IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 16 June 2023, accepted 7 July 2023, date of publication 11 July 2023, date of current version 24 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3294256

SURVEY

A Comprehensive Survey on Software as a Service (SaaS) Transformation for the Automotive Systems

DAVID FERNÁNDEZ BLANCO[1,2], FRÉDÉRIC LE MOUËL[1], TRISTA LIN[2], AND MARIE-PIERRE ESCUDIÉ[3]

[1]Univ Lyon, INSA Lyon, Inria, CITI, EA3720, 69621 Villeurbanne, France
[2]STELLANTIS, 78140 Velizy-Villacoublay, France
[3]Univ Lyon, INSA Lyon, Institut Gaston Berger, 69621 Villeurbanne, France

Corresponding author: David Fernández Blanco (david.fernandez-blanco@insa-lyon.fr)

The research leading to the results presented in this study was supported by Stellantis under the collaborative framework OpenLab VAT@Lyon, involving STELLANTIS and CITI Laboratory (Association Nationale Recherche Technologie (ANRT) contract n°2020/1415).

ABSTRACT Over the last few decades, automotive embedded Information and Communication Technology (ICT) systems have been used to enhance vehicle performance and enrich peopleâ's driving experience, increasing the panel of software features within them. However, even though until now automakers have kept up with the innovation pace in terms of the functionalities that have been offered to passengers, the majority of automakers' efforts have concentrated on bringing in these new functionalities by adding an unceasingly larger set of ECUs. All of this has been done without evolving any of the embedded software architecture consequently, due to budgetary constraints, legislative limitations, retro-compatibility problems, and a lack of awareness of the trending IT innovation. This unbalanced progress has then led to a substantial increase in in-vehicle architectural complexity, which has become a major concern for automakers nowadays as it makes the vehicle repairing process more complex, decreases software traceability and clashes with the objective of having higher business flexibility, modularity, and dynamicity within the vehicles. In this paper, we are going to go through literature, both academic and industrial, and propose a comprehensive study into automotive system transformation. We begin by giving a detailed analysis of the causes of evolution under five axes - i.e., society, business, industry, application, and technical. Then, we discuss the convergence of cars and software life cycles and propose a three-layered analysis of automotive ICT systems consisting of architecture design, software pipelines, and run-time management. Finally, we are going to propose certain detailed guidelines on the evolution perspectives for automotive systems through deriving from the convergence of advances in IT, as well as current and future automotive environmental constraints.

INDEX TERMS Automotive ICT system, software defined vehicles, system architecture, vehicle-to-cloud (V2C)/vehicle-to-anything (V2X), software pipelines, E/E architecture.

I. INTRODUCTION

As the cities and societies evolve to be more connected, accessible, and flexible, the automotive industry and the current perception of mobility services and vehicle ownership are following the same path (cf. §IV). Automobiles have become a part of a bigger and more complex network in which cooperative services and cloud-based applications combined with enhanced onboard computing capabilities are paving the way for innovation, moving towards a fully connected autonomous vehicle.

Over the last few decades, Electronic Control Units (ECUs) have been used to ameliorate vehicle performances and enrich the driving experience by adding a multitude of new

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

innovative services such as Over-The-Air Update Frameworks, Anti-Lock Braking Systems, embedded GPS systems, automated parking systems, and so on. These changes have shifted the value of digital features in vehicles by up to 35- 40% in today's market [1], thus, becoming, for most people, a differential factor when it comes to buying a new vehicle. Henceforth, this trend is expected to continue increasing substantially in the coming years, making embedded vehicle architectures more flexible from a business point of view, enhancing the quantity of applications and customisation possibilities, and opening the existing collaboration frontiers of the automotive sector to new-coming IT companies such as Google, Facebook, Microsoft, or Amazon.

However, even if until now the automotive industry has managed to keep pace with innovation in terms of the different functionalities offered to the passengers, all the efforts have been focused on launching these new functionalities by adding an unceasingly larger set of ECUs without evolving all the embedded software architecture consequently due to budgetary constraints, legislative limitations, retro-compatibility problems or a general lack of awareness regarding trending IT innovation [2]. This unbalanced progress has then, as we can see in Buckl et al. [3], considerably increased the complexity of the system and cost for both developing and maintaining new services, while complicating the path to evolve application development methodology and bring in new innovative software-related business proposals.

In this context, adapting the current Electrical/Electronic (E/E) and Software architectures to the upcoming requirements, without forgetting about the already present base, has become a vital necessity, but also a major challenge for the automotive sector. As could not be otherwise, multiple industrial and academic researchers center themselves nowadays on trying to stage the phases of this transformation. On the one hand, if we look more closely at the industrial research initiatives, we can appreciate an increasing emergence of new standardization groups and projects such as *SOAFEE*, [4], *Eclipse Software-Defined Vehicle (SDV)*, [5], *Eclipse Automotive*, [6], *Android Automotive*, [7], *MIH EV*, [8], *eSync*, [9] or *Connected Vehicle Systems Alliance* (COVESA) [10], previously called *GENIVI Alliance*, trying to establish some inter-company basis for the architectural transformation. However, these solutions are still at an early stage and no clear standards have been proposed as of yet. On the other hand, if we take an in-depth look at the current academic research contributions, considerable effort has been put into the economical and organisational aspects of software architecture transformation [11], the benefits and applicability of upcoming paradigms [12], [13], [14], [15], the evolution of embedded E/E architecture [16], [17], and the evolution of network architecture, either around communication protocols used in automotive embedded systems [18], [19], [20], [21], [22] or network evolvability and adaptability [23], [24], [25]. However, as we will detail in §II, there is a considerable research gap with regard to the study of the automotive ICT architectures as a whole (i.e. both software and E/E architecture, software integration pipelines and run-time control services) as well as the relationships between its layers, which makes it more difficult to apply the contributions to the automotive environment.

Thus, in this paper, we will research the different, complete, or partial, software architecture related proposals from 1986 to early 2023, and give comprehensive guidelines for automotive architecture transformation, focusing on the analysis of software and system development life cycle (i.e., design, development, integration, testing, maintenance, etc.) and the whole panel of evolution causes, including societal evolutions, technical and application evolutions, business evolutions, etc. In addition, this paper has a second objective regarding converging the technical advancements of IT (especially those of Software as a Service (SaaS) / Platform as a Service (PaaS) [26] in Cloud Computing) and the constraints and needs, old or new, of the automotive systems in terms of flexibility, safety, security, dynamicity, etc. Hence, the main contributions of the paper are summarised as follows:

- A detailed analysis of the causes of evolution under five axes: societal evolutions (cf. §IV-A), business evolutions (cf. §IV-B), automaker design process evolutions (cf. §IV-C), the evolution of application profiles and requirements (cf. §IV-D), and technical causes of evolution (cf. §IV-E).
- An analysis of the IT and automotive literature around a three macro-theme classification deriving from the convergence of the vehicle product and software life cycles (cf. §V). Thus, these three macro-themes are Architecture Design (cf. §VI), Software Pipelines (cf. §VII) and Run-time Management (cf. §VIII).
- The presentation of our complete vision for future automotive software architecture models, evolution perspectives and future work for the automotive systems deriving from the convergence between advances in IT and current and future automotive environments and constraints (cf. §IX).

The remainder of this paper is organised as follows: First, Section II gives a high-level analysis of the current literature research trends. Then, Section III presents a detailed scenario of the uses of software in the automotive industry to illustrate the resulting challenges of this transformation. Subsequently, Section IV details the causes leading to this unprecedented automotive IT evolution. After that, Section V discusses the convergence on software life cycle between the vehicles product and software life cycles, proposing a new division in three macro-themes and their associated open issues, that will be explained in more details later in Section VI - Architecture design, Section VII - Software Pipelines and Section VIII - Run-time Management. After that, Section IX will summarise all the studies in a detailed literature-based architecture evolution proposal, as well as detailing future associated works. Finally, Section X will present the conclusions of our study.

II. LITERATURE OVERVIEW

Many articles discuss the distinct aspects of software transformation within automotive systems, both on-board and off-board. In this section, we want to present the different main research trends in automotive ICT systems as well as the evolution of their presence in renamed IT and automotive conferences[1] and journals.[2] The results of this study of over 600 papers spanning the last decade, including some of those presented throughout this paper, are presented in Fig. 1. Thus, in the following paragraphs of this section we will investigate each of these research subjects one by one, presenting their most representative works. After that, we will precisely position the scope of our survey with regard to the rest of the state of the art.

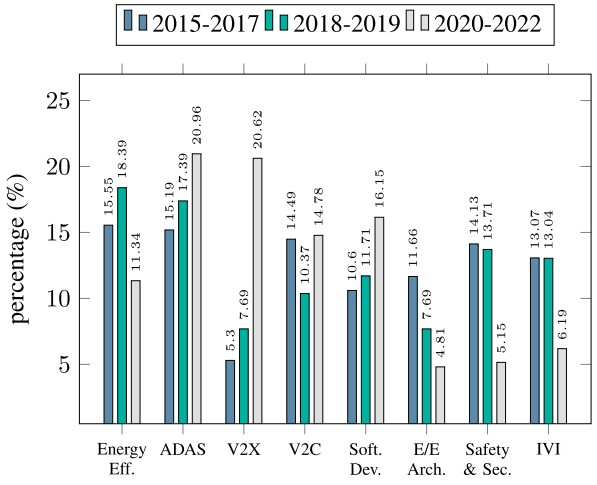


FIGURE 1. Popularity of the different academic research topics interest in the automotive domain since 2015.

From Fig. 1, we can highlight that the sectors whose interest has increased more over the last years are those concerning the V2X (either V2C, V2V or V2I) and the enhanced ADAS systems, closely followed by those focusing on software development & architecture paradigms, which is coherent with the evolution of the business and user requirements and interests presented later on this paper (cf. §IV-B). For the rest of the section, we will give an overview of each of the categories, giving some references that illustrate them.

[1]Among the conferences, we can highlight IEEE/IFIP Working Conference on Software Architecture (WICSA), ACM/IEEE International Conference on Model-Driven Engineering Languages and Systems (MODELS) or the IEEE Conference on Local Computer Networks (LCN) from the IT domain and IEEE Vehicular Technology Conference (VTC), IEEE Vehicular Networking Conference (VNC) or Intelligent Vehicles Symposium (IV) from the automotive domain.

[2]Among the journals, we can highlight IEEE Access and IEEE Transactions on Parallel and Distributed Systems from the IT domain and IEEE Vehicular Technology Magazine, IEEE Transactions on Vehicular Technology, IEEE Transactions on Transportation Electrification, IEEE Transactions on Intelligent Transportation Systems, and Vehicular Communications and IEEE Transactions on Intelligent Vehicles from the automotive domain.

First, the energy consumption and energy efficiency of vehicles has been addressed both at a low-level manner [27], [28], [29], [30] by seeking to optimize the consumption of different parts of the vehicle (i.e. engine, telecommunications unit…) and, also, at a higher level manner through route optimization, collaborative coordination, smart parking optimizations… [31], [32], [33], [34], [35]. Secondly, the software related functional advancements with regard to mobility, the context heterogeneity and the integration of the vehicle with the users, the network and other smart-city infrastructures, have been addressed separately depending on whether the application runs exclusively within the vehicle [36], [37], in coordination with other vehicles or infrastructure (V2X) [38], [39], or in coordination with the Cloud servers (V2C) [40], [41], either the central or edge nodes. Moreover, applications are usually grouped by their final objective such as entertainment (IVI) [42], [43] or driving assistance (ADAS) [44], [45]. Besides, several contributions appeared to target the evolution of software development methodologies, with the adaptation of the classic V-cycles and the agile manifest [46], [47], [48] and philosophies [49], [50]. Third, Security and Safety properties have been widely covered, both isolated and combined, through multiple surveys. In-vehicle has been addressed from a purely software design perspective in [51], [52], and [53], but also from a hardware point of view in [54], [55], and [56]. Similarly, security has also been covered domain by domain, first focusing on the network [57], [58], [59], [60], then on the application design [61], [62]. However, despite the fact that these two properties are usually studied separately, they are often addressed as an ensemble with regard to system threat analysis and risk assessment [63], [64]. Finally, the Hardware (E/E) architecture evolution, including more centralized and performant architectures, has been discussed in multiple surveys before [65], [66], [67], [68], [69], however, we haven't found any contribution that goes into detail about the hardware characteristics of the nodes, only the way to distribute them.

However, despite these advances, little information can be found concerning the study of both the E/E and Software automotive architectures as a whole, while, to the best of our knowledge, the study of integration between these features through the vehicle life cycle and the interactions in between remains completely unexplored. Thus, in this paper, we will focus on covering these missing spots, as well as surveying each of the mechanisms independently, in order to give an extensive view of the transformation progress and options in the automotive sector. In addition, we will add supplementary reviews of proposals from the IT sector, which leads this digital transition, all the while keeping in mind the importance of safety and security concerns required by automotive legislation and certificates. Finally, we will try to propose a theoretical reference for architecture and guidelines for fully dynamic and connected automotive architectures. This is one of the key development trends of the sector and will be explained in more detail in §IV.

III. USE-CASE SCENARIO: SOFTWARE IN THE AUTOMOTIVE INDUSTRY

In this section, we provide a detailed use-case scenario (cf. Fig. 2) depicting the complexity of the software life cycle through all the vehicle software life-cycle phases (i.e., Development, Deployment, and Maintenance). This is a simplification of the life cycle presented in §V, leaving all the economic aspects aside. First, it is worth noting that this scenario was elaborated based on the literature [70], [71] and from the innovation reports from several automaker groups (e.g., STELLANTIS, General Motors, Toyota, BMW…), who all share similar internal processes.

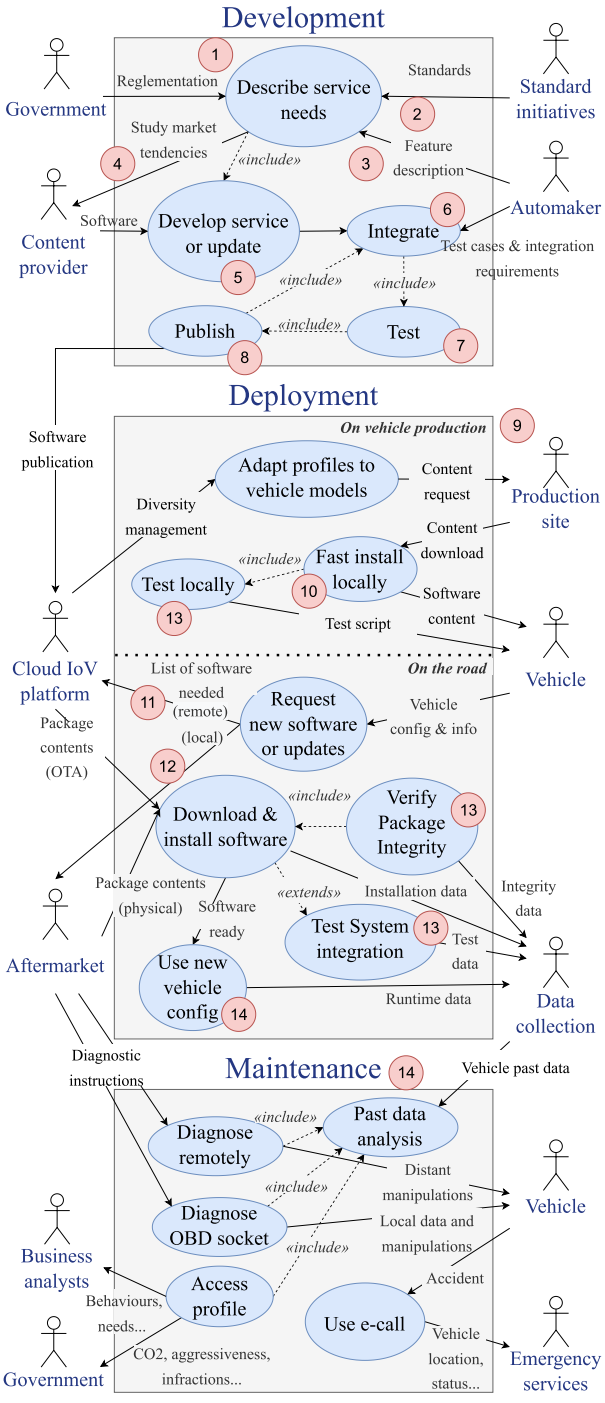


FIGURE 2. Software life-cycle through the vehicle states use case scenario.

To fully illustrate and highlight the complexity of this cycle, we are going to focus on the most frequent case, an application developed by a supplier (or content provider) dependent on specific hardware, shared between multiple car models within the same automaker, and that provides recent updates regularly. Some example applications would be the rear camera controller application or the lidar & radar collection application. In this case, the study market tendencies (cf. Fig. 2–4), automaker demands (cf. Fig. 2–3), and the analysis of the standards (cf. Fig. 2–2) and regulations (cf. Fig. 2–1) are carried out by the supplier before developing (cf. Fig. 2–5) the product. After that, the supplier will work directly with the automaker to integrate everything (cf. Fig. 2–6,7,8) into the rest of the in-vehicle architecture, elaborating the different installations, testing and monitoring instructions specific to this application and for each necessary update, in order to adhere to the security certificates and legislative constraints. The cycle continues after with the deployment phase when these applications are deployed, first, in the vehicle production phase (cf. Fig. 2–9) using local fast installation tooling (cf. Fig. 2–10) and will be updated remotely (or locally in a garage) for the subsequent updates (cf. Fig. 2–11,12). After being successfully installed and tested (Fig. 2–13), the monitoring and data collection instructions for this application will be added to the global maintenance tool (cf. Fig. 2–14), thus, being able to manage its correct functioning on the road. However, this cycle does not always take place like this. The party developing the software can also be the same automaker or even an independent company wanting to sell a product instead of a supplier under a contract. Besides, the software can be deployed at distinct stages (i.e., Over-The-Air on the road, locally in an after-sales garage or on the production line) and with a periodic update that can range from never updating the software once installed to updating it monthly. This causes a hard-to-handle heterogeneity on the integration, certification, installation, and maintenance pipelines, since every application is designed by a different organisation without following a standardised procedure that, in turn, making the in-vehicle systems more complex.

Note that, in this environment, the heterogeneity on the different vehicle variants and user preferences also has a negative impact on the complexity of the system, especially whenever deploying new software. However, even though, up until now, current innovation pace has kept up with user expectations, given the recent evolution of user requirements (cf. § IV-A), this is no longer the case. The role of the vehicle is changing and, thus, business dynamicity (cf. § IV-B) needs to evolve along with it. In this context, and as we mentioned in the introduction, vehicles will need to be able to add all these new applications with different

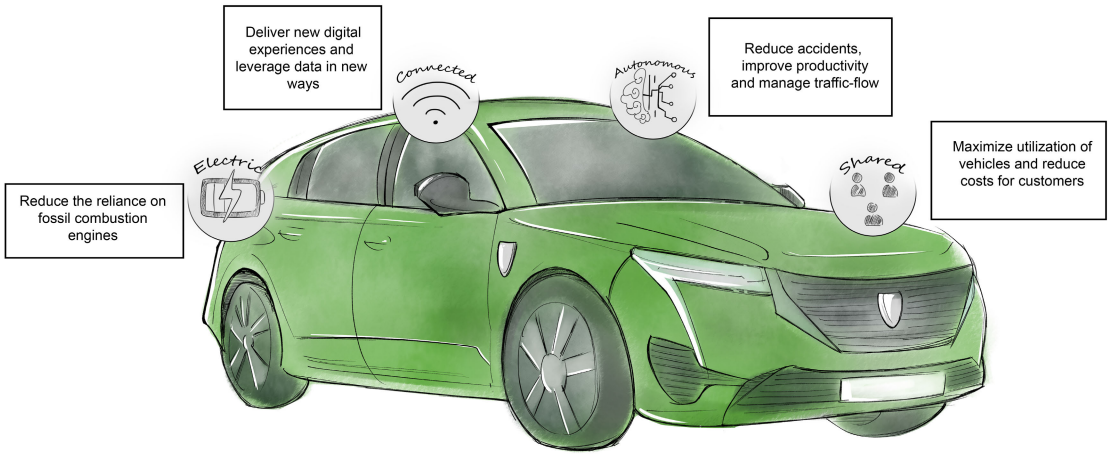


FIGURE 3. Automotive technology mega-trends described through C.A.S.E (Connected Autonomous Shared Electrified) acronym.

profiles (cf. § IV-D), coming from both internal and third party application stores, remotely and with a higher dynamicity (cf. § IV-C), without being limited by the lack of network or hardware capacities, upcoming standards and regulations, integration constraints, nor safety and security needs. This evolution involves a major change in perspective within the vehicle industry. Extra effort will be needed to keep pace with with the software and E/E architecture needs and to simplify software development, delivery, and maintenance, while all the time keeping in mind that critical safety and security levels must be respected (cf. § IV-E).

However, as the need for defining standardised pipelines for application deployment and maintenance is a past challenge for the IT industry [72], [73], this problem has been widely addressed in research on the cloud computer field, first for homogeneous [74], [75], and more recently heterogeneous environments [76]. Thus, the automotive industry should take advantage of these advancements to make its ICT systems and software life cycle management less complex. On the other hand, despite the lack of modern software standards, from this use case we can also highlight that the Automotive industry maintains strong fundamentals regarding how to manage the integration of legacy systems and highly heterogeneous applications coming from completely different suppliers, all the while keeping the system safe and secure [77], [78]. Hence, over the last few decades, in-vehicle software and systems have focused on safety and security and not on dynamicity and flexibility, which is a recent trend for the automotive industry. Maintenance and integration costs are nowadays too high, raising the need for a review into in-vehicle ICT systems.

IV. CAUSES OF EVOLUTION

As we said before, the use of software has enabled vehicles to become more connected, autonomous, and efficient. However, the transformation of the automotive sector is not solely due to technological advancements, but rather a combination of factors. This section will provide a detailed analysis of the causes of evolution in the automotive industry, specifically focusing on societal factors, business factors, automaker design process factors, application profiles and requirements factors, and technical factors. The objective of this section is to give a better understanding of the driving forces behind the transformation of the automotive sector and how they will influence the future of the automotive sector.

A. SOCIETAL EVOLUTIONS

In recent years, and as we previously stated, cities and societies have evolved to be more connected, accessible, and flexible and, with it, the role of the vehicle and the behavior of the new generations of customers have drastically shifted. Even though the nature of these societal changes are multifaceted, at a high level of abstraction, we can highlight some of the principal interesting changes such as a rising interest in mobility instead of car ownership (cf. § IV-A1), a decreasing interest in driving while in the vehicle (cf. § IV-A2), a shift in the general perception of the vehicle (cf. § IV-A3), and a convergence toward a greener vehicle production and usage model (cf. § IV-A4). These four principal interests converge into the industrial guiding megatrends for vehicle transformation *C.A.S.E.* (cf. Fig. 3).

1) FROM CAR OWNERSHIP TO OCCASIONAL USE

The way young generations feel about vehicles in developed countries, mostly those in urban areas, has progressively shifted due to multiple factors. Among these factors, we can include the development of urban transportation networks (bus, subway, train, bicycles, taxis…) in most cities in developed countries [79], [80], increasing car usage restrictions [81], a lack of parking space [82], [83], the environmental impact of using and producing private transport [84], [85], elevated maintenance [86], [87], gasoline [88], insurance costs [89], etc…Indeed, studies [90], [91] show that interest in car ownership has drastically decreased among the

young population strata. For example, new vehicle purchases per-capita for 18- to 54-year-olds in the US has declined more than 15% within the last 20 years. Studies [92], [93], [94] show that financial issues, a lack of need for a vehicle, mostly in urban areas, and a rising concern for the environment are the major causes that delay or rule out having a car or even a driving license. Moreover, studies suggest that car use has reached its peak in the last decade [95], [96] leading to a phenomena of voluntary demotorisation and reduced car ownership, particularly in urban areas [97], [98].

Thus, as car ownership is considered to be more and more of a struggle, motivated by the aforementioned factors, among the younger population [99], on-demand car-ride services (such as Uber, Lyft or Bolt), carpooling services (such as BlaBlaCar or Via) and car-sharing services (such as Citiz, Ouicar or Zipcar) [100], [101], [102], have experienced a rapid growth. This growth has led some traditional automaker companies to join this Mobility-as-a-Service market trend, such as STELLANTIS with the acquisition of Share Now [103] or BMW with Drive Now [104]. This trend of using vehicles as punctual, flexible, shared resources within the smart-city ecosystem is expected to keep growing from its current 8% of total global miles travelled to over 26% in 2030 [94]. Having a car population consisting of a high percentage of publicly shared vehicles would enhance the value of some already present research topics such as dynamic driver profiling [105], pay-as-you-go applications [106] or dynamic software-defined vehicle systems [107], all the while redirecting the business's focus towards mobility instead of vehicle production (cf. § IV-C). However, other studies [108] highlight the extent to which giving up car ownership can be a stigmatising factor for rural areas [92] or developing countries [109], in which there are no real mobility alternatives because of a lack of infrastructure or country instability.

2) DECREASING INTEREST IN DRIVING

In addition to the changes presented previously in this section, which suggest a fall in car ownership among younger generations, people also seem to have changed their behaviour related to car use. Driven by the Fear of Missing Out (FoMO) [110] phenomenon, new generations dislike more and more tasks in which they have to invest all their attention for a long period of time, such as driving or the process of buying the car itself [99]. In addition to that, teleworking democratisation [111] and rising real estate prices within urban areas [112], [113] are leading to an increase in the home-to-work distance people are willing to assume, which can be see with the recent exodus towards peri-urban or rural areas. This distance increase is especially remarkable withing the lower strata of society [114], [115]. As the travel distance increases, which increases as well the massiveness of daily traffic jams [116] and road stress, it also deteriorates the image of driving. Which has lead to what is perhaps one of the biggest differences between the older and recent generations with regards to driving, whilst previous generations perceive driving as a fun activity, recent generations perceive driving as a chore [99], something perfunctory.

In addition, with the evolution of network coverage and the connectivity and the IT capabilities of smart-phones, laptops and the vehicles themselves [16], [117], numerous opportunities for virtual relationships, online shopping and other remote activities, such as working or entertainment, so many activities can now be carried out without physically moving, even when on the road [118]. Therefore, the possibility of multitasking with your smartphone while going to work, school, or elsewhere is viewed as one of the main reasons for taking public transport [119] and is a desired feature for drivers [120], [121]. Semi-and-full-autonomous vehicles will therefore help increase traffic efficiency and travel time, allowing drivers to multitask while on the road, while increasing traffic and energy efficiency [122], etc.

3) SHIFT ON THE VEHICLE SYMBOLISM

Since the invention of cars during the last few decades of the XIXth century to the early XXth century, vehicles were a rare asset [123] and were viewed as the ultimate symbol of wealth, freedom, independence and autonomy [118]. This turned them into a capital status symbol all around the world. However, as mass production spread, focusing on functional cost-efficient vehicles further down the class hierarchy, mere ownership lost its ability to convey distinction. Thus, a clear distinction appears between luxury vehicles, which are nowadays affordable for a small percentage of the population and are still viewed as status symbols worldwide [123], and functional cars. Today, given the advances in mobile phones and internet, smartphones arguably provide as much freedom as cars, offering instantaneous access to information, family, friends, shopping…which, also helped by progressively better urban transport availability, is responsible for fewer trips being made, and, therefore, for the decrease in ownership interest [123] unless we are talking about vehicle necessity (i.e. family needs [124], uncovered rural areas [93], etc.). Hence, we can say that vehicle perception is changing from being a status symbol, towards utility [125]. However, in developing countries, given the elevated price of vehicles and the poor public transport infrastructures, vehicles are still rare and considered by the populations both a necessity and a status symbol [126], [127], [128].

Moreover, users are questioning identification as they now identify less with communities or brands and therefore want their car to be more unique and tailored to their individual needs [129], [130], [131]. According to this concept, products and services should not be adapted to consumer segments, but rather to the preferences of individual consumers. However, customisation also has a negative effect on customer acceptance, as each product needs to be adapted for a specific personality [132], which brings higher costs. Therefore, a balance needs to be found between consumer acceptance and satisfaction of individual customisation. This challenge

is even bigger when we consider the previous trends of shared driving or carpooling services.

4) THROUGH A GREENER VEHICLE PRODUCTION AND USAGE

The 5th and 6th IPCC Climate Change Report [133], [134], as well as the Paris Agreement Goals adopted at COP21 [135], expose the unbalanced footprint within the different main sectors of activity, with regard to resource needs, biodiversity impact, energy consumption, GHG emissions, etc. From all the previously cited factors, we are going to focus in detail on the most frequently studied GHG emissions. The transport sector accounted for 16.2% of the total GHG emitted per year, with the largest part being(11.9%) produced directly from road transport [133], the majority of that coming from passenger travel (i.e., cars, motorcycles and buses). However, this statistic only shows the impact of vehicles once they are on the road, completely ignoring the costs associated with the impact on the rest of the vehicle product life cycle (i.e. production phase, maintenance, end of life, etc.) [136], [137], [138].

On one hand, with regard exclusively to the usage-related impact, technological breakthroughs in the internal combustion engine and migration to other power sources such as electricity or hydrogen fuel are driving down the emissions from vehicle driving [139], [140], [141], [142]. However, in order for these changes to be sustainable in the long term, further technical and societal advancements are required in subjects such as electricity decarbonisation [143], [144] or battery recycling / reuse [141], [145].

On the other hand, with regard to vehicle lifecycle itself, studies show contrasting results on whether extending the vehicle lifetime would positively impact its footprint. Those in favor of extending it contend that by extending their lifetime, we would decrease the production of new vehicles [146], in turn, helping to reduce the CO_2 incidental from manufacturing new products. This argument also goes along with extending vehicle hardware / software repairability and upgradability meaning users would not need to change their vehicles once they are no longer functioning at an optimum level. Contrariwise, those against extending it [147], [148], [149], [150] claim that slowing the pace of product replacement will leave many older, less energy-efficient products in society, which will increase CO2 emissions incidental to product usage. However, in this case, what comes after the end of life of the vehicles is of critical importance increasing the need for higher re-use and shifting from linear to circular economies to be sustainable in the long time [151]. As may be the case, both these options rely on higher software and hardware compatibility and a separate management of hardware and software life cycles, moving closer to plug & play vehicle dynamics.

Finally, the critical environmental situation does not seem to be a differential factor for people when choosing whether to buy a vehicle or not [93]. As we said before, the need to have a car is the principal deciding factor for car ownership. However, despite it not being the main factor for consumers, it is a critical point for both Governments and Automakers, with multiple brands being transformed to full electric and many subsidies proposed by developed countries to motivate the population to move towards greener engines.

B. BUSINESS EVOLUTIONS

As society and user interests evolve through time to become a highly connected, autonomous, shared and electrified ecosystem (cf. § IV-A), new business opportunities arise [152], transforming the whole business ecosystem, which is critical for the traditional automotive manufacturers and suppliers in order to adapt to the current trends to maintain or better their current market position [152], [153], [154]. In this section, we propose an analysis of the business ecosystem focusing separately on two issues, first on product transformation opportunities (cf. § IV-B1) and then, on the company strategic aspects (cf. § IV-B2).

1) PRODUCT TRANSFORMATION

First, we need to highlight the shift in perceptions of the customer. While until now, the customer's role was limited to buying what Automotive manufacturers were offering, studies on co-creation processes [155], [156] have shown that adding digital interactions with the customers, integrating them into the processes of product design, is essential to rapidly sense and respond to changing customer needs, which has become vital for the survival of organisations in the digital age [157].

Secondly, if we take a closer look at the business consequences of societal transformations (cf. § IV-A), both shared-mobility services and autonomous driving mean a substantial change in the value proposition. On one hand, shared mobility changes the scope from delivering a product (the vehicle) towards also delivering a service (mobility). This shift in scope has had significant implications [154], [158], [159], [160], [161] on both the automaker's revenue model, which will likely be altered as these services comprise of pay-per-use pricing schemes, and the partnership model which has needed to be extended towards public transport providers. Moreover, Mobility as a Service redefines the traditional 1:1 relationship between vehicle and customer leading to an n:n type of relationship, which is a potential available business market for new innovative contributions. On the other hand, autonomous driving also substantially changes the value proposal as the customer no longer needs to drive the vehicle. Hence, while on the road, there is an available spot for innovation in both entertainment, comfort and, potentially, in many other sectors.

Finally, we feel it important to highlight that, due to the aforementioned reasons, the integration of the vehicle into Smart City environments and enhanced connectivity evolution trends, vehicles are a major innovation target for new IT-related services [162], [163], [164], whether in

entertainment, comfort, driving assistance etc. These new software-based business opportunities will become a new important form of revenue for both automakers and suppliers, relying once again on subscriptions and pay-per-use economic models. In this context, software deployment systems, such as OTA updates, have an even greater importance than before. Moreover, these services tend to produce a large amount of data [165], [166] that needs to be stored and that can be analysed and used for other business purposes, such as optimisation of vehicle diagnostic processes, custom advertising etc. and that can be exploited economically as is done in multiple other domains [167], [168].

2) STRATEGIC ASPECTS

As software transformation opens the pre-established automaker industry borders and enlarges its evolution perspectives and market opportunities [152], [153], [169], powerful non-traditional players, such as IT companies like Apple or Google, progressively enter into the automotive ecosystem, enforcing competition for both suppliers and the automakers themselves. Thus, traditional players see their position threatened as this transformation implies a major update, not only of their products, but also their internal processes, product life cycles and development methodologies (cf. § IV-C). Besides, traditional players may no longer have the full in-house competences to develop these new IT-related products and services, needing to restructure their workforce or collaborate with other IT-centered companies. Thus, in order to preserve the stability within their companies and comply with legislation, fully restructuring a company workforce and internal processes cannot be immediately done, making inter-company co-operation and outsourcing crucial in the coming years [170], [171], [172], [173], [174].

C. AUTOMOTIVE DESIGN PROCESS EVOLUTIONS

Motivated by societal evolution trends (cf. § IV-A) and business evolution perspectives (cf. § IV-B), automakers need to update their internal development and design processes to match the dynamicity required by the digital age, as they now have a decoupled life cycle between software and hardware and an unexploited innovation panel to cover. In this section, we will first (a) reevaluate the design principles on which automakers are based, and then (b) revisit the progress on their updates in development methodology. It is worth noting that with regard to these points, we will compare these advances to those in the IT industry and comment on their integration possibilities.

1) DESIGN PROCESS RE-EVALUATION: TOP-DOWN VS BOTTOM-UP

As depicted in Fig. 4, a software block can be seen as a mapping between functional (business, user, etc.) requirements (at the top level of abstraction) and the technical bases (at the bottom level of abstraction) that make them possible [175], [176]. This mapping is complex and is done through multiple

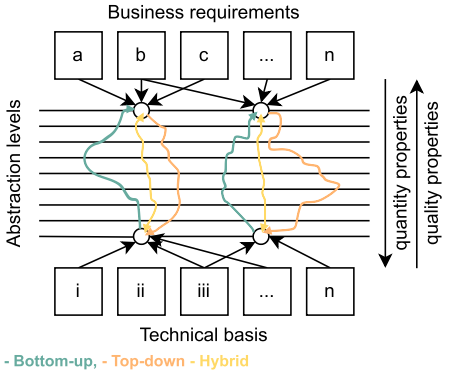


FIGURE 4. Software life-cycle through the vehicle states.

levels of specifications, certificates, algorithms, concepts etc., that can be done through multiple solutions. This mapping can be done from either side [177], [178], [179], either with the functional requirements mapped directly over the technical layer, which is normally called top-down and is the preferred model within the industry, or the technical advancements converted into a product afterwards, which is usually the case for the IT sector. If we focus once again on Fig. 4, the functional requirements at the top represent long-term wishes and dreams whilst the bottom side is made up from the existing real options. Therefore, someone who strongly focuses on the top part of this figure would often fail to deliver a result that can be considered feasible, needing to delay and re-adapt their products, which is often the case in the actual Automotive sector. On the other hand, someone focusing on the bottom part would struggle to stay within budget requirements, producing lots of impractical and redundant things. Thus, the key to long-term success is to keep both perspectives in balance, maintaining both strategic and operational aspects with similar priority. This way, a balanced or hybrid top-down-bottom-up design approach will combine the requirements from both sides to quickly produce useful results and continuously adapt to the changing demands by evolving this solid operational foundation [178], [180], [181]. However, this balance is complicated to establish in the automotive industry, since the time scale for the inter-layer propagation is often delayed due to certificates, legislation validation etc. Therefore, a full revisit of the traditional automotive development methods is needed to enable them to get in line with this trend.

