MULTIDISCIPLINARY SCIENTIFIC JOURNAL
NÚCLEO DO CONHECIMENTO
REVISTA CIENTÍFICA MULTIDISCIPLINAR NÚCLEO DO CONHECIMENTO ISSN: 2448-0959
https://www.nucleodoconhecimento.com.br

# VERIFICATION AND VALIDATION OF EMBEDDED SOFTWARE IN AN AUTOMOTIVE CONTEXT: A SYSTEMATIC LITERATURE REVIEW

**REVIEW ARTICLE**

ARCANJO, Renato Rafael[1], MARTINS, Luiz Eduardo Galvão[2], FERNANDES, Dirceu Lavoisier Graci[3]

ARCANJO, Renato Rafael. MARTINS, Luiz Eduardo Galvão. FERNANDES, Dirceu Lavoisier Graci. **Verification and validation of embedded software in an automotive context: a systematic literature review.** Revista Científica Multidisciplinar Núcleo do Conhecimento. Year. 08, Ed. 09, Vol. 03, pp. 207-250. September 2023. ISSN: 2448-0959, Access link: https://www.nucleodoconhecimento.com.br/computer-science/embedded-software, DOI: 10.32749/nucleodoconhecimento.com.br/computer-science/embedded-software

## ABSTRACT

In automotive context, the embedded software Verification and Validation (V&V) is always a critical step for each project that involves testing solutions for new function, system optimization and compliance with legal requirements. However, automotive software V&V is laborious and time-consuming. Activities such as planning workshops at test tracks and public roads and functionality and durability tests require significant effort and robust coordination. The rigorous management and storage of test results are also a challenge. This review consolidates the state of the art on automotive software V&V to realize the most common standards in the industry and understand current testing concepts. The consolidated knowledge will help in the future development of a flexible V&V framework for embedded automotive software. A Systematic Literature Review (SLR) was performed by searching four digital libraries from 2011 to March 2022. Sixty-two papers were selected, which indicated that the automotive software V&V process is usually based on the ISO 26262 standard and that the software development life cycle V-model is the most common test platform in the automotive domain. Automotive software for a specific domain has been developed to cover a wide variety of vehicles. Variables from specific regions or countries can influence the entire V&V process for automotive software, such as differences in homologation requirements, infrastructure, driver behavior, customer desires, and electromagnetic force interference. The SLR identified specific characteristics of automotive software and regional factors that can affect the V&V process, as well as significant considerations to ensure correct decision-making, resource allocation, and support of team members.

Keywords: Verification and Validation, Automotive software.

## 1. INTRODUCTION

Rapid advances have led automakers to develop and produce vehicles with electronic components and embedded software rather than only hardware parts (HAGHIGHATKHAH *et al.,* 2018b) engineering and integration with various stakeholders are required to meet high reliability requirements (ROSA, 2013) and the current needs of customers and the market, which have resulted in very complex systems and features. Embedded automotive software is usually a low-cost solution that can be quickly implemented compared with hardware development (KRIEBEL *et al.*, 2018). An automotive vehicle is a complex system with many components such as sensors, actuators, and the Electronic Control Unit (ECU), and the hardware components are usually managed by embedded software (GRAF *et al.,* 2013). The increasing complexity of software-controlled systems has driven discussion on the robustness of current frameworks for testing software, especially owing to issues with autonomous cars. For example, faulty Uber software resulted in a fatal accident (EFRATI, 2018).

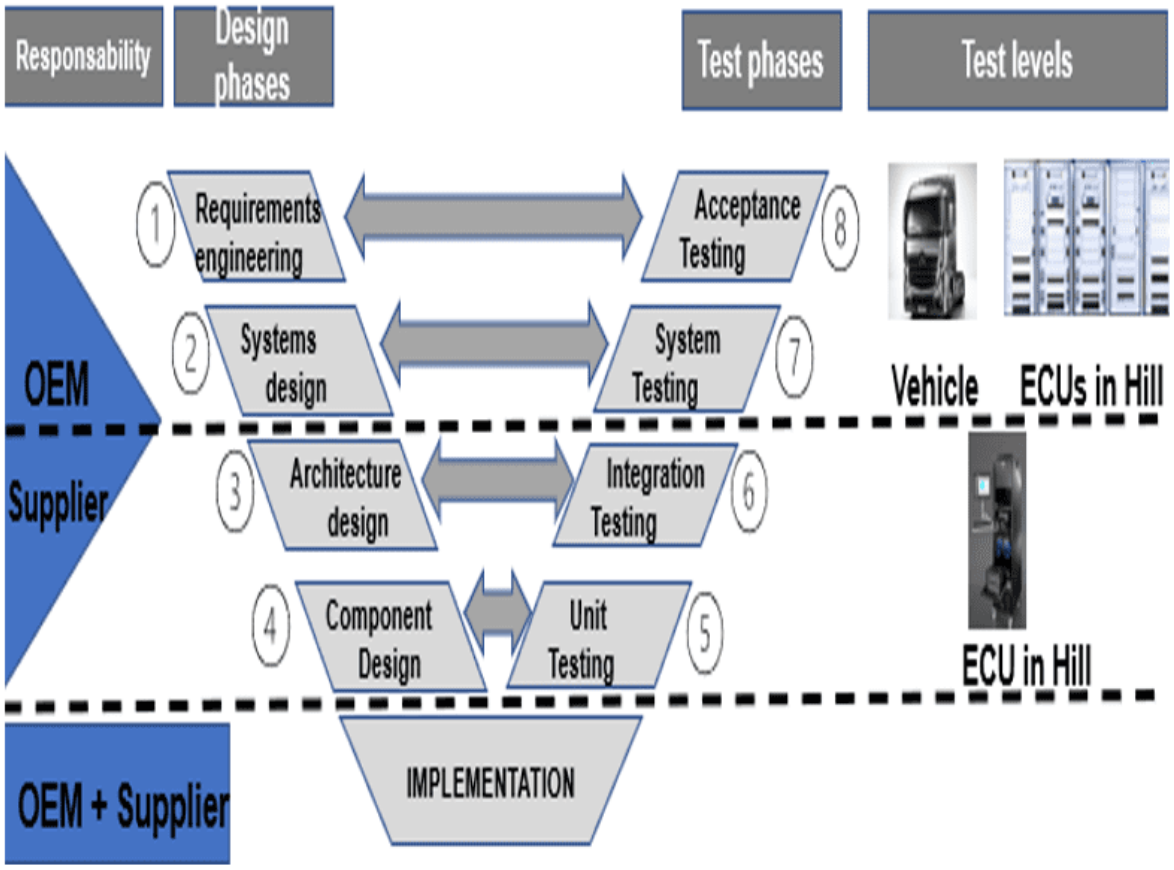
To meet the various requirements of a complex system, software development companies usually follow standards such as OSEK/VDX or AUTOSAR (CHOI, 2018). A key part of such international standards is the Verification and Validation (V&V) process, which must follow a consistent and rigid framework to ensure high reliability, robustness, and customer confidence. Otherwise, software errors can have disastrous consequences, especially in safety-critical software or critical applications (KOVACS, 2021).

(BOULANGER, 2018; DAO, 2018) and Shooman (2012) have noted that at least one-third of recalls related to automotive software is due to software errors, even this reality was from a decade ago, the example of Uber software issue shows that these problems still present nowadays. Thus, a more robust V&V process for automotive software is needed to avoid such problems. Such a process requires significant effort and time, and a wide variety of cases and tests need to be considered. The distributed

software development environment and specific challenges regarding software testing in the automotive sector have led to discussions on the development of a test methodology that addresses all of these issues and ensures reliable software to avoid a high incidence of recalls (JUHNKE, 2018; TICHY, 2018; HOUDEK, 2018).

Owing to its enormous number of features and systems, the basis for automotive software development and testing has been the Software Development Life Cycle (SDLC) V-model (LOCHAU *et al.*, 2014). As shown in **Figure 1**, the V-model encompasses the development lifecycle from end to end: from the elicitation of requirements to acceptance testing. It is divided into two types of phases that are connected to form the shape of the letter V. The design phases (left side) start with requirements engineering and end with implementation step. The test phases (right side) start with acceptance testing and end with implementation. (JUHNKE, 2018; TICHY, 2018; HOUDEK, 2018) noted that organizing the activities of automotive software development and testing requires clarifying the responsibilities of the supplier and Original Equipment Manufacturer (OEM). In addition, the OEM is responsible for phases 1, 2, 7, and 8, the supplier is responsible for phases 3–6, and both are responsible for implementation. From a software testing perspective, the OEM is responsible for high-level testing (e.g., system integration testing or validation in an actual vehicle), while the supplier is responsible for low-level testing (e.g., model or software integration tests) (JUHNKE, 2018; TICHY, 2018; HOUDEK, 2018).

Figure 1. V-model for software development and testing in the automotive




Source: (JUHNKE, 2018; TICHY, 2018; HOUDEK, 2018).

Before a V&V methodology can be developed for automotive software, it is first necessary to define what automotive software is. Durisic *et al.* (2013) defined automotive software as a mixed-critical system that contains safety functions and regular relevant functions for the user. These two types of functions often interact with each other, so testing the interface between them and establishing their integration is a critical step for software development. Choi (2018) defined automotive software as a controllable software for specific vehicle parts normally controlled by an ECU. The operating system is generally developed by following a rigid international standard such as AUTOSAR (2017). However, each ECU is controlled by different software to satisfy varying customer requirements. Hence, V&V must ensure that each ECU complies with the regular operating system and works correctly when interacting with other ECUs. Examples of subdomains with their own ECUs include safety components

(e.g., anti-lock brake system, electronic stability control, active cruise control), the powertrain, and infotainment. Kodali *et al.* (2013) noted that high-end vehicles may have up to 70 ECUs that exchange Controller Area Network (CAN) signals in online mode and control safety-relevant functions or regular user functions, such as the radio or telephone calls. The evolution of autonomous, electric, and hybrid vehicles is expected to increase the complexity of automotive software even further.

To ensure that the V&V process results in low-cost, fast, and reliable automotive software, the tests can be divided into three main blocks:

**Functional tests** are black-box tests that assess if a software meets requirements for a specific set of data (DEVMEDIA, 2012).

**Application tests** assess if the embedded software works with specific vehicle variants and different parameters and applications encountered under real road conditions (THURIMELLA, 2013; BRÜGGE, 2013). In addition, this specific test is also used to evaluate the software interaction with many mechanical parts manufactured by different suppliers and the characteristics that can interact with each other (LIDA *et. al*, 2016). Such tests must cover the considerable variability of vehicles and software among automakers.

**Homologation tests** assess whether software satisfies specific legislation requirements before market launch (STAHL, 2021; DIERMEYER, 2021), such as CONTRAN 519 for brakes or CONTRAN 641 for electronic stability control. Because such tests are normally performed on regular dates, the OEM should conduct a pre-evaluation to confirm the product's reliability before proceeding to the homologation tests for approval.

In contrast to automotive software, the V&V process for regular software is divided into two main blocks (DELAMARO, 2007; JINO, 2007; MALDONADO, 2007):

**Static tests** assess software against a specification or functional model.

**Dynamic tests** assess the overall behavior of software while it is running.

In the automotive industry, static tests check the response of actuators based on the execution of embedded software or driving request, whereas dynamic tests evaluate the dynamic behavior of a vehicle in running situations.

We present a Systematic Literature Review (SLR) in which our main objectives were to identify the most common and efficient types of automotive software tests applied in the automotive industry, flexible frameworks that can be used to manage the test results, and the most relevant standards to the current state of the art for testing embedded automotive software. The article was organized in the following way. Section 2 will demonstrate the background and related items. Section 3 will present the research methodology and describe how it was set up. Moving on to section 4 which will demonstrate the results as well as the RSL analysis. Finally, section 5 will provide the conclusion of this article.

