, with many interactions between components and systems.

One of the contributions of this work was to identify the factors that influence decisions to determine what levels of testing are needed to validate each vehicle function. The identification of these factors allowed the creation of a formal strategy to define all levels of testing. This procedure improves the efficiency of product development, evaluating all test levels at the same time, eliminating gaps and redundancies, allowing the identification of the exact test plan of each function, with the customized application for each vehicle function.

One of the highlights is the functional safety, which is standardized by the ISO 26262 in the automotive industry, inserting a structured form of the definition of the risk level of accident for each function, and the application of the appropriate test levels to mitigate or identify the possible failures of each software function.

This approach provides a structured state-of-the-art methodology, identifying each part of the system, transforming the complex system into a fully identified and organized system. The mapping of the function by system and components built a basis for planning the test strategy and systematically defining what levels are required for each function. This framework facilitates documentation of development activities, which are required by ISO 26262, demonstrating robustness and transparency of all test steps applied in product development.

The proposed methodology to identify the test levels required for each function included a procedural, structured, and reasoned form to define the test strategy, facilitating the identification of duplication and gaps in the test planning.

Among the test levels of this methodology, the acceptance test was systematically included, creating a procedure to discuss client participation in the software tests for functions when appropriate.

All planning and structuring proposed here can be defined in the beginning of the project. It only requires a minimum of information, without the need to have a detailed specification of the components. This allows to carry out early planning and quantify testing costs. In this way, the project approval decision-making process becomes more robust and assertive. Since the costs and time needed to test each project are more precise, helping decision making, whenever the cost and time required for software testing are critical factors in the development process.

B. TRACEABILITY

Traceability is a complex topic when discussing software development. Here, we have brought the discussion to a more practical level. We organized the system view, step by step. We provided a systematic process to identify uniquely each system, component, and function. This facilitates the assignment task of each function role. All the high-level objects of the embedded system have received a structured code, which becomes the DNA of each part of the system, making the complex embedded system traceable.

Two main contributions have been presented: a methodology to identify all functions, the function contributions by systems and components; a method to link the test case of the component, system, vehicle, and acceptance test levels. The proposal, in both cases, defined a unique code model, which has in its structure the configuration identification of the embedded complex system. Additionally, the proposed coding method creates a basis for identifying the documentation and registration of software testing activities that can be easily tracked, for future projects or troubleshooting field failures.

VIII. CONCLUSION AND FURTHER WORK

The automotive domain is already quite complex, but the challenges should increase, focused on innovations such as autonomous cars, increasingly electrified vehicles, and integration of connectivity solutions, such as IoT. Within this scenario, external actions such as "cracker/hacker" in the vehicle system can be a challenge for future systems and may influence the planning of software tests, and it is necessary to study and include this vulnerability factor within the methodology definition of the planning of the test levels required for system approval.

The proposed method provides a structured base of identification and mapping of the embedded system with a unique coding for each part of the automotive system. This could be used as a basis for tracing and modeling future tools of development and documentation of embedded systems, and this framework could also be used as a base to create an SW development database, where all the vehicle development could be stored and with the identification proposed here.

REFERENCES

[1] A. Haghighatkhah, A. Banijamali, O.-P. Pakanen, M. Oivo, and P. Kuvaja, "Automotive software engineering: A systematic mapping study," *J. Syst. Softw.*, vol. 128, pp. 25–55, Jun. 2017, doi: 10.1016/j.jss.2017.03.005.

[2] S. Kriebel, M. Markthaler, K. S. Salman, T. Greifenberg, S. Hillemacher, B. Rumpe, C. Schulze, A. Wortmann, P. Orth, and J. Richenhagen, "Improving model-based testing in automotive software engineering," in *Proc. 40th Int. Conf. Softw. Eng., Softw. Eng. Pract.*, May 2018, pp. 172–180.

[3] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 427–438, Feb. 2013.

[4] X. Ge, F. Yang, and Q.-L. Han, "Distributed networked control systems: A brief overview," *Inf. Sci.*, vol. 380, pp. 117–131, Feb. 2017, doi: 10.1016/j.ins.2015.07.047.

[5] M. Broy, I. H. Kruger, A. Pretschner, and C. Salzmann, "Engineering automotive software," *Proc. IEEE*, vol. 95, no. 2, pp. 356–373, Feb. 2007, doi: 10.1109/JPROC.2006.888386.