2) REVISITING SOFTWARE DEVELOPMENT METHODOLOGY

Classical, traditional or sequential development method- ologies (such as the V-Model [182] or the Waterfall model [183]) have been wandering around IT and industry for the last decades [182], [184], [185]. These models rely on a clear statement: all processes involved during the development of a product are phase-to-phase dependent on each other, needing to end the precedent process before starting the next one. Furthermore, these models also focus on offering

detailed documentation after each of the phases to enhance action traceability. However, as these models were designed for short end-to-end industrial projects, they have considerable flaws in long-term/complex projects in which requirements may evolve overtime. Therefore, as a first step through more adaptable development methodologies, some multi-stage sequential models appeared (such as Rapid-Application Development [186]) which tried to solve the lack of adaptability by separating the whole set of requirements into sub-modules, meaning that each cycle was smaller and easier to manage. However, even though these proposals were able to address the dynamicity problems of sequential models for medium / small projects, they added elevated planning costs and parallel development issues. Consequently, as a way to unify all multi-stage incremental models and offer a more flexible and cost-effective paradigm, a group of developers released, in 2001, the Agile Manifesto, from which most of the modern development models, such as eXtreme Programming [187], Scrum [188], or Kanban [189] have derived. Despite the variations between them, these models are nowadays dominant within most IT companies worldwide and are characterized by the same principles of flexibility, dynamicity, cost effectiveness, adaptation to constant user and client feedback and fast development. However, as these models are mostly designed for software, they don't focus on specific automotive safety and quality standards (such as ISO/IEC 15504, ISO 21434 Cybersecurity, or ISO 26262 Safety SW), and thus need to be revisited by the automotive industry.

More specifically, if we focus now on the automotive domain, as the traditional V-Model was not designed with critical safety and quality in mind, and as Automakers tended to focus on the core competencies when developing cars as a whole, needing to outsource a considerable part of the work, they set up Automotive SPICE (ASPICE) [46]. This was was started as an initiative to determine the capability of the suppliers to fit these high-end requirements and standards [47], [190]. ASPICE can simply be seen as an extension of the V-Model compliant with the previously mentioned automotive standards and certificates. However, as the ASPICE standard was defined prior to the software boom in the automotive systems, it still has an archaic vision of software dynamicity and high coupling between the hardware and software life cycles, which makes the software innovation more complicated [191]. Thus, in this context, new initiatives [46], [191] are pushing towards a merger between IT-oriented Agile models and strong automotive requirements, however there is not yet a clear standard proposal for automakers to follow.

D. EVOLUTION OF APPLICATION PROFILES AND REQUIREMENTS

If we map together now all the societal and business changes (cf. § IV-B & IV-A), the upcoming and mostly research-driven functional evolutions (cf. § II), as well as those already deployed in vehicles (cf. § III), we are able to propose a simple application profile classification using four distinct categories: (1) Static Driving Real-Time Services (SDRTS), (2) Collaborative (V2V/V2X) Services, (3) HMI/ Infotainment/Data Collection Services, and (4) Remote services (V2C). Table 1 shows a detailed description of each of the categories based on their network-related, software-related and hardware-related requirements and characteristics.

1) Static Driving Real-Time Services (SDRTS): This service category was originally made up of a group of small signal-based and safety-critical applications (highly prone to jitters and latency) which were used to handle everything related to the driving of the vehicle. However, with time, these services have been added on top of higher-level complex services that complement the easy control tasks with higher driving enhancement mechanisms (ADAS). These services are normally statically orchestrated since they are needed for the vehicle to work properly and safely, which means they are running most of the time.
2) Collaborative (V2V / V2X) Services: This category consists of highly dynamic collaborative services computed simultaneously in multiple vehicles and infrastructures depending on the environment. These services, coming mainly from the newly shared driving use cases, need high system dynamicity and flexibility since they may need to be put into motion at any moment depending on the vehicle-fleet status. Among these services, we can also find those that are not directly related to driving, which have more relaxed constraints, such as vehicle infrastructure log collection or global traffic optimization algorithms.
3) HMI / Infotainment / Data Collection Services: This third category comprises the most varied set of services and, as you can see in Table 1, we have split them into 3 different subcategories: (i) those focusing on passenger comfort, which are normally low-complex functions such as winding the car window up or down, changing the position of the mirrors…(ii) those focusing on the Infotainment, and therefore those that are closer to the classic IT services, needing normally high networking and computing capabilities and (iii) those related to data collection, either with business, technical or legislation purposes. This last sub-category is similar to the one in the *Remote Services* category but keeps the data stored locally instead of in the cloud.
4) Remote services (V2C): This last kind is composed of all the services that have a direct interaction with the cloud, whether it's an offloaded cloud function, in which case the on-board local function will not need many resources, or if it's a remotely triggered function, such as tele-diagnosis, vehicle configuration…

Therefore, if we now combine the requirements expressed in Table 1 with the aforementioned research evolution

TABLE 1. Automotive software application profiles detailed characteristics.

	Communication classification criteria							
Profiles	Deadline	Reactivity	Data complexity	Data Size	Communication Pattern	QoS level	Protocol	Actors
Static Driving Real-Time Services	Hard	Few μs to ms	-	-	Periodic	Exactly once (2)	-	-
→ Low complex services (ie. sensors, actuators...)			Low (simple values)	Low			Signal-based	Inter-vehicle
→ High complexity services (ie. ADAS, image recon...)			Medium / High	Low / Medium			IP-based	Inter-vehicle (might use cloud support)
Collaborative Services (V2X / V2V)	Hard	-	-	-	-	At least once (1)	IP-based	Intra-vehicles and infrastructure
→ Real-time dynamic collaborative services (ie. shared driving)		1-10ms	Low / Medium	Low	Periodic (changes dynamically)			
→ Not time-constrained services (ie. logging, traffic optimization...)		10-250ms	Low / High	Medium / High	Event-driven			
HMI / Infotainment / Logging Services	Soft / Medium	100 - 250 ms	-	-	-	At most once (0) for streaming and Exactly once (2) the rest	-	-
→ In-vehicle comfort services			Low	Data Size	Sporadic		Signal-based	Inter-vehicle
→ Infotainment / Navigation services			High	High	Sporadic / Periodic		IP-based	Vehicle and Cloud
→ Logging services			Medium / High	Medium / High	Periodic			
Remote Services (V2C)	Soft	Many seconds	Medium	Medium / High	Sporadic	At least once (1) / Exactly once (2)	IP-based	-
→ Cloud offloaded services (ie. inter-software dependency calculation...)								Vehicle to cloud
→ Cloud triggered services (ie. telediagnosis...)								Cloud to vehicle

	Hardware classification criteria						Software life-cycle classification criteria			
Profiles	Hardware coupling	CPU need	RAM need	ROM need	Sensor plug-ins	Software Complexity and Size	Quantity	Dynamicity	Launch time	Delayed launching
Static Driving Real-Time Services	-	-	-	-	-	-	-	-	≤ 100ms	No
→ Low complex services	High	Low	Low	Low / None	Yes	Low	Several	Static		
→ High complexity services	Medium / High	High	High	Low (cache)	Maybe	High	A Score (∼ 20)	OTA-dynamic		
Collaborative Services (V2X / V2V)	Low	Low / Medium	Low / Medium	Low	-	Low	-	-	-	-
→ Real-time dynamic collaborative services					Possible		Many	Highly dynamic	≤ 20-100ms	No
→ Not time-constrained services					No		Few	Dynamic	≤ 500ms	Possible
HMI / Infotainment / Logging Services	-	-	-	-	-	-	-	-	-	-
→ In-vehicle comfort services	High	Low	Low	Low	Yes	Low	Many	Static	≤ 300ms	Possible
→ Infotainment / Navigation services	Medium (screen needs)	High (unless GPU)	High (unless GPU)	High	Not often	High	A Score (∼ 20)	OTA-dynamic	≤ 300ms	Possible
→ Logging services	Low	Medium	Medium	High	No	Medium	Few	Static	≤ 300ms	No
Remote Services (V2C)	Low	-	-	-	-	Low	Few	-	-	-
→ Cloud offloaded services		Medium	Low	Low	No			Dynamic	≤ 300ms	Possible
→ Cloud triggered services		Medium / High	Medium	Low / High	Possible			OTA-dynamic	≤ 200ms	Possible

perspectives (cf. § II), we can hypothesise about future technical requirements and any blocking points associated. The incessant integration of V2X functions raises questions about a higher connectivity of the vehicle with its surroundings and, at the same time, the need for software flexibility and dynamicity to be able to efficiently orchestrate highly dynamic services whenever they are needed, without over-consuming resources along the way. On the other hand, the evolution of the ADAS mechanisms, being suddenly based on complex neuronal networks and combining services from different domains, confirm the need for higher computing processors and more powerful onboard networking. Nonetheless, as more and more ADAS safety critical functions appear, so does the need for inter-service isolation, strong control services and standardised network middle-wares. If we look now at the last category, the software development & architecture paradigms, it raises, once again, the need for standardisation and flexibility, all the while respecting the safety and security needs of the vehicle.

E. TECHNICAL CAUSES OF EVOLUTION

Now, we have exposed the diverse needs coming from new application advancements, societal changes, business opportunities and design & development process evolutions, we can oversee the technical conflicts needing urgent evolution within the current on-board systems. To begin, we need to remind you that embedded ICT architecture is often classified in two layers: hardware architecture and software architecture. While the hardware architecture is composed of all the Electrical/Electronic (E/E) resources deployed in the vehicle

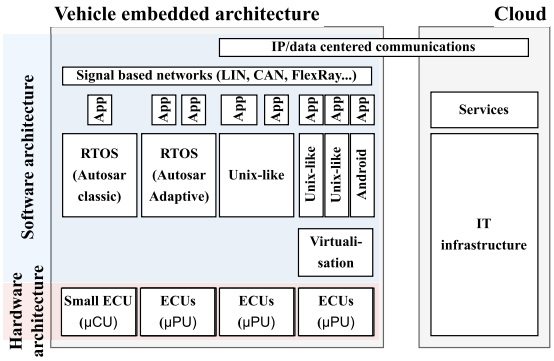


FIGURE 5. Simple overview of the current Vehicle embedded & Cloud arch.

i.e., ECUs, Power Sources, Communication supports…-, the software architecture includes all the high-level factors -such as business strategy, human dynamics, design, development cycle…- and all software specific details - such as its granularity, interactions, control services, execution environment or system security. Fig. 5 shows a high-level simplified vision of each of these layers in current embedded vehicle architecture, and how they interact with the cloud support layer.

1) HARDWARE ARCHITECTURE EVOLUTION CONFLICTS

With the raising levels of vehicle automation cars have progressively needed more computational power, which has ended up with a massive increase in the number of ECUs, which leads to unmanageable system complexity and is a struggle to keep growing at the same pace the application demands. Current cars are equipped with a mix 70 to 100 restrained ECUs, either Microcontrollers (MCUs) or Microprocessors (MPUs), connected through a highly heterogeneous set of communication supports, both wired (CAN, LIN, FlexRay, Ethernet…) and wireless (Bluetooth, LTE-M…) [66], [192], [193], [194]. In addition to the previous problems, the heterogeneity and limited capacities of this infrastructure also present an obstacle regarding resource efficiency, space harness efficiency, and energy consumption. Thus, it seems of the utmost importance that the automotive industry reduces the number of ECUs, by merging various mixed-critical applications into multi-core SOCs and standardizing the network connections.

Moreover, the lack of E/E standards between car models [195], even within the same company, complicates the reuse and optimization of the development processes, both in terms of time, energy, and money, while also having a negative impact on the planet. Adding to this problem, we can also highlight the lack of clear long-term architectural strategy [196] and the need to integrate black-box solutions coming from a wide set of suppliers [66], which hinders the definition of homogeneous standards for all in-vehicle archs.

Finally, as a way to keep the vehicle updated through the years, and to use less energy and resources while having less of a negative impact on the environment, vehicles need to solve the current compatibility problems and integration costs linked to plugging new hardware into its original embedded systems [197], allowing them to add the new trending innovations, reinforcing their original ICT systems.

2) SOFTWARE ARCHITECTURE EVOLUTION CONFLICTS

As the trend is to evolve through a more flexible and customizable architecture, in which each driver could set-up their own vehicle preferences, use some pay-as-you-go subscriptions to core vehicle functionalities and move with their own profile from their personal vehicle to other vehicles, such as their work vehicle or even a rental vehicle, the embedded software architecture still has a lot to go through. To begin with, one of the main problems affecting this evolution is high coupling in the current vehicle ICT systems between the hardware and the software [12], [198], both for internally developed software applications and those coming from suppliers. This coupling also hinders testing requirements, as testing is required for the integration of each software repeatedly for each different hardware variant, as well as limiting the software reuse, resulting in a higher dev. cost.

In addition to that, another serious problem of the current software architecture is linked to the possibility of installing new software on user's demand in a safe and secure way [199], [200], [201], [202]. This problem derives from the aforementioned E/E system heterogeneity, as well as from a lack of standardized middleware and control services, the complexity orchestration in such environments, and the current testing infrastructure and legislation constraints. Besides, the use of virtualization, which may relax this integration stress, is under-exploited, being used exclusively for security and business segregation concerns. Another stumbling block for the current software architecture evolution is the integration of the legacy software components into this new deployment phase, which is critical to ensure an affordable transition from the current to the new generation ICT systems. Finally, from a network point of view, current architecture needs to find a way to standardize the communication for all software blocks, enabling it to guarantee Quality-of-Service (QoS) and safety deadlines despite the software dynamicity [21], [203], all the while remaining focused on the pricing gap between network cables.

Furthermore, we cannot forget the interaction with the new support infrastructures such as the V2X [204], [205] or the V2C [206], [207], [208], [209], either in Classic, Fog or Edge clouds, which can also be used to run in-vehicle functions, acting as a support, while also being a more resource efficient manner to run the necessary soft. However, vehicles need to evolve their connectivity capabilities to be able to match strict deadlines as well as dynamically integrate the available external resources collaboratively into their orchestrator.

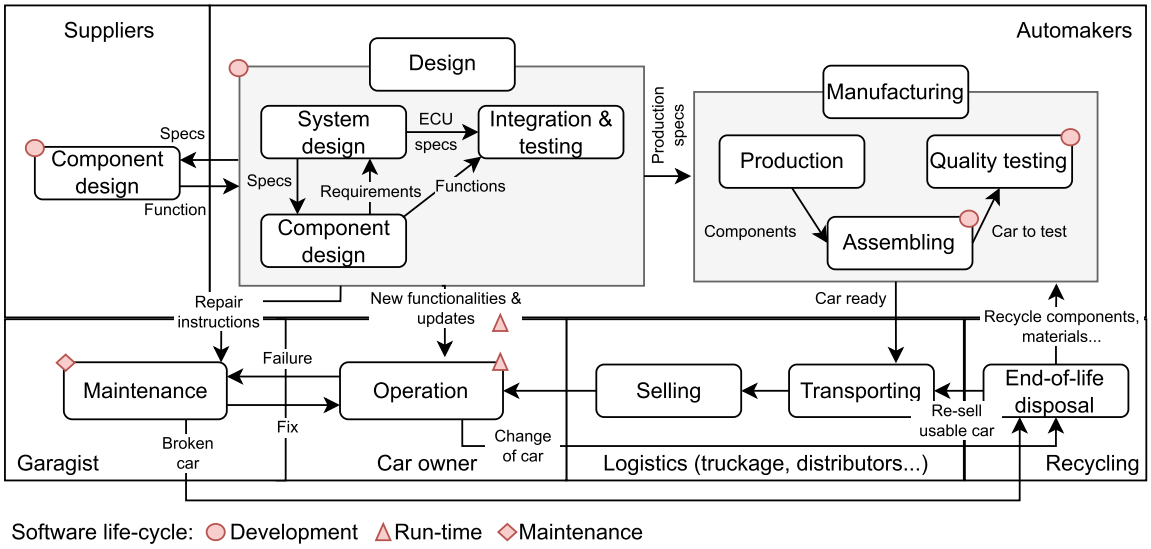


FIGURE 6. Convergence between vehicle product and software life-cycles.

V. AUTOMOTIVE APPLICATION LIFE-CYCLE: CONVERGENCE AND OPEN ISSUES

Due to the rapid integration of software into modern automobiles, the convergence of automotive product and software life cycles has become increasingly important in recent years. As a result, the traditional linear approach to product development has been changed to integrate both hardware and software processes. This section details the result of this convergence between the automotive product and software life cycles, including the challenges that arise from said convergence. We will also examine the open issues that are yet to be addressed in order to achieve a successful integration of hardware and software in the automotive industry.

A. CONVERGENCE OF AUTOMOTIVE PRODUCT AND SOFTWARE LIFE-CYCLES

In this subsection, we will delve into the convergence of the automotive product and software life cycles. We will examine how the traditional product development process is being transformed by the incorporation of software and how new methodologies are being developed to integrate hardware and software. Furthermore, in Fig. 6, we propose a merger between these two life cycles, with the objective of completing the previous use-case scenario (cf. § III).

Thus, if we focus first on surveying the vehicle life cycle from a product point of view, we expand on the literature proposals made in [21], [78], [210], and [211] and also integrate the participants and phases linked to the distribution and end-of-life of the vehicle. Hence, as you can see in Fig. 6, the vehicle life cycle is composed of six different participants - i.e., automakers (or vehicle manufacturer), suppliers, logistics companies (transports, distributors…), car owners, maintenance and recycling plants - and represented as a finite-state machine model, where each state represents a phase in the life cycle of the vehicle - i.e. design, manufacturing, transporting, selling, operation, maintenance and end-of-life disposal. It is worth noting that for each of these states, a player is associated with and has exclusive access to the vehicle during this phase.

Moreover, in Fig. 6, you can also see a high-level mapping of the automotive software life cycle through the different phases of the product life cycle, firstly, consisting of an architecture design phase, a second software development and pipelines phase, and a final run-time management phase. However, even though this mapping makes it easier to understand how the software life cycle is set out through the manufacturing process, it is still unclear in which part the different blocks of the vehicle software (i.e., the software architecture, the software integration pipelines and the run-time management / control services) are implemented or interact with each other. Thus, to complete this section, and as we want to look at the automotive systems mainly from an IT point of view, we will structure the main part of the survey following these three sections:

- A first architecture design trend (cf. § VI) including all the changes grouping the architectural evolutions for all four; E/E architecture (cf. § VI-A), Software architecture (cf. § VI-B), Network architecture (cf. § VI-C) and Internet-of-Vehicles platform (cf. § VI-D).
- A second trend concerning all the software integration pipeline evolutions (cf. §VII) for both initial vehicle production and on-the road management. In this second phase we will detail the changes in the development (cf. § VII-A), delivery (cf. § VII-B), deployment (cf. § VII-C), testing (cf. § VII-D), and orchestration (cf. § VII-E) processes, allowing the software to be dynamically integrated into any vehicle software context.

TABLE 2. Summary of the open issues that need to be addressed through software transformation. Classification made from the aggregated data from the papers [22], [48], [65], [71], [192], [194], [198], [212], [213], [214], [215], [216], [217], [218], [219], [220], [221], [222].

	Issue	Description	Arch.	Pl.	Run-time
(Is1)	Bandwidth bottlenecks	Sudden data volume increase saturates the in-vehicle networks.	✓	∼	✓
(Is2)	Mismatching latencies	Cross-domain critical communication struggles to meet the safety latency requirements.	✓	✓	∼
(Is3)	Complex wiring layer	Wiring complexity & harness weight due to ECU increase has negative effects on maintenance, power performance and price.	✓	✗	✗
(Is4)	Heterogeneous network stack	Heterogeneity hardens the establishment of global comm. standards, which complexifies the network management at application level.	✓	✓	✗
(Is5)	Deficient ECU computing and storage capabilities	New AI, big-data, IVI... applications computing, and storage needs to overpass the in-vehicle ECU characteristics. Moreover, these needs are magnified as the quantity of applications available increases.	✓	✗	✗
(Is6)	High Software / Hardware coupling	High coupling has a negative effect on the system dynamicity, flexibility, adaptability, and scalability, as well as on the software reuse.	✓	✓	∼
(Is7)	Restricted availability	Vehicle availability is not always ensured as network coverage is not sufficient in rural areas, underground parking... Furthermore, battery and garage/industrial constraints also limit the availability of the vehicles when delivering software.	✗	✓	∼
(Is8)	Heterogeneous ECUs and legacy integration	Presence of highly heterogeneous Legacy CUs, MCUs & MPUs means that delivery and maintenance pipelines are more complex.	∼	✓	∼
(Is9)	Increasing Energy cons. & battery limitations	Increasing functionalities might hinder vehicle energy efficiency and produces an over-consuming battery, which may cause early malfunctions.	✓	✓	✗
(Is10)	Economic impact of changing tooling	Changing the in-vehicle architecture and technology stacks require substituting or modifying the tooling chain, which has an enormous economic impact.	∼	✓	✗
(Is11)	Over-consuming Testing	Being able to dynamically adapt the in-vehicle systems to the environment requires increasing the tempo during the testing phase at the installation, or at least after it has been tested once in the vehicle.	✗	✓	✗
(Is12)	Complex multi-provider integration	As more become involved and join the in-vehicle ecosystems, the integration of software will keep increasing both in complexity and cost, unless the in-vehicle architecture flexibility and adaptability steps-up.	∼	✓	∼
(Is13)	Lack of community	The lack of acceptance of external developers as well as the undeveloped sand-boxing, shadowboxing and developed simulation tooling slows the automotive innovation possibilities.	✗	∼	✗
(Is14)	Dynamic certification	The enhanced software dynamicity is a major problem for system security and, more precisely, the certifications, which, at least for now, do not address software context dynamicity.	✗	✓	✓
(Is15)	Few in-vehicle traceability	Current systems do not have sufficient levels of traceability, which hardens and slows down the resolution of unexpected bugs.	✗	✓	✓

· ✓ :*Addressed*, ∼:*Partially addressed*, ✗ :*Not addressed*, **Arch.** :*Architecture design*, **Pl.** : *Software Integration Pipelines*, **Run-time** : *Run-time management services*

- A final trend concerning all the run-time management services (cf. § VIII) that ensure the proper functioning of the systems, as well as the compliance with the different safety and security certificates and legislation. In this section and taking into account that the number of possible Control Services is too extensive, we will then focus on the mechanisms of tackling the security and safety of the in-vehicle and off-board systems and software and those centred on the data.

B. OPEN ISSUES

Prior to beginning the detailed state-of-the-art analysis, in this subsection, Table 2 summarises the open issues that need to be addressed through automotive software transformation. This table includes the issues concerning all four networking, architecture, control, and application and indicates in which of the three macro-trends of this survey (i.e., Architecture in § VI, Software Integration Pipelines in § VII and Control Services § VIII) each of them will be addressed, or partially addressed. However, note that, as there are plenty of issues, this table contain those targeting mostly the global architecture and, although other smaller or less general issues do exist, they are not mentioned in this table but presented individually in the concerned section.

VI. ARCHITECTURE DESIGN

Traditionally, in the automotive sector, all the levels of architecture design were grouped under the definition of E/E Architecture, which was seen as a convergence of electronics hardware, network communications, software applications and wiring into one integrated system. However, this definition is shallow and not precise enough for the current context, in which software and off-board architectures are steadily growing in importance. Thus, in this section and as a way to narrow this definition, and the brief one in § IV-E, we will split the architecture design into four distinct categories - i.e., Electric/Electronic Architecture, Software Architecture, Network Architecture, and Internet-of-Vehicles Platform. For each subsection, we will go through an extensive analysis of the state-of-the-art features and initiate a discussion about the evolution perspectives and threats. Moreover, for each of these categories, we have classified their sub-parts according to their main properties - i.e., processing, in-vehicle communication, energy supply and hardware modularity & reusability for the E/E Architectures, software re-use, dynamicity, flexibility, safety, security, maintainability & complexity for the software architectures, standardisation and data-management for the network architectures and cloud interaction, Vehicle-to-Vehicle, Vehicle-to-Everything interactions for the IoV platform.

A. ELECTRIC/ELECTRONIC (E/E) ARCHITECTURES

To fully address the challenges facing E/E architectures we are going to conduct a four-layered analysis starting by taking an in-depth look at the state of technology presentation of the Electronic control units (ECUs) possibilities, to address the processing capabilities of the system. This is then followed by another analysis of network cables, targeting, this time, the in-vehicle communication properties. Then, we will focus on the external energy, delving deeper into the power supply infrastructures and, finally, we will focus on the hardware modularity and reusability by giving an extensive analysis of the automotive hardware reference architectures and their evolutions. As you will see all through this section it is worth noting that E/E evolutions are especially limited with regard to security & safety certificates (such as ISO 26262 [223] or ISO 21434 [224]).

1) PROCESSING PROPERTIES: ECU EVOLUTION

Given the previously presented context, automakers are forced to evolve their traditional ECUs so that they can deliver more performance, functionality, and flexibility, all the while reducing component cost and size, lowering power consumption and footprint, as well as meeting the stringent requirements regarding reliability, robustness etc. All of this must be done despite the punishing operating conditions as described in the standards, legislation, and certificates. In addition, they are required to operate correctly, without needing to be changed until they have functioned under high material stress conditions (i.e., rain, with elevated temperatures…) over many years. In order to accurately choose the new set of ECUs for their new generation on-board systems, automakers have a wide range of choices to choose from depending on the target functionalities that the systems need to provide.

Firstly, and the most cost-effective and simplest option, we have micro-controllers (MCUs). Micro-controllers typically use on-chip highly constrained flash memories in which its main program is stored and executed. This guarantees higher energy efficiency as well as a shorter start-up period [232], [233], [234]. Most MCUs available on the market have few Mega Bytes of Program memory, which is undoubtedly a limiting factor when trying to operate with more complex applications. However, these kinds of boards are interesting for low-functional use-cases that need low energy consumption and a fast/resilient start, such as the applications within §IV-D - *Static Driving Real-time Services > Low complex services*. Some examples from this category that are dominating the automotive market are the Infineon Aurix [235], [236], the Renesas RH850 [237], [238] or the NXP S32K [239].

On the other hand, if we now turn to more complex and high-end boards, we have microprocessors (MPUs), which don't have the same memory constraints as the MCUs. They separate non-volatile memory such as NAND or serial flash from volatile memory, traditionally DRAM. In the former they store programs and data, whereas in the latter the start-up files are loaded [232], [233], [234]. These kinds of nodes are then, given their higher complexity and number of interfaces, slower to start and have a higher energy consumption, however they do allow significantly more complex functions to run. These MPUs are often complemented with other extra chip-sets (as is also done less often for MCUs), higher computing, network or graphic capabilities becoming Systems-on-Chip (SoCs), or clustered together with other MPUs and MCUs from Chiplets. These boards will then become an interesting choice for both general purpose functions and functions with specific needs, even being able to substitute MCUs in most cases, thanks to tools such as virtualisation. Further in-depth analysis will be carried out on this topic in this paper in §7.3. If we focus on different features, some widespread examples would be the NVIDIA Jetson AGX Xavier [240] for High-Performance Computing (HPCs), Qualcomm Gen2/3/4 [241] for Graphical Processing Units (GPUs) or Mobileye 6-Series [242] for AI-based functions relying on Neuronal Network Processing (NPUs).

Thus, summing up, there is a wide variety of potential ECUs available from different suppliers, already compliant with legislation and certificates, and even in academic research [243], [244], [245], that could be used for the new software enhanced automotive systems. However, their specifications must be carefully chosen considering the trade-off between application/system needs and system monetary and energetic efficiency. It is worth noting that there are other factors that we have not yet discussed and that need to be considered when choosing an ECU, such as the Instruction Set Architecture (ISA) supported (i.e., ARM, RISC-V or CISC-based) [246], [247], [248], the price gaps between different suppliers, production availability [249], etc.

2) IN-VEHICLE COMMUNICATION: NETWORK CABLES EVOLUTION

As the capacities of the in-vehicle ECUs evolve to satisfy the application needs, more data will get produced within the in-vehicle systems for both inter-service collaboration and interactions with the off-board architectures (such as Cloud, infrastructure, or other vehicles - cf. § VI-D). Therefore, in this context, the network architecture and stack (cf. § VI-C), need to be revisited in order to extend the bandwidth requirements, match the application latency constraints, and establish new standards, all the while once again keeping both the costs and energy consumption down. Note that, Table 3 summarises the technology panel for network cables.

First, with regard to the network bandwidth bottleneck issues, it seems clear that Ethernet offers considerably higher options, being able to offer data-rates of multiple GBit/s in comparison with the 20 MBit/s limit of CAN XL, which is the most performant solution available for signal-based networks (i.e., CAN, LIN or FlexRay). Moreover, Ethernet has a bigger community and a wider choice panel within IP higher layer

TABLE 3. Network cable standard detailed comparison. Classification made from the aggregated data from the papers [225], [226], [227], [228], [229], [230], [231].

Tech. stack		Max Data-rate	Nodes	Bus	Standards	TCP/IP comp. standardised	TCP/IP comp. research	PoDL standardised	PoDL research	Wake up capability	Deployed in cars
CAN	LS	125 kBit/s	[2..32]	✓	ISO 11898 CiA 610-1	✗	✓	✗	✓	✓	Yes
	HS	1 MBit/s *cur. 500kBit/s*	[2..110] *cur. 16HS - 24FD*								
	FD	12 MBit/s *cur. 5 MBit/s*									
	XL	20 MBit/s *cur. 10 MBit/s*				✓					No
LIN		20kBit/s	[2..16]	✓	ISO 17987	✗	✗	✗	✗	✓	Yes
FlexRay		10MBit/s	[2..22]	✓	ISO 17458	Yes	✗	✗	✗	✓	Yes
ETH	100base-T1	100 MBit/s	[2]	✗	IEEE 802.3bw	✓	✓	✓ IEEE 802.3bu	✓	✓ OPEN TC10	Yes
	100base-Tx				IEEE 802.3u						Yes
	1000base-T1	1000 MBit/s			IEEE 802.3bp						Yes
	2.5Gbase-T1	2.5 GBit/s			IEEE 802.3ch						∼ (soon)
	5Gbase-T1	5 GBit/s									No
	10Gbase-T1	10 GBit/s									No
	10Base-T1S	10 MBit/s	[2..16] *cur. [2..8]*	✓	IEEE 802.3cg			∼		✓OPEN TC17	∼ (soon)

· ✗ :*Not applied*, ✓ :*Applied*, ∼ :*Partially or similarly applied*, cur.: *Currently* and comp.:*compliance*. For all tables the color-shades have the purpose of highlight the best-fit for each property.
· *LS* :*Low Speed*, *FD* :*Flexible Data-rate*, *HS* :*High Speed*, *XL* :Extra Large and *PoDL* :*Power over Data Lines*

protocols, which means it is a better fit at an application level, especially when managing complex data-structures. However, in order to profit from these high speeds, Ethernet needs a network switch, which significantly increases the cost of the system, which is already higher for Ethernet cables than for signal-based cables, and energy consumption [228]. If enough Ethernet ports are available in ECUs, then it may be able to host the Ethernet switch, however, this solution will be less performant than a dedicated switch.

Secondly, if we focus now on the impact on the processing workloads of both Ethernet and CAN [225], which is the most widespread signal-based network, studies show that even though Eth. communication requires significantly more processor than CAN communication for the interface initialisation, it is considerably more efficient at transmitting, receiving and unpacking messages. Thus, Ethernet has a higher initial cost, which impacts the time within which an ECU is considered fully available, but once the interface initialisation is done, the system will struggle less to match the application latency constraints. Note that both technologies are more than capable of communicating within an acceptable safety range of few μs for small/medium sized messages. However, only Eth. maintains an acceptable level of efficiency for long messages. Furthermore, even though both technologies are appropriate for resilient applications, Ethernet guarantees a 10-fold lower error occurrence than CAN. Moreover, if we look further now into the energy consumption issues, CAN comm. shows clear advantages with almost 50% lower power consumption than Ethernet [225]. However, this energy efficiency gap is less significant as the package size increases, as the Ethernet interface can enter idle or sleep mode for a longer time, or if we use mechanisms such as PoDL [227], [229] whose efficiency and maturity is higher for Ethernet than for CAN.

Finally, and as the transition from the legacy to the new generation systems will happen progressively, it is important to survey the techniques that will help to simplify this transition. Thus, as CAN is the biggest representative of the signal-based networks, it's worth mentioning some papers that present how to bring CAN closer to the IP world. Among these papers, and as you can see in Table 3, some proposals target encapsulating the IP package over the CAN frame [231], for best-effort traffic use cases, or adding bridges to make the encapsulation from CAN to Ethernet [250]. Finally, it is also interesting to take under consideration [251] and [252], which study the integration of CAN networks with Time-Sensitive Networking (TSN) and Ethernet for real-time scheduling.

3) EXTERNAL ENERGY: POWER SUPPLY PLATFORMS EVOLUTION

With the recent advancements in Electric & Hybrid Vehicles, most automakers now have their own modular global power supply platforms to abstract the energetic aspects from the E/E design process itself. This way, E/E designers can make use of the electric platform simply on a dynamic power-on-demand basis to keep the power supply controlled, standardised, and decoupled from the rest of the E/E choices. These modular platforms are usually made up from the chassis of the vehicle including the battery, motor, and power electric system, and are standard and can be reused independently from the vehicle model. Examples of these platforms would be Hyundai Electric Global Modular Platform (E-GMP) [253], Volkswagen Group Premium Platform Electric (PPE) [254], [255], STELLANTIS STLA SMALL/MED/LARGE/FRAME [256], Toyota New Global Architecture (e-TNGA) [257], Volkswagen Modular Electric-drive Toolkit (MEB) [258] or Renault-Nissan Common Module Family (CMF) [258]. Furthermore, some suppliers, such as Foxconn, with their MIH initiative [8], also try to provide solutions for these kinds of power platforms.

Note that as these platforms are at an early stage, they act in such a way that they furnish energy to the ECUs, however, this is expected to evolve in the near future, more in line with a power-on-demand basis. Moreover, as more and more power-hungry functions, ECUs and wires are used within cars, designing new techniques to reduce the energetic impact on the on-board IT systems seems crucial. Considering their negative impact on vehicle autonomy and, with it, some techniques such as ECU degraded operation mode [259], [260], partial network wake-up [261], [262], [263] or highly dynamic software contexts (cf. § VII) will become more important in the coming following years.

4) HARDWARE MODULARITY AND RE-USABILITY: REFERENCE ARCHITEC- TURE EVOLUTION

As we said in §IV-E, as more and more functionalities were added to the car, the in-vehicle systems needed more computational power, which then signified a massive increase in the number of embedded ECUs. This has led to unmanageable system complexity, making it complicated to keep growing at the same pace as application demands with regard to computing, network or energy consumption [16], [17], [264], [265]. Today, most of the automakers' E/E architectures are already in compliance with the E/E Domain Vehicle reference architecture. However, there are still over a hundred low functional MCUs, connected through a highly limited and heterogeneous set of CAN, LIN, FlexRay and, to a lesser degree, Ethernet. If we look further into the progression perspectives set out in the previous state-of-technology analysis and the research trends with regard to upcoming in-vehicle architectures (i.e. Zonal Architectures, also known as ZoA), presented in Fig. 7, there exists a clear trend. They move through more centralised architectures, with fewer high-end function target ECUs both for general purposes and specific uses, - i.e., NPUs, GPUs, HPUs etc. interconnected, at least in the network backbone, through Ethernet to reduce latencies and enhance system bandwidth.

Thus, if we delve further into detail, the now low-functional ECUs would be grouped into higher-end SoCs depending on the functions that they will be handling (e.g., a GPU for the ECU handling the on-board screen, an HPC for the ECU in which ADAS will more than likely be deployed…). These SoCs can then be separated into smaller, more dynamic, virtual ECUs thanks to techniques such as virtualisation (cf. § VII-C), offering more optimal communication management interfaces, multi-threading, fault tolerance tooling, and, most importantly, significantly reducing system complexity and coupling. On the other hand, as far as the cables interconnecting the ECUs are concerned, migrating from signal-based networks, (i.e., CAN, LIN and FlexRay) to data-centred networks (i.e., Ethernet) [22], shall relax the bandwidth and latency stress, allowing for the establishment of homogeneous communication interfaces and standards, reducing once again system complexity [17], [216], [266]. Finally, with regard to the cost & energy efficiency, by drastically decreasing the quantity of ECUs and cables and not having to maintain all the architecture powered on a constant basis, we should reduce the costs/energy consumption or, at least, compensate for the increase in new ECU and wire evolutions.

Furthermore, as the vehicle becomes progressively more integrated with smart-city and cloud infrastructure (cf. § VI-D), some studies [267], [268], [269] suggest that a combination of resources including cloud, infrastructures and vehicles operating together as a hive (i.e. Seamless Architecture) should be the next step in the evolution of Automotive E/E architectures.

B. SOFTWARE ARCHITECTURES

Now that we have gone over the evolution perspectives of the E/E architecture, the software architecture beneath needs to be revisited so as to benefit from these new system capacities in the most optimal way possible. Thus, in this section, we will begin focusing on the main reference architectures coming from an IT point of view, focusing on properties such as software reuse, dynamicity, and flexibility. Subsequently, we will go over the automotive software architectures and proposals, trying to highlight the safety and security aspects that are considered to be vital requirements for these systems. Finally, we will discuss the convergence of both, trying to evaluate how we can combine the aforementioned IT-related properties with automotive needs and how does it impact on the maintainability and complexity of the system.