## 2. BACKGROUND

## 2.1 DEFINITIONS

To set the scope of this SLR, we first define the most relevant terms in alphabetical order:

**Automotive Software Engineering (ASE):** Automotive software has some specific characteristics (PRETSCHNER *et al.*, 2007), such as its heterogeneous nature, the significant number of communication processors, the need to handle large numbers of vehicle variants, and specific requirements related to safety, security, and reliability.

**Automotive Safety Integrity Level (ASIL):** It is part of the ISO 26262 - Functional Safety Standard (SYNOPSYS, 2021) and is used to classify the functional safety of a vehicle. ASIL A has the lowest level of risk to the user, and ASIL D requires the highest level of safety for the user and has a high probability of risk to the user and surrounding people.

**Automotive Open System Architecture (AUTOSAR, 2017):** This is a partnership among many companies in the automotive industry (BAHIG, 2017; EL-KADI, 2017),

such as BMW, Bosch, Daimler, Ford, Volkswagen, and Toyota. AUTOSAR is an open industry standard for automotive architecture. Specific standards rely on informal and semi-formal representation, which is also known as the Unified Model Language (UML), (ARTS, 2015; MOUSAVI, 2015).

**Homologation or Type Approval:** This is a ratification process to ensure that a product or system meets the minimum requirements of a specific standard or regulation of a specific country/continent before commercial production (KILLIAN, 2018; IEEE, 1990). There are two different types of homologations (REDMILL, 2007; ANDERSON, 2007): via a third party (e.g., TÜV Rheinland or Applus+ IDIADA) that confirms that the vehicle/software was tested and meets minimum requirements for approval, and self-certification where the automaker performs a set of evaluations and informs the local government whether the vehicle/software meets minimum requirements.

**Electronic Control Unit (ECU):** This is a generic electronic part of the mechatronic system of a vehicle, and it contains embedded software that controls actuators of a complex system and manages its failures (YOUR MECHANIC, 2015).

**Embedded Software:** This is software written in a specific program language and that is embedded in hardware. An ECU contains specific embedded automotive software to control systems, features, and actuators (TECHOPEDIA, 2021).

**ISO 26262:** This is an automotive functional safety standard that is applied to safety-related systems including one or more electrical and/or electronic (E/E) components (BAHIG, 2017; EL-KADI, 2017). It classifies possible hazards due to malfunction of E/E safety-related systems (ISO 2018) according to ASIL.

**Hardware-In-the-Loop (HIL):** This is a common test used in the automotive sector to evaluate the whole system's behavior on standalone hardware by using simulations to interact with the rest of the vehicle without needing a complete physical system (STAPLES, 2018).

**Software-In-the-Loop (SIL):** This is a common test used in the automotive sector to evaluate the system's behavior by using simulations without the need for a complete hardware system (STAPLES, 2018).

**Systems Modeling Language (SysML):** This is an architecture modeling language for engineering applications, and it supports the V&V process for hardware and software (SysML 2022).

**Unified Modeling Language (UML):** This is a modeling language aimed at defining a standard way to visualize a system's design (UML, 2005).

**Verification & Validation (V&V):** This process evaluates whether a component or system meets functional or customer-desired requirements (IEEE, 1990).

**V-Model:** This is a traditional method used in the automotive for software development and testing and with the objective of integrating software and hardware components (RANA *et al.*, 2013).

## 2.2 RELATED WORK

Four databases were selected to search for articles in March and April 2022 related to the automotive software V&V process: IEEE Xplore, ScienceDirect, SpringerLink, and ACM Digital Library. During this search, two SLRs were found: one on software verification and the other on the validation process for automotive engineering. A third related work was identified because it was cited in several other articles discussing and defining concepts for ASE.

**Table 1** lists the Research Questions (RQs) and motivations of the first SLR by Rajabli *et al*. (2020). For this review, databases such as SpringerLink, IEEE Xplore, ACM Digital Library, and Wiley were consulted. After the ICs and ECs were applied, 200 articles were obtained. Many types of testing techniques, frameworks, and philosophies were identified. A common conclusion was that the ISO 26262 standard is not ideal for the V&V process of autonomous vehicles. This is because the requirements of an autonomous vehicle cannot be defined when test cases are initially

defined because this type of vehicle is very unpredictable. This means that the vehicle response differs depending on the boundary conditions, which is not covered by ISO 26262. The SLR found a consensus that the fault injection method can find many faults in different environments and that a formal method of testing and mutating tests may be necessary for the V&V process of autonomous vehicles. In summary, Rajabli *et al.* (2020) concluded that it is not possible to define all requirements for the V&V process of autonomous vehicles because their response to different environments cannot be predicted. This is one of the biggest challenges for testing this type of vehicle and the overall development process. ISO 26262 should be updated to accommodate guidelines for autonomous vehicle development.

Table 1. RQs and motivation

| Research Question | Motivation |
|---|---|
| **RQ1: What are the relevant V&V requirements for safe and autonomous cars?** | Identify current V&V requirements and regulations for autonomous vehicles. |
| **RQ2: What are the main challenges of software V&V for safe and autonomous cars?** | Identify approaches to meet the safety requirements for autonomous vehicles and determine the weaknesses of current software processes. |
| **RQ3: What are the open issues and opportunities for software V&V of safe autonomous cars?** | Investigate the state of the-art of software V&V for autonomous cars and highlight open issues, promising directions, and opportunities for future research. |

*Source: Rajabli et al. (2020).*

**Table 2** lists the RQs and motivation of the second SLR by Haghighatkhah *et al.* (2017a). They collected 679 articles from multiple research fields. They found substantial literature on ASE in the 1990s after which academic interest declined, although interest has revived from 2007 onward. Research interest in ASE is steadily increasing, they also observed an evolution in article topics over the years. In the 1990s, the most common topics were related to the agreement support group, project support group, and requirement engineering. After 2007, the most common topics were system/software architecture and design, followed by system/software testing. The most common types of studies published were evaluation research, which typically involved implementing a proposed solution and presenting the results obtained via case studies or surveys. The most relevant topics were proposed frameworks and learned lessons, which highlighted the importance of good management of ASE

development and testing. Although the authors demonstrated that the topic of ASE has been gaining relevance in the academic environment, such studies often did not consider practical applicability from an industrial perspective. Therefore, they concluded that topics such as software V&V should have empirical evaluations of proposed methods under actual practical conditions that can happen in the industry domain.

Table 2. RQs and motivation

| Research Question | Motivation |
|---|---|
| **RQ1: What is the intensity of research activity in the ASE context?** | Identify and analyze the intensity of research activities associated with ASE. Note: For this question, intensity was represented by the quantification and analysis of publication trends, authorship information, publication channels, publication titles, and citations in studies relevant to ASE. |
| **RQ2: What is the application most investigated? How has ASE improved over time? Which subtopics are often discussed in these studies?** | Identify and analyze the most common research topics in the ASE literature, their evolution over time, and the automotive applications for which they have been investigated. |
| **RQ3: Which are the types of research methods used in ASE-related publications?** | Identify and classify primary studies based on the type of research and methodology. Note: The research type is nature of the inquiry employed in the study, whereas the research method is the research methodology applied. |
| **RQ4: What are the contributions of the ASE studies?** | Investigate and classify the ASE literature concerning their contribution to an academic environment. |
| **RQ5: What are the gaps in the research and promising future directions for ASE?** | Identify potential research gaps by analyzing the identified research topics and their relevant contributions to the ASE domain. Note: This analysis identifies areas that may require additional investigation and suggest future directions of research on ASE. |

*Source: Haghighatkhah et al. (2017a).*

**Table 3** details the RQ and motivation of the third study by Pretschner *et al.* (2007). They described several characteristics of ASE and distinguished between regular software and automotive software. In particular, one characteristic of automotive software is its heterogeneous nature owing to the different functionalities (e.g., powertrain, infotainment, driver assistance). This heterogeneity results in a long system integration process, a large number of vehicle variants that the software needs to be compatible with, and specific requirements for reliability and safety.

Table 3. RQ and motivation

| Research Question | Motivation |
|---|---|
| **RQ1: What characteristics of ASE are different from regular software?** | Identify the differences between regular software and ASE. |

*Source: Pretschner et al. (2007).*

These studies give a good overview of ASE and automotive software testing. Our SLR provides the benefit of consolidating content in terms of standards, research trends, and peculiarities in the ASE domain and the V&V process for automotive software. This SLR can also be used to help guide future works just like the related works. For example, Rajabli *et al.* (2020) noted that current standards do not cover the V&V process for autonomous cars, Haghighatkhah *et al.* (2017a) noted that the intensity of ASE research should be increased, and Pretschner *et al.* (2007) differentiated between automotive software and regular software.

## 3. RESEARCH METHODOLOGY

Here, we present the main strategies, execution, and motivation of our SLR.

## 3.1 OBJECTIVE

The main objective of this SLR was to address the following topics in the automotive context:

Synthesize the main types of testing embedded automotive software.

Identify tools of the V&V process that coordinate the software testing process and store the results.

Synthesize knowledge in the literature on the limitations of the current automotive software testing process.

Define a cluster of specific automotive software tests or variables for specific applications or countries.

Identify the main differences between automotive software and regular software.

We used a platform called Parsifal (https://parsif.al) to help plan, conduct, and report the SLR. The main deliverables are described in the following subsections.

## 3.2 PLANNING TARGETS

The planning was defined based on key points that we wanted to clarify or obtain knowledge of, as well as the basis for determining the RQs. The main targets are defined in Section 3.1, and they were subdivided into the planning, conduct, and report phases.

## 3.3 PICOC CRITERIA

PICOC is a framework for defining the five key elements of research and comprises the following elements:

**Population:** Automotive V&V methodologies and Automotive software development.

**Intervention:** Innovative processes for testing software during the development process

**Comparison:** Strengths and weaknesses of state-of-the-art test methodologies.

**Outcome:** Defining key elements that differentiate regular software from automotive software, identifying flexible and robust V&V frameworks for testing embedded automotive software, obtaining results quickly, and organizing the results.

**Context:** Automotive industry using embedded automotive software.

## 3.4 RESEARCH QUESTIONS

The RQs were defined based on the objective in Section 3.1 and PICOC criteria in Section 3.3. **Table 4** presents the RQs and their motivations.

Table 4. RQs and motivations of the SLR

| Research Question | Motivation |
|---|---|
| **RQ1: What types of embedded software testing are used in the automotive domain?** | Identify the best practices available for testing embedded software in the automotive domain. |
| **RQ2: What are the most efficient tools for managing software development and testing deliverables?** | Identify methodologies that can be used as a support tool during the automotive software development process. |
| **RQ3: What types of automotive embedded software testing can increase the quality/robustness of the software delivered to the customer?** | Identify testing types that can increase the quality/robustness of the automotive software V&V process. |
| **RQ4: What standards govern the development and testing of embedded automotive software?** | Analyze the advantages and disadvantages of using a standard to approve automotive software for commercialization. |
| **RQ5: How are test cases defined for specific applications and countries in the automotive sector?** | Understand the issues that can occur in a vehicle due to local characteristics. These will initially affect purely mechanical/pneumatic/hydraulic components, sensors, actuators, or hardware in general. Eventually, the software/system will also be affected. Therefore, it is extremely important to identify variables related to the local region that can affect the behavior of an automotive software/system. |
| **RQ6: What are the differences between automotive software and non-automotive software that suggest a need to develop a new test methodology for the former?** | Identify methodologies that can be used as a support tool during the automotive software development process and identify the motivation to develop a specific software for the V&V process. |

*Source: Authors (2022).*

## 3.5 SEARCH STRINGS

Four different databases were searched for this SLR: IEEE Xplore, ScienceDirect, ACM Digital Library, and SpringerLink. The following keywords were derived from the RQs to formulate the initial search strings: Automotive testing software, Automotive software V&V, Special automotive test cases, Automotive software testing for a specific application, Vehicular software testing for a specific country, and software testing for commercial vehicles. In addition, the Boolean operators OR and AND were added to the search strings, which were then applied to search trials of the four databases. However, the selected articles were applicable to all of the RQs. Thus, the generated search string was applied only to IEEE Xplore.