[6] L. Lo Bello, R. Mariani, S. Mubeen, and S. Saponara, "Recent advances and trends in on-board embedded and networked automotive systems," *IEEE Trans. Ind. Informat.*, vol. 15, no. 2, pp. 1038–1051, Feb. 2019, doi: 10.1109/TII.2018.2879544.

[7] S. Maro, J.-P. Steghöfer, and M. Staron, "Software traceability in the automotive domain: Challenges and solutions," *J. Syst. Softw.*, vol. 141, pp. 85–110, Jul. 2018, doi: 10.1016/j.jss.2018.03.060.

[8] A. Pretschner, M. Broy, I. H. Kruger, and T. Stauner, "Software engineering for automotive systems: A roadmap," in *Proc. Future Softw. Eng. (FOSE)*, 2007, pp. 55–71, doi: 10.1109/FOSE.2007.22.

[9] Y. Dajsuren and M. van den Brand, *Automotive Systems and Software Engineering: State of the Art and Future Trends*. Eindhoven, The Netherlands: Eindhoven Univ. Technol., 2019, doi: 10.1007/978-3-030-12157-0.

[10] *Road Vehicles-Functional Safety*, Standard, ISO 26262:2018, ISO, Dec. 2018.

[11] R. Nörenberg, R. Reissing, and J. Weber, "ISO 26262 conformant verification plan," in *Proc. Conf., Informatik, Service Sci.*, 2010, pp. 515–520.

[12] W. Afzal, S. Alone, K. Glocksien, and R. Torkar, "Software test process improvement approaches: A systematic literature review and an industrial case study," *J. Syst. Softw.*, vol. 111, pp. 1–33, Jan. 2016, doi: 10.1016/j.jss.2015.08.048.

[13] V. Garousi, M. Felderer, and T. Hacaloğlu, "What we know about software test maturity and test process improvement," *IEEE Softw.*, vol. 35, no. 1, pp. 84–92, Jan. 2018, doi: 10.1109/MS.2017.4541043.

[14] M. H. Calp and U. Kose, "Planning activities in software testing process: A literature review and suggestions for future research," *Gazi Univ. J. Sci.*, vol. 31, pp. 801–819, Feb. 2019.

[15] D. Flemstrom, D. Sundmark, and W. Afzal, "Vertical test reuse for embedded systems: A systematic mapping study," in *Proc. 41st Euromicro Conf. Softw. Eng. Adv. Appl.*, Aug. 2015, pp. 317–324, doi: 10.1109/SEAA.2015.46.

[16] AUTOSAR Consortium. (2018). *AUTomotive Open System Architecture*. [Online]. Available: http://www.autosar.org

[17] E. Bringmann and A. Krämer, "Model-based testing of automotive systems," in *Proc. 1st Int. Conf. Softw. Test., Verification, Validation*, Apr. 2008, pp. 485–493, doi: 10.1109/ICST.2008.45.

[18] D. Sundmark, K. Petersen, and S. Larsson, "An exploratory case study of testing in an automotive electrical system release process," in *Proc. 6th IEEE Int. Symp. Ind. Embedded Syst.*, Jun. 2011, pp. 166–175, doi: 10.1109/SIES.2011.5953659.

[19] R. Awedikian and B. Yannou, "A practical model-based statistical approach for generating functional test cases: Application in the automotive industry," *Softw. Test. Verification Rel.*, vol. 24, pp. 85–123, Aug. 2012, doi: 10.1002/stvr.1470.

[20] A. Chunduri, R. Feldt, and M. Adenmark, "An effective verification strategy for testing distributed automotive embedded software functions: A case study," in *Proc. Int. Conf. Product-Focused Softw. Process Improvement (PROFES)*, in Lecture Notes in Computer Science, vol. 10027. Cham, Switzerland: Springer, Nov. 2016, pp. 233–248, doi: 10.1007/978-3-319-49094-6_15.

[21] S. S. Somé, "Supporting use case based requirements engineering," *Inf. Softw. Technol.*, vol. 48, no. 1, pp. 43–58, Jan. 2006, doi: 10.1016/j.infsof.2005.02.006.

[22] J. L. Barros-Justo, F. B. V. Benitti, and S. Matalonga, "Trends in software reuse research: A tertiary study," *Comput. Standards Interfaces*, vol. 66, Oct. 2019, Art. no. 103352, doi: 10.1016/j.csi.2019.04.011.