1) IT SOFTWARE REFERENCE ARCHITECTURES: ZOOM ON RE-USE, DYNAMICITY AND FLEXIBILITY

If we begin with Software Product Line (SPL) [270], [271], which was, to the best of our knowledge, the first modern reference architecture targeted to simplify and standardise the software development and deployment process, we have to highlight that this reference architecture was inspired by the classic industrial production lines and individual products, that interact between themselves through a trade-off of the base concept around which these architectures rotate: the variability, which is used to classify each of the features detailing an application as a commonality or a variation. Fig. 8 depicts the structure of SPLs, which consists of three phases: (1) Scoping, in which the company determines what to develop (i.e., which products will be part of the product line and what are their commonalities and variants). (2) Domain engineering, in which the objective is to design the underlying system architecture by characterising the inter-application reusability and pre-scheduling their data exchanges. Furthermore, in this phase, the system will also produce different test-cases, scenarios, documentation, and specifications for each individual product. Finally, (3) Application engineering, in which the output of the two precedent phases will be analysed, the inter-application conflicts resolved, and the development of the final architecture and applications will be done through the creation of reusable templates that will be

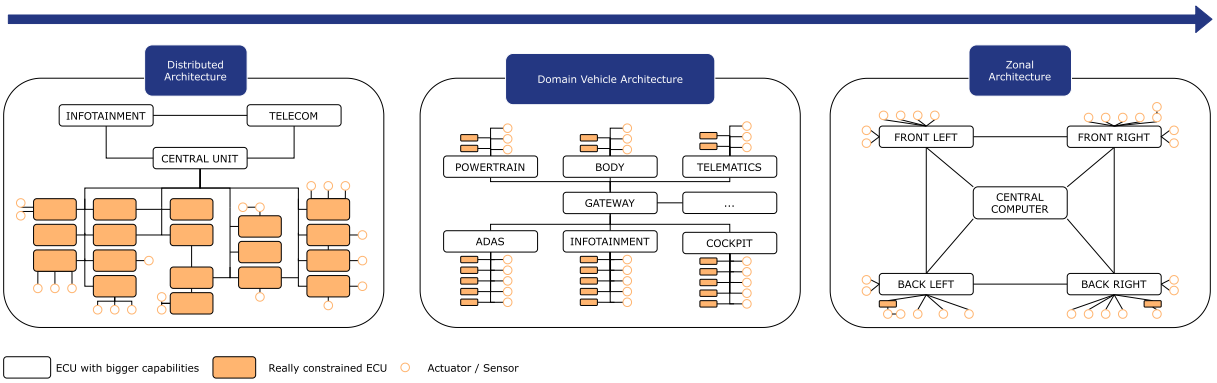
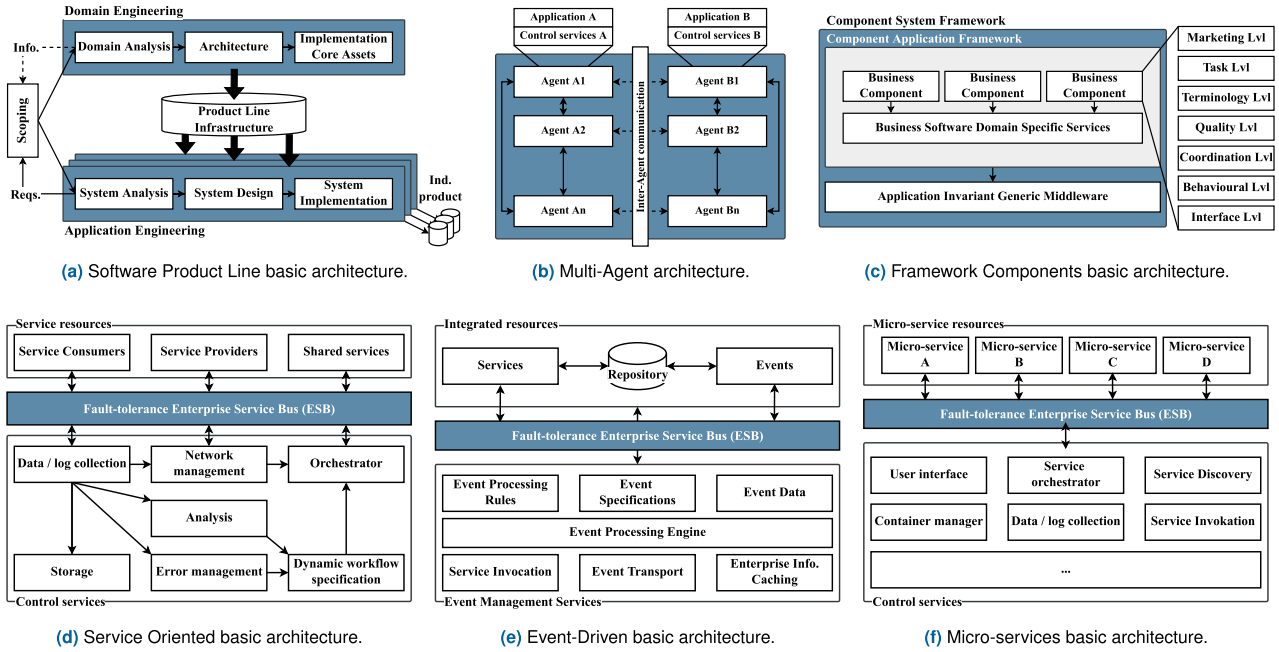


FIGURE 7. On-going transitions for automotive E/E architectures.



(a) Software Product Line basic architecture. (b) Multi-Agent architecture. (c) Framework Components basic architecture.

(d) Service Oriented basic architecture. (e) Event-Driven basic architecture. (f) Micro-services basic architecture.

FIGURE 8. Theoretical software reference architecture schematic figures.

further tested and integrated into the architecture, trying to maintain them so that they are reusable for future iterations or projects.

Secondly, Framework Components Arch. (FCA) [270], [271], relies on the definition of the framework itself, which is basically a tool or a set of tools that provide ready-made components or solutions that can be easily used and customised in order to speed up the development and integration of new software applications. Therefore, the FCA ref. arch. proposes a sort of configurable black box, that already provides all the classes that contain application logic, which can be directly initiated or parameterised to adapt themselves to the needs of the application. However, in addition to these default parameterization possibilities, FCA frameworks ensure their extensibility by defining a clear set of interfaces to add components that were not initially planned. Thus, as we can see in Fig. 8c, this ref. arch. is composed of a component system framework that provides application invariant generic services close to the middleware layer, and a component application framework which provides highly customizable interfaces for the developers to implement their business-specific software over it, hence, enhancing the intelligence of the architecture and its middle-wares. To sum up, FCA paradigm offers a large set of supporting functions that allow the user to only implement the business applications required and forget about the rest of the system control mechanisms, which are developed once and are afterwards common to all the applications.

As the obvious evolution of the previously mentioned FCA, the Service-Oriented Architectures (SOA) [272], [273] try to solve the aforementioned architectural problems in a similar manner: by enhancing the intelligence of the

infrastructure and offering a common management layer for all the different applications running over it. These architectures base themselves on the existence of a fault tolerance service bus or middleware used by all the different services for either simple data passing or inter-service requesting. This will act as an abstraction layer between the services and the physical position of their instances, enhancing the flexibility and, through simple replication, the fault tolerance of the system [274], [275]. Furthermore, and as you can see in Fig. 8c, a typical SOA will be made up of (1) service providers, those who generate and share the data in the system, (2) service consumers, also called service requester sometimes, who are those using this data to operate, (3) a fault-tolerant common comm. bus or middleware, (4) a set of control services running constantly and trying to exploit, coordinate and ensure that both the services and the infrastructure themselves are functioning well, dynamically adapting to system constraint changes over time [18], [276]. Finally, this paradigm also focuses on the importance of reusing and making the services as modular and inter-operable as possible, while always establishing an elevated level of standards and pipelines to be respected by the developers.

In the same scope, Event-driven Architectures (EDA), as you can see in the Fig. 8e, are a variant of the previously presented FCA. However, unlike SOA, which focuses on the services and their behaviour and interactions, the EDA reference architecture will focus on ensuring the real-time reactivity, implementing for it an asynchronous communication bus instead of the classic synchronous bus of SOA [277], [278]. This way, EDA would increase even more the flexibility and agility of SOA, relaxing further the software integration process.

For the final variant we will review the FCA, Micro-services Architecture (MSA) [279], [280], [281] which is built over a collection of smaller, normally mono-functional, independent service units with well-defined interfaces. The goal of this ref. arch. is to exploit to the maximum the reusability and decoupling of software chunks between themselves, allowing for the possibility to to independently deploy and scale the different micro-services dynamically depending on the demands of the rest of the services [282], [283]. This flexibility when developing new services, together with the continuous growing collection of the already well-defined and deployed micro-services in the infrastructure, help developers to work faster and reduce the integration time, which then enables them to home in on the specific business needs.

On the other hand, Multi-Agent Architectures (MAA) this time try to solve the aforementioned problems by enhancing the intelligence of each software application instead of the infrastructure itself. MAA synthesizes contributions from different areas, including artificial intelligence, software engineering and distributed computing to address developing systems that are composed by many dynamically interacting components [270], each of them having their own distributed control mechanisms to coordinate each other. Between these mechanisms we can account for some automotive critical

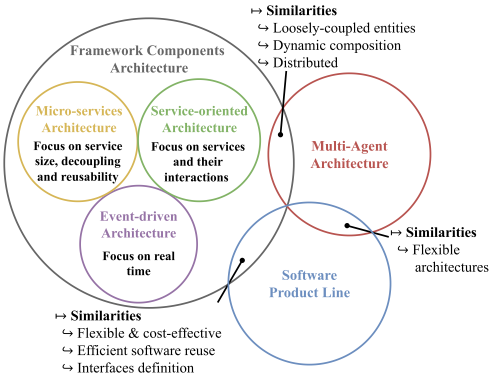


FIGURE 9. Outlook of the studied software architecture paradigms.

mechanisms such as neighbour discovery, fault tolerance, network abstraction or consensus [284]. In this paradigm, and as you can see in Fig. 8b, this architecture model relies on the concept of agents, which are pieces of software able to coordinate themselves in sub-sets so as to establish the necessary control mechanisms without the need for a centralised service to intervene. However, this solution, even if it is a better fit in some cases, implies a redevelopment of the control services each time for each individual piece of software, which is not ideal for a highly variant and participative environment such as vehicle embedded architecture.

To conclude, having already presented in detail all the main reference software architectures, we will compare them and comment on the interesting points of each one for future on-board architectures. Moreover, we will lean on Fig. 9 which visually depicts the similarities and differences between the different paradigms, and Table 4, which proposes a multi-criteria comparison of the reference architectures with regard to all four communication, software, architecture and business centred characteristics. On the one hand, if we isolate all three MAA, SPL and FCA, we can highlight that by making the control services exclusive to each application, as it is for MAA, we make the application development, maintenance, and integration considerably more difficult, having more coupling and less flexibility than SPL and FCA. However, this paradigm is interesting as ensuring the autonomy between the infrastructure and applications can be fascinating in some situations, as is ensuring the fault-tolerance of the control services layer in the FCA variants. In addition, we can also highlight that by the SPL focusing on optimising the software production through templates has a positive effect on decreasing software complexity and granularity and enhancing software reuse but won't be enough to address the complete set of previously exposed issues without adding some extra control layers or mixing it with other reference architectures. On the other hand, all three of SOA, EDA and MSA match the future architecture levels of abstraction, security, and flexibility with slight differences in their focus. Thus, SOA paradigms will propose a higher tolerant and reliable call-driven environment, while MSA focuses less on

TABLE 4. Exhaustive comparison of the aforementioned software architecture paradigms.

Properties		Framework Components Architecture				
		Service-Oriented Architecture	Event-driven Architecture	Micro-services Architecture	Multi-agent Architecture	Software Product Line
Communication-centered characteristics	Comm. management	Common middleware	Common middleware	Common middleware	Application custom	-
	Comm. paradigm	Synchronous (Req/Res)	Asynchronous (Pub/Sub)	Synchronous (Req/Res)	Synchronous (Req/Res)	∼ Depends on app
	Inter-service comm. & discovery	Dynamically guaranteed	Dynamically guaranteed	Dynamically guaranteed	Rare	∼ Depends on app
Software-centred characteristics	Reuse	Opportunistic	Opportunistic	Systematic	Low	Systematic
	Hardware / Software Coupling	Decoupled	Decoupled	Highly decoupled	Coupled	-
	Autonomy	∼ Depends on middleware	∼ Depends on middleware	∼ Depends on middleware	∼ Depends on the other agents	∼ Depends on app
	Granularity	Medium	Medium	Small / mono-functional	Big	Medium
	Development complexity	Low	Low	Low	High	Low
	Software complexity	Multi-threaded	Multi-threaded	Mono-threaded	Multi-threaded	Multi-threaded
	Modification complexity	Medium	Medium	Simple	Hard	∼ Depends on the quantity of templates
Architecture / system characteristics	Flexibility	High	High	Very High	Low	∼ Depends on the quantity of templates
	Dynamicity	Yes, on request	Yes, on real-time	Yes, on request	∼ Depends on app	-
	Architecture Complexity	Medium	High	High	Low	Low
	Fault-tolerance	Yes	Yes	Yes	∼ Depends on app	∼ Depends on app
	Availability	Ensured by the control services	Ensured by the control services	Ensured by the control services	∼ Depends on app	∼ Depends on app
	Deployment complexity	Low	Medium	Very Low	Medium	Medium
	Qty of services	Dozens	Dozens	Hundreds	Couple	Dozens
	Scalability	High	High	Very High	Low	∼ Depends on app
	Scheduling intelligence	Optimal	Optimal	Optimal	No, topology unknown	-
	Focus	Service interactions	Real-time	Service granularity and reuse	Coordinate without control services	Simplify software production
Business characteristics	Time-to-market	Low	Low	Very low	Medium	Low
	Integration cost	Low	Medium	Very Low	Medium	-
	Maintenance cost	Low	Low	Low	Medium	-
	Governance	Standardized collaboration	Standardized collaboration	Relaxed collaboration	Centralized	Centralized

these aspects but offers higher flexibility and collaboration. By reducing the granularity of the systems, and EDA focuses on implementing these control services in an asynchronous and more reactive manner. Furthermore, we can then imagine that future automotive software reference architecture could benefit from many of these paradigms. Using SPL to reduce development complexity and establish standards, SOA to manage all safety, security, scheduling, and criticality that is needed by embedded vehicle architectures, and even EDA or Micro-services to reduce the complexity of the applications or ensure network reactivity of time-constrained services.

2) AUTOMOTIVE SOFTWARE ARCHITECTURES: SAFETY AND SECURITY

In this subsection, we will be taking an in-depth look at the automotive contributions around software arch., with SOA being the one that we focus on the most. Note that Table 5 summarises the main contributions around Software Architectures in the automotive sector, breaking each of them down into sub-domain categories depending on all the subjects addressed in the paper.

If we take a detailed look at this table, we can observe that, at least for now, there are few contributions concerning full

TABLE 5. Classification of the most interesting automotive SOA (complete or partial) propositions.

Solution	Hardware characteristics	Communication characteristics	System safety	System security	Software characteristics	Control services	Metrics & use cases	Economic - political interests
M. Rumez *et al.* [19]	✓	✓	∼	Network only	✗	∼	∼	✗
P. Obergfell *et al.* [289]	✓	∼	∼	∼	✓	∼	✗	✗
M. Mody *et al.* [16]	✓	✓	∼	∼	✗	∼	✓	✗
G. L. Gopu *et al.* [20]	∼	✓	✗	✓	∼	∼	✗	✗
D. Stabili *et al.* [21]	∼	✓	∼	✓	∼	∼	✗	✗
M. Haeberle *et al.* [205]	∼	✓	✓	∼	∼	✗	✓	✗
R. Rotermund *et al.* [290]	✗	✓	✓	∼	✗	✗	✓	✗
M. Çakır *et al.* [18]	✗	✓	∼	Network only	✗	∼	✓	✗
L. Baresi *et al.* [291]	✗	✓	✗	✗	✗	∼	✗	✗
F. Oszwald *et al.* [292]	✗	✓	∼	∼	∼	∼	✗	∼
M. Bellanger *et al.* [13]	✗	✓	∼	∼	∼	∼	∼	∼
T. Hackel *et al.* [293]	✗	✓	✓	✗	∼	✗	✓	✗
O. S. Al-Heety *et al.* [23]	✗	✓	∼	✓	✓	∼	∼	∼
D. E. Sarmiento *et al.* [24]	∼	✓	✓	∼	✓	∼	∼	∼
Q. Y. Zhang *et al.* [25]	✗	✓	✓	∼	✓	∼	✗	✗
T. K. Kuppusamy *et al.* [295]	✗	✓	∼	✓	✓	∼	∼	∼
G. Falco *et al.* [296]	✗	✓	∼	✓	✓	∼	∼	∼
A. Kampmann *et al.* [297]	✗	∼	∼	∼	✓	∼	✗	✗
K. Mansour *et al.* [298]	✗	∼	✓	✓	✓	∼	✗	✗
M. Steger *et al.* [299]	✗	∼	∼	✓	✓	∼	✗	✗
D. K. Nilsson *et al.* [300]	∼	∼	∼	✓	✓	∼	✗	∼
D. K. Nilsson *et al.* [301]	✗	∼	∼	✓	✓	∼	∼	∼
K. Mayilsamy *et al.* [302]	✗	∼	✓	✓	✓	∼	∼	✗
M. Steger *et al.* [303]	✗	∼	∼	✓	✓	∼	∼	✗
B. Liu *et al.* [304]	✗	✗	✗	∼	✓	∼	✗	✗
A. Vetter *et al.* [12]	✗	✗	✗	✗	✓	∼	✗	✓
S. Kugele *et al.* [14]	✗	✗	∼	✗	✓	∼	✓	∼
A. Iwai *et al.* [15]	✗	✗	∼	✗	✓	∼	✓	✓
A. Frigerio *et al.* [17]	✗	✗	✗	✗	✓	✗	✓	∼
S. M. Mahmud *et al.* [305]	✗	∼	✗	✓	✓	∼	✗	∼
V. Cebotari *et al.* [306]	✗	✗	✗	✗	✓	✗	✓	✗
B. Schegolev *et al.* [307]	✗	✗	✗	✗	✗	✗	✓	✓
S. M. Bohloul *et al.* [11]	✗	✗	✗	✗	✗	✗	✗	✓

· Not evoked (red), Evoked (yellow) and Focused (green)

software reference architectures or design patterns within the automotive industry and that we can identify four distinct categories of papers. Firstly, there are those that focus on software architectures and zoom in on the E/E properties, then those giving special importance to communications and networking within the architecture. Subsequently, there are those that focus on application characteristics and granularity, and finally, those focusing on the initial stages, going extensively through the context and use cases to justify architectural evolution. Furthermore, the run-time management of the infrastructures, done through the control services and which will be addressed in §VIII, are often evoked but are never the focal point of the paper, leaving the focus on this field considerably behind when compared to others.

Finally, despite the existence of papers covering software characteristics and relations we haven't found any paper on how to properly manage its life cycle (install / deploy / testing / monitoring), which is a crucial part of any success when implementing SOA architecture (cf. § VII). Moreover, in industry, AUTOSAR SOME-IP is the biggest representative of SOA in the automotive sector. While already being the principal OS standard for the constrained ECUs of vehicles, AUTOSAR released SOME-IP: a SOA-like architecture quite similar to the one proposed in Leguay et al. [303] and, in the same manner, focusing exclusively on the networking capabilities of the architecture and introducing interesting features such as service discovery and network abstraction to the legacy of AUTOSAR-based ECUs. This approach does not address the lack of control services and flexibility in the automotive embedded architectures nor their low dynamicity and adaptability that, as we said before, are key features for the future of the automotive industry. However, it's a good a good foundation from where to start and can be complemented with the addition of control services over it and other features to

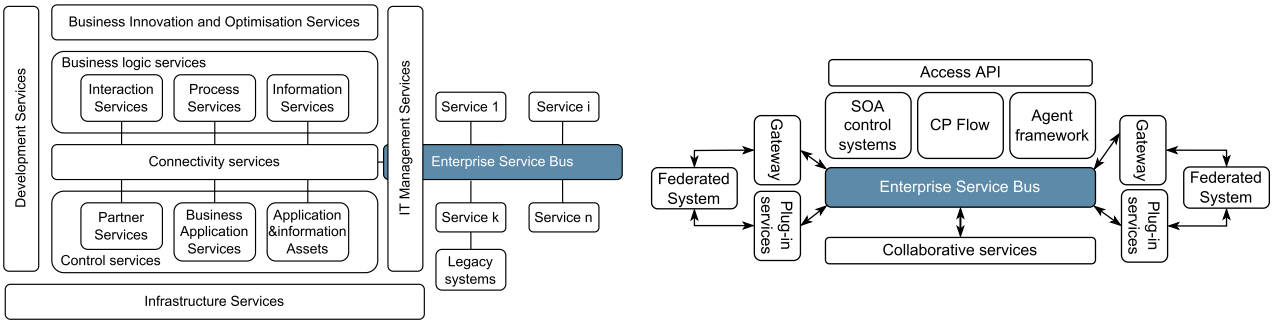


FIGURE 10. IBM WebSphere (left) & OASIS FERA (right) architecture schemes.

achieve the aforementioned capabilities. In this same scope, other proprietary OEMs, and companies (GuardKnox, BMW, TESLA…) have developed or offer SOA frameworks, however, the lack of papers detailing their structure or justifying their choices makes their evaluation and realistic positioning in the ecosystem impossible. It should also be noted that we will be comparing AUTOSAR's SOME-IP solution to other IT solutions in the next section.

3) IT AND AUTOMOTIVE SOFTWARE ARCHITECTURE CONVERGENCE: MAINTAINABILITY AND COMPLEXITY

To cover the lack of maturity of the automotive software architecture research proposals, we are now going to focus on the perspectives of both the classic high-end cloud computing sector, which was the precursor of SOA, and some lightweight SOA implementations from the IoT sector, comparing them over an eight-vertex criteria deriving from §IV and the open issues in §V. This evaluation is discussed through the whole section going one by one through the remarkable representative solutions and summarised in Table 6, which can be found at the end of the section. It is worth noting that, as these solutions are not targeting the automotive sector, we cannot evaluate if they comply with the traditional automotive safety & security certificates & standards, which must surely be worked on if finally integrated into the automotive in-vehicle systems.

If we start with solutions that are targeting cloud computing, the first proposal worth mentioning is IBM Websphere [304], [305], [306]. As we can see in Fig. 10, the core of this architecture is the connectivity services, which provide the infrastructure to manage the Enterprise Service Bus. These connectivity services provide three major resources: transport services, event services, and mediation services. Directly linked to the connectivity services, we can find business logic services that define what the application needs to develop, as well as control services, which are in charge of managing the interaction between services. In addition to that, this architecture also provides development services, modelling, testing, and maintaining the system, business innovation and optimisation services for research and development, sand-boxing, IT management services for security, network, virtualisation, system management and, finally, infrastructure services to manage hardware resource dependencies.

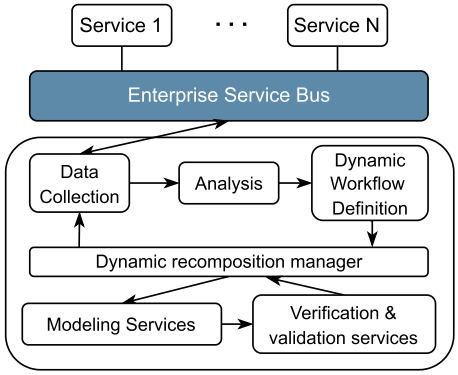


FIGURE 11. PESOI architecture scheme.

Secondly, we have OASIS FERA [307] (Federated Enterprise Reference Architecture) that presents a set of principles and guidelines for the development of an SOA application with loosely coupled collaboration for Federated Architectures.[3] This model, as we can see in Fig. 10, maps the Federated Architecture formed with semi-autonomous decentralised systems into the SOA concept architecture. In this implementation, the federated systems interfere with the rest of the federated systems, SOA control services and external/developer access APIs through a centralised fault tolerance ESB.

Thirdly, we are going to focus on a solution that concentrates on integrating the development and operation cycle into the SOA architecture: the Process-Embedded Service-Oriented Infrastructure (PESOI) [308]. As this solution integrates the development process into the infrastructure, this simplifies the redevelopment of the applications to add new features on run-time.

[3]Federated architecture (FA) is a pattern in enterprise architecture that allows interoperability and information sharing between semi-autonomous de-centrally organised lines of business (LOBs), information technology systems and applications.

This model then implements a control service layer based on the development cycle of the application, in which, we can count (1) modelling services, to offer high SOSE support and services to help the users to construct the architecture model, the process model and the policy model of the target application, (2) verification and validation services, (3) a dynamic re-composition manager, in charge of taking the model and picking up proper services from the service repositories to compose the desired applications, (4) run-time data collection, (5) Analysis, and (6) dynamic workflow definition services in order to monitor the system, adapt and optimise execution after the applications are assembled and deployed (cf. Fig. 11). Thus, this system is able to monitor the behaviour of the service and dynamically reconfigure it for better performance or reliability (or even replace the service) at run-time, which is an very important feature for the automotive industry.

This time, focusing in IoT-like resource restrained devices as in the automotive industry, we have Lenguay et al. [303], in which a proposal and implementation of multi-level SOA-based architecture is described. This is for heterogeneous and dynamic wireless sensor nodes with limited computing, battery, and communication capacity. Afterwards this solution will then create both a new protocol and routing algorithm allowing for dynamic discovery (cf. Fig. 12), network abstraction, fault-tolerance, as well as invoking services in other nodes. However, the absence of nodes with higher computing capabilities to implement a more complex control layer considerably holds back the capacity and possible uses of the system.

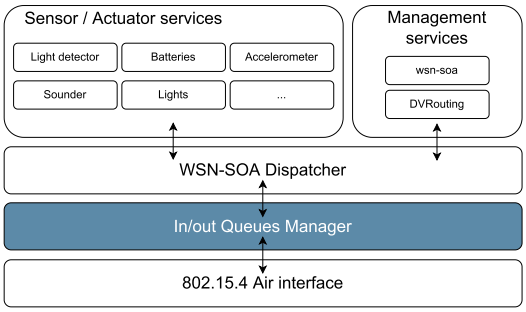


FIGURE 12. Lenguay et al. [303] architecture scheme.

Finally, within the same scope, in Wang et al. [309] they propose a Service-Oriented model for heterogeneous MPSoCs, suggesting a new 3-layered hierarchical model, as we can see in Fig. 13. Thus, the three layers are, in order from the furthest from the hardware, to the closest; (1) the Services Layer, which offers an API for deployment, run-time analysis mechanisms, an orchestrator and scheduler and some data collection services, (2) the Servants Layer, which is in charge of the error management of the different services running and the fault-tolerant communication support among them, and finally, (3) the Physical Layer, which is responsible for providing the requested physical resources to

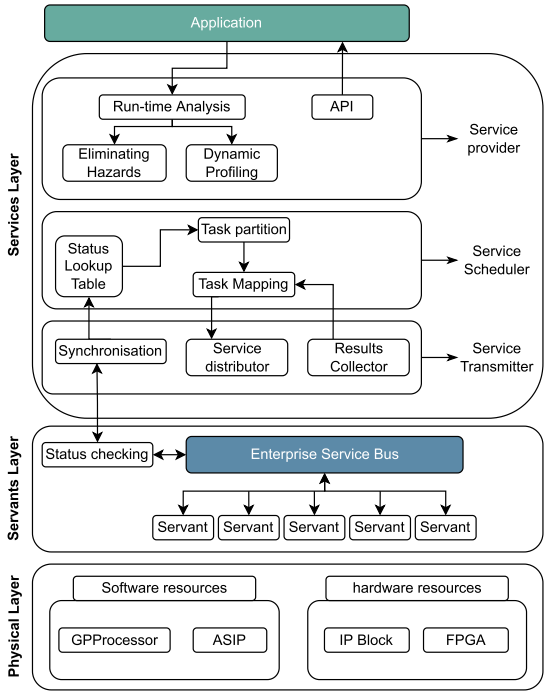


FIGURE 13. Wang et al. [309] architecture scheme.

the different service. Therefore, this solution implements a completely dynamic SOA in which the resources and services are allocated to need and run-time analysis techniques, aiming to reduce hazards and to optimise the execution of the applications.

To conclude, if you compare all the previous detailed solution descriptions, we can highlight some layers that are common to all of them. To begin with, we have the Fault tolerant Communication Layer, usually named Enterprise Service bus, and which will looked at in § VI-C. This is probably the most important layer as it enhances the flexibility of the architecture and cuts the hardware specific couplings, therefore increasing the reusability of the software. Secondly, now that the communication between services has been ensured, the second point we can highlight is the software development pipelines, that we will study in detail in § VII, covering the software installation, orchestration, deployment, and testing within the vehicles. Finally, and more extensive than the two previous layers, we have the control services, to be looked at further in § VIII, which are very varied but share the common goal of ensuring that the software installed and the communication channel are both functioning well through services such as the service registry, orchestrators and schedulers, the status checkers etc. However, we need to add some mechanisms to those ones, allowing them to compensate for the heterogeneity either in terms OS, security and safety constraints, and resources, all of which are inherent to the automotive sector. Furthermore, as detailed in Table 6, we can see that the more complex and complete the control services and architecture proposition are, the lower the maintenance

TABLE 6. Comparison summary of the IT SOA state-of-the-art. Classification made from the aggregated data from the papers [304], [305], [306], [307], [303], [308], [309], [310].

Solution	Flexibility	Dynamicity	Scalability	Dev. cost	Maintenance cost	Integration cost	Complexity	Fault Tolerance	Energy consumption
IBM WebSphere [304]–[306]	Medium	✓	✓	Low	High	Medium	Low	On apps	High ↘
OASIS FERA [307]	Low	✓	✓	Low	High	Medium ↗	Low ↘	On apps	Medium ↗
PESOI [308]	High	✓	✓	Medium ↗	Medium ↘	Low	Medium ↗	On apps	High
J. Leguay *et al.* [303]	Medium	✓	∼	Low ↗	High ↘	Medium	Low	On apps	Low
C. Wang *et al.* [309]	High	✓	∼	High	Low	Low ↘	High	On apps	Low
SOME/IP [310]	Low	∼	∼	Low ↘	High ↗	High	Low ↘	∼ On apps	Low

and integration costs. However, this signifies a high initial cost that, considering the life cycle of the vehicles, should be rapidly compensated for by simpler maintenance and integration cycles.

C. NETWORK ARCHITECTURES

Given the advances of both E/E (cf. § VI-A) and Software architecture (cf. § VI-B), we now need to adapt the network stack so as to fully profit from the new bandwidth enhancements thanks to the higher presence of Ethernet, the new simple E/E architecture board availability, and to match the requirements necessary to have fault-tolerant data-based communication middleware (or Enterprise Service Bus). Thus, in this section we will start by (1) surveying how the network architecture is standardised, by analysing application-level network protocols to be used for this homogeneous data-based communication middleware over IP. After that, (2) we will discuss the rest of the tooling that allows us to enable data-oriented communications over this middleware, which is a key point for a flexible and dynamic abstract communication ESB. Finally, (3) we will discuss how the legacy software components can be integrated into this framework, which is mandatory for the transition.

1) STANDARDISATION: NETWORK PROTOCOLS

To begin the study, if we start with the solutions from the IT domain, where the SOA architectures are already in place, HTTP REST [311], [312] is highlighted. HTTP REST is a communication protocol that defines an ensemble of constraints to be used when creating a web service to establish easy interoperability between services on the internet. This protocol follows a request/response communication pattern in star topology. In this host-centric protocol, the central server will listen for client requests that, on reception, will invoke the different functions available. HTTP implements lots of interesting mechanisms such as TLS security, object compatible payloads, standard response, and error codes. However, the fact that this protocol needs a centralised server to offer its services, linked to the lack of environment service discovery or single point of error, failure prevention mechanisms might be a problem when dealing with safety constrained systems such as vehicle IT architecture.

In the same scope, if we keep looking at web service oriented communications protocols we can find other interesting protocols such as SOAP [313], [314] or WebSocket [315]. SOAP first appeared two years before HTTP REST and provides mostly the same functionalities. The main difference is that SOAP requests are written in a custom XML (or YAML) SOAP format then wrapped in an HTTP message while REST directly uses the HTTP as the application layer protocol. This provides higher bandwidth and processing consumption for the SOAP protocol if compared to REST, which in the context of embedded systems, is a critical point that makes REST more suitable. On the other hand, WebSocket approaches this problem as a lower-end solution. WebSocket is based on the concepts of socket and port and will use the IP address from the device and their application port details in order to establish a bi-directional communication channel, whereas HTTP REST only works as a unidirectional channel. WebSocket is therefore ideal for real-time scalable communication with high data exchange between services but is less performant when handling lots of small requests from different clients. At the same time, as WebSocket is a very low-level restful protocol that manages the mapping itself between services, it needs to know a lot of infrastructure details. This lack of abstraction might limit the possibilities of a flexible SOA arch. in which the services might change their location over time. However, it could be interesting when used in some situations since the comm. cost is lower than REST and the bidirectional socket can offer some advantages.

To reduce overheads and computing requirements of these protocols, the Constraint RESTful Environments (CoRE) working group of IETF developed CoAP [316]. CoAP is a protocol like HTTP REST but with the peculiarity that the headers, methods, and status codes are all binary encoded and that it runs over UDP instead of TCP. This reduces the package overhead but, unfortunately, also transmission reliability. To deal with the lack of reliability of UDP, CoAP implements a functional layer to re-transmit lost or corrupted packages. At the same time, in order to reduce the resource consumption of the protocol even more, they created a new mechanism to reduce the message exchange to retrieve data. While in HTTP REST the data retrieving always implies 2 messages (a HTTP GET request and a HTTP RESPONSE), in CoAP, with the first HTTP GET request, you can add a header flag (observe) to continue receiving changes to the requested resource from the server. Therefore, whenever this resource changes, the server will send the information to the client without the need

for an HTTP GET Request. This implementation was the first step to asynchronous messaging over HTTP, getting closer to a publish/subscribe communication pattern.

Asynchronous messaging allows systems to increase the flexibility of their architecture, which makes this communication pattern rather more suitable for resource constrained devices and non-ideal network conditions than the classic request/response paradigm. Between the asynchronous messaging protocols we can state that three of them stand out from the rest: MQTT [317], AMQP [318], XMPP [319] and DDS [320], [321]. MQTT runs over TCP (which ensures its reliability) with the peculiarity that it has an exceedingly small header and, thus, extremely low package overhead, making it one of the most prominent solutions in constrained environments. MQTT follows the publish/subscribe paradigm with the peculiarity that any communication is centralised through a Broker that will handle message persistence and ensure its arrival to the correct host. However, since it was designed to be the more lightweight possibility, MQTT's weak point is a lack of security and the potential single point of failures. In the same scope, being aware of the security deficiencies of MQTT, there also exist alternatives such as AMQP and XMPP. Even if both implement strong data protection and authentication, XMPP is a bit more secure than AMQP because of the inclusion of stricter security, authentication, privacy, and access control protocol extensions. However, the fact that, at the same time, XMPP offers request/response and publish/subscribe communication services and that they use an inefficient XML payload formatting, makes the resource consumption of XML high compared to both AMQP and MQTT- This will complicate its use on lossy low-power wireless networks and low-specification processors. Even though XMPP is slightly more secure than AMQP, this second protocol has some other interesting points. As one of the main problems in MQTT was the threat that a broker failure would prevent the entire system from working, AMQP developed a new async. peer-to-peer mechanism that makes the pub./sub. pattern more flexible and robust, eliminating the threat coming from the centralised broker single point of failure. However, as in XMPP, all these features denote an increase in network, power consumption, processing, and memory reqs., making it a potential problem if implemented in highly constrained devices.

Finally, if we look at other domain specific solutions we can find OPC UA [322], [323] (coming from the industry 4.0) and SOME/IP [324], [325] (in the automotive industry). To begin, OPC UA offers both publish/subscribe and request/response mechanisms based on TCP/IP. OPC UA, and as is required for industrial context, focuses on real time and safety of machine-to-machine communications. As this protocol is widely adopted in the industry 4.0, it offers interesting interoperability options that can lead to new business models and interactions between industry and vehicles. Secondly, SOME/IP is an open standard messaging protocol formalised by AU- TOSAR as a reaction to the inclusion of software-oriented architectures in the automotive system. SOME/IP offers a complete communication standard including 3 communication patterns (publish/subscribe, request/response and fire and forget (request without response)) that communicate via a centralised broker over TCP/IP. At the same time, it offers a service discovery protocol to deal with any abstraction between network and data providers and some basic security features. One final positive characteristic of AUTOSAR is the fact that it has already been widely adopted by the automotive OEMs which eases the transition to a software-oriented architecture. However, AUTOSAR has the same problems as the rest of the broker centralised communication protocols, as well as the fact that it does not handle data objects or serialise a method invocation payload.

Hence, if we look further into detail in Table 7, it seems clear that the combination of multiple communication protocols can be useful to cover all the different automotive use cases. To begin, among the communication protocols that include publish/subscribe patterns, the most remarkable solutions would be MQTT, SOME/IP, DDS, AMQP and ROS2.