Next, a second set of different keywords was defined: Managing automotive test results, Coordination tools for automotive software test results, No failures in automotive software testing by management control, Quality improvement of automotive test results by management tools, Automotive testing management, Vehicular software testing, and Test result management. The Boolean operators OR and AND were again used in the search string to focus on RQ2 and RQ3, which are related to specific tools, coordination, and quality improvement of software testing. In the end, this search string was applied only to ScienceDirect.

Third, a different search string was defined with different keywords: Automotive software, Automotive software concept, and Main differences of automotive software. The Boolean operators OR and AND were added to try to answer the remaining RQs. However, it was not possible to select articles that answered the last RQ, and this search string was applied only to ACM Digital Library. Therefore, an additional search string had to be generated by adding two keywords to the previous string (Automotive software, Specific) and using only the Boolean operator AND. This search string was applied to SpringerLink, which allowed us to find articles in the four databases to answer all six RQs. The search strings are defined in **Table 5**. An additional filter was added to only include articles published after 2012 to ensure that the SLR focused on recent developments. Other research databases were not considered owing to time constraints.

Table 5. Search strings

| Source | String |
|---|---|
| **IEEE Xplore** | ("Automotive testing software" OR "Automotive software V&V") AND ("Special automotive software test cases" OR "Automotive software testing for a specific application" OR "Vehicular software test for a specific country" OR "Software testing for commercial vehicles") |
| **ScienceDirect** | ("Managing automotive tests results" OR "Coordination tools for automotive software tests results" OR "No failures in automotive software testing by management control" OR "Quality improvements on automotive tests results by managements tools") AND ("Automotive testing management") AND ("Vehicular software testing") AND ("tests results management") |
| **ACM Digital Library** | (Title: automotive software definition) OR (Title: automotive software concept) OR ((Title: main differences of automotive software) AND (Title: gaps in the automotive test)) |
| **SpringerLink** | ("Automotive software") AND ("Specific testing") |

*Source: Authors (2022).*



RC: 148861
Available in: https://www.nucleodoconhecimento.com.br/computer-science/embedded-software

## 3.6 INCLUSION AND EXCLUSION CRITERIA

Next, the titles and abstracts of the collected articles were read. **Table 6** defines six Inclusion Criteria (ICs), and **Table 7** defines four Exclusion Criteria (ECs) with their respective justifications.

Table 6. Inclusion criteria

| ID | Inclusion Criteria | Justification |
|---|---|---|
| **IC1** | Automotive software testing | To focus on studies in the automotive domain |
| **IC2** | V&V for an automotive safety-relevant software | To ensure that normally safety-relevant processes meet specific requirements that can be used to create an automotive testing process |
| **IC3** | Studies in English | To verify the global tendency of studies for automotive software V&V process |
| **IC4** | Primary studies | To obtain the start of the art in terms of automotive software |
| **IC5** | Automotive software validation by ISO 26262 | To evaluate the benefits and weaknesses of the most commonly applied standard in the automotive industry |
| **IC6** | Validation of safety and driver assistance systems | To cover most of the subdomains in a vehicle |

*Source: Authors (2022).*

Table 7. Exclusion criteria

| ID | Exclusion Criteria | Justification |
|---|---|---|
| **EC1** | Not primary studies | To find the maximum possible state of the art in the domain |
| **EC2** | Articles published before 2011 | To find the most up-to-date articles in the automotive domain |
| **EC3** | Abstracts that do not mention automotive software V&V or similar terms | To focus on studies of automotive software V&V |

*Source: Authors (2022).*

## 3.7 QUALITY CRITERIA

**Table 8** lists the Quality Criteria (QCs) that were applied as an additional filter to the selected articles. The questions posed by the QCs each had three possible answers with different weights. An answer of “yes” was weighted with 1 point, an answer of “partially” was weighted with 0.5 points, and an answer of “no” was weighted with zero points. There were seven questions, so the highest and lowest scores possible were seven and zero points, respectively. Articles with a score of four or more points answered at least half of the QCs in the affirmative and thus were classified as relevant

for the SLR. Articles with a score of less than 4 points were rejected. Finally, the articles were checked if they passed at least one of the defined criteria. If an article met at least one IC, it was allowed to pass along to the next step of the evaluation step. If an article met at least one EC, it was excluded from the SLR. At the end of the selection process, 62 articles were selected.

Table 8. Quality criteria

| QC | Quality Criteria |
|---|---|
| QC1 | Are the aims clearly stated? |
| QC2 | Do they answer the RQ? |
| QC3 | Are the methods for automotive software testing clearly defined? |
| QC4 | Are the methods able to identify errors at the beginning of the automotive software development process software? |
| QC5 | Can the methods be used for different subsystems in the automotive domain? |
| QC6 | Do they define how to test an automotive software for a specific region or application? |
| QC7 | Does it use a tool to coordinate software test/model-based deliverables? |

*Source: Authors (2022).*

## 3.8 DATA EXTRACTION

The Data Extraction (DE) phase took up most of the SLR. The metadata included the title, library, year, source of publication, and other information. All 62 articles were read in detail to extract the data listed in **Table 9** and answer the RQs.

Table 9. Data extraction

| ID | Data Extraction (DE) |
|---|---|
| DE1 | Title |
| DE2 | Library |
| DE3 | Year |
| DE4 | Source of publication |
| DE5 | Author |
| DE6 | Country |
| DE7 | Classification of paper |
| DE8 | Type of software testing |
| DE9 | Main method |
| DE10 | Techniques used for implementation |
| DE11 | Study validation |
| DE12 | Validation context |
| DE13 | What are the most relevant difficulties when using this methodology? |
| DE14 | What are the benefits due to the use of this methodology? |
| DE15 | Is it possible to determine an increase in the identification of software errors? |
| DE16 | Which regulations are used as a basis for the tests? |

*Source: Authors (2022).*

## 3.9 THREATS TO VALIDITY

The Parsifal platform helps geographically distributed researchers work on the same SLR. The results for different authors and databases were compared to reduce the inherent biases caused by differences in the knowledge and field of expertise of authors of the selected articles. Four threats to validity were evaluated to increase the robustness of the SLR: construct validity, internal validity, conclusion validity, and external validity.

### 3.9.1 CONSTRUCT VALIDITY

Construct validity involves the correct definition of tools used to measure the object of study. Different interpretations of terms and constructs can lead to wrong conclusions. For example, differing understandings of the term “homologation” could lead to the wrong interpretation. As noted in Section 2.1, we used the definitions of Killian (2018) and the IEEE (1990) for this term. As another example, we used the definition of Pretschner *et al.* (2007) for ASE. We created Section 2.1 to ensure a consistent definition of terms and construct validity of the SLR.

### 3.9.2 INTERNAL VALIDITY

Internal validity is concerned with whether consistent results can be obtained from extracted data (WOHLIN *et al.*, 2000). We focused specifically on database selection, search string definition, and the ECs and ICs. With regard to the databases, we tried to reduce research bias by only searching IEEE Xplore, ScienceDirect, ACM Digital Library, and SpringerLink. This means that other relevant works may be present in other databases, or a simple adjustment in the defined search string could obtain other relevant articles. A second threat was the definition of the search string of each database. We carried out some pilot searches to define and clarify the RQs as much as possible. However, there is always a possibility that a search string was missing a key element. The third threat to increasing the robustness of the data was including all ICs and ECs. Despite our careful consideration, this work is not free of mistakes, and the results may be incomplete.

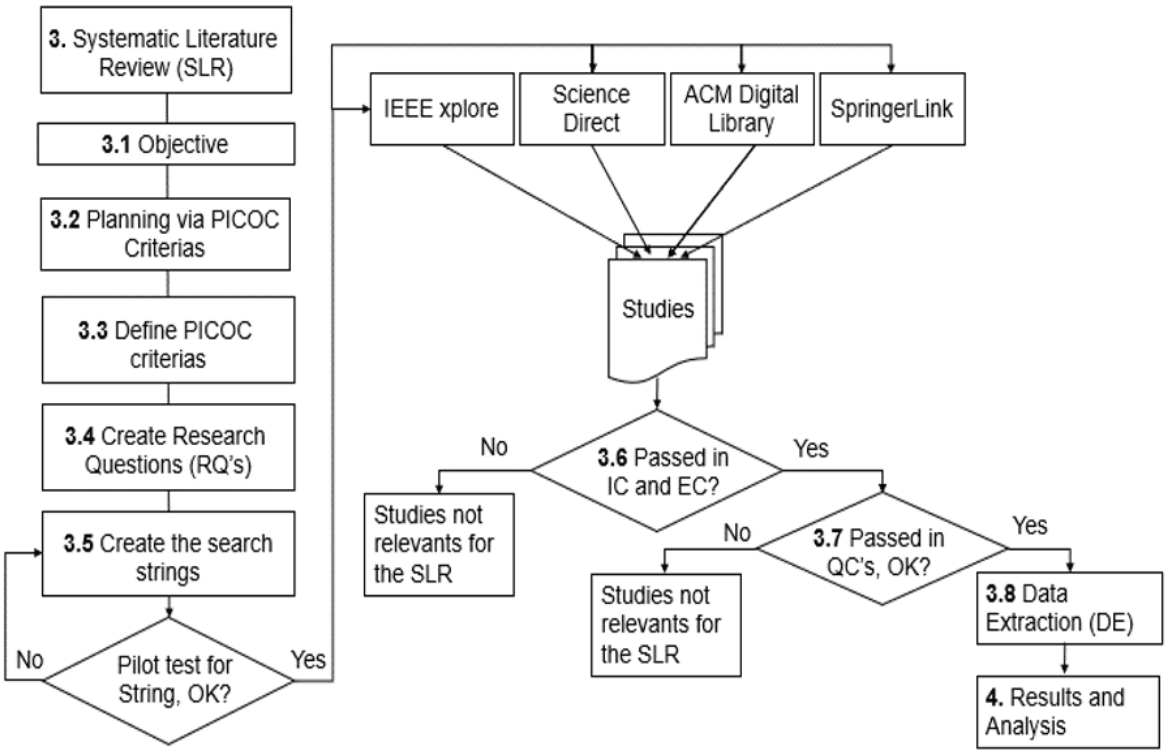
### 3.9.3 CONCLUSION VALIDITY

Conclusion validity concerns whether a study reaches the correct conclusion from rigorous and repeatable treatment (WOHLIN *et al.* 2000). We concluded the characteristics of ASE and testing based on searches of scientific databases and the technical aspects of each article. To reduce bias, we tried to use several references to come to the same conclusion. Thus, all conclusions that we drew can be traced to the extracted data.

### 3.9.4 EXTERNAL VALIDITY

External validity concerns whether the obtained results can be generalized to other environments with appropriate robustness. As discussed in Section 3.9.2, all efforts were made to minimize research bias for both internal and external validity. The selected articles from the four databases were published by different researchers of varying nationalities and institutes. Therefore, readers can analyze the applicability of our results in different environments and contexts. After all threats to validity were evaluated, we obtained 62 articles that passed the filters, which we used to extract data and answer the RQs. **Figure 2** presents the flowchart of the protocol we followed for the SLR.