[23] A. M. Pérez and S. Kaiser, "Bottom-up reuse for multi-level testing," *J. Syst. Softw.*, vol. 83, no. 12, pp. 2392–2415, Dec. 2010, doi: 10.1016/j.jss.2010.07.028.

[24] D. Graham, R. Black, and E. van Veenendaal, *Foundations of Software Testing: ISTQB Certification*, 4th ed. Boston, MA, USA: Centage Learning EMEA, 2019.

[25] A. Spillner, T. Linz, T. Rossner, and M. Winter, *Software Testing Practice: Test Management: A Study Guide for the Certified Tester Exam ISTQB Advanced Level*, 1st ed. San Rafael, CA, USA: Rocky Nook, 2007.

[26] J. Schäuffele and T. Zurawka, *Automotive Software Engineering*, 2nd ed. Warrendale, PA, USA: SAE International, 2016.

[27] G. J. Myers, C. Sandler, and T. Badgett, *The Art of Software Testing*, 3rd ed. Hoboken, NJ, USA: Wiley, 2011.

[28] J. Zander, I. Schieferdecker, and P. J. Mosterman, Eds., *Model-Based Testing for Embedded Systems*, 1st ed. Boca Raton, FL, USA: CRC Press, 2012.

[29] J. Zander, "Model-based testing of embedded systems in the automotive domain," Ph.D. dissertation, Dept. Elect. Eng. Comput. Sci., Tech. Univ. Berlin, Berlin, Germany, 2009.

[30] H. W. Neukirchen, "Languages, tools, and patterns for the specification of distributed real-time tests," Ph.D. dissertation, Dept. Math. Natural Sci. Faculties, Georg-August-Univ., Göttingen, Germany, 2004.

[31] Z. R. Dai, "An approach to model-driven testing with UML 2.0, U2TP and TTCN-3," Ph.D. dissertation, Dept. Elect. Eng. Comput. Sci., Tech. Univ. Berlin, Berlin, Germany, 2006.

[32] C. Hood, S. Wiedemann, S. Fichtinger, and U. Pautz, *Requirements Management: The Interface Between Requirements Development and All Other Systems Engineering Processes*. Germany: Springer, 2008.

[33] *IEEE Standard for System, Software, and Hardware Verification and Validation*, IEEE Standard 1220, 2016.

[34] E. van Veenendaal, Ed., "Standard glossary of terms used in software testing," Int. Softw. Test. Qualification Board, The Netherlands, Tech. Rep. Version 2.1, Apr. 2010.

[35] B. Beizer, *Black-Box Testing: Techniques for Functional Testing of Software and Systems*. Hoboken, NJ, USA: Wiley, 2008.

[36] M. Vierhauser, J. Cleland-Huang, J. Burge, and P. Grunbacher, "The interplay of design and runtime traceability for non-functional requirements," in *Proc. IEEE/ACM 10th Int. Symp. Softw. Syst. Traceability (SST)*, Montreal, QC, Canada, May 2019, pp. 3–10.

[37] M. Milosevic, M. Z. Bjelica, T. Maruna, and N. Teslic, "Software platform for heterogeneous in-vehicle environments," *IEEE Trans. Consum. Electron.*, vol. 64, no. 2, pp. 213–221, May 2018, doi: 10.1109/TCE.2018.2844737.

[38] J. Kamga, J. Herrmann, and P. Joshi, "Deliverable D-MINT automotive case study-daimler, deliverable 1.1, deployment of model-based technologies to industrial testing," Eur. Commission-CORDIS, Tech. Rep., 2007.

[39] E. Lehmann and A. Krämer, "Model-based testing of automotive systems," in *Proc. IEEE ICST*, Lillehammer, Norway, vol. 8, Apr. 2008, pp. 485–493.

[40] Q. V. E. Hommes, "Assessment of the ISO 26262 standard: Road vehicles—Functional safety," in *Proc. SAE Government/Ind. Meeting*, Jan. 2012, pp. 1–19.

[41] P. Liggesmeyer and M. Trapp, "Trends in embedded software engineering," *IEEE Softw.*, vol. 26, no. 3, pp. 19–25, May/Jun. 2009, doi: 10.1109/MS.2009.80.

[42] M. Butenuth, R. Kallweit, and P. Prescher, "Vehicle-in-the-loop real-world vehicle tests combined with virtual scenarios," *ATZ Worldwide*, vol. 119, no. 9, pp. 52–55, Sep. 2017.