If we try to use the most resource efficient solution, despite security issues, MQTT offers the lowest resource consumption and compatibility for Android, Austosar Adaptive and Linux. In addition to that, MQTT is a widespread solution in the IT world, which is an advantage when customers are looking for new solutions at a lower price. From another point of view, if what we seek is a solution that is compatible with both the old ECUs and the new ones, SOME/IP, as it's the AUTOSAR solution, is the only one capable of ensuring compatibility with both AUTOSAR Classic and the other Linux-based operating systems. However, SOME/IP is far from being optimal in terms of other criteria, having low security and higher development and maintenance costs than MQTT.

From a security point of view, both solutions offer basic authentication, encryption, and TLS techniques. However, other communication protocols like ROS2, AMQP and DDS have stronger additional mechanisms in exchange for a higher resource consumption. Finally, if we look from a flexibility point of view, once again DDS, AMQP and ROS2, as they are P2P publish/subscribe solutions, offer a higher flexibility on the system design. On the other hand, if we shift our focus to communication protocols following the request/response scheme, we have HTTP REST, WebSocket, CoAP, OPC UA and SOME/IP. Among these solutions, HTTP REST is the most frequently used in the IT world, which presents an advantage in terms of development tools and support. At the same time, it is compatible with all but AUTOSAR Classic and offers decent security and flexibility at an assumable resource cost. In the same scope, CoAP is an optimisation of HTTP REST for constrained environments. However, with the libraries and optimisations made in REST by the community, the resource consumption of REST is only slightly superior to CoAP.

TABLE 7. Application layer network stack comparison.

Protocols		Signal-based	HTTP REST	SOAP	WebSocket	CoAP	MQTT	AMQP	XMPP	DDS	ROS2	OPC UA	SOME/IP
Standard		-	RFC 6690	W3C	RFC6455	RFC7252	OASIS	OASIS	RFC6210	OMG	OSRF	OPC	ISO17215
Paradigm	Request/Response	✗	✓	✓	✓(bidirectional)	✓	✗	✗	✗	✗	✗	✓	✓
	Publish/Subscribe	✗	✗	✗	✗	∼	✓	✓(P2P and Broker)	✓	✓(P2P)	✓(P2P)	✓	✓
	Fire & Forget	✗	∼	∼	✗	✓	✗	✗	✗	✗	✗	✗	✓
	Static subscribe	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Architecture	Topology	1-1	1-N	1-N	1-N	1-N	N-1-N	N-1-N or N-N	N-1-N	N-N	N-N	N-1	1-1
	Broker	No need	No need	No need	No need	No need	Centralized	Centralized/Distributed	Centralized	Distributed	Distributed	Centralized	Centralized
	Multicast	✓	✗	✗	✗	✓	✓	✓	✓	✓	✓	✓	✗
	Asynchronous	✗	✗	✗	✗	∼	✓	✓	✓	✓	✓	✓	✓
	Dynamic discovery	✗	✗	✗	✗	✗	✗	✓(device)	✗	✓(service)	✓(service)	✓(service)	✓(service)
	Network abstraction	✗	✗	✗	✗	∼	∼	✓	∼	✓	✓	∼	∼
Interface	Payload data format	Binary	JSON/XML/YAML	XML/YAML	Not specified	JSON/XML	Not specified	Not specified	XML	JSON/XML	JSON/XML	JSON	Binary
	Serialization	✗	✓	✓	✓	✓	✓	∼	∼	✓	✓	✓	✗
	Interface description	-	REST IDL	CORBA IDL	Web IDL	Not defined	Not defined	Not defined	XMPP Stanza	OMG IDL	OMG IDL	OMG IDL	Franca IDL
Design	Complexity	Low	Medium ↘	Medium	Medium ↗	Medium ↗	Low ↗	Medium	High	High	High	Medium ↗	Low ↗
	Flexibility	Low	Medium	Medium	Low ↗	Medium ↗	Medium ↗	High	Medium ↗	High	High	High	Medium ↗
	Adaptability	Low	Low ↗	Low ↗	Low	Low ↗	Medium ↗	High	Medium ↗	High	High	High ↘	Medium ↗
	Resilience	Low	Low ↗	Low ↗	Low	Medium	Medium	Medium ↗	Medium	High	High	Medium ↗	Medium
	Resource Consumption	Low	Medium	Medium ↗	Low ↗	Medium	Low ↗	Medium ↗	High	Medium ↗	Medium ↗	Medium ↗	Medium
	Implementation cost	Low	Medium	Medium ↗	Medium	Medium ↗	Low	Medium	Medium ↗	Medium ↗	High	High	Medium
	Development cost	Medium ↗	Low	Low ↗	Low ↗	Medium	Low	Medium	Medium	Medium	Medium	Medium ↗	Medium
	Maintenance cost	High	Low	Low ↗	Medium	Medium	Low	Low ↗	High	Medium	Medium	Medium	Medium
	QoS Aware	✗	✗	✗	✗	✓(Limited)	✓(3 levels)	✓(3 levels)	✗	✓	✓	✗	✓
System	Autosar Classic	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
	Autosar Adaptive	✓	✓	∼	✓	∼	∼	✗	✗	✓	✗	∼	✓
	Linux-like	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Android	∼	✓	✓	✓	✓	✓	✓	✓	✓	∼	✓	∼
Security	TLS/DTLS	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Authentication	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	∼
	Encryption	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	∼
	Access Control	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	∼
	Extra security	Low	Medium	Medium	Medium	Medium ↗	Low ↗	Medium ↗	High	Medium ↗	High	High	Low ↗

However, if resource efficiency is a priority, WebSocket is the most interesting solution. Nevertheless, this protocol needs low-level knowledge of the infrastructure, which blocks the network abstraction concept of SOA unless an abstraction connection manager is implemented over this network stack at application level. After that, when focusing on security, it is clear that OPC UA is the most adequate solution. However, the development and maintenance costs and the resource consumption of the protocol are higher than the other ones. Finally, as in the case of the publish/subscribe paradigm, SOME/IP is the only fully compatible solution with the AUTOSAR classic and Linux-based ECUs.

2) DATA-MANAGEMENT: THROUGH A DATA-CENTRIC COMMUNICATI- ON MIDDLEWARE

As the aforementioned evolution trends of both E/E and software architecture suggest an increasing break in domain boundaries (i.e., ADAS, Power-train…), a democratization of Ethernet as the main transport support, a dynamic and seamless perception of the software location, as well as a steady rise the complexity of deterministic. Whilst deterministic comm. is a key challenge to match real-time constraints of the automotive sector [326], something that we will discuss in depth later in § VI-C3, in this section we are going to focus on the migration from static network mapping to more flexible and dynamic data-centric mechanisms. Remember that, in the FCA sub-variants, and specifically in SOA, systems are represented as a set of services that are either data producers, data consumers or both at the same time.

Today, as the in-vehicle systems and development processes mostly ECU / signal-based development approaches, the complexity and data maintenance of both signal and bus message databases requires a lot of time and money [327]. The high software coupling to both network and hardware means that multiple functions, such as driver presence, are implemented several times for specific customer functions on different ECUs when they could easily be communal functions provided as a service and reused by all ECUs. Moreover, nowadays, we cannot underestimate that an unnecessarily high bus load can be observed because of signals being sent without receivers on most of the CAN buses [328]. Therefore, the implementation of an Ethernet-based data centric middleware will not only allow for a reduction in overall system complexity and comply with the flexibility and dynamicity proposed by the software architecture evolution trends (cf. § VI-B), but it will also help cope with the problems of function reuse and unnecessary network consumption. For that matter, the only source of communication overhead will be the one added by the middleware, which is predictable and not rooted in inadequate tooling or insufficient communication management processes. This shall help maintain the system determinism.

On the one hand, if we look closer at the proposals surrounding data-centric communication middleware for the automotive industry, what stands out is that most of them rely on the aforementioned DDS tack, with the publish/subscribe comm. pattern playing a central role in this matter. The DDS functions either alone with some containerisation and custom tooling [328], or as an extension of a network layer for either SOME-IP [327] or ROS 2 [329]. As a complement of these studies, [326] presents, in a theoretical way, a study of the desired properties for future data-centric real-time comm. middleware, focusing on how resources are shared, priorities managed, and data consistency can be ensured.

On the other hand, with the objective currently not only being about adding a coordination layer but, also, to change the host- centric philosophy TCP/IP and move towards an information centric philosophy, some papers [330], [331] chose to delegate the network abstraction to a lower OSI level. Thus, the network routing protocols of this solution will no longer generate routing tables containing the addresses of the connected devices, but instead of the data that these connected devices can provide [332]. However, since there is no widespread standard as of yet, Information Centric Architecture implementation has been approached in diverse ways (Data-Oriented Network Architecture, PubSub, Named Data Networking...). Even though different implementations have small subtleties, they all focus on the data (Named Data Object) and its storage, mobility, authenticity, integrity, and security [332], [333]. ICN approaches can be seen as an ensemble of data publishers and data requesters connected through the NDO they share and the abstraction routing mechanisms. However, even though the potential of this new network arch. paradigm is interesting, the maturity of the technology makes its integration into the in-vehicle systems more complicated.

3) LATENCY: WITH REGARD TO APP. CONSTRAINTS AND DEADLINES

As the comm. capabilities of most of the ECUs within the in-vehicle arch. rise more and more, having a precise view of traffic flows, the nature of the delays (called ''latency'') and variation in travel time (called ''jitter'') has become crucial for both understanding system interactions and matching safety requirements in the different apps. The control and prioritisation of traffic flows can be done at multiple levels - i.e., app. level, data-link level, hardware level...

At the application level, this feature can be done either by adding time-sensitive extensions for the network stack protocols [334], [335], such as those in Table 7, or by implementing some high-level global-schedulers [336]. However, all these solutions mean additional system constraints, such as needing to use a common network stack, which has a negative impact on system flexibility. Moreover, as the scheduling takes place during the application layer, it adds further delays for decapsulating the packages, which also increments the jitter as it strongly depends on the CPU use of the nodes.

On the other hand, hardware-level, real-time traffic flow control can keep treatment delays down and maintain network stack flexibility, solving the main problems that arise from application level real-time communication services. Some examples of these are presented in [337] and [338]. However, these solutions are suddenly too expensive to be implemented within cars and have a real risk of buffer overflow or of ignoring some traffic norms due to its prioritisation. Besides, these solutions limit the dynamicity and adaptability of the system since the hardware and software don't follow the same timescale in their life cycle.

Therefore, being aware of these problems in the automotive industry, both research and industry position themselves by carrying out this traffic-flow control directly over the data-link layer, following the standards described in IEEE 802.1Qca,AB,Qbv,Qci...by the IEEE Time Sensitive Networking (TSN) standardisation group [339], [340], [341], [342]. TSN, therefore, proposes a set of standards featuring time synchronisation, traffic scheduling, gate control, frame preemption...that enables it to have a fine grain control over traffic prioritisation, by reserving specific bandwidths for each traffic, and allows for a precise calculation of latency and jitter. However, even though nowadays TSN is still elevated, it will certainly be deployed cheaper over MAC in the next years. Note that TSN can be complemented with higher level application QoS solutions [343], [344], [345], or even other techniques such as DPDK [346], [347] to overlap the kernel and enhance network bandwidth capabilities.

D. IoV PLATFORMS

The Internet of Vehicles (IoV) is a network born from the integration of vehicles to the smart city and whose objective is to interconnect cars (V2V), pedestrians(V2P), cloud computing infrastructure (V2C) and parts of the urban infrastructure (V2I), such as charging stations, roads, traffic lights, etc. to make transportation more autonomous, safer, faster, more efficient, more eco-friendly, etc. In this section, we will split the IoV platform into two categories, i.e., Vehicle-to-Cloud (V2C) and Vehicle-to-Everything (V2X), and subsequently detail the advances and evolution perspectives of both.

1) VEHICLE-TO-CLOUD (V2C)

As the levels of autonomy and functionality of vehicles increase, vehicles are expected to execute a wide range of computations over short periods. Given the limited on-board battery capability and computation capacity of the in-vehicle systems, offloading, the more power-sapping and time-consuming computation tasks to other more powerful servers, from the Cloud Architectures, has significantly improved the performance of many applications, such as intelligent driving, cruise-control assistance or log collection [348], [349], [350], [351]. Cloud computing is therefore defined, by the US National Institute for Standards and Technology (NIST) in [352], as a model that enables ubiquitous, convenient,

and on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provided and released with minimal mgmt effort or service provider interaction. Moreover, the latest advances in the Cloud comp. sector, driven by the Internet-of-Things explosion, allow us to complete this definition as there are now more possibilities concerning where to place and how to manage Cloud nodes.

If we begin by focusing on the location of these nodes, Table 8 summarises the four big trends regarding this matter. First, we can highlight that Central Cloud Comp. nodes, which are those traditionally used by most of the industry nowadays, are placed in data centers extremely far from the vehicles, which has a negative impact on the latency and energy consumption for this solution. However, they also have the advantage of working globally and offering mostly unlimited resources and, thus, they are preferable when being used for global non-real-time overly complex calculus or data storage. Following on closely, Edge Comp. nodes are placed in smaller data centers at the edge of the network, which tries to even the trade-off between latency and quantity of resources. On the other hand, Fog, and Mist comp., the first based in local antennas or buildings and the second in the IoT surroundings, have a similar objective of scarifying some calculus performance to gain considerably in latency. These nodes are worth using when the calculus does not have exceptionally long to run, as the gain in latency will compensate for the restrained computing capabilities. However, it is worth noting that all three, Edge, Fog, and Mist computing solutions, have limited mobility support, which means that we would need to add high-level mechanisms, such as data or function prefetching, to make the data follow the vehicles when in movement.

On the other hand, if we now focus on the nature and mgmt. of how the cloud computing platform is used, we need to highlight several different modes - i.e., On-premises, Infrastructure-as-a-Service (IaaS) [354], [371], Platform-as-a-Service (PaaS) [372], Function-as-a-Service (FaaS) [373] and Software-as-a-Service (SaaS) [374], [375]. As cloud computing architectures are often split into nine distinct levels - i.e., network, storage, servers, virtualisation, operating system, container technologies, run-time, application, and data - we are also going to rely on this definition to characterise the different management modes. Fig. 14 defines, for each of the previously introduced modes, which part is carried out by the Automakers and which by the Cloud provider. Note that, the more things that are dealt with by the Automakers (On-premises / IaaS), the more complex and time-prone to manage it is, but also more configurability is offered. On the other hand, by letting the cloud provider manage most of the layers, Automakers would gain in agility, while also easing the complexity of the system, thus being able to focus their efforts on more business-related issues. Furthermore, if we look closer at the network stack used for V2C communication, studies [376], [377] show that this communication

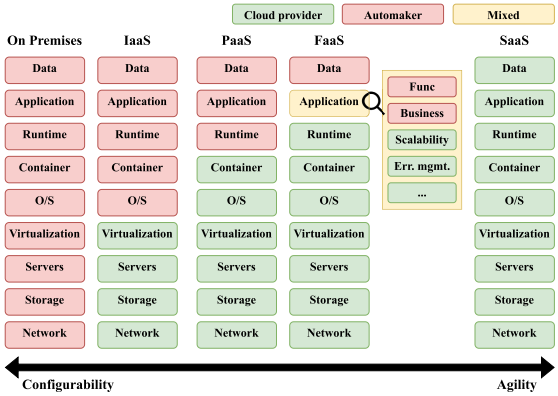


FIGURE 14. Cloud computing management possibilities.

normally takes place through standard LTE/5G wireless networks.

In conclusion if we give a brief overview of current Cloud computing use in the automotive sector, as you can see once again in Table 8, most of the Automakers use On-premises/IaaS/SaaS based Central Cloud Computing solutions. However, this trend is beginning to reverse due to academia proposals suggesting infrastructures closer to the car, which should most likely happen in the industry in the following years, while maintaining the Central Cloud Infrastructures for overly complex calculus and big-data storage. Finally, with regard to the management mode, all management modes that are already in use should continue to coexist as they have done until now, integrating little by little both PaaS and FaaS into the ecosystem.

2) VEHICLE-TO-EVERYTHING (V2X)

Vehicle-to-Everything communication is essential to maintain a shared information layer throughout all the vehicles, pedestrians, and infrastructure. In this layer, the position, status, trajectories, and obstacles for all smart-city participants are grouped together, with the objective of increasing road safety, helping users and vehicles throughout their journey, adapting the energy supply, parking spaces, etc. Thus, in this section, we are going to explain in detail some of the main V2X use cases, i.e., vehicle-to-Vehicle (V2V), Vehicle-to-Pedestrian (V2P) and Vehicle-to-Infrastructure (V2I).

a: VEHICLE-TO-VEHICLE (V2V) ARCHITECTURES

As V2V encompasses a wide range of functionalities and use cases, in this section, we will order them according to the distance and communication technology used to interact with one another. If we begin with those solutions of short V2V ranges that are the most widespread. Their standardisation relies on the extension of the WIFI standard (IEEE 802.11a) to support the ad-hoc mode (IEEE 802.11p [378]) made back in 1999 by the American Federal Communication Commission (FCC), allowing for the use of highly dynamic Vehicular Ad-hoc NETworks (VANET). This protocol can then control any high mobility conditions caused by

TABLE 8. Comparison of cloud computing solutions according to the distance where they are placed.

Features	Central Cloud Computing	Edge Computing	Fog Computing	Mist Computing
IT examples	[353], [354], [355]	[356], [357], [358]	[359], [360]	[361], [362]
Automotive examples	[348], [349], [350], [351]	[363], [364]	[365], [366], [367], [368]	[369], [370]
Distance to vehicle	Far	Medium	Close	Very close
Sevice Coverage	Global	Regional	Local	Local
Comm. Latency	High	Medium	Low	Very low
Maturity	Industry Consolidated	Industry In Consolidation	Academia Consolidated	Academia In Consolidation
Node capacities	Unlimited	High	Medium	Low
Energy consumption	Very High	High	Medium	Low

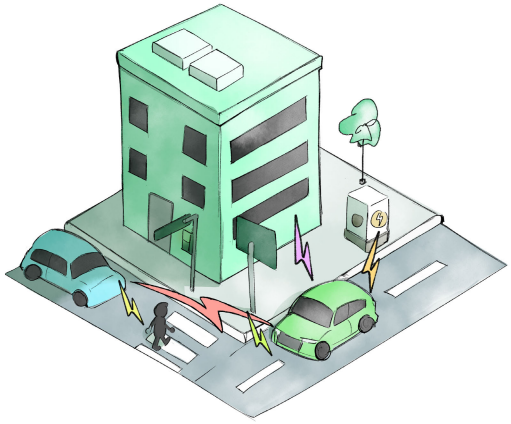


FIGURE 15. IoV Interactions for smart cities.

high speeds coming from multi-path reflections, and Doppler shifts obstructions [379], [380], thanks to techniques such as the Distributed Congestion Control (DCC) mechanism or the Multiple Access with Collision Avoidance mechanism (CSMA-CA). Therefore, in these networks the vehicles will constantly transmit and receive information from all other vehicles within their transmission range. VANETS are suddenly being used for either safety-oriented applications, such as driving assistance [381] (i.e., lane changing, emergency braking or cooperative collision avoidance), information [382] (i.e., speed limit or road work areas), and warnings [383] (i.e., post-crash notifications, road conditions or collision alerts), non-safety-oriented applications for some traffic controls [384], [385], and some comfort and infotainment applications [386], all the while being aware of the limited data-rate and security limitations of the protocol.

In this same scope, with the objective of updating the VANET standard and adapting to the data-centric vehicle enhancements, a new protocol appears that makes a few changes to the IEEE 802.11p extension. This protocol is known as Dedicated Short Range Communication (DSRC) [387], [388] protocol and, among its main improvements we can highlight: the addition of network congestion control for high-density networks, a significant reduction in communication overhead (which remains unsuitable for big-data ultra-low latency applications), a fair management algorithm for transmission coordination between vehicles to reduce the network noise, and the addition of additional security mechanisms. Some examples of applications running DSRC are lane change detection [389] or emergency collision avoidance [390]. However, both DSRC and VANET share their bandwidth not only with other vehicles but with several applications from the smart-city, which is a limiting factor for them when trying to achieve higher data-rates or SLA ratios. In addition, the security of these two protocols is weak compared to the more powerful LTE-based protocols available.

Thus, in order to explore other uncongested frequencies, some studies [391], [392], [393] focus on the adoption of an emergent technology known as Visible Light Communication (VLC). This targets a completely unexplored communication method: light. Some studies have shown interesting possibilities arising from this technology for short range driving assistance applications in V2V clusters; however, this technology is still not mature enough and most of the research is dedicated to increasing its performance, decreasing the noise from other lights or the sun, compensating for weather conditions, etc. Moreover, VLC, as well as previous standards, cannot guarantee service availability, as they need other vehicles in proximity to enable their services.

On the other hand, with the V2V services needing higher availability despite the presence of vehicles nearby, higher security or, simply a higher data-rate, academia [379], [394], [395] suggests 5G must be supported by future V2V architectures, assisting the current standards when they are not able to match the requirements of the application. 5G aims to support wireless communications with high reliability, ultra-low latency, and ultra-high throughput, offering several

interesting features such as Proximity Services (ProSE), and allowing for awareness to detect nearby devices without continuously emitting data, or data managing services. Literature suggests that 5G networks shall be increasingly used in the coming years for various existing and new use-cases such as global traffic management, V2V update spreading, infotainment services, etc.

b: VEHICLE-TO-PEDESTRIAN (V2P) ARCHITECTURES

Pedestrians, cyclists and motorists, who are usually grouped under Vulnerable Road Users (VRUs), account for a big part of traffic fatalities (e.g., in 2020 USA National Highway Traffic Safety Administration (NHTSA) declared 1,674 pedestrian and 355 cyclist fatalities, compared with 10,626 traffic fatalities, which represents around 20% of the country's traffic fatalities [396])). V2P architectures englobe both crash prevention architectures and those assisting both VRUs and vehicles to increase their travel efficiency [397], [398], [399], often named convenience applications (i.e., ride-sharing, green light for bicycles or travel information for VRUs). Typically, V2P architectures involve periodic exchanges of messages among vehicles and VRUs, either directly, through ad-hoc communication technologies such as VANET or DSRC, or indirectly, through infrastructure such as cellular technology. It is normal that VRUs smartphones often play a key role in the mechanism. In addition, these architectures often operate in three distinct phases: detection, tracking and trajectory prediction, and action. Some examples of V2P architectures are [400], [401], and [402].

c: VEHICLE-TO-INFRASTRUCTURE (V2I) ARCHITECTURES

Communication between vehicles and infrastructure [403], [404], [405] encompasses the interactions between vehicles (V2R) and roads, charging stations (V2G), and houses and buildings (B2H & B2B). On the one hand, Vehicle-to-Road (V2R) [406], [407], [408] architectures, which have been already slightly addressed by the V2P indirect mechanisms, aim to facilitate the next step for driverless vehicles by implementing mechanisms for Advanced Driver Assistance (ADAS), such as traffic light cycle details, potential road hazard alerts, vehicle congestion monitoring, global traffic organisation mechanisms, etc. On the other hand, all three V2G, V2H & V2B [409], [410], [411], [412] focus on different sources of power, mostly EV vehicles, discussing charging station coverage, energy eff. of the charging process itself or the parallel connection between the vehicle and the charging network for other vehicle maint. purposes. At this juncture, it is worth noting that, some of the aforementioned protocols, such as OPC-UA (cf. § VI-C), will grow in importance in the near future.

In this section, we have covered the main architectural aspects of the automotive in-vehicle ICT architectures (i.e., hardware, software, network, and external architectures). However, the architecture design is, as we saw in §V, just one part of the life cycle. In the next section, we will focus on how the elements that we have just presented interact with each other and the different data fluxes within these architectures by delving deeper into the software pipelines.

VII. SOFTWARE DELIVERY PIPELINES

Studies [218], [219], [220] have shown that, given the big changes coming to automotive industry motivated by both industry and society (cf. § IV), the frequency of both new system/application releases and updates will significantly increase in the next five to ten years, making application life cycles far more dynamic than nowadays. This acceleration of software in the automotive industry has already started, leading to a higher level of faults in software components which accounted for almost half of the vehicle recalls last year [1], the highest level for the last few decades. Thus, as a counter effect, bug fixes and security threats correction will also carry more weight than nowadays, which could lead to even higher software dynamicity and, with it, an increase in the need to revisit the software delivery pipelines, either centralised, through V2C, or distributed, through V2X (mostly V2V). Moreover, as cars move further towards software customisation, these faults will inevitably increase, as the number of software and hardware context variants rises. Therefore, being able to ensure and define a dynamic and flexible way to bring and update the systems and software, both during the production phase, garage and on the road, is essential so as to address the challenges of future architectures.

In this section, we will go through all of the software delivery pipelines, which for us are composed of five independent parts; development (cf. § VII-A), delivery (cf. § VII-B), deployment (cf. § VII-C), testing (cf. § VII-D) and orchestration (cf. § VII-E). It should also be taken into consideration that the inter-service data-based communication has already been covered in § VI-C by the communication middleware. Moreover, unlike in the previous section, we will focus now on the sub-features of each part instead of on their properties.

A. DEVELOPMENT

In this section, we take an in-depth look at the first part of software delivery pipelines, software development. We will start by extensively focusing on the definition of software and what it consists of. Afterwards, as the development is at a very multifaceted stage, we will go through the software development methodologies, focusing on how they are adapting to on-going evolutions.

1) SOFTWARE CHARACTERISTICS

As software packages can take many forms in the automotive systems, depending on their final target node or purpose, first we need to clarify which kinds are, for us, those that are the most important and, thus, those that we are going to use as the base for the following sections. It is also worth noting that these packages are usually composed of source code files, configuration instructions, testing instructions, various meta-data files containing their certificate, dependencies etc. The five kinds of software are as follow:

(a) Application Configuration packages are composed of, as indicated by their name, a set of run-time/post-installation parameter changes. These parameters include examples such as driver profile changes, deep learning algorithm optimisations, or parameter updates for regulatory compliance. These updates strongly condition vehicle behaviour of the vehicle without requiring any software changes, Due to this, these update types do not require the relevant software components to shut down, only for the vehicle to stop.
(b) Security / Key packages are composed of configurations allowing us to ensure the security of the on-board systems. These are usually done during vehicle production, for the initial set of rules and keys, and are only updated as and when there is a case of a newly discovered security threat.
(c) Firmware packages which include main system software that controls the underlying hardware. Thus, to both install and update these packages, a complete restart and re-flash of the ECU are required. Afterwards, the soft. must be fully tested and, if there are any errors, switched back to the previous firmware version through the use of techniques such as dual banking. This form of updates is used in Actuators and Sensors.
(d) Software packages including the installation of application components. These packages may contain the whole software or a partial/incremental software chunk, also known as Delta (Δ) software packages. Note that the size of the partial updates is typically close to the aforementioned Firmware packages. It is also important to bear in mind that for both full and delta software packages, the installation process must be performed when the vehicle is shut down. This process must also be tested afterwards and allowed to roll back if the system does not operate properly following the update. SOTA usually takes place over Unix-like systems, typically in infotainment or telematics ECUs.
(e) Media file packages, which include some multi-media files such as Global Navigation Satellite Systems (GNSS) maps, custom images, sounds, or videos for the In-Vehicle Infotainment (IVI). Note that these updates are considerably heavier than those described above. However, some academic proposals [413], [414], [415] suggest various solutions to decrease their size by splitting them into more-periodic incremental packages or by narrowing the package content for the user interests, e.g. downloading only your country for GNSS, only your language for audio/video/images etc.

Moreover, it is also worth noting that one of the key challenges facing software development in the automotive industry is the need for elevated levels of reliability and safety. Automotive software must be able to perform reliably under a wide range of conditions, including extreme temperatures, humidity, and vibration. Thus, multiple mechanisms are integrated during the development phase to match these strong safety requireents.

2) SOFTWARE DEVELOPMENT METHODOLOGIES

Traditional or sequential software development methodologies have been in around the industry over the last few decades [182], [184], [185]. These models, as we said in § IV-C, base themselves on a clear statement: all processes involved during the development of a product are phase-to-phase dependent on each other, needing to end th previous process before starting the next. These proposals also ensure to offer detailed documentation after each of the phases to enhance the action traceability. From traditional dev. methodologies, we can highlight two; the first one is the V-model, which is a classic sequential model, whereas the second one, Rapid Application Dev. (RAD) model is in-between the classic sequential models and the modern agile philosophy.

V-model [182] (cf. Fig. 16), where V stands for verification and validation, is one of the most widespread industrial development methodologies thanks to its simplicity and robustness for projects where the initial set of requirements stays static, which has generally been the case for traditional industries until recently. However, even if the V-model has evolved through the years, nowadays as there are so many different variants we are going to focus on a standard classic variant. V-model is an extension of another classic model, the waterfall model [183], in which the progress of a project is seen as flowing steadily downwards through the phases of conception, initiation, analysis, design, implementation, testing, and maintenance, and it proposes a review after each phase so as to evaluate whether to continue or discard the project. Further adding to this, V-model adds product testing in parallel with each corresponding phase of dev. This way, the V-model still benefits from a fast set-up and management simplicity due to the rigidity and previously fixed specific deliverables and review process from the waterfall model. Parallel and exhaustive testing through each of the phases enhance the chances of success of the project, while simultaneously saving a lot of time which allows the dev. teams to re-adapt code that has already been tested and validated, in turn avoiding the downward flow of defects. However, the flaws in long/complex projects from these models, in which requirements may evolve over time, have a negative impact on the adaptability and evolvability of the software and limits the adaptability and innovation capacity desired by the evolution of the users and business requirements.

On the other hand, one of the first proposals we have that works through a multi-stage software development methodology is RAD [186] (cf. Figure 16). The RAD development process starts then by splitting the complete set of requirements into sub-modules, this way each cycle becomes smaller and easier to manage. Also, in this model, the first functional version of the software will come with the first module, with all subsequent releases viewed as new functionalities. This allows the client to evaluate the viability of the solution sooner, create new functionalities that become essential at different moments, etc. In this model, each iteration is thought to take up to 2 or 3 months. Furthermore, this software development model shall only be applied if the project is

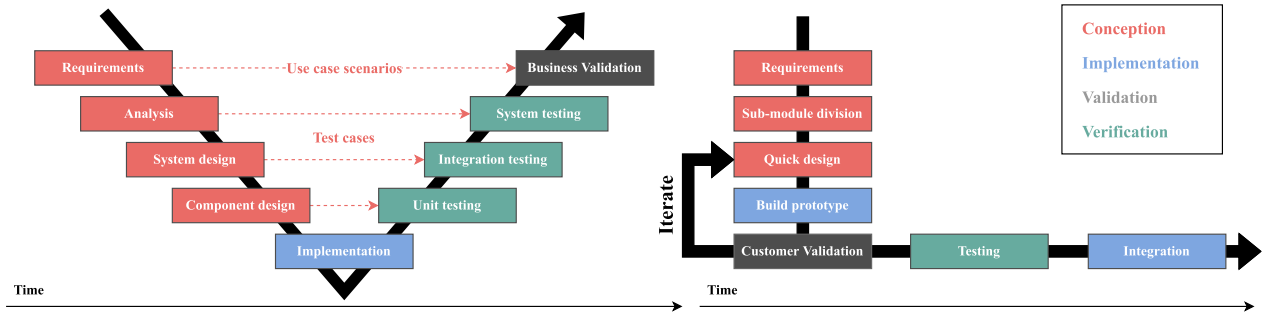


FIGURE 16. The V-model & the Rapid Application Development (RAD) model basic conceptual schemes.

divided into exceedingly small and independent sub-modules. This way it will reduce global development time, increase the reusability of the components, and reduce the chance of rollback and integration issues. However, this model needs developers with high-level planning, modelling, and developing skills, who are able to reduce time and costs through automated code generation and software reuse. This solution is not considered suitable for parallel development nor teams with a high number of staff. Nonetheless, it brings interesting opportunities with regard to software reuse and incremental development, both of which are key points for increasing innovation pace, all the while reducing development costs.

In 2001, and as s a way to unify all the aforementioned methods and develop a common and more flexible paradigm, a group of developers released the Agile Manifesto and, with it, the Agile development model, which is once again based on Incremental model logic. In Agile's approach to software development, work is carried out in small phases, based on collaboration, adaptive planning, prompt delivery, continuous improvement, regular customer feedback, and frequent redesign, all of which result in the development of software increments that are delivered in successive iterations in response to the ever-changing customer requirements. Within the models deriving from the Agile manifesto, we can find several methodologies including Extreme Programming (XP), Scrum, Kanban, Lean, FDD (Feature-Driven Development), Crystal, DSDM (Dynamic Systems Development Method) etc. However, in this paper we are only going into detail about the first three, which in turn, are the most widespread of them all.

Scrum [188], is the most common representative of the agile-based software development methodologies. It focuses on offering high project dynamicity and total flexibility all the while keeping the resources, budget, and delivery constraints low. To achieve this flexibility, Scrum signifies having a cross-functional team where every person contributes towards the best design solution and defines a clear set of roles so as to split the tasks efficiently. These roles are as follows: the Product Owner, who is responsible for defining, prioritising, and communicating the project requirements and its evolution; the Scrum Master, who manages the day-to-day team interactions and removes impediments to development and helps improve the process, development team and software product being developed; and the Development team, which is responsible for executing the tasks allocated within the deadlines. However, to maintain control and to fully profit from the team and client feedback, Scrum proposes splitting the development into small periods (which shall be the in the form of two weeks) called Sprints after which a new version will be deployed and a meeting with high-level managers and clients is set so as to evaluate and correct any necessary details as fast as possible. Also, in order to reduce futile meeting time, and as the planning is done by the PO, Scrum proposes 15-to-30-minute structured reunions on a daily basis in which the product development team members communicate and evaluate the progress status of software development and briefly discuss any blocking points or obstacles. However, it also has some drawbacks such as the lack of a definitive end-date, which often leads to scope creep, an elevated risk of failure if the individuals are not committed or cooperative enough, or the frustration of the development team when a task is blocked for a lengthy period pending client approval. Unfortunately, these drawbacks can compromise the speedy and efficient advance of the project as a whole.

As the objective is to adapt the Agile Manifesto to smaller / less cross-functional teams, XP [187], this bases itself on simplicity and continuous iterations with the clients. In this methodology, requirements are presented as scenarios by users then split into a series of small tasks. As the scenarios are extremely specific, XP aims to implement where strictly necessary for each feature, making it larger when required by the customer's needs. For that, the XP development process starts by developing and agreeing on the acceptance tests through feedback loops and only starts the real development once this is done. However, the cost effectiveness of this programming methodology relies on well-defined processes, version management and constant, clean, and understandable coding and refactoring. Finally, with the current objective being the enhancement of the task workflow visualisation and limiting the parallel work in progress during software development processes, we have Kanban [189]. This methodology aims to facilitate the delivery of software products just-in-time, and therefore maximise productivity.

In short, influenced by the IT sector and attractive flexibility, dynamicity, cost effectiveness, adaptation to constant user and incremental client feedback, and the fast development capacities brought by the agile manifest, automotive industry software development methodology is abandoning little by little traditional solutions. However, in order to fully profit from these upcoming opportunities, it once again raises questions about the capacity of the vehicles to dynamically adapt in a safe and secure manner, as well as their ability to run all these new-coming features on highly restrained and heterogeneous embedded nodes ECUs without exploding the complexity of the system.

B. DELIVERY

Nowadays, software deployment in the Automotive sector is highly conditioned by the place in which this software deployment takes place, with different mechanisms and constraints existing whether it takes place in production after-sales, or in maintenance garages [416], [417], [418]. Furthermore, this deployment process takes place through different means depending on the size and safety impact of the packages to be installed. On the one hand, software and configurations for small software factory/garage deployment can be deployed physically, through the OBD-II port [419], [420], [421], or through the USB port for bigger-sized packages both in production, garages and after-sales. On the other hand, for all low-to-medium sized packages, this deployment can also be done directly and remotely with Over-The-Air (OTA) [422], [423], which is currently being implemented by most Automakers as it allows for the saving of millions of dollars, minimising repair delays, and reducing the environmental impact deriving from update campaigns. However, even though until now, the OTA update frameworks are not systematically used for all update any size and safety issues they may be caused if dealt with incautiously must be considered, studies continue to show [1], [424] that OTA importance will likely keep growing in the near future. In this subsection, we will begin by briefly going through the deployment possibilities and limitations of physical deployment means, - i.e., OBD-II and USB deployment, as they are already widely used by all automakers and, thus, overly complicated to change due to the economic impact that it would imply. Subsequently, we are going to focus hard on the state of the art of Over-The-Air software delivery, which is currently the main challenge faced by software deployment.