Figure 2. Flowchart of the SLR protocol based on the Parsifal tool



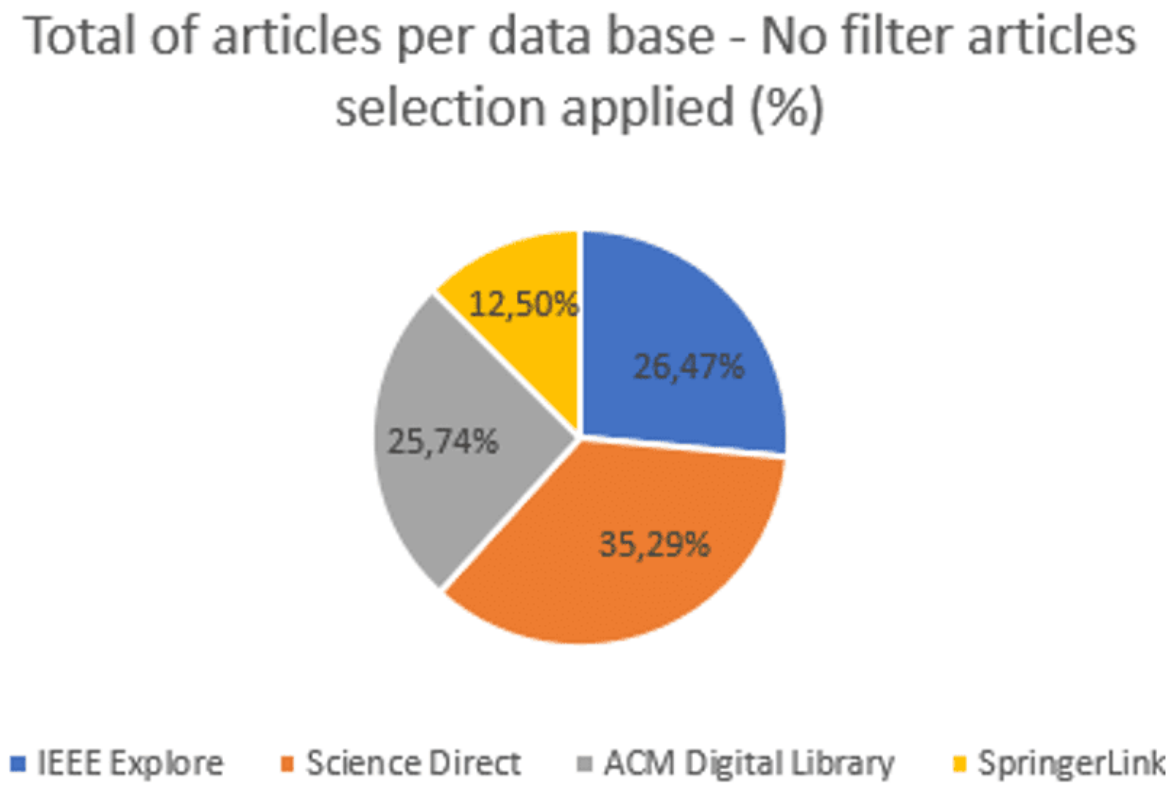

Source: Authors (2022).

## 4. RESULTS AND ANALYSIS

## 4.1 APPLIED PROTOCOL AND EXTRACTED DATA

Applying the automatic search strings (Section 3.5) to the four research databases returned 408 articles: 108 from IEEE Xplore (26.47% of the total), 144 from ScienceDirect (35.29%), 105 from ACM Digital Library (25.74%), and 51 from SpringerLink (12.50%). More papers were obtained from ScienceDirect because we fine-tuned the search string based on our experiences with the other databases. **Figure 3** shows the percentage of articles found from each database.

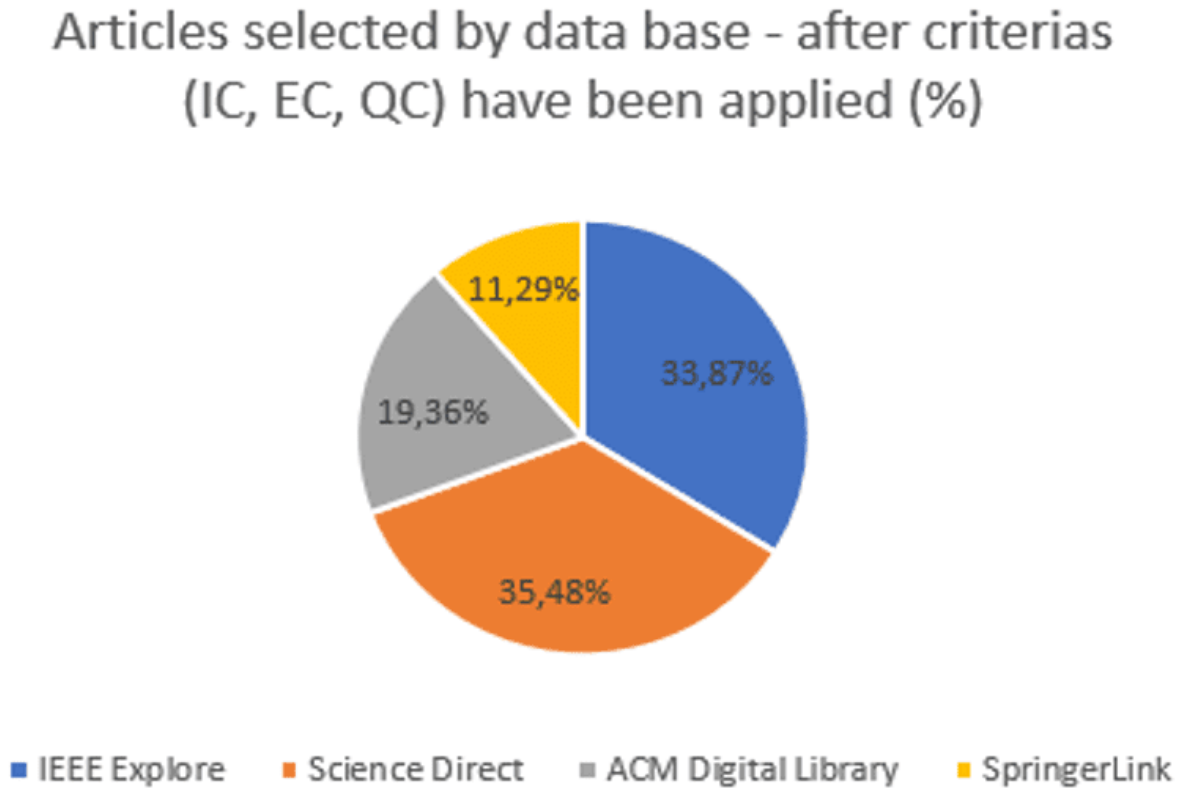
Figure 3. Percentage of papers found in each research database




Source: Authors (2022).

Applying the ECs, ICs, and QCs (Sections 3.6 and 3.7) returned 62 articles: 21 from IEEE Xplore (33.87% of the total), 22 from ScienceDirect (35.48%), 12 from ACM Digital Library (19.36%), and seven from SpringerLink (11.29%). **Figure 4** shows the percentage of articles from each database after the criteria were applied.

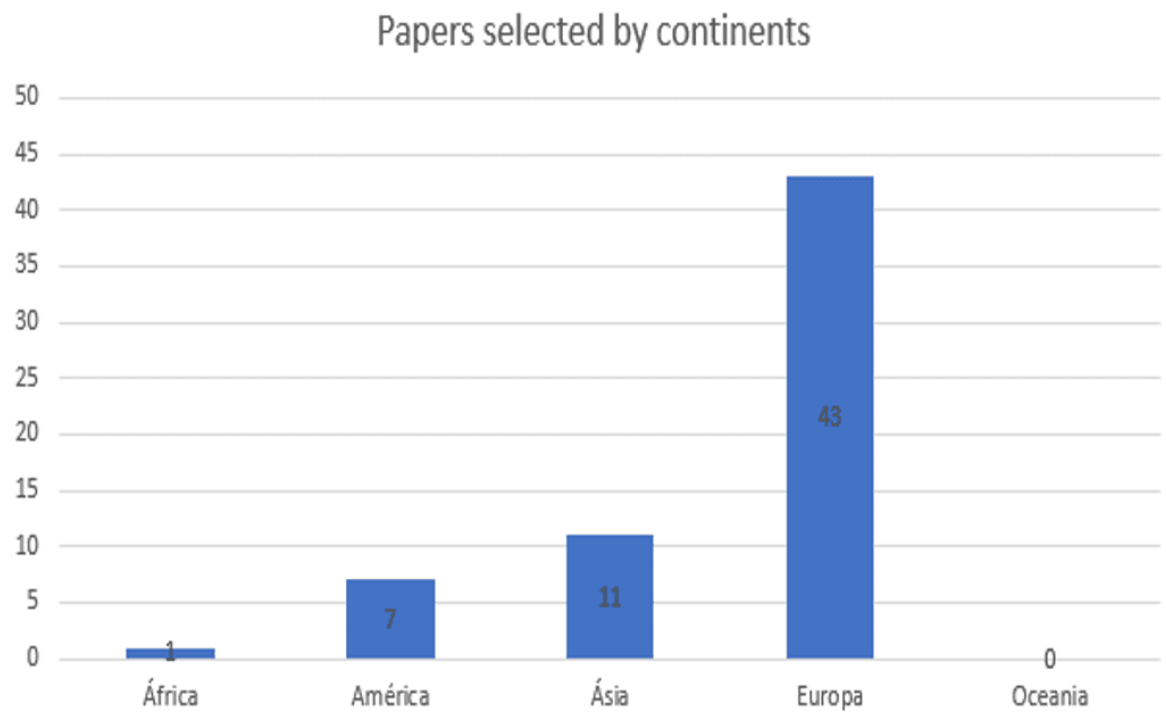
Figure 4**.** Percentage of papers selected from each research database




Source: Authors (2022).

**Figure 5** shows the distribution of selected papers by continent. The most were from Europe at 43, followed by 11 from Asia. The Americas were represented by seven articles, one was from Africa, and no articles were from Oceania. The high number of articles from Europe can be attributed to many companies having their research and development centers there, such as Daimler, BMW, Bosch, ZF, Valeo, and Saab, Ulrich (2021) states that these companies are running against the time and investing huge amounts of money to develop software standards to face the emerging challenges in the auto industry, particularly in the robust German auto industry. Therefore, researchers from Europe are producing numerous publications about the development and results achieved in this domain compared with researchers from other continents.