[43] E. K. Lee, M. Gerla, and G. Pau, "Internet of vehicles: From intelligent grid to autonomous cars and vehicular fogs," in *Proc. IEEE Internet of Things (WF-IoT), World Forum*, Sep. 2016, pp. 241–246.

[44] P. Rempel, P. Mader, and T. Kuschke, "An empirical study on project-specific traceability strategies," in *Proc. 21st IEEE Int. Requirements Eng. Conf. (RE)*, Jul. 2013, pp. 195–204, doi: 10.1109/RE.2013.6636719.

[45] A. M. Perez and S. Kaiser, "Reusing component test cases for integration testing of retarding embedded system components," in *Proc. 1st Int. Conf. Adv. Syst. Test. Validation Lifecycle*, Sep. 2009, pp. 1–6, doi: 10.1109/VALID.2009.19.

KLEBER NOGUEIRA HODEL received the bachelor's degree in electrical engineering from the FEI University Center, in 2003, and the master's and Ph.D. degrees in electrical engineering from the Universiade de Sao Paulo (USP), in 2008 and 2018, respectively. He is a Manager of New Technologies at Mercedes Benz Chassis Worldwide, since 2004, and a Professor at Fatec Santo André, since 2008. He has experience in electrical engineering, with emphasis in automotive electronics, systems and electronic controls, mainly working on the following theme: automotive/industrial electronics.

JOSÉ REINALDO DA SILVA received the bachelor's degree in physics from the Federal University of Bahia, in 1980, the master's degree in physics from the Federal University of Pernambuco, in 1985, the professional master's degree in interdisciplinary computer science from the Mills College, Oakland, CA, USA, and the Ph.D. degree in computer engineering from the Universiade de Sao Paulo (USP), in 1992. He was a Postdoctoral Researcher in computer science and systems engineering and design from the Department of Computer Science and the Department of Systems Design Engineering, University of Waterloo, Canada. He is currently an Associate Professor III at the Department of Mechatronics Engineering, Polytechnic School, USP. His research interests include engineering design: modeling and requirements analysis, formal methods for verification, knowledge engineering, formal modeling, petri nets, intelligent systems: artificial intelligence applied to automatic planning and scheduling, service science: service design, service life cycle, formal modeling, ''manufacturing service'' and its application in new architectures for industry 4.0 oriented to service. The applications of interest are in manufacturing automation, information systems and services.

LEOPOLDO RIDEKI YOSHIOKA received the B.Sc. degree in electronic engineering from the Instituto Tecnologico de Aeronautica (ITA), in 1984, and the M.Sc. and Ph.D. degrees in electronic engineering from the Tokyo Institute of Technology (TITech), in 1988 and 1991, respectively. He was an Assistant Professor with the Department of Information Processing, TITech, from 1991 to 1992, and a Research and Development Coordinator at COMPSIS Computers and Systems, from 1997 to 2010. He is currently an Assistant Professor with the Department of Electronic Systems Engineering, Escola Politecnica, Universiade de Sao Paulo (USP). His research interests include intelligent transportation systems (ITS), automotive embedded systems, and engineering innovation.

JOÃO FRANCISCO JUSTO received the B.Sc. and M.Sc. degrees in physics from the Universiade de Sao Paulo (USP), in 1988 and 1991, respectively, and the Ph.D. degree in nuclear engineering from the Massachusetts Institute of Technology, in 1997. He was a Visiting Associate Professor at the University of Minnesota, from 2007 to 2008. He is currently a Full Professor at the Escola Politécnica, USP. He has experience in computational modeling of nanomaterials and embedded electronics.

MAX MAURO DIAS SANTOS (Senior Member, IEEE) received the Ph.D. degree in industrial engineering from the Universidade Federal de Santa Catarina, Brazil, in 2004. He was a Postdoctoral Fellow in electrical engineering with the Universidade de Aveiro, Portugal, from 2005 to 2006. His career has spanned both academia and industry in the field of automotive, automation, and control. He held many positions in industry and academia, such as at IBM, Unileste, Volvo, and GM. Since 2013, he has been an Associate Professor with the Department of Electronics, Universidade Tecnológica Federal do Parana-Ponta Grossa, Brazil. His professional and scholarly works allowed to be produced in more than 100 journals and conference publications. His teaching and research interests include embedded systems, automotive systems, industrial automation, and autonomous systems. He is an Associate Editor of IEEE TRANSACTIONS ON INTELLIGENT VEHICLES.