1) PHYSICAL SOFTWARE DEPLOYMENT FRAMEWORKS

Even though The standard OBD-II port was first conceived to gain access to on-board diagnostics, it still allows automakers and other third parties to develop software updates, and even software embedded dongles, which interact safely and naturally with in-vehicle systems. Even though these packages that are to be installed rely on, as they also will do for most of the cases both in USB and OTA deployment, certificates, basic encryption libraries and a pre-loaded set of keys to secure this software transaction and prevent most of the attacks [324], [421]. Physical updates have some interesting capabilities as they allow, prior to having a full on-board system up and running and configured, the deployment of basic software and configuration, including the initial set of proprietary keys, which enables the rest of the software to have a secure update. Moreover, even though the OBD-II port data rate is considerably low, USB port allows us to bring in big Media file packages [425], [426], [427], such as GNSS maps, in an easier way than if we were using unstable OTA connections. It is worth noting that, OBD-II is nowadays considered to be the preferred technology for on-production software/configuration deployment, since only a really basic set of configurations are deployed in this phase due to time restrictions (cf. Fig. 17) and given the economic cost of changing the safety/quality control tooling already implemented and deployed in all factories and garages. Thus, in order for OTA deployment frameworks to be successfully integrated into the chain, they need to preserve, at least for now, the standards allowing the use of pre-existing OBD-II-based diagnostic/safety tooling.

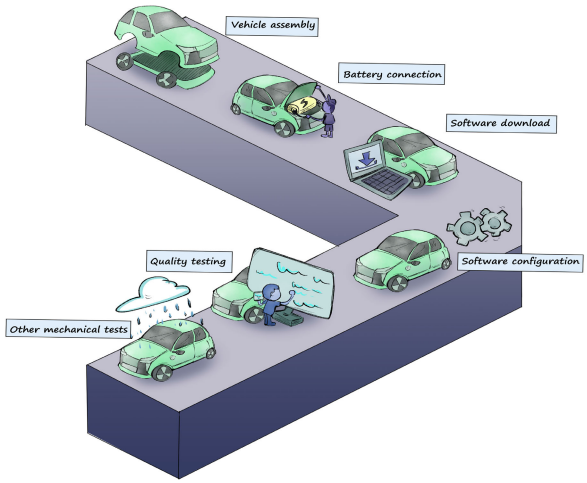


FIGURE 17. Factory software delivery process.

2) OVER-THE-AIR SOFTWARE DEPLOYMENT FRAMEWORKS

OTA software deployment frameworks are an attractive attack surface for malicious users, as there is no need to directly access the vehicle hardware as you needed to do with the physical software deployment frameworks. These malicious users can operate multiple attacks on OTA software deployment frameworks, such as the classic Denial of Service and man-in-the-middle attacks [290]. It is worth noting that, if malicious software is introduced within a vehicle, it will be a critical threat for the passengers' safety and, potentially, a vector to access other devices connected to the cars, such as smartphones, Bluetooth headsets etc., which in turn will have an enormous impact on the passenger's privacy, or even their economy. In this subsection, we will delve into the

state-of-the-art OTA deployment frameworks and evaluate them according to package security, authenticity, and integrity preservation.

The first group of solutions guarantees the authenticity of data using symmetric encryption, asymmetric encryption, or both. In these solutions, the integrity of the data is reliant upon decryption. Interesting examples that represent symmetric key-based OTA frameworks include Mahmud et al. [300] and Mansour et al. [293]. In both studies, a secure software update framework is detailed, based on sharing an initial set of link keys among automakers, vehicles, and software suppliers, which is then used for encrypting both software and communications. In addition, they propose other mechanisms to enhance security and complex transmission traceability, such as time-hopping randomisation [300], or detecting potential errors and malicious behaviour, such as remote diagnostic tooling [293]. However, even if the computing requirements needed to perform the encryption are low, as are the set of keys is directly included in the vehicle by automakers, the impact of a key being compromised, thus making it impossible to link a message to its sender directly, poses a significant threat that does not match the requirements of safety and security-critical frameworks. In addition, neither implements a content integrity verification mechanism, therefore making it impossible to detect maliciously modified packages if an authorised key is compromised.

Hence, to solve the problems linked to the risk of compromising a static set of symmetric keys and to improve message traceability, Steger et al. [294] propose a solution in which an asymmetric key is used to secure unicast communication, in addition to a symmetric multi-cast key from the service centre to several cars, thus enabling parallel updates. In this case, the only keys that will be shared are the public keys, making identity fraud difficult. However, this solution is again at potential risk of key dangers since the triggering action is centralised and there is no distributed network to ensure software package authenticity

Similarly, the approach proposed by Mayilsamy et al. [297] involves combining asymmetric encryption and a well-known cryptography field, steganography. Their study proposes a solution that integrates software files encrypted by an asymmetric encryption algorithm (RSA in this case) hidden along the edge region of the cover image of the update using steganography. This self-verifiable stego-image would then be adequate for safe storage and transmission. However, the danger of long RSA keys (2048 bit in this case) being compromised remains an open challenge, and the costs and storage needs associated with using stego images are also unsuitable for the highly restrained nodes within the automotive industry.

Further considering the integrity of package content instead of heuristic solutions for security during the transmission, we are going to outline discuss hash-based solutions. Based on this technique, Nilsson and Larson [295] propose a secure OTA firmware update protocol for connected vehicles based on dividing software and then hashing and encrypting each chunk. However, this division and hashing process appears inefficient in terms of computing and energy consumption compared to using software packages as a whole. Nilsson et al. [296] propose an alternative infrastructure in which a trusted portal calculates the hash of the whole software package and places it at the end of the message so the receiver can check the veracity of the message. However, in both approaches, having a centralised authority in charge of distributing keys is highly vulnerable to attack, a single point of failure, and identity usurpation attacks. Within the same scope as [296], we can consider the approach proposed by Kuppusamy et al. [290]. This solution secures key storage at a lower level by using Secure Hardware Module technology to handle key management. Their study also proposes an OTA framework that distributes updated software to ECUs in the form of images (containing collections of code and data) and metadata (containing image-related files such as the size of the file, the image hash, creation date, author, etc.). In addition, Kuppusamy et al. [290] suggest having different keys introduced in the hardware to verify the encryption of different files. However, this solution is vulnerable to rollback attacks due to a lack of proper verification mechanisms during software update installation. This system also suffers from the same issue as those in the previously discussed approaches of being centralised rather than distributed.

Through optimising hashing solutions by adding distributed middleware and some new optimisation mechanisms, some studies propose using blockchain-based solutions. In the blockchain, software packages are linked to each other immutably and spread through the different nodes integrating the system; in this case, it is always possible to use a simple majority vote approach to detect malicious or erratic introductions. Thus, this technology guarantees data integrity, complete traceability, and higher consistency and security than the aforementioned techniques. However, there are also some drawbacks-for example, blockchain-based solutions do not allow, or barely allow, for the modification of past data, rely on the secrecy of private user keys, require substantial amounts of storage and significant computational resources (for the Proof of Work and other cryptographic mechanisms). On top of that, their relationship regarding future legislation and regulations remains uncertain. Nonetheless, in recent years, increasing efforts have been expended to develop new techniques which reduce the resource consumption of this solution and allow its deployment in the IoT world, such as pruning and new, less resource-consuming proofs.

In the automotive sector, multiple papers have proposed the use of blockchain-based OTA update frameworks. However, most of these, such as Steger et al. [298], are based on the Proof-of-Work mechanism, while others such as Witanto et al. [428] and Mtetwa et al. [429] focus on other parts of the implementation such as peer-to-peer exchange or transmission details, respectively. However, the resource consumption constraints of embedded vehicle architecture suggest that Proof-Of-Work algorithms are unsuitable for such systems, hence indicating a key limitation to these

TABLE 9. Comparison of the over-the-air frameworks found in the state-of-the-art.

		Integrity & authenticity	For trust-less dynamic environments	Secure	Resource consumption	Traceability	On-board storage needs
Symmetric or Asymmetric key based (CA-like)	S. Mahmud *et al.* [300]	Very low	No	Very low	Low	No	Very Low
	K. Mansour *et al.* [293]	Very low	No	Very low	Low	∼	Very Low
	M. Steger *et al.* [294]	Very low	No	Low	Low	No	Very Low
	K. Mayilsamy *et al.* [297]	Medium	∼	Medium	High	∼	High
Hash-based	D. K. Nilsson *et al.* [295]	Medium	No	Medium	High	∼	Medium
	D. Oka *et al.* [296]	Medium	No	Medium	Medium	∼	Medium
	T. K. Kuppusamy *et al.* [290]	Medium	∼	High	Medium	No	Medium
Blockchain-based	M. Steger *et al.* [298]	High	Yes	High	Very High	Yes	Very High
	E. N. Witanto *et al.* [428]	High	Yes	Very High	Very High	Yes	High
	N. S. Mtetwa *et al.* [429]	High	Yes	High	Very High	Yes	High
	D. Fernandez Blanco *et al.* [430]	Very High	Yes	High	Medium	Yes	Medium

∼ : More or less compliant.
Note that this comparison is theoretical and done analyzing the papers and claims.

proposals. Additionally, all the cited blockchain solutions follow traditional public blockchain schemes, which we consider unpractical in the automotive software development context, where the blockchain publishers are varied and with different interests (i.e., companies that need to use this chain as a mean to sell their software, internal automaker developers…). Therefore, other solutions such as Blanco et al. [430] take these factors into consideration and focus on reducing the computing, energetic and storage needs of blockchain-based software deployment frameworks by proposing a proof-of-authority based multi-provider collaborative software deployment framework that is both public to publish and privately managed by the Automakers.

C. DEPLOYMENT

As the software has already been safely deployed in the vehicles by this stage, in this section we are going to focus on the software deployment/install process. Firstly, we are going to look at the Firmware deployment process, then, less specifically at a single hardware, the OS deployment process and, finally, making an abstraction of the hardware resources below, the software virtualisation deployment process.

1) FIRMWARE

Even though Firmware started, mostly for MCUs, as a simple combination of a hardware device and some computer instructions and data residing in the read-only memory (ROM) of the hardware device [431], nowadays, firmware is stored in the Electrically Erasable Programmable Read-Only Memory (EEPROM), which is divided into program and data memory, which, in turn, enables it to be updated after being deployed. With regard to the normal operation cycle,

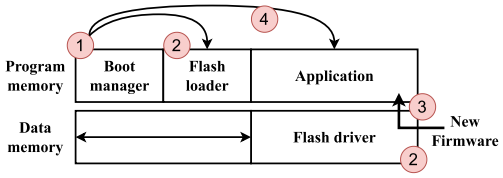


FIGURE 18. Schematic view of the Firmware flashing process.

traditionally, the program memory contains an application to run a flash-loader which contains the instructions on how to re-flash a new application to the application memory, and, finally, a boot manager, which will determine whether it is operating in flash-loader mode or application mode, always choosing application mode if a valid application is available. Furthermore, the data memory will contain a flash-driver agent which will be able to coordinate a new installation or update with the flash-loader. Thus, when wanting to install a new firmware or update the already installed one, as you can see in Fig. 18 the boot manager will start on flash-loader mode, which will trigger the flash-driver. Once the flash-driver is active it orchestrates the installation of the new firmware application with the application memory. Once installed, it will re-boot in application mode. It is worth noting that if the node has no EEPROM memory, or if it is too small, the flash-driver will be loaded onto the Random Access Memory (RAM) or directly with the flash-loader [431], [432], [433].

Moreover, in the automotive sector, as firmware usually handles safety-critical components, some techniques need to be implemented over the traditional mechanism. The first technique we want to focus on is the Secure Boot

feature whose objective is to increase protection tampering attacks. This feature complements the mechanisms presented in the § VII-B by loading an initial set of keys to the ROM memory that are only known by the Automaker and used to guarantee software integrity [434], [435], [436]. Furthermore, with the objective now of reducing installation time and increasing safety by always keeping a backup of the previous firmware version, we have A/B, also called dual-banking, mechanisms. This is when the ECU contains two identical partitions so that one can be updated and tested while the other is running. This helps to make the process as seamless and problem-free as possible. This way, once the firmware is installed and tested in the idle partition, the car only needs to restart the MCU to boot up the new application memory, while leaving the other one as a backup [436], [437]. Finally, the last extra-mechanism that we want to highlight concerns the delta-firmware updates [436], [438], [439], which are used to reduce the size of the patch, consuming less bandwidth and storage within the cars. The former is extremely important considering that MCUs are mostly connected through low data-rate CAN networks nowadays (cf. § VI-A). In delta updates, differential compression algorithms, such as bsdiff [440], rsync [441] or xdelta [442], are used to create a patch that only contains the part of the software that has changed. Even though these advancements speed up and enhance the security and safety of firmware flashing, it is always necessary to, at least, restart the MCU, which is not possible to be done during run-time, which, therefore becomes a complicated factor for in-vehicle dynamicity and flexibility. In addition, firmware updates are, by definition, highly coupled to hardware and thus, considerably harder to reuse. Moreover, firmware installation does not support, to the best of our knowledge, multiple simultaneous applications deployed at the same time.

2) OPERATING SYSTEMS

Operating systems, similar in many ways to Firmware, but more complex and less hardware specific, are a set of software that normally runs in kernel mode, providing the application developers with a clean abstract set of resources that can be exploited and used to manage hardware resources associated with enhancing the capabilities of the devices. Appearing in the 1950s for mainframes, their evolution and flexibility has led to a complete palette of choices depending on system constraints (processing, memory management, networking…) and preferences (real-time support, security and safety, power consumption…). In this part, we will mainly focus on the Operating Systems that are able to run in resource-constrained devices, which therefore excludes classic PC Operating Systems such as Ubuntu or Windows and the server-oriented OSs. This, combined with the aforementioned automotive requirements and limitations, will orientate our research towards those deployed in (1) IoT / sensor nodes, (2) embedded systems or (3) automotive nodes, while also carrying out a survey of those focusing on (4) real-time. In order to compare the different systems with each other, throughout this section we will be bearing in mind certain aspects such as OS architecture and kernel, scheduler characteristics, programming model, languages permitted, real-time support, memory management techniques, security and safety evaluation, power consumption, and multimedia support (Audio/Video). An extensive ranking of all the OSs mentioned in this section is detailed in Table 10.

a: IoT / SENSOR NODE OPERATING SYSTEMS

Sensor nodes and IoT are small-powered computers with built-in radio, suddenly deployed in high quantities that operate simple calculus and possess poor memory capacity. These systems run small, but complete, operating systems that normally follow event-driven programming models in order to respond to external events or to periodically take measurements based on internal interruptions. The major problem concerning this system is how to extend the battery lifetime as much as possible because of the high number of sensors and the difficult physical access for substitution. Thus, these kinds of OS are simple and small, implementing sleep-awake periods and other energy efficiency techniques. Among these solutions we can include TinyOS [443], Contiki [444], MantisOS [445] and LiteOs [446].

First, TinyOS is an open-source non-Linux based OS that was explicitly designed for this use case: low-end IoT devices. This solution has a monolithic kernel and uses a component-based arch. that depends on the requirements of the application. This reduces the size of the code needed to set up the hardware. In addition to that, this solution follows an event-driven programming concurrency model which consists of split-phase interfaces, deferred computation, and asynchronous events. TinyOS also implements other techniques for updating and efficiently scheduling soft. on demand. However, this solution implements poor security mechanisms to keep the OS overhead to the minimum.

Following the same logic of lightweight non-Linux OS, we have the Contiki option: a modular architecture with an unchangeable kernel that allows the app. modules and subsystems, which will make up the Contiki system, to be downloaded. In addition to that, it adds a system of polling to check update availability, some slightly safer security mechanisms (TLS, DTLS available) and the possibility to support both event-driven programming models and multi-threading. However, the energy consumption of this solution is slightly higher than TinyOS.

After that, regarding a cross Linux/Windows environment solution, we can turn our attention to MantisOS. MantisOS is a microkernel-based solution that does not follow the event-driven programming model it (only allows multi-threading). Adding more security compared with the past two solutions and various semaphores to deal with the inter-thread comm. (which implies higher energy cons.), MantisOS presents itself as an interesting solution for applications with more sensitive data (or with higher criticality). However, none

of these solutions comply with the norms without adding extra security and safety mechanisms.

Finally, we have LiteOS, which is the most used Linux-based solution over sensor nodes. This Operating System was conceived to provide an UNIX-like environment for IoT developers and to supply programmers with familiar programming paradigms such as a hierarchical file system developed using LiteC and a UNIX-like shell. Thus, just as with linux, it has multi-threading capabilities and a higher security and safety level (complying with ASIL D) than its predecessors. The best part of this solution is its power consumption, which is rather similar to TinyOS.

b: EMBEDDED SYSTEMS OPERATING SYSTEMS

Embedded systems are computer systems that have a dedicated function within a larger mechanical or electronic system. These kinds of systems usually don't allow users to install new software and have strong real-time computing constraints. In addition to that, as these systems form part of a larger system, safety and security is sometimes required. Thankfully, as user software installations are not allowed, the possibility of having untrusted software running on your system is reduced, which acts as the first security barrier. In this type of OSs, we should focus onQNX Neutrino [447], WindRiver VxWorks [448], [449], Embedded Linux (uCLinux) [450], Android Things [451] and Raspbian [452].

To begin, uCLinux is an open-source Linux-based OS that extends the classic Linux kernel to enable its operation in micro-controllers without MMU (memory management unit). As in the classic Linux, uCLinux follows a monolithic architecture to which you can add different CPU platforms and file systems that are more adapted to embedded solutions. At the same time, uCLinux supports multi-threading programming, a wide set of networking and communication protocols as well as using a priority-based preemptive scheduler to manage call executions. However, the security mechanisms that uCLinux implements are not completely secure and the energy consumption of the Operating System could be lower, despite the kernel sleep mode possibility.

Then we have Raspbian, Android Things and QNX Neutrino, all of which are based on the classic Linux Kernel. Despite their similarity to uCLinux, all these solutions need to have, or benefit strongly from having, an MMU to operate under. Raspbian is a solution conceived for RPi hardware that has lower energy consumption and higher security while operating on their specific boards. However, when it operates over other kinds of hardware, its performance and energy consumption are mostly similar to those of uCLinux. With even safer security mechanisms, we have QNX Neutrino. QNX Neutrino is a solution focused on the security and reliability for the automotive and medical industry. However, this linux-based solution is less optimal in terms of energy consumption, but it is compliant with the automotive ISO 26262 D level without needing any modifications.

Finally, we have the leader of the smartphone market, Android who have presented Android Things to conquer the OS ecosystem of the embedded system by offering the development tools, the wide android store, and the strength of the big Android developing community. This port of Android manages to keep power consumption low, even lower than the solutions that have already been presented, despite the security of the system which was already representative of the Smartphone Android OS. Finally, as the only remarkable non-Linux embedded OS, we have WindRiver VxWorks. Widely used by the aeronautical, automobile and telecommunications industry, this system is an alternative if you need a real-time safe (ASIL D) operating system without an excessively high energy consumption.

c: AUTOMOTIVE OPERATING SYSTEMS

Currently targeting those Operating Systems designed or adapted to recognise the Automotive on-board node needs and use-cases, we can find Autosar Classic OS [453] and Autosar Adaptive OS [454], [455]. These target devices that don't require visual interaction with the user and Windows Embedded Automotive 7 [456] and Android Automotive OS [457] for the vehicle entertainment system. For starters, nowadays Autosar Classic is the most widespread OS choice for the OEMs, which are progressively transitioning to Autosar Adaptive. Autosar Classic has a decent energy consumption and lots of interesting features that have been developed over the years to adapt to upcoming automotive needs. However, the biggest features missing from Autosar Classic, and that are already covered by Autosar Adaptive, are the file mgmt. system, the security mechanisms, and the lack of compatibility with the incoming Linux libraries. To conclude, as a way to conquer the IVI systems in the vehicle, Android (Android Automotive OS) and Windows (Windows Embedded Automotive) both launched their OS for entertainment in the automotive industry. However, since Android had already won the smartphone market and the Google Play Store brings a lot of new apps and developers, it also won the automotive market, leaving windows embedded automotive as a dead-end project. As you can see in Table 10, it is worth noting that even though some of the aforementioned technologies have not yet been targeting the automotive systems, Automakers have managed to implement them in the on-board systems.

d: REAL-TIME OPERATING SYSTEMS (RTOS)

As their name implies, these Operating systems focus on the real-time reactivity of the nodes. Many of these are found in industrial process controls, avionics, military, and similar application areas. Thus, these systems must provide absolute guarantees that a certain action will occur by a certain time. Even if real-time constraints will also appear within some of the embedded system and automotive OSs, these are not as precise time-wise as RTOS. In this category, we can find various solutions focusing on low-end

devices such as FreeRTOS [458], RIOT [459], Mynewt [460] and others that focus on medium/high-end devices such as Nucleus RTOS [461], Green Hills Integrity [462] and TizenRT [463].

As one of the main representative OSs for low-end device RTOS, FreeRTOS is an open-source non-Linux real-time OS for low-end devices. It supports an extensive variety of hardware architectures, which makes it an excellent choice for heterogeneous environments. This solution is based on a rather small micro-kernel that supports multi-threading and includes a preemptive scheduler. Therefore, the small size of the kernel makes this solution very scalable, simple, easy to use, highly portable and with an incredibly low energy consumption. However, the security of this Operating System relies only on a lightweight OpenSSL substitute: WolfSSL, which means it is indirectly adapted to the automotive environment requirements without extra development work. However, with the appearance of the SafeRTOS extension, it gains ASIL D security and safety pre-certification, making its adaptation to the automotive industry much easier. In the same scope, we have RIOT which is another open-source non-Linux based OS. RIOT supports many interesting functionalities such as hardware abstraction, interruption handling, memory management, IPC and good reliable synchronisation, performance, scalability, and an energy consumption similar to FreeRTOS. On the other hand, if we are searching for an open-source Linux-based real-time representative solution, there is Apache Mynewt OS. This solution aims to bring a linux environment (and the developing advantage that this implies) to low-end devices that have limited memory and storage capabilities and that need to operate for a long time under power constraints. With the same level of functionalities, the consumption of this solution is similar to the ones aforementioned, however, the granular power control mechanism that it implements allows us to reduce consumption a great deal when some functionalities are not useful for the application. However, this solution has poor security capabilities that must be reworked and redesigned in order to bring this solution to the automotive world.

If we consider RTOS for higher-end IoT devices, we have Siemens Nucleus RTOS. This OS was designed for industry (medical, aerospace, automotive…) and thus matches all the security requirements imposed by the ISO 26262 D level. It also offers interesting features such as a power manager, 64-bit support, support for heterogeneous computing multi-core System on a Chip (SOC) processors, and some domain partitioning and isolation techniques. All of this makes this OS an interesting solution for the automotive high-end IoT nodes. In the same range, we have GreeHills Integrity OS. This Linux-based solution offers scalable run-time environments with secure partitions, safe (ASIL D), secure embedded multi-core virtualisation, fast boot and advanced development tools that lower development costs and reduce the time to market. Finally, we have Samsung's solution: Tizen (for high-end devices) and TizenRT (for low-end devices). Tizen was designed and introduced by Samsung for their mobile devices, wearable devices, and smart TVs. Therefore, the energy consumption and multimedia capabilities of the OS are remarkably interesting for the automotive industry. However, as they are not designed for a safe-critical domain, the security capabilities are far from optimal.

To sum up, if we look at Table 10, we can see that most of the operating systems matching both the consumption and security requirements are Linux-based, which helps to decrease the development complexity. For the low-end sensor nodes on the system LiteOS, if real time is not required, FreeRTOS and Autosar Adaptive show themselves to be the most complete solutions. However, due to its compatibility with Autosar Classic and all the current infrastructure, Autosar Adaptive seems like the best choice for, at least, the transition to a more centralised architecture. If we look now at higher-end solutions, we have more choice. Nucleus RTOS (if a non-Linux system is preferred), Integrity OS, VxWorks and QNX Neutrino are safe and secure Operating Systems designed for the automotive industry. All of them will be adequate and the choice will depend mostly on the pricing and support that the companies are willing to offer to the OEMs. On the other hand, Android Automotive OS is an interesting solution for the IVI since the application store and development community is by far the most active. However, its safety is lacking, and it needs extra development for its conception errors to be compensated.

3) SOFTWARE VIRTUALISATION

Thus, in order to make the installation and development of new applications more flexible, dynamic, reusable, and simple, we usually put the applications over an Operating System, as they are no longer firmware applications but software applications, which is already the case for the Automaker MPU nodes. Even though we usually use firmware for the MCUs due to their restrained resources, there are other possibilities as the micro-kernel OSs presented above in Table 10, such as RIOT, MantisOS or FreeRTOS. Moreover, when installing the software over an operating system, there are other mechanisms on top of this to increase system flexibility and adaptability even more. The most common way to do this is virtualisation, which presents itself as a technology able to combine or divide computing resources. This allows them to present one or many operating environments using methodologies like hardware and software partitioning or aggregation, partial or complete machine simulation, emulation, time-sharing etc. In other words, visualisation allows a single ECU of any kind to run sets of code independently and in isolation from other sets. Note that virtualisation is already a critical feature in other sectors such as cloud computing or artificial intelligence and provides higher resource orchestration, easier environment maintenance, less expensive sand-boxing, the ease with which to maintain multiple execution environments with low resource

TABLE 10. Exhaustive comparison of the different operating systems analysed.

Operating System	Architecture	Scheduler	Programming model	Real-time	Dyamic memory	File System	ASIL compliance	Energy Consumption	Multimedia support	Linux-based	For automotive	Targeted device
TinyOS	Monolithic	Cooperative	Event-driven	✗	∼	Single level	-	Low	✓	✗	✗	Low-end
MantisOS	Microkernel	Preemptive	Multithreading and semaphores	✗	✓	Single level	-	Medium ↘	✓	✗	✗	Low-end
Contiky	Modular	Cooperative	Multithreading and event-driven	✓	✓	Single level	-	Low ↗	✓	✗	✗	Low-end
Huawei LiteOS	Modular	Preemptive	Multithreading	✗	✓	Hierarchical Unix-like	ASIL D	Low ↘	✗	✓	✗	Low-end
Amazon FreeRTOS	Microkernel	Preemptive	Multithreading, mutexes and semaphores	✓	✓	Hierarchical	ASIL D (SafeRTOS)	Low	✓(static image only)	✗	✗	Low-end
RIOT	Microkernel	Preemptive	Multithreading	✓	✗(kernel) ✓(Apps)	Hierarchical	-	Low	✓	✗	✗	Low-end
Apache Mynewt	Modular	Preemptive	Multithreading	✓	✓	Hierarchical Unix-like	-	Low	✗	✓	✗	Low-end
Autosar Classic	Monolithic	Preemptive	Multithreading	✓	✓	Hierarchical	-	Low	✓	✗	✓	Low-end
Autosar Adaptive	Microkernel	Preemptive	Multithreading	✓	✗	Hierarchical Unix-like	ASIL D	Low ↗	✓	✓	✓	Low-end
uClinux	Monolithic	Preemptive	Multithreading	✓	✓	Hierarchical Unix-like	-	Medium ↗	✓	✓	✗	High-end
Nucleus RTOS	Monolithic	Preemptive	Multithreading	✓	✓	Hierarchical	ASIL D (SafetyCert)	Medium	✓	✗	✓	High-end
Green Hills Integrity OS	Microkernel	Preemptive and cooperative	Multithreading	✓	✓	Hierarchical Unix-like	ASIL D	Medium ↘	✓	✓	✓	High-end
Tizen	Modular and Monolithic	Preemptive and cooperative	Multithreading	✓ (TizenRT)	✓	Hierarchical Unix-like	-	Medium ↗	✓	✓	✗	High-end
Raspbian	Monolithic	Preemptive	Multithreading	✓	✓	Hierarchical Unix-like	-	Medium (on raspberries)	✓	✓	✗	High-end
Android Things	Modular	Preemptive and cooperative	Multithreading	✗	✓	Hierarchical Unix-like	-	Medium ↘	✓	✓	✗	High-end
QNX Neutrino	Microkernel	Preemtive	Multithreading	✓	✓	Hierarchical Unix-like	ASIL D	Medium ↗	✓	✓	✓	High-end
WindRiver VxWorks	Monolithic	Preemtive	Multithreading and semaphores	✓	✓	Hierarchical Unix-like	ASIL D	Medium	✓	✓	✓	High-end
Windows Embedded Automotive 7	Hybrid kernel	Preemtive	Multithreading	✓	✓	Hierarchical Tree Structure (NTFS)	-	Medium ↗	✓	✗	✓	High-end
Android Automotive OS	Modular	Preemptive and cooperative	Multithreading	✓	✓	Hierarchical Unix-like	-	Medium	✓	✓	✓	High-end

consumption, techniques to facilitate the deployment of apps multiple systems, etc.

Even if most of the visualisation techniques present similar advantages and environments to the end user, the levels of abstraction in which they operate, and the underlying architecture tend to vary extensively. Therefore, depending on the abstraction in which they operate, the virtualisation techniques can be classified, as set out in Table 11, into the following: (1) virtualisation at the instruction set architecture level, (2) at the hardware abstraction layer, (3) at the operative system level, (4) at the library level and (5) at the application level. Finally, we will also address the virtualisation evolution perspectives over (6) MCU nodes.- In addition, as you can see, the layer in which the virtualisation is applied implies a quite different performance for the solution, being able to identify patterns within all the solutions at each level.

a: VIRTUALISATION AT ISA LEVEL

ISA Level virtualisation consists of emulating an ISA completely with middleware. Then, an emulator will transmit the instructions to a set of native instructions and then execute them on the available hardware. This technique presents advantages when dealing with multiple platforms since there is no binding between the guest and the host operative system. However, the performance of this solution is far from optimal. Among the virtualisation solutions that come from this logic, Bochs [464], Crusoe [465], QEMU [466] and Bird [467] all stand out. It is worth noting that the inefficiency of this solution comes from the overhead added by the system call translator, with each of the aforementioned solutions having diverse ways to approach this problem. The studies show that QEMU and Crusoe are those with the highest level of efficiency.

b: VIRTUALISATION AT HAL LEVEL

These solutions try to exploit the similarities between the architectures of the guest and host platforms so as to cut down on interpretation latency. Thus, this technique helps to map virtual resources to physical resources and uses the native hardware for computations in the virtual machine. This mapping and the resource management process is managed by a control software called Hypervisor. There are multiple ways to implement HAL virtualisation, however, in this paper, we will focus on the two most widespread: Full-Virtualisation, in which the guest OS are fully independent, both from the rest of the host and guest OSs, and are built either on the top of the physical hardware (a - Type I or Bare-Metal Hypervisor) or on top of an existing OS (b - Type II or Hosted Virtualisation), and (c -) Para-virtualisation, in which the guest OS kernel must be modified to act as a bridge between the different apps. and the hardware resources.

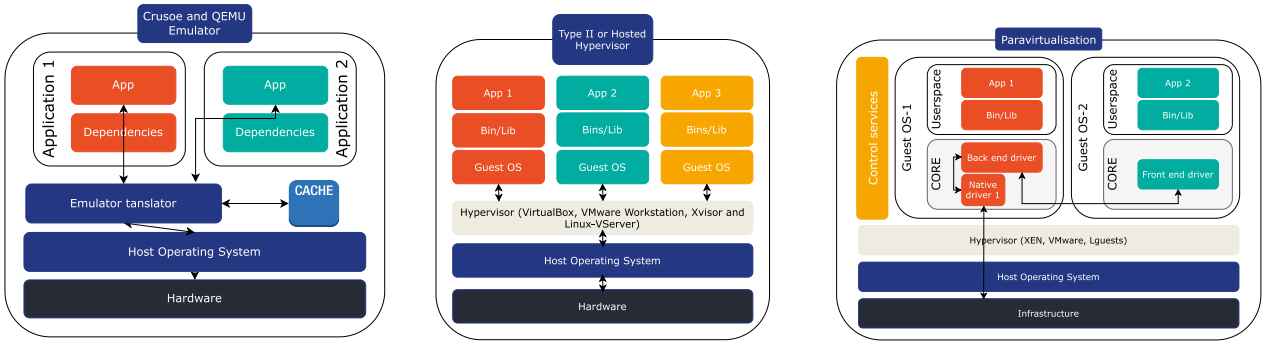
TABLE 11. Exhaustive comparison of the different virtualisation solutions.

			Security		Performance				
Layer	Technique	Solution	Safety Possibilities	Isolation	Efficiency	Implementation complexity	Flexibility	Scaling	Boot Time (overhead)
ISA Level	Emulation	Bochs	Low↗	Medium	Very Low	Medium ↗	High ↘	Medium ↗	High
	Emulation	Crusoe	Low↗	Medium ↗	High	Medium ↗	High ↘	Medium ↗	Medium
	Emulation	QEMU	Low↗	Medium ↗	High	Medium ↗	High ↘	Medium ↗	Medium
	Hybrid Emulation and Direct Hardware Execution	BIRD	Medium	Medium ↗	Medium↗	High	High↘	Medium	Medium ↗
HAL Level	Type I Hypervisor	Linux KVM	High	High ↗	High ↗	Medium	Medium	Medium	Low
	Type I Hypervisor	VMWare ESXi	High	High ↗	High	Medium ↗	Medium	Medium	Low
	Type I Hypervisor	XEN	High	High ↗	High ↗	High	Medium ↗	High ↘	Low↘
	Type I Hypervisor	Hyper-V	High	High↗	High↗	Medium↗	Medium	Medium	Low
	Type II Hypervisor	VirtualBox	Medium↗	Medium↗	Medium↗	Medium ↘	Medium	Medium ↗	Medium ↘
	Type II Hypervisor	VMware Workstation	Medium↗	Medium↗	Medium↗	Low↗	Medium	Medium ↗	Medium↗
	Type II Hypervisor	Xvisor	Medium↘	Low↗	Medium↗	Medium	High	High↘	Medium↗
	Type II Hypervisor	Linux VServer	Medium↗	Medium↗	High↗	High↘	Medium↗	Medium ↗	Low
	Para-virtualsation	XEN (ParaVirt)	Medium	Medium↘	High↗	High	Medium↘	High	Low
	Para-virtualsation	VMware (ParaVirt)	Medium↘	Medium↘	High	Medium ↗	Medium↘	High	Low
	Para-virtualsation	Lguest	Medium↘	Medium↘	High	High↘	Medium↘	Medium↗	Low
OS Level	Containerisation	Docker	Low	Low↗	Very High	Low↗	Very High	Very High	Very Low
	Containerisation	Linux Containers (LXC)	Low↗	Low↗	Very High↘	Medium↗	Very High	Very High	Very Low
	Containerisation	Linux Containers (LXD)	Low↗	Low↗	Very High↘	Medium↘	Very High	Very High	Very Low
	Containerisation	CoreOS Rocket (RKT)	Medium	Low↗	Very High	Low↗	Very High	Very High↗	Very Low↗
	Containerisation	Windows Containers	Low↗	Low↗	Very High↘	Medium	Very High↘	Very High↘	Very Low↗
	Containerisation	Podman	Medium	Low↗	Very High↗	Medium	Very High	Very High	Very Low
	Zones	Solaris Containers	High	High	High	Low↗	High	Medium↗	Low↘
	Jails	Jails	Medium↗	High	High↗	Medium	High	High	Low↗
Library Level	Linux to Windows	WINE	Low	Low	Low↗	Low	Medium↘	Low	High
	Linux to Windows	WABI	Low	Low	Low	Low↗	Low↗	Low	High
	Linux to Windows	MainWin	Low	Low	Medium	Low↗	Low↗	Low	Medium↘
	Inter Linux	LxRun	Low↗	Low	Medium↗	Medium↘	Medium	Medium	Medium
App. Level	Java	JVM	-	Low	Medium	-	-	-	Medium
	C / C++	Microsoft .NET CLI	-	Low	Medium↗	-	-	-	Medium↘
	Perl	Parrot	-	Low	Medium↗	-	-	-	Medium↘
MCU Virt.	Full-Virtualisation inspired	ARM TrustedZone-M	High	Medium	High↘	Medium↗	Medium	Medium	Medium
	Full-Virtualisation inspired	PRPL HV	Medium	Medium	Medium↗	High	Medium	Medium↘	Medium
	Containerisation inspired	Femto-Containers	Medium↘	Medium	High	High	High	Medium↗	Medium↘

(a) *Type I or Bare-Metal Hypervisor.* In Type-I hypervisors, the hypervisor is placed directly over the hardware. It is small as its main task is sharing and managing hardware resources between the different OS running in the machine. The major advantages of this solution are that it achieves a performance at almost 100% of its capabilities and that any existing problem in one of the different guest OS running on the hypervisor will never affect any other guest. We are going to analyse and consider the following as representative solutions to this model,Linux KVM [468], VMware ESXi [469], XEN [470] and Hyper-V [471].

(b) *Type II or Hosted Virtualisation.* In Type-II hypervisors, the hypervisor runs over a host Operating System instead of directly over the hardware. Even though it runs over a host OS, the hypervisor can support other OSs above it. This technique is easier and faster to configure and allows for better specification policies. However, the performance of these kinds of hypervisors is slightly slower than Type I (while still performing better than emulators). Another disadvantage of this technique is that the guest OSs are dependent on the host OS for its operations and thus, any problem with the host will affect all the other hypervisor guests too. We are going to review the following as representative solutions to this model: VirtualBox [472], VMware Workstation Player [469], Xvisor [473] and Linux-VServer [474].

(c) *Para-virtualisation.* In para-virtualisation, as previously stated, the guest OS kernels need to be modified so as to function as a bridge between the other guest OSs, the applications, and the hardware resources. In that way, guests are able to recognise the presence of the para-virtualisation hypervisor and communicate through it with the hardware resources in a more efficient manner. To enhance this efficiency,



(a) ISA Level virt. (Crusoe/QEMU) **(b)** Type II Full-Virtualisation example. **(c)** Para-virtualisation example.