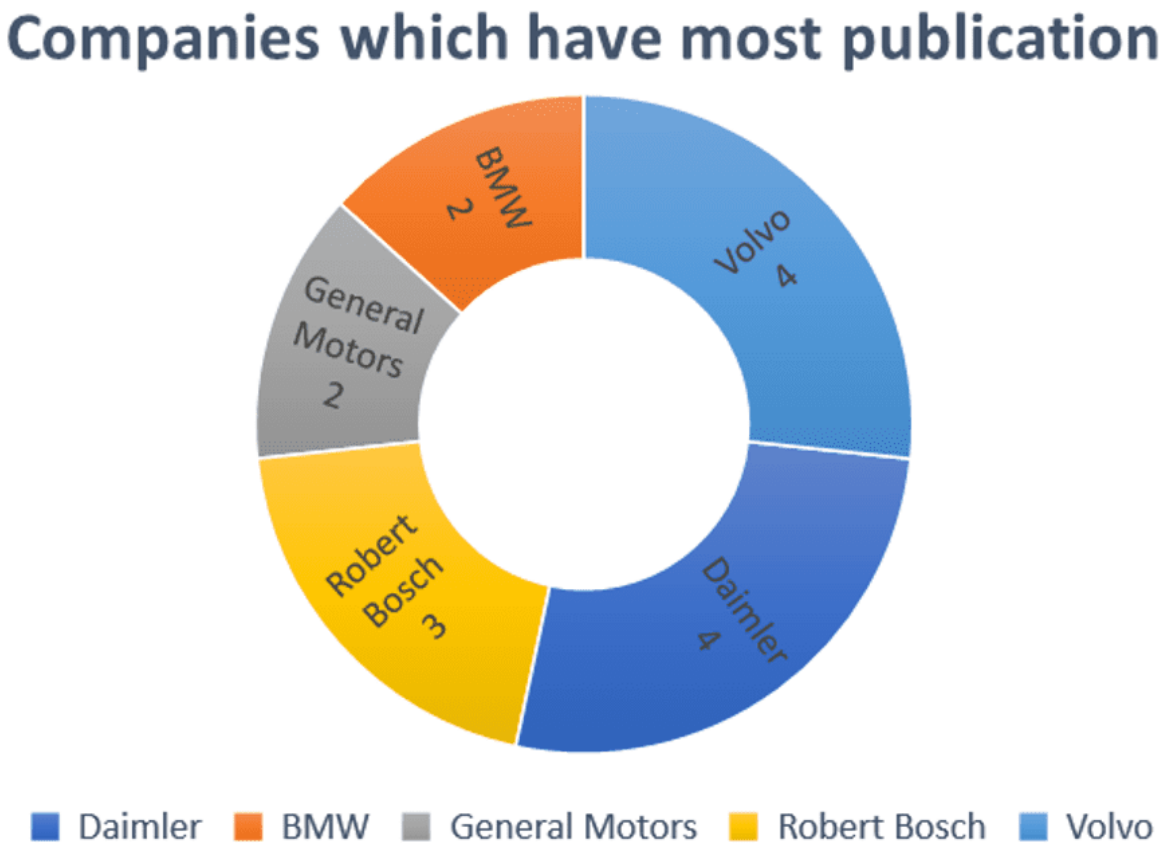
Figure 5. Distribution of selected papers by continent




Source: Authors (2022).

We also ranked the top five companies involved in studies on V&V automotive software. **Figure 6** shows that Daimler was involved in four of the selected articles (FREY *et al.*, 2016, (JUHNKE, 2018; TICHY, 2018; HOUDEK, 2018)., KÖNIG *et al.*, 2023, BRAUN *et al.*, 2014). Volvo was also involved in four of the selected articles (ANTINYAN *et al.* 2020; DURISIC *et al.*; 2013; RANA *et al.* 2014, 2016). Robert Bosch GmbH was third with three articles (BURTON *et al.*, 2012; NEUROHR, 2021; BRAUN *et al.*, 2014). Fourth and fifth were BMW with two articles (KRIEBEL, 2018; DEICKE *et al.*, 2012) and General Motors also with two articles (PETRENKO *et al.*, 2015; NEUROHR, 2021). Other companies such as FEV Europe GmbH, SMR Automotive Mirrors Stuttgart GmbH, Toyota, ZF AG, and Hyundai Motor were also involved with articles (KRIEBEL, 2018; KLENDAUER *et al.*, 2012; YAMAGUCHI *et al.*, 2016; NEUROHR, 2021; SEO, 2012; CHOI, 2012; YANG, 2012). These results indicate that a small circle of companies is involved in researching the topics covered by this SLR, and most of them are in Europe. This agrees with the results shown in **Figure 5**.

Figure 6. Ranking of companies with the most published articles regarding the automotive software V&V process




Source: Authors (2022).

Next, we categorized the articles according to topics. **Table 10** indicates that the most common research domain among the selected articles was simulation/calculation for the automotive software V&V process (17.74%), followed by ASE (16.13%) and software/testing for the next generation of vehicles (14.52%). Other domains included improvement of the ASE process (9.68%) and a single article on tools for the V&V process (1.61%).

Table 10. Distribution of articles by research domain

| Domain | Number of articles | Percentage (%) |
|---|---|---|
| **Model testing** | 2 | 3.23 |
| **ASE** | 10 | 16.13 |
| **Future of automotive software/testing** | 4 | 6.45 |
| **Improvement in the ASE process** | 6 | 9.68 |
| **Simulation/calculation for automotive software V&V** | 11 | 17.74 |

| | | |
|---|---|---|
| **Functional safety/ISO 26262/ FUSA** | 7 | 11.29 |
| **Software testing for next-generation vehicles** | 9 | 14.52 |
| **Tools for the V&V process** | 1 | 1.61 |
| **Software/testing management** | 5 | 8.06 |
| **Requirement validation** | 4 | 6.45 |
| **Others** | 3 | 4.84 |

*Source: Authors (2022).*

## 4.2 DISCUSSION

Before we answer the RQs, we need to define and explain some principles regarding the software development models, test definitions, types of tests, and legislation. The following information is based on the data extraction of the selected papers.

V&V is a group of activities to assess the quality and reliability of a software, especially in safety domain applied in different environments (BERNARDI *et al.*, 2020, AHANGARI *et al.*, 2018). In short, it is a process with the aim of proving that the designed software does what it was developed to do. However, even a robust testing framework cannot ensure that the software is free of errors. Therefore, a good testing framework is expected to find the maximum number of mistakes before the software is released to the customer. Software development is a complex process that requires a wide range of skills and interpretation, similar to other engineering activities. Hence, software development always has the possibility of errors. To minimize the risk of errors before release to the customer, a series of activities is carried out: validation, verification, and testing (DELAMARO, 2007; JINO, 2007; MALDONADO, 2007). V&V activities are divided into two main categories: static tests do not necessarily need working software and they can be based on a functional model or even specifications, while dynamic tests run the actual software and not just a reference model. Multiple other authors have also described software testing in the same manner (SZALAY *et al.*, 2021; IQBAL, 2020; JORDAN *et al.*, 2019; NEUROHR, 2021; DURISIC, 2013; DELAMARO, 2007; JINO, 2007; MALDONADO, 2007). Szalay *et al.* (2021) noted that V&V is an evolutionary activity in the automotive industry because of the increasing number of embedded software technologies providing new features.

Rana *et al.* (2014) unit testing aims to evaluate the interaction of the smallest part of a software or internal flow limit component. In short, unit testing focuses on verifying the internal processing logic and structures within a component and/or module. In an automotive environment, such tests are usually performed at the beginning of the software project by the supplier itself, and the automaker receives reports describing the final status of the test.

Pretesting is a type of test that aims to find errors before the software is even implemented. This is to prevent failure from only being identified in dynamic tests, which will require more budget and time to correct. Such tests require documents that can represent the software in a preliminary manner or simpler structure, such as the source code, specifications, or schematic model. Some studies have demonstrated that pretesting is more effective at discovering defects than many software tests (SOMMERVILLE, 2011). In the automotive context, pretesting is most often based on the specifications of the system or embedded software. High-level schematics of the system architecture and the source code are also helpful.

Performance tests evaluate the performance of software within an integrated context, and they are often linked to stress testing (SOMMERVILLE, 2011). For vehicle systems with dynamic variables such as the vehicle speed, mass, and wear of mechanical components, software needs to be evaluated by considering all possible variations of the system and in the external environment (e.g., asphalt conditions, temperature). In the automotive context, performance testing is necessary to check if the software and/or system meet diverse demands, especially regarding safety and the powertrain.

Regression tests aim to verify that corrections to the software do not generate new errors, which would indicate that the software has regressed (DEVMEDIA, 2012). Regression tests are necessary for software development because new releases often incorporate new functions. In the automotive industry, software is commonly released several times, so regression tests are necessary.

The acceptance test is the final stage of testing before a system or product is delivered to the customer (SOMMERVILLE, 2011). This test is usually performed under real-

world conditions. Alpha tests offer the embedded software/system to a limited number of customers for testing, and beta tests offer the software to a large number of customers for testing.

A relevant topic is the software development process models commonly used in other industries, which are often also applied in the automotive context to embedded software development. (PRESSMAN, 2011; MAXIM, 2011) noted that software development process models have the main objective of bringing order to a disorganized process. However, some researchers have argued that order is not the natural state of an activity and that an unorganized environment can stimulate a high level of creativity, which increases profits and generates self-organization. However, a total lack of organization can lead to chaos. Thus, finding a middle ground is necessary. The models usually used in software development are the waterfall models, incremental and reuse-oriented software engineering (SOMMERVILLE, 2011).

The waterfall model considers as the fundamental activities, specification, development and evolution. For each of these phases, they are represented in different ways, such as specification requirements, software design and testing. The benefits include consistency with other engineering process models and the production of documentation at each stage. The disadvantages include the need to make commitments at the beginning of the project, which makes it difficult to implement changes according to customer requests.

Incremental development integrates specification, development, and validation activities. The system or software is designed and created along with a series of versions with improvements in each version, and functionality is also added to previous versions. The benefits include a lower cost than other models and more flexibility to make changes during the software development process. The disadvantages include the black-box nature of the development process, which is opaque and difficult to measure or manage.

Reuse-oriented software engineering is based on reusable components. When developing a system, instead of starting a creation process from scratch, the process

will focus on incorporating components of an existing system. The benefits include a reduction in the amount of software development, which allows for earlier delivery. The disadvantages include an inability to meet customer requirements/needs.

## 4.3 ANSWERS TO RESEARCH QUESTIONS

### 4.3.1 RQ1: WHAT TYPES OF EMBEDDED SOFTWARE TESTING ARE USED IN THE AUTOMOTIVE DOMAIN?

RQ1 was aimed at identifying the best/standard practices in automotive software available in the market. Most innovations in the automotive sector are now in software, which has increased the complexity of automotive systems. According to Durisic *et al.* (2013), automotive software includes safety control, driver assistance, and driver comfort features in the same execution space (i.e., hardware/ECU). Robust testing is required to ensure that automotive software works properly. (JUHNKE, 2018; TICHY, 2018; HOUDEK, 2018) argued that testing is the most relevant part of automotive software development because failure in the field will damage the automaker's brand and have financial and societal impacts. They divided the responsibilities for testing between the OEM and supplier based on the V-model of the development process (**Figure 1**). The supplier is responsible for tests at lower levels of integration while the OEM is responsible at higher levels of integration. This division of responsibilities can be used to explain some different types of testing available (RANA *et al.*, 2014, TECHOPEDIA, 2021, DURISIC, 2013):

**Unit Testing:** This tests the smallest part of software in terms of the data flow, data sequence, and integration. In the automotive industry, the supplier typically performs unit testing, and the results are sent to the automaker as a report (RANA *et al.*, 2014).

**Integration Testing:** This type of test decomposes the system architecture to visualize the data flow and relationships between ECUs in the same network or compare alternative pathways in different networks. Using a graph model is effective at identifying potential failures when the embedded software is running. Machine test models evaluate the integration between components and ECUs by checking the data

flow. Message sequence charts evaluate the message sequence to evaluate whether sending/receiving it will result in an integration error (FRAGAL *et al.*, 2018).

**SYSTEM TESTING:** This test uses similar or real environmental conditions in which the software will be running for the final customer, and it is generally performed on an actual prototype vehicle (MOSTAFA *et al.*, 2018).

**ACCEPTANCE TESTING:** The automaker normally delivers test vehicles to limited groups of customers to receive feedback on whether their needs are met (DIVAKARLA, 2016; EMADI, 2016; RAZAVI, 2016).

Besides the tests based on the V-model, a considerable number of other tests were found in the literature:

**REGRESSION TESTING:** This test is essential for evaluating whether corrections made to software have added errors, especially regarding integration (LOCHAU, 2014; PEKARIC, 2019; SAUERWEIN, 2019; FELDERER, 2019).