FIGURE 19. Virtualisation examples.

the hypervisor will distribute control over the different highly requested hardware materials (hard disk, network interface, CPU…) among all the guests. Therefore, by installing a set of drivers, the guest OS that has control of a component (in this case OS-1) is able to efficiently redistribute and schedule the call executions from the other concurrent guests (in this case OS-2). Para-virtualisation technologies are more efficient and performant than any of the aforementioned virtualisation techniques. However, their efficiency comes from creating new intra-guest dependencies and reducing the isolation between concurrent operative systems, therefore creating new hardware / software coupling constraints and possible security vulnerabilities. Multiple projects such as XEN or VMware have para-virtualisation solutions. However, since they share the same logic concerning the solutions presented before, by adding the concept of system call redistribution they all see their performance increase as well as their intra-guest isolation decrease. Another interesting para-virtualisation solution is Lguest [475]. This offers a lightweight hypervisor built into Linux kernel and that does not provide any fancy features that other hypervisors do, but it does allow for an easy development and test environment.

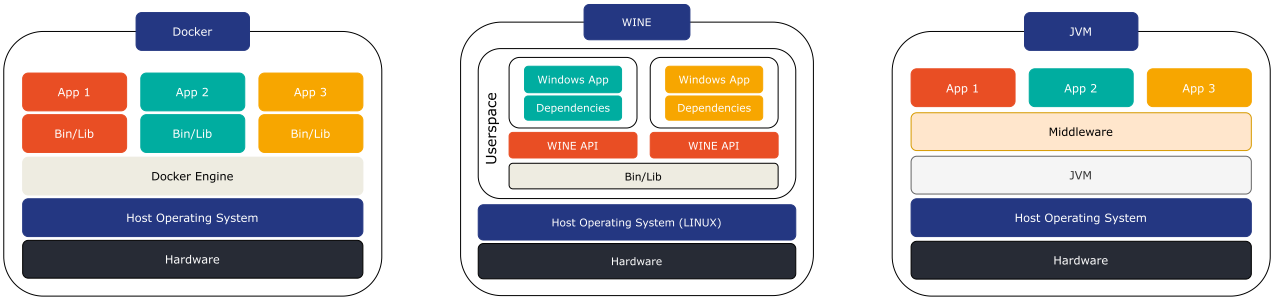
c: VIRTUALISATION AT OS LEVEL

If we now review virtualisation the Operative System Level, these solutions tend to have a high degree of isolation, as well as support for different OSs and applications without requiring rebooting and can carry out complicated dual boot setup procedures with minimal risk and easy maintenance. In OS level virtualisation, the kernel allows for the existence of isolated user space instances. Such instances can be Containers, Zones, Virtual Private Servers, virtual environments, virtual kernels or jails. A program running inside one of these instances will only be aware of the content and devices assigned to it and not all the resources available in the whole machine.

If we focus on the main implementation, Containers run over a host OS (Linux-based normally) which needs a containerisation control engine. Contrary to the classic hypervisor-based solutions, this technique does not need the guests to run their own OS but it is the host kernel that allows for multiple isolated user-space instances to which the necessary libraries and bins are added. This is much more performant, with a lower boot time while being considerably more lightweight. In addition to that, this technology implements diverse isolation mechanisms, resource-management features that limit the impact the containers have on the others, network abstraction mechanisms etc. This technique allows for the execution of mono-functional apps. in isolated environments at a reasonable resource cost. However, containerisation is less flexible than full virtualisation since you cannot have different host OS and guest OS for your apps. Finally, with regard to security, it is a misconceived thought that containers ensure security boundaries like VMs do. Containerisation is more like packaging and delivery mechanisms and the security of the system relies not on the containers but on the host OS. Containerisation is a trending subject these days and therefore, multiple, and heterogeneous solutions appear. The most widespread solutions are are Docker [476], Linux Containers (LXC & LXD) [477], [478], Rkt [479], [480], Windows Containers [481] and Podman [482].

d: VIRTUALISATION AT LIBRARY LEVEL

This technique relies on the principle that applications are programmed using a set of APIs exported by a group of user-level library implementations. Such libraries are designed to hide the OS related specific details so as to keep it simpler for programmers to use. Thus, in this technique, virtualization is done above the OS layer by producing a different virtual environment through exposing different but equivalent binary interfaces to be able to emulate the application binary interfaces and application program interfaces needed to run an application. Examples of these applications. are WINE [483], WABI [484], LxRun [485] and Visual MainWin.



(a) OS Level virt. (Docker) (b) Lib level virt. (WINE) (c) App level virt. (JVM)

FIGURE 20. Virtualisation examples.

e: VIRTUALISATION AT THE APPLICATION LEVEL

In this technique, the idea is to create a virtual device at application level that can behave like a machine to a set of applications, just like any other machine. This virtual machine includes registers, stack…and the byte code that the application requires to run together with this abstract machine. Subsequently, the abstract machine will interpret the compiled byte-code into machine instructions. By using this technique, applications can be run in multiple different machines and distributions with the same expected behaviour. In other words, these applications are aligned with the philosophy of Write Once Run Anywhere. The main representative technologies of these virtualisation techniques are JVM [486], Microsoft .NET CLI [487] and Parrot [488]. These three technologies profit from the compiling process to generate files containing VM commands that run the code for Java (JVM), C++ (.NET CLI) and Perl (Parrot). As these machines are very finely grained, the performance is close to the natively compiled program if used correctly but it can considerably slow the performance if not developed according to the recommendations.

f: MCU VIRTUALISATION

All the aforementioned techniques consume, each one in its own measure, a high amount of processing time due to the need to virtually emulate hardware. That, in turn, poses a problem for the current automotive embedded architecture, mainly composed of low-cost micro-controllers. Until now, in these resource-constrained environments, the partitioning of mixed-criticality systems has been mainly implemented through federated architectures (by adding a physical separation of several subsystems across different MCUs). However, with the exponential rise in functions, this physical separation becomes impractical and inefficient in terms of size, weight, power, and cost. Therefore, the concept of virtualization becomes interesting to ensure safe and security-critical functions that operate at a lower complexity and cost. Hence, the increasing demand of MCU-based virtualization is leading the academic researchers and industrial developers to put significant effort into developing lightweight virtualization solutions. This way, we can find some interesting solutions such as, ARM TrustZone Assisted Hypervisor [489], [490], PRPL-HYPERVISOR for MIPS micro-controllers [491] and Femto-Containers [492].

D. TESTING

As more safety-critical car functions rely on software (i.e., self-driving functions, V2X…), testing the new upcoming applications in real live traffic has become dangerous and has already caused fatalities. Thus, in this context, exhaustive virtual testing offers a more cost-efficient and safer alternative compared to operational tests [502]. This virtual testing can be done in multiple ways, however, for this paper, we are going to focus on three of them. We are going to rank them from the most to least decorrelated with regard to the vehicle environment, and they are as follows: shadow-mode testing, sand-boxing, and model-based simulations.

First of all, model-based simulations allow for the development of high-level test models than can be used from the early stages of the development process, being able to integrate not only the software constraints, but also the electrical and mechanical aspects entwined. These models allow us to find a common functional understanding and validation early in the design phase, all the while improving the communication within development and engineering teams, reducing the time-to-market through component reuse, and strong validation prior to the implementation. Model- based simulation provides a development process from requirements-to-code, ensuring that the implemented systems are complete and behave as expected. Some examples of these testing techniques are [493] and [494]. However, as the number of apps and variants rise, creating suitable complete test scenarios is becoming increasingly laborious and difficult to ensure that they are complete in all situations. Nonetheless, to deal with this complexity, a recent research trend suggested using automatic test-case generation based on search-based procedural contents [502], [503] or machine learning analysis from real situations [504], [505], [508]. If we look closer at model-based automatic testing in the IT sector, we believe it is fundamental to highlight the possibilities that exist that help reinforce learning as a way to continuously revisit the tests, adapting granularly to the incoming applications and

TABLE 12. Summary of the testing mechanisms.

Mechanism		Objectives	Conception & Development	Reinforcement	Post-install
[493], [494]	Model-based testing simulation	Development of high-level test models to be used in early stages of the development process, integrates multi-disciplinary constraints.	✓	∼	✗
[495], [496], [497], [498]	Sand-boxing	Creation of automotive-like environments to ease the development and testing in real conditions.	✓	✓	✗
[499], [500], [501]	Shadow-mode testing	Introducing the software to be tested directly in the car but in an isolated environment.	∼ (late dev.)	✓	✗
[502], [503], [504], [505]	Automatic test-case generation	Study all the possibilities through past-data analyis to establish a *full coverage* testing procedure.	✓	✓	✗
[506], [507]	Validation & verification	Running the pre-generated testing procedures and asserting the correct behavior of the vehicle.	✗	∼	✓

· ✓ :*Used in this phase*, ✗ :*Not used*, ∼ :*Not the main technique for this phase but potentially usable*.

consequently enhancing the on-board system safety and security [509], [510].

Secondly, the objective of sand-boxing is to create a close-to-reality software environment, making it easier to both test and develop new software, integrating these environments much easier with classic CI/CD tooling such as [511], [512], [513]. Even though sand-boxing does not seem to have the same importance now in the automotive systems as it does in the IT sector, there are some interesting examples in literature [495], [496], [497], [498]. Moreover, sand-boxing will not only help to ease the development and testing of new applications but also to make the automotive sector easier to access for new players. However, just like it did for model-based simulations, the incessantly increasing number of applications and variants makes it more and more complex to cover all the test cases within the sand-boxing environment, taking longer and longer to test all the available possibilities.

Finally, shadow mode testing, first introduced by Tesla on their vehicle fleet, and which relies heavily on virtualisation tools. Shadow mode consists of introducing a software to be tested directly in the car, but in an isolated environment, in which this software can collect all the necessary inputs of all the software in the environment without intervening in the operation of the car [514]. This way, the software tested registers the actual behaviour and what the system may have reacted under real-case circumstances. Thus, considering the magnitude of the car fleets, this mechanism allows us to rapidly cover most of the different application combinations, behaviours, and software contexts, while also detecting errors that could not have been foreseen otherwise. Moreover, these statistics that have been gathered can later be used to refine the simulation testing phase or as part of the previously presented reinforced learning mechanisms. Even though, this technique has barely been explored until now [499], [500], [501], if compared with the precedent techniques, we believe it can be a key-mechanism to lessen future automotive integration and development complexity.

On the other hand, addressing not how the testing criterion is defined but if the software has been successfully integrated into the in-vehicle systems, the verification & validation mechanisms mainly consist of [506], [507], and [507], simply following a testing procedure, which is compliant with the safety and security certifications, that comes with the installation package. In addition, this testing is not only done after installation but also throughout the control service § VIII over time to detect any potential unexpected failures, allowing us to bring the car to a safe-state mechanism, even if this state usually implies a temporary reduction of the functionalities. It is worth noting that this process is similar for testing software in factories, after-sales, and on the road.

E. ORCHESTRATION

Having already gone through how to safely deploy, install and test the in-vehicle software, the next step, and the one we are addressing in this section, is how to decide into which of the ECUs in the system each application shall be deployed. Contrary to the precedent sections, in which each application could be treated separately, orchestration must be evaluated from a global point of view in order to guarantee that we individually fulfill, in the most optimal way, all application resource requirements (i.e., computing, storage, bandwidth…) but also collectively all the functional latency and safety requirements for every functionality [515]. For example, when we deploy an ADAS function, such as the Anti-lock Braking System (ABS), the function dealing with this functionality needs to be able to command braking under extremely low-latency constraints, which might not be possible if the distance to the software dealing with brake management is too far away. In brief, it can be said that the overall performance of in-vehicle systems critically depends on the different service resource allocation.

Until now, as the number of applications to be installed was very restrained and the update / new functionality delivery periodicity followed long-time periods, this application orchestration was done by the Automaker's workforce during the design phase [216]. They directly assigned the software components to a certain computing unit and then testing and certification of the correct operation of this given mapping

was carried out. Furthermore, this mapping was manually revisited by the Automakers when a new functionality had a significant impact on the system. However, revisiting and certifying this software mapping study for each variant or significant system update is time-consuming and, as the number of applications and functionality delivery periodicity rise, cannot be maintained over time. As you can see in Fig. 21, the inclusion of dynamic context-dependent functionalities, such as the V2X functions [516] punctually used and depending on the driving situation, has meant that the traditional approach of statically orchestrating hardware and network resources is no longer optimal for all the different vehicle states and environmental variations. It is important to note that this is even more noticeable if we think of more futuristic cases such as seamless inter-vehicle cloud assisted architectures [517], [518], [519] in which any vehicle could be requested to run another function of any other vehicles at any given time. Therefore, in this context, it seems that going from static orchestration to dynamic orchestration should solve the dynamicity and variability issues, as was already the case for the Cloud Computing sector.

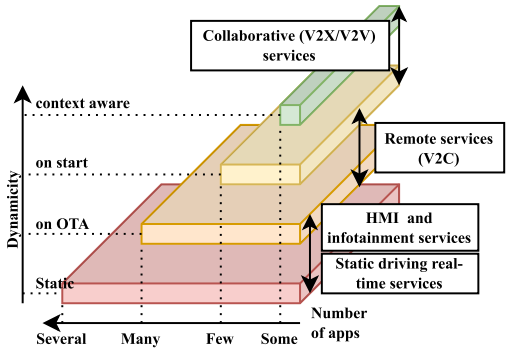


FIGURE 21. In-vehicle applications dynamicity levels.

On the one hand, we begin with those proposals targeting the orchestration of resources with networking as decisive criteria. First, Khalili et al. [520] study the radio resource allocation in a multi-cell orthogonal frequency-division multiple access (OFDMA) cellular network, focusing on locally minimising energy consumption through a variable relaxation, memorisation and minimisation (MM) method. On a similar scope, targeting once again a centralised infrastructure dealing with multiple parallel requests, Kuang et al. [521] investigate the joint problem of partial task offloading and resource allocation between mobile devices and the Edge Computing infrastructures with the objective of minimising the execution delay and energy consumption from the network point of view and proposes a multi-layered algorithm combining genetics and flow-shop scheduling for the upper layer and convex optimisation techniques for the lower level. If we take a more in-depth look at those contributions, targeting the Automotive sector, Li et al. [522] and Khan et al. [516] both address the issues of existing competition for comm. among mobile users when offloading tasks to the Edge, the first through a simplification and an auction based offloading algorithm, and the second through a multi-factor prioritisation function and a budgetary scheduler. With the same objective, but now within intelligent transportation systems, Mittal et al. [517] proposes a centralised multi-factor prioritisation mechanism that calculates the offloading orchestration based on vehicular movements along urban roads and their expected trajectory. Finally, and as the only proposal that we have found with reference to in-vehicle networks, Hu et al. [523] study a holistic scheduling problem for handling real-time apps. in time-triggered in-vehicle networks, proposing a new, highly-flexible scheduling algorithm called Unfixed Start Time, and, to compensate for the conflicts with both a rescheduling and a backtracking mechanism.

On the other hand, now we are going to focus on the proposals that use the computing characteristics of the nodes as decisive criteria. Note that, to the best of our knowledge, there are no automotive-specific contributions on this matter, but we can learn from the domain of cloud / edge computing. First, Xu et al. [524] propose a computation offloading method for cloud-edge computing in which a non-dominated sorting genetic algorithm is employed to address the multi-objective optimisation problem that will help reduce both the execution delay and energy consumption. Secondly, Yu et al. [525] consider the scenario where multiple mobile users offload duplicated computation tasks to the network edge, sharing the computation results among themselves to develop fine- grained collaborative offloading strategies with caching enhancements. This is done to minimise the overall execution delay on the mobile terminal side, through coalition game formation algorithms. Finally, Xu et al. [526] study service catching and replication using a mobile edge heterogeneous computing arch. through Lyapunov's optimisation and Gibbs's sampling to reduce the delay and spatial demand coupling of the systems.

Finally, we now focus on the contributions mixing both computing and network as decisive criteria. First of all, if we start with those not conceived for the automotive industry, the first, and in our opinion most interesting proposal comes from is Gholami et al. [515]. This proposes a framework for resource orchestration for micro-services based 5G applications in a dynamic, heterogeneous, multi-tiered computer and network fabrics by analysing the end-to-end application requirements. Moreover, Wang et al. [530] and Xing et al. [531] also target this issue, firstly through an alternating direction method of multipliers to reduce the computation complexity and, secondly, through a heuristic scheme based task assignment algorithm. Shifting our focus now to consensus and security, Guo et al. [528] use blockchain technology for adaptive resource allocation and computation offloading in future wireless networks, where the blockchain works as an overlaid system to provide management, offloading consensus, and control functions. Finally, between those specifically targeting the automotive industry, even though they target the external networking and not the in-vehicle systems, Vu et al. [527] address the

TABLE 13. Exhaustive comparison of the different dynamic orchestration & scheduling techniques analyzed.

Solution		Automotive centred	Computing	Networking	Storage	Algorithm	Optimising factor	Distributed	Node profiles	Safety contraints
A. Khalili *et al.*	[520]	✗	✗	✓	✗	Majoriaztion minimization method	Energy	✗	Edge & Mobile Devices	✗
Z. Kuang *et al.*	[521]	✗	✗	✓	✗	Genetic multi-layer algorithm	Delay & Energy	✗	Edge & Mobile Devices	∼ (delay)
L. Li *et al.*	[522]	✗	✗	✓	✗	Auction-based approach	Energy	✓	Edge & Mobile Devices	∼ (delay & inter.)
M. I. Khan *et al.*	[516]	✓	✗	✓	✗	Multi-factor prioritization and budgetary scheduler	Network congestion	✓	Inter-Vehicles	✓
M. Hu *et al.*	[523]	✓	✗	✓	✗	Earliest/Latest Start time and Rescheduling with Offset Modification	Quality-of-service	✗	In-vehicle	✓
S. Mittal *et al.*	[517]	✓	∼	✓	✗	Centralised multi-factor prioritization	Flexibility & Delay	✗	Inter-Vehicles, Edge and Central Cloud	✓
X. Xu *et al.*	[524]	✗	✓	✗	✗	Non-dominated Sorting Genetic Algorithm III	Delay & Energy	✗	Edge and Central Cloud	✗
S. Yu *et al.*	[525]	✗	✓	✗	✗	Coalitional Game Formation	Delay	✗	Edge & Mobile Devices	∼
J. Xu *et al.*	[526]	✗	✓	✗	✗	Lyapunov optimization and Gibbs sampling	Service heterogeneity, Spatial Demand Coupling & Delay	✓	Edge & Mobile Devices	✗
D. Vu *et al.*	[527]	✓	✓	✓	✗	Maximum Weight Matching	Computing & Networking	✗	Fog & Vehicles	✓
A. Gholami *et al.*	[515]	✗	✓	✓	✗	Multi-objective optimization problem	Computing and Bandwidth efficiency	✗	Edge, Central Cloud & IoT	✗
F. Guo *et al.*	[528]	∼	✓	✓	✗	Deep Neuronal Network	Performance	✓	Edge & Vehicles / Mobile Devices	∼
L. P. Qian *et al.*	[529]	✓	✓	✓	✗	Many-to-one matching algorithm	Delay & Cost	✗	Edge & Vehicles / Mobile Devices	∼
Y. Wang *et al.*	[530]	✗	✓	✓	✗	Alternating Direction Method of Multipliers	Quality-of-Service	✗	Edge & Mobile Devices	∼
H. Xing *et al.*	[531]	✗	✓	✓	✗	Heuristic-based algorithm	Latency	✗	Edge & Inter-Mobile Devices	✗
X. Wang *et al.*	[532]	∼	✓	✓	✓	Deep Neuronal Network, Federated learning	Performance	✗	Edge, Central Cloud & Mobile Devices	✗
Y. Zhou *et al.*	[533]	∼	✓	✓	✓	Alternating direction method of multipliers	Utility	✗	Edge and Central Cloud	✗

popularity and load balancing issues of Fog nodes depending on their location and contents. They propose a dynamic resource orchestration scheme which harmonises resource allocation through maximum weight matching for connected vehicles. This migrates the offloaded services among fog nodes depending on the computing and network resources available and the location of the fog node latency-wise. Secondly, Qian et al. [529] investigate the non-orthogonal multiple access (NOMA) enabled multi-access edge computing platforms, with the objective of minimising a system-wise cost that accounts for the overall delay in finishing total computation workload. Thirdly, making use of artificial intelligence, Wang et al. [532] use deep reinforcement learning techniques and a federated learning framework with the mobile edge systems to optimise the mobile edge computing, caching, and communication. Furthermore, Zhou et al. [533] tackle this problem where they once again use alternating direction method of multipliers algorithms.

VIII. RUN-TIME MANAGEMENT

Having addressed the challenges from both architecture and software delivery pipelines, we now need to focus on the services that ensure that the systems and all of the software deployed on them run smoothly. These services are called, as we mentioned in § VI-B, control services. However, as there is an infinitude of possible control services, we are going focus on those we feel are a priority to comply with the automotive requirements, mostly in terms of safety and security. Therefore, this section is composed of three parts; the first is about data-centred control services, the second is to do with security-centred control services and the last one focuses on safety-centred control services.

A. DATA-CENTRED CONTROL SERVICES

These control services allow for the collection, analysis, and distribution of data to optimise the performance, efficiency or even safety of the vehicle. In this section, first of all, we will go over the state-of-the-art data collection solutions proposed in both the automotive and IT sectors, then move towards how this collected data is shared and used. Finally, we will focus on how to safely store this data. Note that Table 14 summarises all the data-centred contributions listed in each subsection to give a graphic outlook of the section results.

1) DATA COLLECTION

The collection of vehicle data is a promising research field, being one of the main motivations for the increase of in-vehicle cameras, software, and sensors that gather valuable info. about cars and driving patterns. In this first subsection, we will be focusing on how this new data is being collected by vehicles, as well as the context in which it is being collected. Afterwards, we will briefly go through data collection in the IT sector to illustrate the rest of the panel choice.

To illustrate the advancement of these control services in the automotive sector, we focus on one of the most advanced data collection use cases: digital forensics, which targets the collection of data to enhance traceability and accountability of delinquent activities. Thus, in this context, we highlight Alexakos et al. [534], whose paper tirelessly goes through the challenges of data collection and security and proposes an attack attribution and forensics readiness tool for the IoV systems. For other purposes, we can find other interesting contributions such as Sheeny et al. [535], who target the collection of automotive sensors for weather profiling, Xun et al. [536] who propose a framework

TABLE 14. Summary of the solutions around data-centred control services.

Solution	Purpose	Data collection	Data sharing	Data storage	Automotive
C. Alexakos *et al.* [534]	Digital forensics	✓	✗	✗	✓
M. Sheeny *et al.* [535]	Weather profiling	✓	✗	∼	✓
Y. Xun *et al.* [536]	Driver profiling	✓	✗	∼	✓
B. Major *et al.* [537] & J. Kocic *et al.* [538]	ADAS	✓	∼	✗	✓
D. LLorca *et al.* [539]	Traffic monitoring	✓	∼	✗	✓
F. Giobergia *et al.* [540] & T. Tiedmann *et al.* [541]	Predictive maintenance	✓	✗	✗	✓
G. Beier *et al.* [165]	Environmental	✓	✗	✗	✓
M. Rahim *et al.* [542]	New application	✓	✗	✗	✓
Y. Liu *et al.* [543], T. Li *et al.* [544] & W. Mo *et al.* [545]	IoT sensor network applications	✓	∼	✗	✗
M.T. Nguyen *et al.* [546] & T. Tuor *et al.* [547]	Mobile sensor network applications	✓	✗	✗	✗
K. A. Khaliq *et al.* [548] & T. Wang *et al.* [549]	Fog/Edge/Cloud computing applications	✓	∼	✗	✗
Y. Jiang *et al.* [550]	System & application monitoring	✓	✗	∼	✗
M. Waltereit *et al.* [551], X. Feng *et al.* [552] & A. D. Sathe *et al.* [553]	Interactions with the cloud infrastructure	✓	✓	✗	✓
J. Xiong *et al.* [554]	User-data privacy	✗	✓	✗	✓
G. Rathee *et al.* [555] & Z. Su *et al.* [556]	System security	✗	✓	∼	✓
Q. Chen *et al.* [557] - [558] & Y. Fu *et al.* [559]	V2C applications	✗	✓	✗	✓
T. Hidayat *et al.* [560] or N. Eltayieb *et al.* [561]	Data encryption	✗	✓	✗	✗
M. Shen *et al.* [562] or C. Ge *et al.* [563]	Cloud computing applications	✗	✓	∼	✗
H. Jin *et al.* [564] & X. Cheng *et al.* [565]	Other safety critical applications (medical)	✗	✓	✗	✗
X. Du *et al.* [566] & B. Shen *et al.* [567]	Data location	✗	✓	∼	✗
H. Guo *et al.* [568], M. Li *et al.* [569] & N. Vinzenz *et al.* [570]	Data authenticity	✗	∼	✓	✓
A. Mohammad *et al.* [571]	Blockchain challenges for data collection	✗	∼	✗	✓
R. Jabbar *et al.* [572]	Blockchain for inter-vehicle comm.	✗	∼	✓	✓
F. Morano *et al.* [573]	Driver data	✗	✗	✓	✓
S. Jain *et al.* [574]	Future applications	✗	✗	✓	✓
H. Tabrizchi *et al.* [575], J. Wu *et al.* [576] & I. Odun-Ayo *et al.* [577]	Cloud computing applications	✗	∼	✓	✗
M. Wang *et al.* [578] & O. Khalaf *et al.* [579]	IoT system applications	✗	∼	✓	✗
R. Li *et al.* [580] & W. Liang *et al.* [581]	Blockchain-based storage	✗	∼	✓	✗
C. Gyhorodi *et al.* [582] & Z. Lv *et al.* [583]	Relational data storage	✗	✗	✓	✗
S. Bradshaw *et al.* [584] & J. Yang *et al.* [585]	Non-relational data storage	✗	✗	✓	✗

· ✓ :*Addressed*, ✗ :*Not addressed*, ∼ :*Partially addressed*. Focusing on data collection, data sharing & data storage.

to use data collection for driver profiling authentication, Major et al. [537] and Kocić et al. [538] focusing on using data collection for ADAS purposes such as the generation of cross-vehicle omni-view middleware for autonomous vehicles, Llorca et al. [539] who use data collection for traffic monitoring purposes, Giobergia et al. [540] and Tiedmann et al. [541] who propose using this data for some predictive maintenance frameworks and, finally, Beier et al. [165] who suggest making use of data collection for environmental purposes. Finally, Rahim et al. [542] summarise, in a more general way, how new services might interact with the data collection services and how they would access this data.

On the other hand, if we focus now on the proposals coming from the IT sector, data-collection is addressed for IoT in Liu et al. [543], Li et al. [544] and Mo et al. [545] and more generally for mobile sensor networks and distributed systems respectively in Nguyen [546] and Tuor et al. [547]. It is also extensively addressed regarding fog/edge/cloud computing in Khaliq et al. [548] and Wang et al. [549]. Finally, one solution of data-collection for monitoring purposes that is worth mentioning is from Jiang et al. [550], which proposes a push/pull/alarm monitoring framework for micro-services architecture, being able to cover both system and software.

2) DATA SHARING

As the number of connected cars grows, the amount of data produced will increase respectively. This will be used to make vehicles safer, more fuel-efficient, with better in-vehicle

navigation and with higher traffic congestion management capabilities. Moreover, this data is also expected to enable better diagnostics, maint. and monitoring, as well as a higher understanding of the security and safety issues which will help automakers detect and develop better services and options for their customers based on their real habits.

In this case, with regard to automotive systems, the solutions we found tackle this subject either by focusing on the privacy aspects involved in the data-sharing process, such as Xiong et al. [554], the security aspects of the sharing transaction, such as Rathee et al. [555] or Su et al. [556], or the interactions with the cloud infrastructure, such as Chen et al. [558] and [557] or Fu et al. [559]. Furthermore, within the last category, many solutions focus both on data collection and data sharing, such as Waltereit et al. [551], Feng et al. [552], and Sathe and Deshmukh [553]. On the other hand, with regard to the IT domain, we can highlight multiple surveys focusing on security through the data sharing process, which are either focused on encryption characteristics - i.e., Hidayat and Mahardiko [560] or Eltayieb et al. [561] - or on the use of blockchain technology with this purpose, both in cloud computing Shen et al. [562] or Ge et al. [563], or in other security-critical domains such as medicine - i.e., Jin et al. [564] or Cheng et al. [565]. Finally, we would like to highlight some papers that focus on the efficiency of data-sharing with regard to data placement and replications Du et al. [566] or Shen et al. [567].

3) DATA STORAGE

Having already detailed the immense value of data collection and sharing both intra-and-inter vehicles and the increasing amount of data produced, in this section we aim to discuss how to safely store this data, as it might be interesting for the development of future models, the creation of new business opportunities, etc. In this context, first of all we look at the study of automotive systems, Guo et al. [568], Li et al. [569], and Vinzenz and Eggendorfer [570] propose ways to securely store and preserve data authenticity through distributed storage systems such as blockchain [568]. Moreover, we can find many other data-storage proposals centred around the advantages of Blockchain when ensuring data authenticity and enhancing traceability in contexts other than digital forensics, such as Mohammad et al. [571] which surveys the blockchain challenges for data-collection in the automotive sector, Jabbar et al. [572] who propose a solution for inter-vehicle communication, Morano et al. [573] who suggest a solution for driver data collection or Jain et al. [574] who discuss which future directions this promising technology may take in the future. In addition, it is worth noting note that all the blockchain-based solutions presented before, such as those concerning software delivery (cf. § VII-B), could also be also included in this section with minor data-structure adjustments.

Finally, with regard to the contributions on the IT domain, many contributions tackle the data storage in cloud computing - i.e., Tabrizchi and Kuchaki Rafsanjani [575], Wu et al. [576] or Odun-Ayo et al. [577] - or in IoT systems - i.e., Wang et al. [578] or Khalaf and Abdulsahib [579]. Moreover, we can also point out many contributions concerning the form of storage either using the blockchain - i.e., Li et al. [580] or Liang et al. [581] -, classic databases - i.e., Győrödi et al. [582] or Lv et al. [583] - or non-relational databases - i.e., Bradshaw et al. [584] or Yang et al. [585].

B. SECURITY-CENTRED CONTROL SERVICES

Security has been shown to be one of the most desirable properties for automotive systems. Thus, in this section, we will discuss, firstly, two of the most important security-centred run-time control services - i.e., Access Control Policies and State Management. Afterwards, we will go on to discuss security framework evolutions and threats.

1) ACCESS CONTROL POLICIES

Access control policies are important in automotive in-vehicle architectures to ensure that only authorised users can access and use certain features and functions of the vehicle. These policies can help to guarantee the safety and security of the driver and passengers, as well as prevent unauthorised use or theft of the vehicle. Traditionally, the access control model can be divided into three categories: Attribute-based access control (ABAC), role-based access control (RBAC), and access control tech. based on usage control (UCON).

Firstly, if we focus on the solutions found in RBAC and ABAC models, which is the second most common in the automotive domain, we can find many proposals. Among the ABAC models, we can highlight; Kumar et al. [586], who propose a tree-based certificate access management framework for large-scale communications within connected vehicles, Kim et al. [587], who suggest an integration of ABAC models for AUTOSAR, Rumez et al. [588], who propose a general discussion of ABAC integration in automotive systems or Farooq et al. [589] who discuss the use of hardware for ABAC authentication through RFID chips. In addition, others also focus on introducing dynamicity properties to the ABAC models either for the new generation of rules, as proposed by Gupta et al. [590], or for the resolution of ABAC conflicts, as discussed in Asif et al. [591] and Lachmund and Hengst [592]. On the other hand, for the RBAC model, Veitas and Delaere [593] are a good representative of the contributions in this scope regarding data confidentiality, integrity, and availability with the objective of recovering an exact situation following the occurrence of an event or on demand. In addition to these, from the IT sector, we can highlight many surveys on this matter such as Ren et al. [594], Qiu et al. [595], Hu et al. [596] or Zhu et al. [597] or even certain papers focusing on the combination of both RBAC and ABAC as in Long and Yan [598] or Al-Alaj et al. [599].

On the other hand, those based on UCON are focused on regulating the usage of the resources, rather than just guaranteeing, or denying access to them. In a UCON access

control policy, access to a resource is granted based on whether the user's request satisfies a set of usage conditions, rather than just a set of static access control rules. This way, UCON policies allow the definition of more fine-grained and context-aware rules, guaranteeing higher flexibility and dynamicity. Some examples of UCON access control frameworks in the automotive industry are Terzi et al. [600], which discusses a way of dealing with identity management and authorization through the authenticity characteristics of blockchain, or Kaur et al. [601], Lupu and Lupu [602] and Chen and Yin [603], that propose using bio-metric parameters for access control. On the other hand, from the IT sector, we can highlight an exhaustive survey published by Guoping and Wentao [604], some contributions surrounding the use of blockchain for access control such as Maesa et al. [605], Ali et al. [606] or Tang et al. [607] and, finally, a variety of contributions regarding flexibility and dynamicity of the generation of complex access rules in Jajodia et al. [608] and Lachmund and Hengst [592].

2) STATE MANAGEMENT

State management in automotive in-vehicle systems refers to the process of managing and tracking of the different states or modes of the system as it interacts with the driver, passengers, and external environment. In simpler terms, it involves the control and coordination of contrasting functions, features, and components of the in-vehicle system. Some examples of states that can be managed in automotive in-vehicle systems include:

- Power states: These states are related to the power management of the electronic systems of the vehicle. For example, the system may need to manage the power state of the infotainment system, air conditioning system, or other subsystems based on battery level or user preferences. Some examples of this are shown in Wang and Gladwin [609] or Zhong et al. [610].
- Driver assistance states: In-vehicle systems may have various modes for driver assistance, such as adaptive cruise control, lane-keeping assistance, or emergency braking. State management is crucial in ensuring that these systems are functioning correctly and providing the right level of assistance to the driver. Some examples of this kind of state management can be found in Al-Saadi et al. [611], Sajadi-Alamdari et al. [612] or Iglesias et al. [613].
- Communication states: Many modern in-vehicle systems support communication protocols such as Bluetooth, Wi-Fi, or cellular networks. Managing the state of these communication systems is crucial for ensuring reliable connectivity and data transfer between the vehicle and external devices. Some examples of this can be seen in Joshi et al. [614] or Hontani and Higuchi [615].
- User interface states: The in-vehicle system may have different states for its user interface, such as menu navigation, input methods, or screen layouts. Managing these states is essential to provide a smooth and intuitive user experience. Some examples of this can be seen in Braun et al. [616] and [617], Urooj et al. [618] or Kern and Schmidt [619].

Therefore, in general, state management in automotive in-vehicle systems requires a combination of hardware and software solutions. This can involve sensors, controllers, and software algorithms that work together to monitor and control distinct aspects of the system. Some examples of more general approaches for state management, this time coming from non-specific IT solutions, can be found in Nabi et al. [620], Sharma and Santharam [621] or Ahmed et al. [622].

3) CYBER-SECURITY THREATS AND FRAMEWORKS

The enhanced connectivity and software capabilities of new generations of vehicle has widened the range of cyber-attacks, and this has become one of the major challenges for Automakers each time a new functionality is introduced to keep the system secure and able to guarantee the passengers' safety. Thus, multiple papers have researched possible attacks, exploits and vulnerabilities of future vehicles; Teichmann et al. [623], Cui et al. [624], Parkinson et al. [625], Humayed et al. [626], Buinevich and Vladyko [627], Gupta et al. [628], Sommer et al. [629] and Bazzi et al. [630]. Other papers such as Chowdhury et al. [631], Pisarov and Mester [632], Jang and Shin [633] and Pascale et al. [634] further extend this threat analysis while at the same time also proposing some countermeasures. Furthermore, cybersecurity is also a trending topic in the IT sector from which we can highlight; Zhuang et al. [635] tackling cybersecurity in smart grid systems, Shaukat et al. [636] using machine learning for enhancing system security, Nikander et al. [637] in agriculture, and the one which the closest to the automotive domain, Andrade et al. [638] in IoT.

C. SAFETY-CENTRED CONTROL SERVICES

The main control services targeting in-vehicle software safety are Fault tolerance and Load balancing, with both being extensively addressed in the Automotive sector, as well as in IT which was already required by the current systems. Moreover, if we consider safety with regard to passenger-basis, we also need to include system monitoring and diagnostics.

1) FAULT TOLERANCE

On the one hand, fault tolerance allows the systems to continue operating properly when one or more failures occur within some of its components. Fault tolerance has been generally researched in the Automotive sector in Cheng et al. [639] and Bhat [640], heterogeneous systems are focused on in Lex et al. [641], deep neuronal networks are detailed in Malekzadeh [642], a conceptual threat approach is looked at in Gräßler et al. [643], and, then specifically for the Autosar systems in Bhat et al. [644]. In addition, in the IT sector, it has been widely addressed concerning two similar sectors, such as cloud computing in Kumari and Kaur [645]

and Bharany et al. [646] and distributed systems in Gupta and Vaidya [647] and Xiao et al. [648].