**APPLICATION TESTING:** This test is relevant for covering a wide range of vehicle variants (ARRIETA *et al.*, 2019). The main aim is to ensure that a specific vehicle configuration (i.e., combination of functions, sensors, and actuators) is usable in a specific application (e.g., mining, off-road, military) or region. (THURIMELLA, 2013; BRÜGGE, 2013) linked application engineering with application testing: the former focuses on developing individual systems for a specific market/application, and the latter assesses the developed system.

**MODEL-IN-THE-LOOP (MIL):** This test is a basic simulation that evaluates the interaction between the embedded software and the defined plant model (ARRIETA *et al.*, 2019).

**SOFTWARE-IN-THE-LOOP (SIL):** This test is a simulation based on the source code, where the plant is also digital and can be changed by lines of code (ARRIETA *et al.*, 2019; LI *et al.*, 2019). Therefore, the interaction between the embedded software and plant is more accessible than when the plant model is physical. Moreover,

(SCHROEDER *et al.*, 2016) states that before performing any SIL test, it is necessary to assess and validate the quality of the model due the industry is growing the complexity of the software and reliability of the testing results must be ensured.

**HARDWARE-IN-THE-LOOP (HIL):** In this test, the embedded software is implemented in a physical ECU and real-time infrastructure. HIL is usually used to validate non-functional requirements, such as performance and time constraints (CHEN *et al.*, 2019).

**REACTIVE TEST:** This test is designed to consider the unpredictability of real environments and evaluates the ability of the software to react to the system outputs (ARRIETA *et al.*, 2019).

**MODEL-BASED TESTING (MBT):** It is a formal technique, where test cases are derived from a model that describes what will be validated (ANAND *et al.*, 2013; PEKARIC, 2019; SAUERWEIN, 2019; FELDERER, 2019). MBT often uses incomplete testing approaches to correct the software because it is based on a black-box model for which the input and output behaviors are evaluated.

**HOMOLOGATION:** Automotive software is normally developed to comply with some regulatory domain, such as a specific ISO standard or local legislation (KRIEG *et al.*, 2013). Examples include ISO 26262 for functional safety or CONTRAN 641/16 for ESP homologation. Before a product or automotive software is launched on the market, it has to meet the minimum legislative requirements (SZALAY *et al.*, 2021; STAHL, 2021; DIERMEYER, 2021).

**VEHICLE-IN-THE-LOOP (VIL):** This test is performed in an artificially controlled and virtual environment to consider different sensor types and to simulate different traffic and road conditions (SZALAY *et al.*, 2021).

**FAULT INJECTION:** This test intentionally injects a fault in the software to simulate an error and evaluate its robustness (KRIEG *et al.*, 2013). Faults are injected by changing the programming code or external causes such as radiation.

**TEST BY SPECIFICATION:** This test introduces agile concept by evaluating the software in the early phases of development. The use of this approach will reduce the risk of requirement inconsistencies and ambiguities, thereby reducing the development costs at all (WIECHER, 2019; GREENYER, 2019; KORTE, 2019).

**STRESS TESTING:** This test evaluates the reliability and robustness of the software (NALIC *et al.*, 2020). It is important for checking the robustness of the embedded software at handling errors in critical situations. Stress testing is also called endurance testing, and it is used to evaluate the operating limits of software. When the limit is exceeded, the system is expected to respond with a software error message.

Other types of tests are available in the automotive domain. However, the abovementioned tests were mentioned the most often in the selected articles and were considered the state of the art by the selected authors.

### 4.3.2 RQ2: WHAT ARE THE MOST EFFICIENT TOOLS FOR MANAGING SOFTWARE DEVELOPMENT AND TESTING DELIVERABLES?

RQ2 was aimed at identifying efficient tools currently in use worldwide to support and manage software development and testing. This is relevant owing to the increased complexity of automotive systems and innovative solutions being implemented via software; thus, software has become the most crucial infrastructure in recent years (RODRIGUEZ, 2019; PIATTINI, 2019; EBERT, 2019). Kuhrmann *et al.* (2016) considered management one of the most important parts of successful development of automotive software.

(WOHLRAB, 2020; KNAUSS, 2020; PELLICCIONE, 2020) noted that the high degree of heterogeneity of automotive systems and functions presents a significant challenge to software development in all phases, including testing. Such challenges are typically multidisciplinary and include fields such as mechanics, electrical, pneumatics, and software. Unfortunately, there is no common discipline to facilitate the integration of these fields. (WOHLRAB, 2020; KNAUSS, 2020; PELLICCIONE, 2020) described

balancing the alignment and diversity of requirements as critical to a successful testing phase of automotive software. They emphasized alignment as a powerful tool to facilitate integration and ensure quality and a common language within a company or between external partners and suppliers. The development process for automotive software involves teams of engineers from diverse disciplines and regions. The Requirement Information Model (RIM) is a useful tool for balancing the alignment and diversity of requirement engineering. The interviews of (WOHLRAB, 2020; KNAUSS, 2020; PELLICCIONE, 2020) with collaborators from three different OEMs indicated that involving key stakeholders in the creation/adjustment of a RIM for a specific project will bring the benefit of a common language and compliance with standards. Other authors also mentioned benefits such as a high degree of integration between teams, increased quality of deliverables, precise definition of tasks, and detailed specifications of the developed functions that will be useful in the testing phase (SZALAY *et al.*, 2021; IQBAL, 2020; JORDAN *et al.*, 2019; NEUROHR, 2021; DURISIC, 2013; LOCHAU, 2014).

(KASOJU, 2013; PETERSERN, 2013; MÄNTYLÄ, 2013) stated that there is no evidence on how the automotive industry tests software nowadays. Furthermore, there are few studies on the strengths and challenges of tests in the automotive software domain. They listed several challenges that the industry faces: a non-unified testing process, short testing time available, lack of support from stakeholders, and resource constraints. They proposed Evidence-Based Software Engineering (EBSE) as one of the best solutions for management of the automotive software development and testing processes. EBES comprises the following steps:

1. Obtain a piece of evidence for a problem and formulate a question.
2. Define the best evidence that will answer the question from step 1.
3. Critically assess the response (i.e., evidence) given to solve the problem and analyze the context in which this solution will be implemented.
4. Evaluate the previous steps/evidence critically.

However, EBSE is only viable if a stakeholder/coordinator can organize the following activities:

Test process management: Managing various activities within the test process;

Test artifacts and assets organization: Organizing test documentation, providing testing tools, and ensuring a safe and available test environment;

Competence management: Allocating human resources with the necessary skills to perform specific tests;

Defect management: Identifying opportunities to detect software errors earlier and managing/supporting different people working together;

Requirements management: Analyzing and defining changes in requirements that can reduce testing times and improve the robustness of test cases.

(KASOJU, 2013; PETERSEN, 2013; MÄNTYLA, 2013) concluded that EBSE is an efficient approach to automotive software development/testing but requires at least one individual to manage the above activities daily. No single tool will provide a magical solution.

Rana *et al.* (2016) suggested that the Bayesian inference method, where past projects are utilized to specify prior information, could potentially provide reasonable prediction accuracy with only a small amount of data from the current project. This makes the Bayesian inference method especially useful for making predictions early in the project timeline, which will help with planning and allocating human and test resources. Predicting the number of defects discovered each week during software development is essential for prioritizing test resources, making release decisions, and ultimately releasing software without flaws.

### 4.3.3 RQ3: WHAT TYPES OF EMBEDDED AUTOMOTIVE SOFTWARE TESTING CAN INCREASE THE QUALITY/ROBUSTNESS OF THE SOFTWARE DELIVERED TO THE CUSTOMER?

RQ3 was aimed at defining specific types of testing to increase the robustness of automotive software. However, we were unable to identify a particular test to improve

the quality or robustness of software. Choi (2018) noted that automotive software typically uses the same operating system across multiple ECUs according to international standards such as OSEK/VDX or AUTOSAR, and the control logic of the ECU depends on the specific application. Therefore, significant testing is required to ensure reasonable quality/robustness, much testing is needed during an SDLC such as the V-model (**Figure 1**). Rana *et al.* (2014) reported a considerable number of Software Reliability Growth Models (SRGMs) that answer two critical questions for the development of embedded software: Is the software ready for release, and how much effort should be planned for the development of a specific embedded software in the automotive context? They proposed two SGRMs for improved efficiency depending on the model used for the software development process: the Gompertz model for the V-model or agile development process and the logistic model for the waterfall development process. Both models increase efficiency in terms of finding errors, but they are limited by their dependence on experienced software engineers to ensure accurate results.

Petrenko *et al.* (2015) noted that, even though many methodologies/frameworks are available for generating test cases from random inputs, an experienced software engineer is necessary to evaluate the reliability of the generated data and tests. They used the term “tester-in-the-loop” to describe the necessity of such individuals for improving the results of simulations and real tests. Neurohr *et al.* (2021) focused on the challenges of applying current software testing methodologies in the automotive industry to autonomous vehicles. In general, current guidelines define testing requirements at the beginning of the software project to improve the robustness and quality of the software. However, autonomous vehicles require significant amounts of testing in a real environment to ensure their robustness, even so due high costs and test tracks which do not represent the reality of the application of these vehicles, the simulation tests are also mandatory for these vehicles (PARRA *et al.,* 2020).

In conclusion, a lengthy V&V process is required to ensure the robustness and quality of automotive software. No single test is available that can ensure the safety, quality, and robustness of the software.

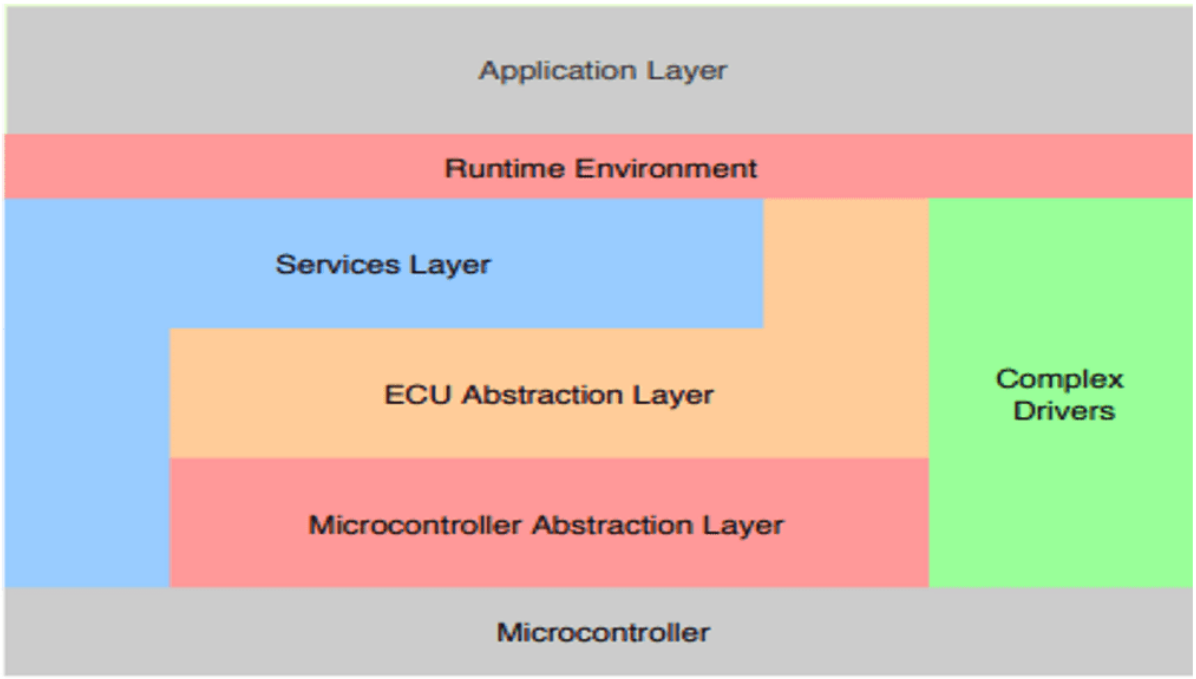
### 4.3.4 RQ4: WHAT STANDARDS GOVERN THE DEVELOPMENT AND TESTING OF EMBEDDED AUTOMOTIVE SOFTWARE?