2) LOAD BALANCING

On the other hand, load balancing is the process of distributing a set of tasks or net workload over a set of computing units or cables, with the aim of making their overall system more efficient. In IT, we highlight some static algorithms such as Round Robin, MIN-MIN or MAX-MIN, described in Kaur and Luthra [649] and Mesbahi and Rahmani [650], and some dynamic algorithms such as the honeybee foraging behaviour algorithm, throttled algorithms or ant colony algorithms, described respectively in Thapliyal and Dimri [651], Patel and Rajawat [652] and, Nishant et al. [653]. Load balancing is also surveyed in the Automotive sector between both vehicles and cloud (V2C) in Liu et al. [654], Majumder et al. [655], vehicles and vehicles (V2V) Wu et al. [656], and within a single vehicle architecture Ahmed et al. [657] and Syed et al. [658].

3) SYSTEM AND SOFTWARE MONITORING

Software and system monitoring in cars is an essential part of modern vehicle technology. With the increasing complexity of car systems and the growing importance of software in controlling these systems, monitoring has become a critical aspect of ensuring vehicle safety and performance. This topic has barely been explored within the automotive sector, with Yemelyanov et al. [659], Charette [660], and Hameed et al. [661] and [662] providing the most representative and traditional proposals. Moreover, other contributions spend time explaining how to use machine learning and artificial intelligence for software monitoring such as Norouzi et al. [663], Kuutti et al. [664] and Georgakos et al. [665] or discussing the effects of software aging Parnas [666] or Costa et al. [667]. On the other hand, from IT, we can highlight some papers about software monitoring in cloud computing, such as Aktas [668], Spring [669], Tamburri et al. [670] or Shao et al. [671], in IoT, such as Maksymyuk et al. [672] or Razzag [673], or even more safety critical domains such as Industry from Rakhmonov and Kurbonov [674] or He et al. [675].

4) VEHICLE DIAGNOSTICS

In modern cars, system diagnostics are typically performed by the onboard diagnostic (OBD) system. The OBD-II system uses a series of sensors and diagnostic codes to detect malfunctions and issues with various components within the car. When a problem is detected, the system generates a diagnostic trouble code (DTC) that can be read using a scan tool. This makes it easier for technicians to diagnose and repair problems with a car. OBD-II systems are typically located under the dashboard, near the steering column. Some examples of this are discussed in McCord [676], Malekian et al. [420] or Rimpas et al. [677]. Furthermore, and more recently, as a way to reduce the environmental impact of hardware repairs, to detect future problems sooner, and to speed up garage visits, many solutions explore how to carry out vehicle diagnostics remotely such as Mesgarpour et al. [678] or Singh et al. [679]. Others, extend this idea by introducing machine learning for analysing as detailed in Theissler et al. [680] or technologies such as digital twin in Bhatti et al. [681].

IX. DISCUSSION

Society and, more extensively, vehicles follow more connected, accessible, and flexible perspectives, increasingly integrating software-based functionalities, enhancements, and optimisations. However, as we showed in previous sections, adapting the architecture, delivery pipelines and run-time management services, without forgetting about the already existing base, has become not only a major necessity, but also a major challenge for the automotive sector. Thanks to these enhancements, vehicles will not only be more prone to innovation but also considerably less complex, easing the costs of maintenance, integration and, even, development.

The results presented in the earlier sections are numerous and diversified, covering different disciplines and topics. We understand how important it is to work on future in-vehicle and off-board ICT systems and it raises numerous open questions and challenges, including technical, societal, and economical issues. Our research provides us with the scope to complete this paper by give some engineering guidelines (cf. Fig. 22) about how to combine all the aforementioned sections, completing the results of the research in the previous Fig. 5 in §IV, which can be completed with Fig. 22 and to which we can comment layer by layer:

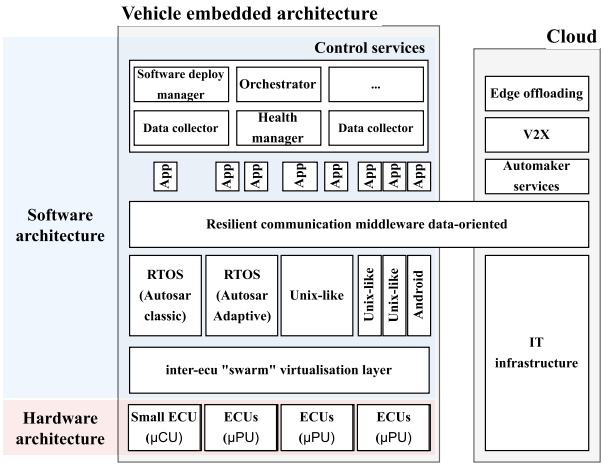


FIGURE 22. Simple overview of our vision, based on the survey results, of future vehicle embedded & cloud architecture.

Architecture Design: If we focus first on the E/E architecture, it seems clear that in the coming years there is a lot of work to go through in terms of how to organise the nodes within our architectures. It is an open challenge to decide whether it is better to go through zonal architectures or central architectures like Tesla's, as well as discussing the

fundamental issue regarding which kind of boards (i.e., GPU, HPCs…) to use for each case [66]. In addition, there are also various concerns regarding the integration and cohabitation of legacy and new software in terms of hardware, as it must be able to support both [231]. Finally, further research is needed on the analysis of other ISA level architectures (RISC-V, CISC, ARM…) to match safety, cost, and performance constraints [246]. Secondly, regarding software architecture, further work needs to be done on how to combine the different paradigms so as to fully profit from the advantages arising from all of them [682]. Moreover, the adaptation and implementation of an IT-like complete complex software architecture should be done so as to evaluate its performance and overhead for the automobile sector. Thirdly, with regard to network architecture, the key research point will be the scheduling and the respect of the latencies in a data-centred communication middleware [18]. Finally, IoV architecture needs to be explored much more, as most of the current contributions don't delve deep enough into the implementation and do not profit enough from the possibilities of integrating both smart cities and vehicles [683].

Software Delivery Pipelines: First, with the enhanced networking capabilities of vehicles, software delivery faces a new revolution, needing to integrate OTA within their classic software management cycle, and then focusing on how to ensure the safety, security and data authenticity becomes of vital importance, as the scope of this mechanism and its attack surface are both enormous is huge [430]. Secondly, further research is needed on software deployment so as to, first of all pick the appropriate Operating system and, secondly, choose the right installation method that will favor flexibility, isolation… [492]. Thirdly, testing faces many open issues with regard to dynamic certification and the testing process itself that also needs further work [506]. Finally, additional research needs to be done on how to orchestrate, both efficiently and dynamically, the in-vehicle applications depending on the software context and also the nearby users [518].

Run-Time Management Services: Firstly, with regard to data-centred control services, further research is needed to integrate the new data collected from the development of new services [534] and, as the data has high privacy implications, additional work is also needed concerning data privacy and authenticity [554]. Moreover, with regard to security-centred control services, the new technologies added to the vehicle have several implications concerning the security issues of the whole system, and thus, much effort has been made to find and solve [684] them. Finally, safety-centred control services need further integration with the modern technologies, in order to apply current knowledge to the new applications and cases [639].

A. FUTURE TRENDS

As the automotive industry is constantly evolving, there are several areas that are likely to see significant dev. in the coming years. Here are a few potential areas of focus for future work in the automotive industry.

1) ELECTRIC VEHICLES (EVs)

The transition to electric vehicles is already underway, but there is still a lot of work to be done to make EVs more affordable, efficient, and convenient. Future work in this area might focus on developing new battery technologies [141], improving charging infrastructure [685], [686], and increasing the range of electric vehicles [687].

2) AUTONOMOUS VEHICLES

Self-driving cars are becoming more common, but there are still technical and regulatory hurdles that need to be overcome before they become mainstream. Future work in this area might focus on developing more advanced sensors and software [688], [689], as well as working with regulators to establish guidelines for autonomous vehicles [690].

3) SUSTAINABILITY

As concerns about climate change and environmental impact grow, there is a growing focus on making cars more sustainable. Future work in this area might include developing more fuel-efficient engines [691], increasing the use of renewable materials in car construction, reducing the environmental impact of manufacturing processes [692], [693], and changing car sharing possibilities [160], [694].

4) PERSONALIZATION

Consumers are increasingly looking for cars that can be customised to fit their individual needs and preferences. Future work in this area might focus on developing more modular car designs that can be easily modified to meet specific customer requirements [695], as well as developing new interfaces and technologies that allow drivers to customise their driving experience [696], [697].

5) NEW USE-CASES

Furthermore, other work will be done on creating new services that interact with the current unexplored or barely explored smart city, such as:

- Predictive maintenance: By analysing data from various sensors and systems in a vehicle, software can help predict when maintenance is needed, reducing downtime and repair costs [698], [699], [700].
- Emergency response & Healthcare services: Vehicles can be fitted with emergency response equipment, such as defibrillators, oxygen tanks, and first-aid supplies, to create mobile emergency response units that can quickly respond to medical emergencies or other crises [701], [702]. This can be particularly useful in rural or underdeveloped areas where access to healthcare is limited.
- Tourism: Vehicles can be used to provide guided tours, either through traditional tour companies or through new

models, such as self-driving tour buses or mobile audio guides [703].

- Urban farming: Vehicles can be transformed into mobile urban farms, providing fresh produce to urban areas where access to fresh food is limited [704].
- Education & Mobile offices: Vehicles can be adapted to provide educational services, such as mobile classrooms or libraries, providing educational resources to underdeveloped communities [705]. Moreover, with with the rise of remote work, many people are looking for ways to work from anywhere. Vehicles can be fitted with workspace features, such as Wi-Fi, power outlets, and comfortable seating, to create mobile offices that allow people to work while on the go [706], [707].
- Delivery services: As e-commerce continues to grow, there is an ever-increasing demand for delivery services. Vehicles can be adapted to provide last-mile delivery services, either through traditional delivery companies or through new delivery models, such as autonomous vehicles [708].
- Entertainment: Vehicles can be fitted with entertainment features, such as high-end sound systems, large displays, and gaming consoles, to create mobile entertainment centers that can be used for events, road trips, or even as mobile movie theatres [709], [710].

X. CONCLUSION

Software in the automotive system is becoming increasingly important and is a major factor for clients when it comes to deciding which vehicle to buy. However, even if until now the automotive industry has kept up with the innovation pace in terms of functionalities offered to the passengers, much effort has been made to bring about these new functionalities by adding an unceasingly larger set of ECUs without evolving all the embedded software architecture consequently due to budgetary constraints, legislative limitations, retro-compatibility problems, or lack of awareness regarding trending IT innovation. This unbalanced progress has then, as we can see in § IV, considerably increased the complexity and cost of the system for developing and maintaining new services, hardening the path for evolving the application development methodology and bringing about new innovative software-related business proposals.

Our paper has provided a comprehensive overview of the current state-of-the-art technologies in Software-as-a-Service transformation for automotive systems. Through an extensive review of literature, we have compared our paper to others in the same scope and identified the unique contributions of our work. Our use-case scenario highlights the importance of software in the automotive industry and the challenges faced by automakers when adopting Software-as-a-Service solutions. We have identified societal, business, design process, application profiles and requirements, and technical factors as the main catalysts of evolution towards Software-as-a-Service in the automotive industry. The automotive application life cycle and its convergence with Software-as-a-Service have been discussed in detail. We have highlighted open issues in this area, including the need for standardisation, security, and scalability. The architecture design, software delivery pipelines, and run-time management services have been examined in detail, providing a comprehensive understanding of the technical aspects of Software-as- a-Service for the automotive industry.

In conclusion, our research has provided a comprehensive understanding of Software-as-a-Service transformation for the automotive industry. The discussion and future works section identifies key areas for future research and gives our engineering insights by converging all the survey information in one theoretical high-level architecture proposal. Overall, our work contributes to the ongoing efforts to transform the automotive industry through the adoption of Software-as-a-Service solutions.

REFERENCES

[1] R. N. Charette, "How software is eating the car," *IEEE Spectr.*, Jun. 2021.

[2] K Reports. (2017). *The Digitalisation of the UK Automotive Industry*. Financial Report. [Online]. Available: https://assets.kpmg/content/dam/kpmg/uk/pdf/2017/04/The-digitalisation-of-the-U.K.-automotive-industry.pdf

[3] C. Buckl, A. Camek, G. Kainz, C. Simon, L. Mercep, H. Stähle, and A. Knoll, "The software car: Building ICT architectures for future electric vehicles," in *Proc. IEEE Int. Electric Vehicle Conf.*, Mar. 2012, pp. 1–8.

[4] Arm. (2022). *Scalable Open Architecture for Embedded Edge (SOAFEE) Project*.[Online]. Available: https://www.soafee.io

[5] The Eclipse Foundation. (2022). *Software-Defined Vehicle Working Group*. [Online]. Available: https://www.sdv.eclipse.org

[6] The Eclipse Foundation et al., "The eclipse automotive top-level project," 2022. [Online]. Available: https://projects.eclipse.org/projects/automotive

[7] Design For Driving. (2022). *Android Automotive OS (AAOS)*. [Online]. Available: https://developers.google.com/cars/design/automotive-os

[8] MIH Consortium. (2022). *Mobility in Harmony EV Ecosystem*. [Online]. Available: https://www.mih-ev.org

[9] The eSync Alliance, *A Multi-Company Initiative for OTA Updates and Diagnostics*, 2022. [Online]. Available: https://esyncalliance.org

[10] BMW Group. (2022). *Connected Vehicle Systems Alliance (COVESA)*. [Online]. Available: https://www.covesa.global

[11] S. M. Bohloul, "Service-oriented architecture: A review of state-of-the-art literature from an organizational perspective," *J. Ind. Integr. Manage.*, vol. 6, no. 3, pp. 353–382, Sep. 2021.

[12] A. Vetter, P. Obergfell, H. Guissouma, D. Grimm, M. Rumez, and E. Sax, "Development processes in automotive service-oriented architectures," in *Proc. 9th Medit. Conf. Embedded Comput. (MECO)*, Jun. 2020, pp. 1–7.

[13] M. Bellanger and E. Marmounier, "Service oriented architecture: Impacts and challenges of an architecture paradigm change," in *Proc. Eur. Congr. Embedded Real Time Softw. Syst. (ERTS)*, 2020, pp. 1–11.

[14] S. Kugele, P. Obergfell, M. Broy, O. Creighton, M. Traub, and W. Hopfensitz, "On service-orientation for automotive software," in *Proc. IEEE Int. Conf. Softw. Archit. (ICSA)*, Apr. 2017, pp. 193–202.

[15] A. Iwai and M. Aoyama, "Automotive cloud service systems based on service-oriented architecture and its evaluation," in *Proc. IEEE 4th Int. Conf. Cloud Comput.*, Jul. 2011, pp. 638–645.

[16] M. Mody, J. Jones, K. Chitnis, R. Sagar, G. Shurtz, Y. Dutt, M. Koul, M. Biju, and A. Dubey, "Understanding vehicle E/E architecture topologies for automated driving: System partitioning and tradeoff parameters," *Electron. Imag.*, vol. 30, no. 17, p. 358, Jan. 2018.

[17] A. Frigerio, B. Vermeulen, and K. G. W. Goossens, "Automotive architecture topologies: Analysis for safety-critical autonomous vehicle applications," *IEEE Access*, vol. 9, pp. 62837–62846, 2021.

[18] M. Çakir, T. Häckel, S. Reider, P. Meyer, F. Korf, and T. C. Schmidt, "A QoS aware approach to service-oriented communication in future automotive networks," in *Proc. IEEE Veh. Netw. Conf. (VNC)*, Dec. 2019, pp. 1–8.

[19] M. Rumez, D. Grimm, R. Kriesten, and E. Sax, "An overview of automotive service-oriented architectures and implications for security countermeasures," *IEEE Access*, vol. 8, pp. 221852–221870, 2020.

[20] G. L. Gopu, K. V. Kavitha, and J. Joy, "Service oriented architecture based connectivity of automotive ECUs," in *Proc. Int. Conf. Circuit, Power Comput. Technol. (ICCPCT)*, Mar. 2016, pp. 1–4.

[21] D. Stabili, L. Ferretti, and M. Marchetti, "Analyses of secure automotive communication protocols and their impact on vehicles life-cycle," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, Jun. 2018, pp. 452–457.

[22] J. Hemdutt, "Ethernet communication design with service oriented architecture (SOA)," M.S. thesis, Thapar Inst. Eng. Technol. (TIET), Patiala, Punjab, 2019.

[23] O. S. Al-Heety, Z. Zakaria, M. Ismail, M. M. Shakir, S. Alani, and H. Alsariera, "A comprehensive survey: Benefits, services, recent works, challenges, security, and use cases for SDN-VANET," *IEEE Access*, vol. 8, pp. 91028–91047, 2020.

[24] D. E. Sarmiento, A. Lebre, L. Nussbaum, and A. Chari, "Decentralized SDN control plane for a distributed cloud-edge infrastructure: A survey," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 1, pp. 256–281, 1st Quart., 2021.

[25] Q. Zhang, X. Wang, M. Huang, K. Li, and S. K. Das, "Software defined networking meets information centric networking: A survey," *IEEE Access*, vol. 6, pp. 39547–39563, 2018.

[26] C. M. Mohammed, "Sufficient comparison among cloud computing services: IaaS, PaaS, and SaaS: A review," *Int. J. Sci. Bus.*, vol. 5, no. 2, pp. 17–30, 2021.

[27] L. Tartakovsky, M. Gutman, and A. Mosyak, "Energy efficiency of road vehicles–trends and challenges," in *Energy Efficiency: Methods, Limitations and Challenges*. Hauppauge, NY, USA: Nova Science Publishers, 2012.

[28] C. Lv, J. Zhang, Y. Li, and Y. Yuan, "Mechanism analysis and evaluation methodology of regenerative braking contribution to energy efficiency improvement of electrified vehicles," *Energy Convers. Manage.*, vol. 92, pp. 469–482, Mar. 2015.

[29] J.-S. Chen, "Energy efficiency comparison between hydraulic hybrid and hybrid electric vehicles," *Energies*, vol. 8, no. 6, pp. 4697–4723, May 2015.

[30] M. Weiss, K. C. Cloos, and E. Helmers, "Energy efficiency trade-offs in small to large electric vehicles," *Environ. Sci. Eur.*, vol. 32, p. 46, Mar. 2020.

[31] N. Jovicic, G. Boskovic, G. Vujic, G. Jovicic, M. Despotovic, D. Milovanovic, and D. Gordic, "Route optimization to increase energy efficiency and reduce fuel consumption of communal vehicles," *Thermal Sci.*, vol. 14, pp. 67–78, Jun. 2010.

[32] A. Wasserburger, A. Schirrer, and C. Hametner, "Stochastic optimisation for the design of energy-efficient controllers for cooperative truck platoons," *Int. J. Intell. Transp. Syst. Res.*, vol. 20, no. 2, pp. 398–408, Aug. 2022.

[33] A. Sciarretta and A. Vahidi, *Energy-Efficient Driving of Road Vehicles*. Cham, Switzerland: Springer, 2020.

[34] M. Khalid, K. Wang, N. Aslam, Y. Cao, N. Ahmad, and M. K. Khan, "From smart parking towards autonomous valet parking: A survey, challenges and future works," *J. Netw. Comput. Appl.*, vol. 175, Feb. 2021, Art. no. 102935.

[35] T. Lin, H. Rivano, and F. L. Mouël, "A survey of smart parking solutions," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 12, pp. 3229–3253, Dec. 2017.

[36] M. Q. Khan and S. Lee, "Gaze and eye tracking: Techniques and applications in ADAS," *Sensors*, vol. 19, no. 24, p. 5540, 2019.

[37] I. Raouf, A. Khan, S. Khalid, M. Sohail, M. M. Azad, and H. S. Kim, "Sensor-based prognostic health management of advanced driver assistance system for autonomous vehicles: A recent survey," *Mathematics*, vol. 10, no. 18, p. 3233, 2022.

[38] J. M. L. Domínguez and T. J. M. Sanguino, "Review on V2X, I2X, and P2X communications and their applications: A comprehensive analysis over time," *Sensors*, vol. 19, no. 12, p. 2756, Jun. 2019.

[39] N. S. Pearre and H. Ribberink, "Review of research on V2X technologies, strategies, and operations," *Renew. Sustain. Energy Rev.*, vol. 105, pp. 61–70, May 2019.

[40] R. Bindu, M. P. Sejal, and H. Chetan, "A survey paper on evolution of VANET towards IOV," in *Proc. Int. Conf. Opt. Wireless Technol.* Cham, Switzerland: Springer, 2023, pp. 99–113.

[41] T. Rosenstatter, T. Olovsson, and M. Almgren, "V2C: A trust-based vehicle to cloud anomaly detection framework for automotive systems," in *Proc. 16th Int. Conf. Availability, Rel. Secur.*, Aug. 2021, pp. 1–10.

[42] A. Moiz and M. H. Alalfi, "A survey of security vulnerabilities in Android automotive apps," in *Proc. IEEE/ACM 3rd Int. Workshop Eng. Cybersecurity Crit. Syst. (EnCyCriS)*, May 2022, pp. 17–24.

[43] Z. Yu, D. Jin, X. Song, C. Zhai, and D. Wang, "Internet of Vehicle empowered mobile media scenarios: In-vehicle infotainment solutions for the mobility as a service (MaaS)," *Sustainability*, vol. 12, no. 18, p. 7448, Sep. 2020.

[44] M. Mishra and A. Kumar, "ADAS technology: A review on challenges, legal risk mitigation and solutions," in *Autonomous Driving and Advanced Driver-Assistance Systems (ADAS)*. Boca Raton, FL, USA: CRC Press, 2021.

[45] C. Hoyos, B. D. Lester, C. Crump, D. M. Cades, and D. Young, "Consumer perceptions, understanding, and expectations of advanced driver assistance systems (ADAS) and vehicle automation," in *Proc. Hum. Factors Ergonom. Soc. Annu. Meeting*. Thousand Oaks, CA, USA SAGE, 2018, pp. 1–5.

[46] M. Noureldin, A. Ghalwash, L. Abd-Ellatif, and M. ElGazzar, "Blending agile methodologies to support automotive SPICE compliance," *J. Software, Evol. Process*, vol. 34, no. 1, pp. 1–16, Jan. 2022.

[47] R. Klendauer, A. Hoffmann, J. M. Leimeister, M. Berkovich, and H. Krcmar, "Using the IDEAL software process improvement model for the implementation of automotive SPICE," in *Proc. 5th Int. Workshop Co-Operative Human Aspects Softw. Eng. (CHASE)*, Jun. 2012, pp. 66–72.

[48] B. Katumba and E. Knauss, "Agile development in automotive software development: Challenges and opportunities," in *Proc. Product-Focused Softw. Process Improvement, Int. Conf. (PROFES)*. Cham, Switzerland: Springer, 2014, pp. 1–15.

[49] S. Kanti Datta, M. Irfan Khan, L. Codeca, B. Denis, J. Härri, and C. Bonnet, "IoT and microservices based testbed for connected car services," in *Proc. IEEE 19th Int. Symp. A World Wireless, Mobile Multimedia Networks (WoWMoM)*, Jun. 2018, pp. 14–19.

[50] D. Yu and A. Xiao, "The digital foundation platform—A multi-layered SOA architecture for intelligent connected vehicle operating system," 2022, *arXiv:2210.08818*.

[51] E. G. Leaphart, B. J. Czerny, J. G. D'Ambrosio, C. L. Denlinger, and D. Littlejohn, "Survey of software failsafe techniques for safety-critical automotive applications," in *Proc. SAE Tech. Paper Ser.*, Apr. 2005, pp. 1–19.

[52] G. Xie, Y. Li, Y. Han, Y. Xie, G. Zeng, and R. Li, "Recent advances and future trends for automotive functional safety design methodologies," *IEEE Trans. Ind. Informat.*, vol. 16, no. 9, pp. 5629–5642, Sep. 2020.

[53] M. Rabe, S. Milz, and P. Mäder, "Development methodologies for safety critical machine learning applications in the automotive domain: A survey," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2021, pp. 129–141.

[54] J.-H. Oetjens, "Safety evaluation of automotive electronics using virtual prototypes: State of the art and research challenges," in *Proc. 51st ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2014, pp. 1–6.

[55] A. Nardi and A. Armato, "Functional safety methodologies for automotive applications," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2017, pp. 970–975.

[56] S. Marchese and J. Grosse, "Formal fault propagation analysis that scales to modern automotive SoCs," in *Proc. Design Verification Conf. Exhib. DVCON Eur.*, 2017, pp. 1–6.

[57] C. Miller and C. Valasek, "A survey of remote automotive attack surfaces," in *Proc. Black Hat USA*, 2014, pp. 1–12.

[58] S. Jadhav and D. Kshirsagar, "A survey on security in automotive networks," in *Proc. 4th Int. Conf. Comput. Commun. Control Autom. (ICCUBEA)*, Aug. 2018, pp. 1–6.

[59] A. Oyler and H. Saiedian, "Security in automotive telematics: A survey of threats and risk mitigation strategies to counter the existing and emerging attack vectors," *Secur. Commun. Netw.*, vol. 9, no. 17, pp. 4330–4340, 2016.

[60] C. Bernardini, M. R. Asghar, and B. Crispo, "Security and privacy in vehicular communications: Challenges and opportunities," *Veh. Commun.*, vol. 10, pp. 13–28, Oct. 2017.

[61] V. H. Le, J. den Hartog, and N. Zannone, "Security and privacy for innovative automotive applications: A survey," *Comput. Commun.*, vol. 132, pp. 17–41, Nov. 2018.

[62] J. Hao and G. Han, "On the modeling of automotive security: A survey of methods and perspectives," *Future Internet*, vol. 12, no. 11, p. 198, Nov. 2020.

[63] J. Dürrwang, K. Beckers, and R. Kriesten, "A lightweight threat analysis approach intertwining safety and security for the automotive domain," in *Proc. Int. Conf. Comput. Saf., Rel., Secur.* Cham, Switzerland: Springer, 2017, pp. 305–319.

[64] S. Chockalingam, D. Hadžiosmanović, W. Pieters, A. Teixeira, and P. V. Gelder, "Integrated safety and security risk assessment methods: A survey of key characteristics and applications," in *Proc. Int. Conf. Crit. Inf. Infrastructures Secur.* Cham, Switzerland: Springer, 2016, pp.

[65] H. Zhu, W. Zhou, Z. Li, L. Li, and T. Huang, "Requirements-driven automotive electrical/electronic architecture: A survey and prospective trends," *IEEE Access*, vol. 9, pp. 100096–100112, 2021.

[66] H. Askaripoor, M. Hashemi Farzaneh, and A. Knoll, "E/E architecture synthesis: Challenges and technologies," *Electronics*, vol. 11, no. 4, p. 518, Feb. 2022.

[67] S. Graf, S. Reinhart, M. Glaß, J. Teich, and D. Platte, "Robust design of E/E architecture component platforms," in *Proc. 52nd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2015, pp. 1–6.

[68] R. Obermaisser, P. Peti, and F. Tagliabo, "An integrated architecture for future car generations," in *Real-Time Systems*. Cham, Switzerland: Springer, 2007.

[69] I. Collin, "Visualizing complex systems: Variability in E/E architecture," Uppsala Universitet, Uppsala, Sweden, Tech. Rep., 2019.

[70] L. J. Moukahal, M. Zulkernine, and M. Soukup, "Towards a secure software lifecycle for autonomous vehicles," in *Proc. IEEE Int. Symp. Softw. Rel. Eng. Workshops (ISSREW)*, Oct. 2021, pp. 371–377.

[71] M. Staron, *Automotive Software Architectures*. Cham, Switzerland: Springer, 2021.

[72] M. Moravcik, P. Segec, and M. Kontsek, "Overview of cloud computing standards," in *Proc. 16th Int. Conf. Emerg. eLearning Technol. Appl. (ICETA)*, Nov. 2018, pp. 395–402.

[73] S. Ortiz Jr., "The problem with cloud-computing standardization," *Computer*, vol. 44, pp. 13–16, Jul. 2011.

[74] A. M. Caulfield, E. S. Chung, A. Putnam, H. Angepat, J. Fowers, M. Haselman, S. Heil, M. Humphrey, P. Kaur, J. Kim, D. Lo, T. Massengill, K. Ovtcharov, M. Papamichael, L. Woods, S. Lanka, D. Chiou, and D. Burger, "A cloud-scale acceleration architecture," in *Proc. 49th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2016, pp. 1–13.

[75] Z. Zhang, C. Wu, and D. W. Cheung, "A survey on cloud interoperability: Taxonomies, standards, and practice," in *Proc. ACM Perform. Eval. Rev. (SIGMETRICS)*, 2013, pp. 13–22.

[76] E. Lin, L. Xu, S. Bramhavar, M. M. de Oca, S. Gorsky, L. Yi, A. Groetsema, and J. Chou, "Global optimization of data pipelines in heterogeneous cloud environments," 2022, *arXiv:2202.05711*.

[77] I. Mugarza, I. Yarza, I. Agirre, F. Lussiana, and S. Botta, "Safety and security concept for software updates on mixed-criticality systems," in *Proc. 5th Int. Conf. Syst. Rel. Saf. (ICSRS)*, Nov. 2021, pp. 171–180.

[78] J. Dobaj et al., "Towards a security-driven automotive development lifecycle," *J. Softw., Evol. Process*, p. e2407, 2021.

[79] A. E. Minelgait, R. Dagili, and G. E. Liobikien, "The usage of public transport and impact of satisfaction in the European union," *Sustainability*, vol. 12, no. 21, p. 9154, 2020.

[80] O. Boreiko, V. Teslyuk, N. Kryvinska, and M. Logoyda, "Structure model and means of a smart public transport system," *Proc. Comput. Sci.*, vol. 55, pp. 75–82, Jan. 2019.

[81] L. Zhang, R. Long, and H. Chen, "Do car restriction policies effectively promote the development of public transport?" *World Develop.*, vol. 119, pp. 100–110, Jul. 2019.

[82] C. Roman, R. Liao, P. Ball, S. Ou, and M. de Heaver, "Detecting on-street parking spaces in smart cities: Performance evaluation of fixed and mobile sensing systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 7, pp. 2234–2245, Jul. 2018.

[83] J. Parmar, P. Das, and S. M. Dave, "Study on demand and characteristics of parking system in urban areas: A review," *J. Traffic Transp. Eng. English Ed.*, vol. 7, no. 1, pp. 111–124, Feb. 2020.

[84] B. Cox, C. Bauer, A. Mendoza Beltran, D. P. van Vuuren, and C. L. Mutel, "Life cycle environmental and cost comparison of current and future passenger cars under different energy scenarios," *Appl. Energy*, vol. 269, Jul. 2020, Art. no. 115021.

[85] A. Aguilera and J. Cacciari, "Living with fewer cars: Review and challenges on household demotorization," *Transp. Rev.*, vol. 40, no. 6, pp. 796–809, Nov. 2020.

[86] A. Steen, C. Veders, M. Herkes, J. Keiren, F. Tillema, J. Slomp, W. van Stralen, and W. Koeleman, *Workshop on the Move: How Can Car Maintenance and Repair be Future-Proofed in a Disruptive Time?* Arnhem, The Netherlands: HAN Automotive Research, 2020.

[87] R. Logtenberg, J. Pawley, and B. Saxifrage, "Comparing fuel and maintenance costs of electric and gas powered vehicles in Canada," 2 Degrees Inst., 2018.

[88] L. G. T. Carpio, "The effects of oil price volatility on ethanol, gasoline, and sugar price forecasts," *Energy*, vol. 181, pp. 1012–1022, Aug. 2019.

[89] S. Raphael and L. Rice, "Car ownership, employment, and earnings," *J. Urban Econ.*, vol. 52, no. 1, pp. 109–130, Jul. 2002.

[90] C. J. Kurz, G. Li, and D. J. Vine, "The young and the carless? The demographics of new vehicle purchases," Board Governors Federal Reserve Syst. (US), 2016.

[91] G. Lyons, P. Hammond, and K. Mackay, "The importance of user perspective in the evolution of MaaS," *Transp. Res. A, Policy Pract.*, vol. 121, pp. 22–36, Mar. 2019.

[92] L. Haldimann, H. Juby, and B. Warner, "Young people and driving licences: Who hasn't got one (yet) and why?" *Flux*, vols. 119–120, nos. 1–2, pp. 5–24, Jan. 2020.

[93] Y. Demoli, M. Sorin, A. Villaereal, and G. Varro, "Ecological conversion vs automobile addiction. An analysis of the discrepancies between environmental attitudes and car use among low-income households in suburban and rural areas," *Flux*, vols. 119–120, nos. 1–2, pp. 41–58, Jan. 2020.

[94] Morgan Stanley Reports Staff. (2016). *Auto Industry is Ripe for Disruption*. Financial report. [Online]. Available: https://www.morganstanley.com/ideas/car-of-future-is-autonomous-electric-shared-mobility

[95] P. Goodwin and K. Van Dender, "'Peak car'—Themes and issues," *Transp. Rev.*, vol. 33, pp. 243–254, May 2013.

[96] C. Focas and P. Christidis, "Peak car in Europe?" *Transp. Res. Proc.*, vol. 25, pp. 531–550, Jan. 2017.

[97] D. Baehler and P. Rérat, "Between ecological convictions and practical considerations–profiles and motivations of residents in car-free housing developments in Germany and Switzerland," *Geographica Helvetica*, vol. 75, no. 2, pp. 169–181, 2020.

[98] J.-M. Deleuil, E. Barbey, and A. Sintès, "Le dévoiturage ou la Ville sans (SA) voiture : Mobilités plurielles, services numériques et vie de quartier," *Flux*, vol. 108, no. 2, pp. 80–87, Jun. 2017.

[99] L. Eliot, "The reasons why millennials aren't as car crazed as baby boomers, and how self-driving cars fit in," Forbes, 2019.

[100] M. V. Ciasullo, G. Maione, C. Torre, and O. Troisi, "The growth of car-pooling: Insights from a social media investigation," in *Proc. Excellence Services Int. Conf.*, 2017, pp. 1–14.

[101] E. Petersen, Y. Zhang, and A. Darwiche, "Modeling car sharing and its impact on auto ownership: Evidence from Vancouver and Seattle," in *Proc. Efficient Transp.-Managing Demand Conf. Exhib. Transp. Assoc. Canada (TAC)*, 2016, pp. 1–7.

[102] B. Means and J. A. Seiner, "Navigating the uber economy," in *Proc. UCDL Rev.*, 2015, pp. 1–36.

[103] STELLANTIS. (2018). *Free2move Becomes a World Leader in Mobility with Acquisition of Share Now*. [Online]. Available: https://www.stellantis.com/en/news/press-releases/2022/july/free2move-becomes-a-world-leader-in-mobility-with-acquisition-of-share-now

[104] BMW Group. (2018). *DriveNow Becomes Wholly-Owned Subsidiary of BMW Group*. [Online]. Available: https://www.press.bmwgroup.com/global/inproceedings/detail/T0278280EN/drivenow-becomes-wholly-owned-subsidiary-of-bmw-group

[105] A. Ouardini, I. E. O. Ech-Chahedy, A. Bouhoute, I. Berrada, and M. E. Kamili, "Analyzing driving behavior: Towards dynamic driver profiling," in *Proc. Int. Conf. Ad Hoc Netw.* Cham, Switzerland: Springer, 2021, pp. 179–190.