RQ4 was aimed at elaborating the main standards available for guiding the development and testing of embedded automotive software. Many of the selected articles cited ISO 26262 as a reference standard (CHOI, 2018; SZALAY *et al.*, 2021; IQBAL, 2020; JORDAN *et al*, 2019; NEUROHR *et al.*, 2021; DURISIC, 2013; LOCHAU, 2014; ANTINYAN, 2017; STARON, 2017; WIECHER, 2019; GREENYER, 2019; KORTE, 2019; VÖST *et al.*, 2016; FERNANDEZ *et al.*, 2015; BERNARDI *et al.*, 2016). Szalay *et al.* (2021) argued that the safety goals of the V&V process should be defined according to ISO 26262. This standard requires practical tests of the whole vehicle to prove that the entire system/vehicle meets safety requirements. Evidence of the test results (e.g., test reports, fault tree analysis) must be checked during the safety validation. This is a good indication that ISO 26262 is a mandatory tool for the V&V process of embedded automotive software development. However, Neurohr *et al.* (2021) noted that ISO 26262 does not adequately cover testing of autonomous vehicles. One interesting avenue of future research will be proposing adjustments to ISO 26262 so that it can accommodate autonomous vehicles.

Another standard mentioned by the selected articles (BAHIG, 2017; EL-KADI, 2017; PEKARIC, 2019; SAUERWEIN, 2019; FELDERER, 2019) was Automotive Open System Architecture (AUTOSAR), which is an open standard for ECUs in the automotive industry. AUTOSAR uses informal specifications or a semiformal representation (i.e., UML), but this opens the door to error. **Figure 7** shows the layers of the AUTOSAR architecture (AUTOSAR, 2008), which can be used as a basis for creating a standard process for OEMs and suppliers. The **application layer** contains various applications /features for each ECU. The **runtime environment** facilitates communication with the software. The **services layer** provides system services such as memory, diagnostics, and communication. The **ECU abstraction layer** contains an application program interface to access peripheral devices without connecting with the operating system. The **microcontroller abstraction layer** contains an internal driver to access the operating system and internal peripherals. **Complex drivers** are

available to apply functionality that is not currently part of AUTOSAR. The **microcontroller** is a driver to communicate with all layers and peripherals.

Figure 7. AUTOSAR architecture




Source: (AUTOSAR, 2008).

### 4.3.5 RQ5: HOW ARE TEST CASES DEFINED FOR SPECIFIC APPLICATIONS AND COUNTRIES IN THE AUTOMOTIVE SECTOR?

RQ5 is crucial because many software development projects are split among different teams around the world. Beecham *et al.* (2017) has noted that universities are preparing the next generation of software engineers to work in a global environment. However, they did not clarify how automotive software can be validated to run in any part of the world. Being the embedded software according to (SEO, 2012; CHOI, 2012; YANG, 2012) determined as customized targeting hardware also including specific features, being optimized to work on resource constraints. When executed in its target environment the software will have systemic limitations relative to general software operating in a host environment. External inputs can influence the embedded software, taking communication protocols as an example. For example, a GPS feature can be

influenced by the quality of 3G data transmission and the terrain's topography. Thus, to define test cases and validation for different applications and countries, it is necessary to analyze the environmental conditions, topography, and infrastructure that will interact with the final product. Practical tests are necessary to confirm the behavior of the software in a real environment. ISO 26262 requires that automotive software for safety-critical features be tested in a real environment to ensure no faults when functioning in a vehicle (SZALAY *et al.*, 2021). Kim *et al.* (2016) noted that testing software in a vehicle on a physical roadway is usually the last step of the software development process with the aim of confirming that the vehicle works properly in an actual environment and meets traffic/local regulations. Therefore, in addition to the environmental conditions, the traffic/local regulations and driver behavior are additional considerations for defining specific test cases or types of testing depending on the application or region.

### 4.3.6 RQ6: WHAT ARE THE DIFFERENCES BETWEEN AUTOMOTIVE SOFTWARE AND REGULAR SOFTWARE THAT SUGGEST A NEED TO DEVELOP A NEW TEST METHODOLOGY FOR THE FORMER?

RQ6 was aimed at elaborating the peculiarities of automotive software relative to regular software. Such aspects may be relevant to developing a new testing methodology or adjusting an existing framework for automotive software. Durisic *et al.* (2013) defined automotive software as a mixture of safety and comfort functions that are executed by the same hardware/ECU but managed by a unique software for each ECU. Other authors (SEO, 2012; CHOI, 2012; YANG, 2012; VOGELSANG, 2020) have noted that automotive software is typically influenced by many external conditions such as the road pavement, external temperature, electromagnetic forces, traffic, driver behavior, aerodynamics, and weather. Choi (2018) noted that the V&V process in the automotive domain only focuses on application software under the assumption that the supplier validated the operational software. This assumption may lead to the OEM not findings errors caused by the interaction between the operational software and application software.

Another important aspect is the traditional development process of automotive software in OEMs. This process is conducted by solid legacy systems, which raises issues related to very optimized/split resources and extremely difficult relationships between OEMs and suppliers (VOGELSANG, 2020). In addition, automotive software development is usually split into different domains such as the brake system, powertrain, and chassis covered by a specific team to deliver features such as airbags, cruise control, ABS, ESP, and the start-stop system. Antinyan *et al.* (2020) noted the importance of homologation, which leads to different automotive software using the same ECUs to meet specific local requirements. Yamaguchi *et al.* (2016) concluded that the V&V process for automotive software still faces many challenges because some systems have increasingly sophisticated functions, which has led to a complicated integration process and difficulty with meeting specific standards. These issues indicate that it may be worthwhile to develop a V&V methodology specific to the automotive software domain.

## 5. CONCLUSION

The V&V process for embedded automotive software is complex, time-consuming, and expensive. However, it is an essential step to ensure the reliability of the overall vehicle system. This SLR clarified the increase in implemented solutions in the automotive domain. However, the V&V process is still a challenge owing to the unique characteristics of automotive software, such as a wide variety of configurations/variants and influence of external conditions. No single type of test can guarantee the robustness/quality of automotive software before release, which means that a combination of tests must be performed to ensure the expected quality. Good management and integration are critical factors for the successful development of automotive software to ensure efficient coordination between stakeholders and tasks, resource allocation, and test case optimization. Finally, ISO 26262 is a widely used standard to guide the development of automotive software, but it needs adjustment to adequately cover the needs of software for autonomous vehicles.

Our SLR indicates that further progress in the V&V process for embedded automotive software will require more investment in software management and the development

of a flexible framework for software testing and storing documentation. Future work may involve a more extensive SLR considering aspects such as cybersecurity, autonomous vehicles, and electrified vehicles.

## REFERENCES

AHANGARI, H., *et al.* **Analysis of design parameters in safety-critical computers**. IEEE Transactions on Emerging Topic in Computing, 2018.

ANAND, S., *et al.* **An orchestrated survey of methodologies for automated software test case generation.** Journal of Systems and Software, v. 86, n. 8, p. 1978–2001, 2013.

ANTINYAN, V.; STARON, M. **Rendex:** A method for automated reviews of textual requirements. Journal of Systems and Software, v. 131, p. 63–77, 2017.

ANTINYAN, V. **Revealing the complexity of automotive software.** In Proceedings of the 28th A.C.M. joint meeting on European software engineering conference and Symposium on the foundations of software engineering, 1525–1528, 2020.

ARRIETA, A., *et al.* **Search-based test case prioritization for simulation-based testing of cyber-Physical system product lines.** Journal of System and Software, v. 149, p. 1–34, 2019.

ARTS, T.; MOUSAVI, M. R. **Automatic consequence analysis of automotive standards (AUTO-CAAS)**. In 2015 First International Workshop on Automotive Software Architecture (WASA), p. 35-38, 2015.

AUTOSAR. **AUTOSAR enabling continuous innovation.** AUTOSAR. ORG. Available from: https://www.autosar.org/about/history/. Access on: 09, May,2021.

AUTOSAR GbR. **Technical Overview [Technical report].** 2008.

BAHIG, G.; EL-KADI, A. Formal verification of automotive design in compliance with ISO 26262 design verification guidelines. **I.E.E.E. Access**, v. 5, p. 4505–4516, 2017.

BEECHAM, S., *et al.* Preparing tomorrow's software engineers for work in a global environment. **I.E.E.E. Software**, v. 34, n. 1, p. 9–12, 2017.

BERNARDI, P., *et al.* Development flow for on-line core self-test of automotive microcontrollers. **I.E.E.E. Trans Computers**, v. 65, n. 3, p. 744–754, 2016.

BERNARDI, P., *et al.* Software-based self-test techniques for dual-issue embedded processors. **IEEE Transactions on Emerging Topics in Computing**, v. 8, n. 2, 464–477, 2020.

BOULANGER, J. L.; DAO, V. Q. Requirements engineering in a model-based methodology for embedded automotive software. In: **IEEE International conference on research, innovation and vision for the future in computing and communication technologies**, 2008.

BRAUN, P., *et al.* **Guiding requirements engineering for software-intensive embedded systems in the automotive industry**. Computer Science-Research and Development, p. 21-43, 2014.

CHEN, S., *et al.* A novel integrated simulation and testing platform for self-driving cars with hardware in the loop. **I.E.E.E. Transactions on Intelligent Vehicles**, v. 4, n. 3, p. 425–436, 2019.

CHOI, Y. A configurable V&V framework using formal behavioral patterns for OSEK/VDX operating systems. **Journal of System and Software**, v. 137, p. 563–579, 2018.

DEICKE, M., *et al.* Simulation of hardware specific components of ecu software in virtual verification. **ATZ Elektronik**, v.7, p. 52-55, 2012.

DELAMARO, M.; JINO, M.; MALDONADO. J. **Introdução ao teste de software**. Campos, 2007.

DEVMEDIA. **Testes funcionais de Software**. Available from: https://www.devmedia.com.br/testes-funcionais-de-software/23565#3. Access on: 30, aug. 2021.

DIVAKARLA, K. P.; EMADI, A.; RAZAVI, S. N. Journey mapping–A new approach for defining automotive drive cycles. **IEEE Transactions on Industry application**, v. 52, p. 5121-5129, 2016.

DURISIC, D., *et al.* Measuring the impact of changes to the complexity and coupling properties of automotive software systems. **Journal of System and Software**, v. 86, n. 5, p. 1275–1293, 2013.

EFRATI, A. **Uber finds deadly accident likely caused by software set to ignore objects on road**. The Information, 2018. Available from: https://www.theinformation.com/articles/uber-finds-deadly-accident-likely-caused-by-software-set-to-ignore-objects-on-road?shared=56c9f0114b0bb781. Access on: 10, feb. 2022.

FERNANDEZ, S. M., *et al.* A survey on the benefits and drawbacks of AUTOSAR. In: **First International Workshop on Automotive Software Architecture (WASA)**, 2015.

FRAGAL, V. H., *et al.* Extending HSI test generation method for software product lines. The **Computer Journal**, 62, n. 1, p. 109-129, 2018.

FREY, S., *et al.* How software architects drive connected vehicles. **ACM Digital Library**, v. 33, n.6, p. 41-47, 2016.

GRAF, S., *et al.* IVaM: implicit variant modeling and management for automotive embedded systems. In: **International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)**, 2013.

HAGHIGHATKHAH, A., *et al.* Automotive software engineering: A systematic mapping study. **Journal of System and Software**, v. 128, p. 25–55, 2017a.

HAGHIGHATKHAH, A., *et al.* Improving the state of automotive software engineering. **I.E.E.E. Software**, v. 34, n. 5, p. 82–86, 2017b.

IEEE. Institute of Electrical and Electronics Engineers. **IEEE standard glossary of software engineering terminology**. IEEE standard 610.12-1990.

IQBAL, D. Requirement validation for embedded systems in automotive industry through modeling. **I.E.E.E. Access**, v.8, p. 8697-8719, 2020.

ISO. ISO 26262-1:2018. **Road vehicles–functional safety**. Available from: https://www.iso.org/standard/68383.html#:~:text=Road%20vehicles%20%E2%80%94%20Functional%20safety%20%E2%80%94%20Part%201%3A,installed%20in%20series%20production%20road%20vehicles%2C%20excluding%20mopeds. Access on: 09, May, 2021.

JORDAN, C., *et al.* Framework for flexible, adaptive support of test management by means of software agents. **IEEE Robotics and Automation Letters**, v.4, n.3, p. 2754-2761, 2019.

JUHNKE, K.; TICHY, M.; HOUDEK, F. Challenges concerning test case specifications in automotive software testing: Assessment of frequency and criticality. In: **44th Euromicro conference on software engineering and advanced applications (SEAA)**, v. 29, n. 1, p. 39–100, 2018.

KASOJU, A.; PETERSEN, K.; MÄNTYLA, M. V. Analyzing an automotive testing process with evidence-based software engineering. **Information and Software Technology**, v. 55, n. 7, p. 1237–1259, 2013.

KILLIAN, R., **What is a TUV certification**? Available from: what is TUV certification? —1000Bulbs.com blog, 2018. Access on: 09, apr. 2021.

KIM, B. G., *et al.* Testing autonomous vehicle software in the virtual prototyping environment. **I.E.E.E. Embedded Systems Letters**, v. 9, n. 1, p. 5–8, 2016.

KLENDAUER, R., *et al.* Using the IDEAL software process improvement model for the implementation of automotive SPICE. In: **5th International workshop on co-operative and human aspects of software engineering (CHASE)**, 2012.

KODALI, A., *et al.* Fault diagnosis in the automotive electric power generation and storage system (EPGS). **IEEE/ASME Transactions Mechatronics**, v. 18, n. 6, p. 1809–1818, 2013.

KÖNIG, S., *et al.* Flexible scheduling of diagnostic tests in automotive manufacturing. **Flex Services Manufacturing Journal**, v. 35, n. 2, 320–342, 2023.

KOVACS, E. **NIST tools finds errors in complex safety-critical software.** Available from: https://www.security-week.com/nist-tool-finds-errors-complex-safety-critical-software, Access on: 15. Apr. 2021.

KRIEBEL, S., *et al.* Improving model-based testing in automotive software engineering. In: **Proceedings of the 40th international conference on software engineering: software engineering in practice track (ICSE-SEIP2018)**, pp 172–180, 2018.

KRIEG, A., *et al.* Power and fault emulation for software verification and system stability testing in safety critical environments. **IEEE Transactions on industrial informatics**, v. 9, n. 2, 1199–1206, 2013.

KUHRMANN, M., *et al.* Flexible software process lines in practice: A metamodel-based approach to effectively construct and manage families of software process models. **Journal of systems and software**, v. 121, p. 49–71, 2016.

LIDA, T., *et al.* PLE for automotive braking system with management of impacts from equipment interactions. In: **Proceedings of the 20th international systems and software product line conference**, p. 232-241, 2016.

LI, X., *et al.* ParallelEye-CS: A new dataset of synthetic images for testing the visual intelligence of intelligent vehicles. **I.E.E.E. Transactions on Vehicular Technology**, v. 68, n. 10, 9619–9631, 2019.

LOCHAU, M., *et al.* Delta-oriented model-based integration testing of large-scale systems. **Journal of Systems and Software**, v. 91, p. 63–84, 2014.

MOSTAFA, S. A., *et al.* An agent-based inference engine for efficient and reliable automated car failure diagnosis assistance. **IEEE Access**, v.6, p. 8322-8331, 2018.

NALIC, D., *et al.* Stress testing method for scenario-based testing of automated driving systems. **I.E.E.E. Access**, v. 8, p. 224974–224984, 2020.

NEUROHR, C., *et al.* Criticality analysis for the verification and validation of automated vehicles. **I.E.E.E. Access**, v. 9, p. 18016–18041, 2021.

PARRA, A., *et al.* Validation of a real-time capable multibody vehicle dynamics formulation for automotive testing frameworks based on simulation. **I.E.E.E. Access**, v. 8, p. 213253–213265, 2020.

PEKARIC, I.; SAUERWEIN, C.; FELDERER, M. Applying security testing techniques to automotive engineering. In: **Proceedings of the 14th international conference on availability, reliability and security**, pp 1–10., 2019.

PETRENKO, A., *et al.* Model-based testing of automotive software: some challenges and solutions. In: **52nd ACM/EDAC/IEEE Design automation conference (DAC)**, 2015.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de Software uma abordagem profissional**. Porto Alegre, 2011.

PRETSCHNER, A., *et al.* **Software engineering for automotive systems:** A roadmap. Future of software engineering. 2007.

RAJABLI, N., *et al.* Software verification and validation of safe autonomous cars: A systematic literature review. **I.E.E.E. Access**, v. 9, p. 4797–4819, 2020.

RANA, R., *et al.* Analysing defect inflow distribution of automotive software projects. In: **Proceedings of the 10th international conference on predictive models in software engineering**, p. 22-31, 2014.

RANA, R., *et al.* Analyzing defect inflow distribution and applying bayesian inference method for software defect prediction in large software projects. **Journal of Systems and Software**, v. 117, p. 229–244, 2016.

RANA, R., *et al.* Early verification and validation according to ISO 26262 by combining fault injection and mutation testing. **International conference on software technologies**, v. 457, p. 164-179, 2013.

RANA, R., *et al.* Selecting software reliability growth models and improving their predictive accuracy using historical projects data. **Journal of systems and software**, v. 98, p. 59–78, 2014.

REDMILL, F.; ANDERSON, T. **Retrieved from SysML.org. Safer systems.** SysML.org.Systems. ML OPEN Source Project–What is SysML? Who created SysML? Available from: https://sysml.org/?msclkid=45dcb1bece6211eca1163fb9a4534962. Access on: 07, may. 2022.

RODRIGUEZ, M.; PIATTINI, M.; EBERT, C. Software verification and validation technologies and tools. **I.E.E.E Software**, v. 36, n. 2, p. 13–24, 2019.

ROSA, L. **Testes de software de controle para sistemas embarcados.** UFRGS LUME repositório digital. Available from: monografia-VF+ (ufrgs.br). Access on: 09, Mar,2020.

SCHROEDER, J., *et al.* Unveiling anomalies and their impact on software quality in model-based automotive software revisions with software metrics and domain experts.

In: **Proceedings of the 25th international symposium on software testing and analysis**, p. 154-164, 2016.

SEO, J., CHOI, B.; YANG, S. W. Lightweight embedded software performance analysis method by kernel hack and its industrial field study. **Journal of System and Software**, v. 85, n. 1, p. 28–42, 2012.

SHOOMAN, M. L. Bohrbugs, Mandelbugs, Exhaustive Testing and Unintended Automobile Acceleration. In: **IEEE 23rd international symposium on software reliability engineering workshops,** 2012.

SOMMERVILLE, I. **Engenharia de software**. São Paulo, Brazil, 2011.

STAHL, T.; DIERMEYER, F. Online verification enabling approval of driving functions—implementation for a planner of an autonomous Race Vehicle. **I.E.E.E. Open Journal Intelligent Transportation Systems**, v. 2, p. 97–110, 2021.

STAPLES, G. **Is Hardware in the Loop (HIL) and software in the Loop (SIL) testing?.** Electric RC AIRCRAFT GUY, LLC—RC, Arduino, programming, & electronics: what is Hardware in the Loop (HIL) and software in the Loop (SIL) testing?, 2018. Available from: ElectricRCAircraftGuy.com. Access on: 09, may. 2021.

SYNOPSYS. **What is ASIL (Automotive Safety Integrity Level)?.** Available from: https://www.synopsys.com/automotive/what-is-asil.html. Access on: 09, apr. 2021.

SZALAY, Z. Next generation x-in-the-loop validation methodology for automated vehicle systems. **I.E.E.E. Access**, v. 9, p. 35616–35632, 2021.

TECHOPEDIA. **Embedded Software.** TECHOPEDIA. Available from: https://www.techopedia.com/definition/29944/embedded-software#:~:text=Embedded%20software%20is%20a%20piece%20of%20software%20that,constraints%20because%20of%20the%20device%E2%80%99s%20limited%20computing%20capabilities. Access on: 09, apr. 2021.

THURIMELLA, A. K.; BRÜGGE, B. A mixed-method approach for the empirical evaluation of the issue-based variability modeling. **Journal of System and Software**, v. 86, n. 7, p. 1831–1849, 2013.

ULRICH, K. **A corrida tecnológica na indústria automobilística**. 2021. Available from: https://www.dw.com/pt-br/a-corrida-tecnol%C3%B3gica-na-ind%C3%BAstria-automobil%C3%ADstica/a-56760849. Access on: 10, jan. 2022.

UML. **What is UML**. Available from: https://www.uml.org/what-is-uml.htm?msclkid=7f64613bce6411ec96121937444defa6, 2005. Access on: 07, may. 2022.

VOGELSANG, A. Feature dependencies in automotive software systems: extent, awareness, and refactoring. **Journal of System and Software**, v. 160, 2020.

VÖST, S. & WAGNER, S. Trace-based test selection to support continuous integration in the automotive industry. In: **IEEE/ACM international workshop on continuous software evolution and delivery (CSED)**, 2016.

WIECHER, C.; GREENYER, J.; KORTE, J. Test-driven scenario specification of automotive software components. In: **22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)**, 2019.

WOHLIN, C., *et al.* Experimentation in Software Engineering: an introduction. Kluwer academic publishers. Software engineering practices in the US and Japan. **I.E.E.E. Computers**, v. 00, p. 57–66.2000.

WOHLRAB, R.*;* KNAUSS, E; PELLICCIONE, P. Why and how to balance alignment and diversity of requirements engineering practices in automotive. **Journal of Systems and Software**, v. 162, 2020.

YAMAGUCHI, T., *et al.* Combining requirement mining, software model checking and simulation-based verification for industrial automotive systems. In: **Formal methods in computer-aided design (FMCAD)**, 2016.

YOUR MECHANIC. **What is a vehicle's electrical control unit? Your mechanic**. Available from: www.yourmechanic.com/article/what-is-a-vehicle-s-electrical-control-unit. Access on: 09, apr. 2021.

Submitted: July 7, 2023.

Approved: August 1, 2023.

[1] Master's student in Technological Innovation, MBA in Project Management, Graduate in Electrical Engineering – Emphasis in Electronics. ORCID: https://orcid.org/0000-0002-0068-3820. Currículo Lattes: http://lattes.cnpq.br/2662261891056820.
[2] Advisor. Post-doctorate in Software Engineering for Safety-Critical Systems, PhD in Electrical Engineering, Master's in Information Technology, Specialization in Systems Analysis, Degree in Control and Automation Engineering (Mechatronics) and Degree in Data Processing. ORCID: https://orcid.org/0000-0002-7266-5840. Currículo Lattes: http://lattes.cnpq.br/0203910403476737.
[3] Doctorate in Mechanical Engineering, Master's degree in Mechanical Engineering, Bachelor's degree in Mechanical Engineering. ORCID: https://orcid.org/0009-0009-5225-2633. Currículo Lattes: http://lattes.cnpq.br/0800909771825237.