[106] T. A. Litman, "Pay-as-you-drive vehicle insurance: Implementation, benefits, and costs," Tech. Rep. 06-1796. 2006.
[107] C. Jiacheng, Z. Haibo, Z. Ning, Y. Peng, G. Lin, and S. X. Sherman, "Software defined Internet of Vehicles: Architecture, challenges and solutions," *J. Commun. Inf. Netw.*, vol. 1, no. 1, pp. 14–26, Jun. 2016.
[108] J. Cacciari, L. B. Chevallier, and J. Crisp, "Beyond the mobility biographies, the social production of travel choices: Socialization, space and social relations," *Flux*, vols. 119–120, pp. 59–72, Jan. 2020.
[109] L. D. Olvera, D. Plat, P. Pochet, and K. Riley, "Access to the car in the cities of sub-saharan Africa: Practices and users in dakar," *Flux*, vols. 119–120, pp. 73–89, Jan. 2020.
[110] A. Tandon, A. Dhir, I. Almugren, G. N. AlNemer, and M. Mäntymäki, "Fear of missing out (FoMO) among social media users: A systematic literature review, synthesis and framework for future research," *Internet Res.*, vol. 31, no. 3, pp. 782–821, May 2021.
[111] F. Contreras, E. Baykal, and G. Abid, "E-leadership and teleworking in times of COVID-19 and beyond: What we know and where do we go," *Frontiers Psychol.*, vol. 11, Dec. 2020, Art. no. 590271.
[112] A. Gupta, V. Mittal, J. Peeters, and S. Van Nieuwerburgh, "Flattening the curve: Pandemic-induced revaluation of urban real estate," *J. Financial Econ.*, vol. 146, no. 2, pp. 594–636, Nov. 2022.
[113] S. Stein, *Capital City: Gentrification Real Estate State*. London, U.K.: Verso Books, 2019.
[114] C. Li, Y. Zhang, and Y. Chai, "Do spatial factors outweigh institutional factors? Changes in influencing factors of home-work separation from 2007 to 2017 in Beijing," *J. Transp. Geography*, vol. 96, Oct. 2021, Art. no. 103201.
[115] C. Ebner, M. Schmidthaler, and P. Brandstätter, "Digitalization of labor: Evolutionary innovation or game changing developments in the way we work?" in *Proc. Cross-Cultural Bus. Conf. (CCBC)*, 2021, pp. 178–217.
[116] L. M. Clements and K. M. Kockelman, "Economic effects of automated vehicles," *Transp. Res. Rec.*, vol. 2606, no. 1, pp. 1–19, 2017.
[117] A. J. Reid, "A brief history of the smartphone," in *The Smartphone Paradox*. Cham, Switzerland: Springer, 2018.
[118] C. Bayart, N. Havet, P. Bonnel, and L. Bouzouina, "Young people and the private car: A love-hate relationship," *Transp. Res. D, Transp. Environ.*, vol. 80, Mar. 2020, Art. no. 102235.
[119] T. E. McMillan, "The relative influence of urban form on a child's travel mode to school," *Transp. Res. A, Policy Pract.*, vol. 41, no. 1, pp. 69–79, 2007.
[120] F. Kröger, "Automated driving in its social, historical and cultural contexts," in *Autonomous Driving*. Cham, Switzerland: Springer, 2016.
[121] I. Panagiotopoulos and G. Dimitrakopoulos, "An empirical investigation on consumers' intentions towards autonomous driving," in *Transp. Res. C, Emerg. Technol.*, vol. 95, pp. 773–784, Oct. 2018.
[122] V. Kolarova, F. Steck, and F. J. Bahamonde-Birke, "Assessing the effect of autonomous driving on value of travel time savings: A comparison between current and future preferences," *Transp. Res. A, Policy Pract.*, vol. 129, pp. 155–169, Nov. 2019.
[123] D. Gartman, "Three ages of the automobile: The cultural logics of the car," in *Theory, Culture & Society*. London, U.K.: SAGE Publications, 2004.
[124] J. Meissonnier and C. Richer, "Les routines automobiles à l'épreuve des. Perturbations. Comprendre les résistances au changement à partir de récits d'usagers dans La métropole lilloise," *Flux*, vols. 119–120, no. 1, pp. 25–40, Jul. 2020.
[125] X. Jin et al., "The impacts of emerging mobility options and vehicle technologies on travel behavior," 2020.
[126] E. Pojani, V. Van Acker, and D. Pojani, "Cars as a status symbol: Youth attitudes toward sustainable transport in a post-socialist city," *Transp. Res. F, Traffic Psychol. Behaviour*, vol. 58, pp. 210–227, Oct. 2018.
[127] S. Y. He and J. Thøgersen, "The impact of attitudes and perceptions on travel mode choice and car ownership in a Chinese megacity: The case of Guangzhou," *Res. Transp. Econ.*, vol. 62, pp. 57–67, Jun. 2017.
[128] R. Luke, "Car ownership perceptions and intentions amongst south African students," *J. Transp. Geography*, vol. 66, pp. 135–143, Jan. 2018.
[129] G. Da Silveira, D. Borenstein, and F. S. Fogliatto, "Mass customization: Literature review and research directions," *Int. J. Prod. Econ.*, vol. 72, no. 1, pp. 1–13, Jun. 2001.
[130] H.-H. Lee and H. Moon, "Perceived risk of online apparel mass customization: Scale development and validation," in *Clothing Textiles Res. J.*, vol. 33, no. 2, pp. 115–128, 2015.
[131] T. Sakao, W. Song, and J. Matschewsky, "Creating service modules for customising product/service systems by extending DSM," *CIRP Ann.*, vol. 66, no. 1, pp. 21–24, 2017.
[132] V. S. C. Tunn, R. Fokker, K. A. Luijkx, S. A. M. De Jong, and J. P. L. Schoormans, "Making ours mine: Increasing consumer acceptance of access-based PSS through temporary product customisation," *Sustainability*, vol. 11, no. 1, p. 274, Jan. 2019.
[133] A. Woodward, K. R. Smith, D. Campbell-Lendrum, D. D. Chadee, Y. Honda, Q. Liu, J. Olwoch, B. Revich, R. Sauerborn, and Z. Chafe, "Climate change and health: On the latest IPCC report," *Lancet*, vol. 383, no. 9924, pp. 1185–1189, 2014.
[134] P. Arias, N. Bellouin, E. Coppola, R. Jones, G. Krinner, J. Marotzke, V. Naik, M. Palmer, G.-K. Plattner, and J. Rogelj, "Climate change: The physical science basis," in *Proc. Work. Group14 I 6th Assessment Rep. Intergovernmental Panel Climate Change*, 2021, pp. 1–21.
[135] M. F. Bashir, M. Benjiang, H. I. Hussain, M. Shahbaz, K. Koca, and I. Shahzadi, "Evaluating environmental commitments to COP21 and the role of economic complexity, renewable energy, financial development, urbanization, and energy innovation: Empirical evidence from the RCEP countries," *Renew. Energy*, vol. 184, pp. 541–550, Jan. 2022.
[136] X. Ou, X. Zhang, and S. Chang, "Scenario analysis on alternative fuel/vehicle for China's future road transport: Life-cycle energy demand and GHG emissions," *Energy Policy*, vol. 38, no. 8, pp. 3943–3956, 2010.
[137] M. Melaina and K. Webster, "Role of fuel carbon intensity in achieving 2050 greenhouse gas reduction goals within the light-duty vehicle sector," *Environ. Sci. Technol.*, vol. 45, no. 9, 3865–3871, 2011.
[138] S. T. Pauliuk, "Reconciling sectoral abatement strategies with global climate targets: The case of the Chinese passenger vehicle fleet," *Environ. Sci. Technol.*, vol. 46, no. 1, pp. 140–147, 2012.
[139] N. Holjevac, F. Cheli, and M. Gobbi, "Multi-objective vehicle optimization: Comparison of combustion engine, hybrid and electric powertrains," *Proc. Inst. Mech. Engineers, D, J. Automobile Eng.*, vol. 234, nos. 2–3, pp. 469–487, Feb. 2020.
[140] S. F. Tie and C. W. Tan, "A review of energy sources and energy management system in electric vehicles," *Renew. Sustain. Energy Rev.*, vol. 20, pp. 82–102, Apr. 2013.
[141] L. S. Martins, L. F. Guimarães, A. B. Botelho Junior, J. A. S. Tenório, and D. C. R. Espinosa, "Electric car battery: An overview on global demand, recycling and future approaches towards sustainability," *J. Environ. Manage.*, vol. 295, Oct. 2021, Art. no. 113091.
[142] M. Handwerker, J. Wellnitz, and H. Marzbani, "Comparison of hydrogen powertrains with the battery powered electric vehicle and investigation of small-scale local hydrogen production using renewable energy," *Hydrogen*, vol. 2, no. 1, pp. 76–100, Jan. 2021.
[143] A. Dirnaichner, M. Rottoli, R. Sacchi, S. Rauner, B. Cox, C. Mutel, C. Bauer, and G. Luderer, "Life-cycle impacts from different decarbonization pathways for the European car fleet," *Environ. Res. Lett.*, vol. 17, no. 4, 2022, Art. no. 044009.
[144] P. Ahmadi, "Environmental impacts and behavioral drivers of deep decarbonization for transportation through electric vehicles," *J. Cleaner Prod.*, vol. 225, pp. 1209–1219, Jul. 2019.
[145] L. Gaines, K. Richa, and J. Spangenberger, "Key issues for Li-ion battery recycling," *MRS Energy Sustainability*, vol. 5, no. 1, p. 12, May 2018.
[146] A. Cabrera Serrenho and J. M. Allwood, "Material stock demographics: Cars in great Britain," *Environ. Sci. Technol.*, vol. 50, no. 6, pp. 3002–3009, Mar. 2016.
[147] I. Rüdenauer, C.-O. Gensch, R. Grießhammer, and D. Bunke, "Integrated environmental and economic assessment of products and processes," *J. Ind. Ecology*, vol. 9, no. 4, pp. 105–116, Oct. 2005.
[148] D. V. Spitzley, D. E. Grande, G. A. Keoleian, and H. C. Kim, "Life cycle optimization of ownership costs and emissions reduction in US vehicle retirement decisions," *Transp. Res. Part D, Transp. Environ.*, vol. 10, no. 2, pp. 161–175, 2005.
[149] N. Truttmann and H. Rechberger, "Contribution to resource conservation by reuse of electrical and electronic household appliances," *Resour., Conservation Recycling*, vol. 48, no. 3, pp. 249–262, 2006.
[150] A. Curran, "Extending product life-spans: Household furniture and appliance reuse in the UK," *Longer Lasting Products*. Evanston, IL, USA: Routledge, 2016.
[151] S. Boulos et al., "The durability of products: Standard assessment for the circular economy under the eco-innovation action plan," Rep. Eur. Commission, DG Environ., 2015.

[152] A. Hanelt et al., "Digital transformation of primarily physical industries-exploring the impact of digital trends on business models of automobile manufacturers," 2015.

[153] T. Kessler and C. Buck, "How digitization affects mobility and the business models of automotive OEMs," in *Phantom Ex Machina*. Cham, Switzerland: Springer, 2017.

[154] M. Rachinger, R. Rauter, C. Müller, W. Vorraber, and E. Schirgi, "Digitalization and its influence on business model innovation," *J. Manuf. Technol. Manage.*, vol. 30, no. 8, pp. 1143–1160, Dec. 2019.

[155] S. Sarasvuo, A. Rindell, and M. Kovalchuk, "Toward a conceptual understanding of co-creation in branding," *J. Bus. Res.*, vol. 139, pp. 543–563, Feb. 2022.

[156] M. L. Cheung, G. Pires, P. J. Rosenberger III, W. K. S. Leung, and M. K. Chang, "The role of social media elements in driving co-creation and engagement," *Asia Pacific J. Marketing Logistics*, vol. 33, no. 10, pp. 1994–2018, Oct. 2021.

[157] J. Füller, H. Mühlbacher, K. Matzler, and G. Jawecki, "Consumer empowerment through Internet-based co-creation," *J. Manage. Inf. Syst.*, vol. 26, no. 3, pp. 71–102, Dec. 2009.

[158] P. Desyllas and M. Sako, "Profiting from business model innovation: Evidence from pay-as-you-drive auto insurance," *Res. Policy*, vol. 42, no. 1, pp. 101–116, 2013.

[159] G. Seiberth and W. Gründinger, "Data-driven business models in connected cars, mobility services and beyond," in *Proc. BVDW Res.*, 2018, pp. 1–8.

[160] J. J. Yun, X. Zhao, J. Wu, J. C. Yi, K. Park, and W. Jung, "Business model, open innovation, and sustainability in car sharing industry—Comparing three economies," *Sustainability*, vol. 12, no. 5, p. 1883, 2020.

[161] J. Schwanholz and S. Leipold, "Sharing for a circular economy? An analysis of digital sharing platforms' principles and business models," *J. Cleaner Prod.*, vol. 269, Oct. 2020, Art. no. 122327.

[162] P. Wells, "Sustainable business models and the automotive industry: A commentary," *IIMB Manage. Rev.*, vol. 25, no. 4, pp. A200–A201, Dec. 2013.

[163] R. Bohnsack, H. Kurtz, and A. Hanelt, "Re-examining path dependence in the digital age: The evolution of connected car business models," *Res. Policy*, vol. 50, no. 9, 2021, Art. no. 104328.

[164] S. Sarasini and O. Langeland, "Business model innovation as a process for transforming user mobility practices," in *Environ. Innov. Societal Transitions*, vol. 39, pp. 229–248, Jun. 2021.

[165] G. Beier, J. Kiefer, and J. Knopf, "Potentials of big data for corporate environmental management: A case study from the German automotive industry," *J. Ind. Ecology*, vol. 26, no. 1, pp. 336–349, Feb. 2022.

[166] O. Schumann, M. Hahn, N. Scheiner, F. Weishaupt, J. F. Tilly, J. Dickmann, and C. Wöhler, "Radarscenes: A real-world radar point cloud data set for automotive applications," in *Proc. IEEE Int. Conf. Inf. Fusion (FUSION)*, Nov. 2021, pp. 1–8.

[167] D. R. Clough and A. Wu, "Artificial intelligence, data-driven learning, and the decentralized structure of platform ecosystems," in *Academy of Management Review*. Briarcliff Manor, NY, USA: Academy of Management Briarcliff Manor, 2022.

[168] K. Birch, D. Cochrane, and C. Ward, "Data as asset? The measurement, governance, and valuation of digital personal data by big tech," *Big Data Soc.*, vol. 8, no. 1, Jan. 2021, Art. no. 205395172110173.

[169] Stout Staff. (2020). *Automotive Defect & Recall Report*. [Online]. Available: https://www.stout.com/en/insights/report/2020-automotive-defect-and-recall-report

[170] J. Cieśla, "The theory and practice of testing the quality of outsourcing services in the automotive industry," in *Proc. Conf. Quality Prod. Improvement (CQPI)*, 2021, pp. 1–12.

[171] K. C. Felser, "The impact of digital technologies on it sourcing strategies in the German automotive industry," in *Handbook of Research on Digital Transformation, Industry Use Cases, and the Impact of Disruptive Technologies*. Hershey, PA, USA: IGI Global, 2022.

[172] S. Prawesh, K. Chari, and M. Agrawal, "Industry norms as predictors of IT outsourcing behaviors," *Int. J. Inf. Manage.*, vol. 56, Feb. 2021, Art. no. 102242.

[173] P. Munten, J. Vanhamme, F. Maon, V. Swaen, and A. Lindgreen, "Addressing tensions in coopetition for sustainable innovation: Insights from the automotive industry," *J. Bus. Res.*, vol. 136, pp. 10–20, Nov. 2021.

[174] M. Zahir and M. Rabah-Rabbou, "Innovation and knowledge transfer in SME-large firm relationships: The automotive subcontracting sector," *Revue Manag. Innov.*, no. 2, pp. 67–86, 2021.

[175] M. Pizka and A. Bauer, "A brief top-down and bottom-up philosophy on software evolution," in *Proc. 7th Int. Workshop Princ. Softw. Evol.*, 2004, pp. 1–9.

[176] P. A. Sabatier, "Top-down and bottom-up approaches to implementation research: A critical analysis and suggested synthesis," *J. Public Policy*, vol. 6, no. 1, pp. 21–48, Jan. 1986.

[177] V. R. Basili, G. Caldiera, H. D. Rombach, "The goal question metric approach," in *Encyclopedia of Software Engineering*. 1994, pp. 528–532.

[178] R. Capilla, F. Nava, and A. Tang, "Attributes for characterizing the evolution of architectural design decisions," in *Proc. 3rd Int. IEEE Workshop Softw. Evolvability*, Oct. 2007, pp. 1–14.

[179] E. Céret, S. Dupuy-Chessa, G. Calvary, A. Front, and D. Rieu, "A taxonomy of design methods process models," *Inf. Softw. Technol.*, vol. 55, no. 5, pp. 795–821, 2013.

[180] C. Erbas and B. C. Erbas, "Modules and transactions: Building blocks for a theory of software engineering," *Sci. Comput. Program.*, vol. 101, no. 1, pp. 6–20, 2015.

[181] I. Schnabel and M. Pizka, "Goal-driven software development," in *Proc. 30th Annu. IEEE/NASA Softw. Eng. Workshop*, Apr. 2006, pp. 1–9.

[182] B. Liu, H. Zhang, and S. Zhu, "An incremental V-model process for automotive development," in *Proc. 23rd Asia–Pacific Softw. Eng. Conf. (APSEC)*, Dec. 2016, pp. 225–232.

[183] K. Petersen, C. Wohlin, and D. Baca, "The waterfall model in large-scale development," in *Proc. Int. Conf. Product-Focused Softw. Process Improvement*, 2009, pp. 1–11.

[184] N. I. Gross and P. Svasta, "Development and validation of the wheel speed sensor interface based on V-model method," in *Proc. IEEE Int. Spring Seminar Electron. Technol. (ISSE)*, Jun. 2022, pp. 1–12.

[185] I. Graessler and J. Hentze, "The new V-model of VDI 2206 and its validation," *Automatisierungstechnik*, vol. 68, no. 5, pp. 312–324, May 2020.

[186] P. Beynon-Davies, C. Carne, H. Mackay, and D. Tudhope, "Rapid application development (RAD): An empirical review," *Eur. J. Inf. Syst.*, vol. 8, no. 3, pp. 211–223, Sep. 1999.

[187] J. T. Bell, "Extreme programming," Thinking Innov., 2001.

[188] K. Schwaber and J. Sutherland, "The Scrum guide. The definitive guide to Scrum," 2011.

[189] M. L. Junior and M. G. Filho, "Variations of the Kanban system: Literature review and classification," *Int. J. Prod. Econ.*, vol. 125, no. 1, pp. 13–21, May 2010.

[190] A. Löcken, S. S. Borojeni, H. Müller, T. M. Gable, S. Triberti, C. Diels, C. Glatz, I. Alvarez, L. Chuang, and S. Boll, "Towards adaptive ambient in-vehicle displays and interactions: Insights and design guidelines," in *Proc. Automot. User Interfaces Workshop*. Cham, Switzerland: Springer, 2017, pp. 1–12.

[191] E. Edwar, S. Sameh, and I. Sobh, "ASPICE applicability on new automotive technologies (AI)," in *Proc. Eur. Conf. Softw. Process Improvement*. Cham, Switzerland: Springer, 2022, pp. 430–440.

[192] C. Plappert, F. Fenzl, R. Rieke, I. Matteucci, G. Costantino, and M. De Vincenzi, "SECPAT: Security patterns for resilient automotive E/E architectures," in *Proc. 30th Euromicro Int. Conf. Parallel, Distrib. Network-Based Process. (PDP)*, Mar. 2022, pp. 255–264.

[193] V. Bandur et al., "Aspects of migrating from decentralized to centralized E/E architectures," SAE Tech. Paper, 2022.

[194] P. Pelliccione, E. Knauss, R. Heldal, S. Magnus Ågren, P. Mallozzi, A. Alminger, and D. Borgentun, "Automotive architecture framework: The experience of Volvo cars," *J. Syst. Archit.*, vol. 77, pp. 83–100, Jun. 2017.

[195] V. Bandur, G. Selim, V. Pantelic, and M. Lawford, "Making the case for centralized automotive E/E architectures," *IEEE Trans. Veh. Technol.*, vol. 70, no. 2, pp. 1230–1245, Feb. 2021.

[196] P. Wallin and J. Axelsson, "A case study of issues related to automotive E/E system architecture development," in *Proc. 15th Annu. IEEE Int. Conf. Workshop Eng. Comput. Based Syst.*, Mar. 2008, pp. 1–7.

[197] M. H. Farzaneh and A. Knoll, "An ontology-based plug-and-play approach for in-vehicle time-sensitive networking (TSN)," in *Proc. IEEE 7th Annu. Inf. Technol., Electron. Mobile Commun. Conf. (IEMCON)*, Oct. 2016, pp. 1–8.

[198] F. Sagstetter, M. Lukasiewycz, S. Steinhorst, M. Wolf, A. Bouard, W. R. Harris, S. Jha, T. Peyrin, A. Poschmann, and S. Chakraborty, "Security challenges in automotive hardware/software architecture design," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2013, pp. 458–463.

[199] B. Zheng, H. Liang, Q. Zhu, H. Yu, and C. Lin, "Next generation automotive architecture modeling and exploration for autonomous driving," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2016, pp. 53–58.

[200] A. K. Singh, M. Shafique, A. Kumar, and J. Henkel, "Resource and throughput aware execution trace analysis for efficient run-time mapping on MPSoCs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 1, pp. 72–85, Jan. 2016.

[201] P. K. Sahu and S. Chattopadhyay, "A survey on application mapping strategies for network-on-chip design," *J. Syst. Archit.*, vol. 59, no. 1, pp. 60–76, Jan. 2013.

[202] M. Mehrara, T. Jablin, D. Upton, D. August, K. Hazelwood, and S. Mahlke, "Multicore compilation strategies and challenges," *IEEE Signal Process. Mag.*, vol. 26, no. 6, pp. 55–63, Nov. 2009.

[203] M. Haeberle, F. Heimgaertner, H. Loehr, N. Nayak, D. Grewe, S. Schildt, and M. Menth, "Softwarization of automotive E/E architectures: A software-defined networking approach," in *Proc. IEEE Veh. Netw. Conf. (VNC)*, Dec. 2020, pp. 1–8.

[204] M. Hasan, S. Mohan, T. Shimizu, and H. Lu, "Securing vehicle-to-everything (V2X) communication platforms," *IEEE Trans. Intell. Vehicles*, vol. 5, no. 4, pp. 693–713, Dec. 2020.

[205] H. Zhou, W. Xu, J. Chen, and W. Wang, "Evolutionary V2X technologies toward the Internet of Vehicles: Challenges and opportunities," *Proc. IEEE*, vol. 108, no. 2, pp. 308–323, Feb. 2020.

[206] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge computing for autonomous driving: Opportunities and challenges," *Proc. IEEE*, vol. 107, no. 8, pp. 1697–1716, Aug. 2019.

[207] Z. Lv, D. Chen, and Q. Wang, "Diversified technologies in Internet of Vehicles under intelligent edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 4, pp. 2048–2059, Apr. 2021.

[208] C. Lin, D. Deng, and C. Yao, "Resource allocation in vehicular cloud computing systems with heterogeneous vehicles and roadside units," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3692–3700, Oct. 2018.

[209] Y. Leng and L. Zhao, "Novel design of intelligent Internet-of-Vehicles management system based on cloud-computing and Internet-of-Things," in *Proc. Int. Conf. Electron. Mech. Eng. Inf. Technol.*, Aug. 2011, pp. 3190–3193.

[210] R. Rana, M. Staron, J. Hansson, and M. Nilsson, "Defect prediction over software life cycle in automotive domain state of the art and road map for future," in *Proc. IEEE Int. Conf. Softw. Eng. Appl. (ICSOFT-EA)*, Apr. 2014, pp. 1–19.

[211] A. Mestre and T. Cooper, "Circular product design. A multiple loops life cycle design approach for the circular economy," *Design J.*, vol. 20, no. 1, pp. S1620–S1635, Jul. 2017.

[212] M. Broy, "Challenges in automotive software engineering," in *Proc. 28th Int. Conf. Softw. Eng.*, May 2006, pp. 1–14.

[213] S. Furst, "Challenges in the design of automotive software," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2010, pp. 1–3.

[214] T. Liebetrau, U. Kelling, T. Otter, and M. Hell, "Energy saving in automotive E/E architectures," Infineon Technol., Neubiberg, Germany, Tech. Rep. D-85579, 2012.

[215] S. Kugele and G. Pucea, "Model-based optimization of automotive E/E-architectures," in *Proc. 6th Int. Workshop Constraints Softw. Test., Verification, Anal.*, May 2014, pp. 1–8.

[216] K. Jo, J. Kim, D. Kim, C. Jang, and M. Sunwoo, "Development of autonomous car—Part I: Distributed system architecture and development process," *IEEE Trans. Ind. Electron.*, vol. 61, no. 12, pp. 7131–7140, Dec. 2014.

[217] 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.

[218] E. Abazi, "Practicing continuous integration in a multi-supplier environment for the development of automotive software," 2019.

[219] H. Guissouma, H. Klare, E. Sax, and E. Burger, "An empirical study on the current and future challenges of automotive software release and configuration management," in *Proc. 44th Euromicro Conf. Softw. Eng. Adv. Appl. (SEAA)*, Aug. 2018, pp. 1–9.

[220] K. Juhnke, M. Tichy, and F. Houdek, "Challenges concerning test case specifications in automotive software testing: Assessment of frequency and criticality," *Softw. Quality J.*, vol. 29, no. 1, pp. 39–100, Mar. 2021.

[221] V. Kesri, A. Nayak, and K. Ponnalagu, "AutoKG—An automotive domain knowledge graph for software testing: A position paper," in *Proc. IEEE Int. Conf. Softw. Test., Verification Validation Workshops (ICSTW)*, Apr. 2021, pp. 234–238.

[222] P. Reichenpfader, M. Driussi, and F. Pölzlbauer, "Towards certification of software-intensive mixed-critical systems in automotive industry," in *Proc. Workshop Crit. Automot. Appl., Robustness Saf. (CARS)*, 2016, pp. 1–9.

[223] I Staff. (2018). *ISO 26262-2:2018*. [Online]. Available: https://www.iso.org/fr/standard/68384.html

[224] (Aug. 2021). *ISO/SAE 21434:2021*. [Online]. Available: https://www.iso.org/fr/standard/70918.html

[225] A. Reindl, D. Wetzel, N. Balbierer, H. Meier, M. Niemetz, and S. Park, "Comparative analysis of can can FD and Ethernet for networked control systems," in *Proc. Embedded World Conference Digital*, 2021, pp. 1–8.

[226] D. Umehara and T. Shishido, "Controller area network and its reduced wiring technology," *IEICE Trans. Commun.*, vol. E102.B, no. 7, pp. 1248–1262, 2019.

[227] T. Matsushita, D. Umehara, and K. Wakasugi, "Poster: Power over data lines for CAN using AMI code," in *Proc. IEEE Veh. Netw. Conf. (VNC)*, Dec. 2016, pp. 1–2.

[228] L. L. 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.

[229] Open Alliance. (2022). *Automotive Ethernet Sleep/Wake-Up*. [Online]. Available: https://www.opensig.org/tech-committees/tc10-/

[230] N. D. Zervas et al., "Designing a CAN-to-TSN Ethernet gateway," in *Proc. 17th Int. CAN Conf. (iCC)*, 2020, pp. 129–133.

[231] P. Lindgren, S. Aittamaa, and J. Eriksson, "IP over CAN, transparent vehicular to infrastructure access," in *Proc. 5th IEEE Consum. Commun. Netw. Conf.*, 2008, pp. 1–6.

[232] F. Gaillard, "Microprocessor (MPU) or microcontroller (MCU)? What factors should you consider when selecting the right processing device for your next design," 2013, p. 14. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/MCU_vs_MPU_Article.pdf

[233] M. Villari, A. Celesti, and M. Fazio, "Towards osmotic computing: Looking at basic principles and technologies," in *Proc. Conf. Complex, Intell., Software Intensive Syst.* Cham, Switzerland: Springer, 2017, pp. 1–13.

[234] K. R. Raghunathan, "History of microcontrollers: First 50 years," *IEEE Micro*, vol. 41, no. 6, pp. 97–104, Nov. 2021.

[235] S. Krishnadas, "Concept and implementation of AUTOSAR compliant automotive Ethernet stack on Infineon Aurix Tricore board" 2016.

[236] Infineon. *AURIXT TC39x-B User Manual*. Feb. 2021. [Online]. Available: https://www.infineon.com/dgdl/Infineon-AURIX_TC39x-UserManual-v01_00-EN.pdf?fileId=5546d462712ef9b7017182d371dc1d95

[237] Y. Xu, P. Ou, and Y. Li, "Design of vehicle gateway automatic test system based on CANoe," in *Proc. Chin. Autom. Congr. (CAC)*, Nov. 2019, pp. 1433–1437.

[238] *RH850/F1KH User's Manual: Hardware (R01UH0684EJ0110)*, Renesas, Tokyo, Japan, 2018.

[239] NXP. (2021). *S32K1xx Data Sheet*. [Online]. Available: https://www.nxp.com/docs/en/data-sheet/S32K-DS.pdf

[240] Nvidia. (2019). *Jetson AGX Xavier Series System-on-Module Data Sheet*. [Online]. Available: https://ausdroid.co/wp-content/uploads/2020/08/Jetson-AGX-Xavier-Series-Datasheet.pdf

[241] Qualcomm. (2020). *Snapdragon 8 GEN2—Mobile Platform Data Sheet*. [Online]. Available: https://www.qualcomm.com/content/dam/qcomm-martech/dm-assets/documents/Snapdragon-8-Gen-2-Product-Brief.pdf

[242] Mobileye. (2022). *Mobileye 6-Series User Manual*. [Online]. Available: https://www.c2sec.com.sg/Files/UserManualMobileye6.pdf

[243] S. Liu, J. Tang, Z. Zhang, and J. Gaudiot, "Computer architectures for autonomous driving," *Computer*, vol. 50, no. 8, pp. 18–25, 2017.

[244] H. Chishiro, K. Suito, T. Ito, S. Maeda, T. Azumi, K. Funaoka, and S. Kato, "Towards heterogeneous computing platforms for autonomous driving," in *Proc. IEEE Int. Conf. Embedded Softw. Syst. (ICESS)*, Jun. 2019, pp. 1–8.

[245] S. Liu, J. Tang, Z. Zhang, and J.-L. Gaudiot, "CAAD: Computer architecture for autonomous driving," 2017, *arXiv:1702.01894*.

[246] D. Yokoyama, B. Schulze, F. Borges, and G. Mc Evoy, "The survey on ARM processors for HPC," *J. Supercomput.*, vol. 75, no. 10, pp. 7003–7036, Oct. 2019.

[247] E. Blem, J. Menon, T. Vijayaraghavan, and K. Sankaralingam, "ISA wars: Understanding the relevance of ISA being RISC or CISC to performance, power, and energy on modern architectures," *ACM Trans. Comput. Syst.*, vol. 33, no. 1, pp. 1–34, Mar. 2015.

[248] M. N. Asghar, "A review of ARM processor architecture history, progress and applications," *J. Appl. Emerg. Sci.*, vol. 10, no. 2, p. 171, Dec. 2020.

[249] S. Pennisi, "The integrated circuit industry at a crossroads: Threats and opportunities," *Chips*, vol. 1, no. 3, pp. 150–171, 2022.

[250] H. Ekiz, A. Kutlu, and E. Powner, "Design and implementation of a can/Ethernet bridge," in *Proc. IEEE Int. Conf. Commun.*, Sep. 1996, pp. 1–9.

[251] F. Luo et al., "Routing and security mechanisms design for automotive TSN/CAN FD security gateway," SAE Tech. Paper, 2022.

[252] J.-L. Scharbarg, M. Boyer, and C. Fraboul, "CAN-Ethernet architectures for real-time applications," in *Proc. IEEE Conf. Emerg. Technol. Factory Autom.*, 2005, pp. 1–19.

[253] Hyundai Motor Company. (Sep. 2022). *Introducing the E-GMP*. [Online]. Available:https://www.hyundai.com/worldwide/en/brand-journal/ioniq/e-gmp-revolution

[254] Oliver Hoffmann. (2021). *Premium Platform Electric (PPE)*. [Online]. Available: https://www.audi.com/content/dam/gbp2/company/investor-relations/events-and-presentations/investor-presentations/2021/2021-07-07-UBS-PPE-deep-dive.pdf

[255] Z. Zhongming, L. Linong, Y. Xiaona, Z. Wangqiang, and L. Wei. (2021). *New Auto: Volkswagen Group Set to Unleash Value in Battery-Electric Autonomous Mobility World*. [Online]. Available: https://www.volkswagenag.com/en/news/2021/07/new-auto–volkswagen-group-set-to-unleash-value-in-battery-elect.html

[256] *Four Full BEV Platforms to Support Market & Customer Needs*, Stellantis' EV Day Presentation, 2021.

[257] B. Defay, "Toyota prepare une plateforme multi-énergies pour l'europe," 2021. [Online]. Available: https://www.auto-moto.com/hybride-rechargeable/toyota-prepare-plateforme-multi-energies-leurope-11439

[258] *One Platform. Multiple Models*, Volkswagen Staff, Wolfsburg, Germany, 2021.

[259] D. Guilbert, M. Guarisco, A. Gaillard, A. N'Diaye, and A. Djerdir, "FPGA based fault-tolerant control on an interleaved DC/DC boost converter for fuel cell electric vehicle applications," *Int. J. Hydrogen Energy*, vol. 40, no. 45, pp. 15815–15822, Dec. 2015.

[260] R. R. Sabbella and M. Arunachalam, "Functional safety development of motor control unit for electric vehicles," in *Proc. IEEE Transp. Electrific. Conf. (ITEC-India)*, Dec. 2019, pp. 1–6.

[261] P. Weiss, A. Weichslgartner, F. Reimann, and S. Steinhorst, "Fail-operational automotive software design using agent-based graceful degradation," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2020, pp. 1169–1174.

[262] A. A. Syed, S. Ayaz, T. Leinmüller, and M. Chandra, "Dynamic scheduling and routing for TSN based in-vehicle networks," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Jun. 2021, pp. 1–6.

[263] D. Håkansson and J. Martinsson, "Driver intention prediction for optimal vehicle wake-up," 2020.

[264] M. Traub, A. Maier, and K. L. Barbehön, "Future automotive architecture and the impact of IT trends," *IEEE Softw.*, vol. 34, no. 3, pp. 27–32, May 2017.

[265] S. Jiang, "Vehicle E/E architecture and its adaptation to new technical trends," SAE, Warrendale, PA, USA, Tech. Paper 2019-01-0862, 2019.

[266] S. Saidi, S. Steinhorst, A. Hamann, D. Ziegenbein, and M. Wolf, "Special session: Future automotive systems design: Research challenges and opportunities," in *Proc. Int. Conf. Hardw./Softw. Codesign Syst. Synth. (CODES+ISSS)*, Sep. 2018, pp. 1–7.

[267] N. B. Truong, G. M. Lee, and Y. Ghamri-Doudane, "Software defined networking-based vehicular adhoc network with fog computing," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, May 2015, pp. 1202–1207.

[268] R. Shrestha, R. Bajracharya, and S. Y. Nam, "Challenges of future VANET and cloud-based approaches," *Wireless Commun. Mobile Comput.*, vol. 2018, May 2018, Art. no. 5603518.

[269] R. Qun and S. M. Arefzadeh, "A new energy-aware method for load balance managing in the fog-based vehicular ad hoc networks (VANET) using a hybrid optimization algorithm," *IET Commun.*, vol. 15, no. 13, pp. 1665–1676, Aug. 2021.

[270] I. Nunes, C. Lucena, P. Alencar, and D. Cowan, *Software Product Lines, Service-Oriented Architectures and Multi-Agent Systems*. Citeseer, 2009.

[271] A. Helferich, G. Herzwurm, S. Jesse, and M. Mikusz, "Software product lines, service-oriented architecture and frameworks: Worlds apart or ideal partners?" in *Proc. Int. Conf. Trends Enterprise Appl. Archit.* Cham, Switzerland: Springer, 2006, pp. 1–9.

[272] K. B. Laskey and K. Laskey, "Service oriented architecture," *Wiley Interdiscipl. Rev., Comput. Statist.*, vol. 1, no. 1, pp. 101–105, 2009.

[273] S. Pulparambil and Y. Baghdadi, "Service oriented architecture maturity models: A systematic literature review," *Comput. Standards Interfaces*, vol. 61, pp. 65–76, Jan. 2019.

[274] P. Bianco, R. Kotermanski, and P. Merson, "Evaluating a service-oriented architecture," Softw. Eng. Inst., Carnegie Mellon Univ., 2007.

[275] Y.-H. Wang and J. C. Liao, "Why or why not service oriented architecture," in *Proc. IEEE Int. Conf. Services Sci., Manag. Eng. (IITA)*, Sep. 2009, pp. 1–10.

[276] N. Niknejad, W. Ismail, I. Ghani, B. Nazari, M. Bahari, and A. R. B. C. Hussin, "Understanding service-oriented architecture (SOA): A systematic literature review and directions for further investigation," *Inf. Syst.*, vol. 91, Jul. 2020, Art. no. 101491.

[277] H. Taylor, A. Yochem, L. Phillips, and F. Martinez, *Event-Driven Architecture: How SOA Enables the Real-Time Enterprise*. London, U.K.: Pearson Education, 2009.

[278] B. M. Michelson, *Event-Driven Architecture Overview*. Boston, MA, USA: Patricia Seybold Group, 2006.

[279] N. Mateus-Coelho, M. Cruz-Cunha, and L. G. Ferreira, "Security in microservices architectures," *Proc. Comput. Sci.*, vol. 181, pp. 1225–1236, 2021.

[280] L. De Lauretis, "From monolithic architecture to microservices architecture," in *Proc. IEEE Int. Symp. Softw. Rel. Eng. Workshops (ISSREW)*, Oct. 2019, pp. 93–96.

[281] J. Kazanavičius and D. Mažeika, "Migrating legacy software to microservices architecture," in *Proc. IEEE Open Conf. Electr., Electron. Inf. Sci. (eStream)*, 2019, pp. 1–12.

[282] J. Ghofrani and D. Lübke, "Challenges of microservices architecture: A survey on the state of the practice," in *Proc. ZEUS*, 2018, pp. 1–70.

[283] T. Salah, M. J. Zemerly, C. Y. Yeun, M. Al-Qutayri, and Y. Al-Hammadi, "The evolution of distributed systems towards microservices architecture," in *Proc. IEEE Int. Conf. Internet Technol. Secured Trans. (ICITST)*, Dec. 2016, pp. 318–325.

[284] R. Bordini and A. E. F. Seghrouchni, *Multi-Agent Programming: Languages, Tools and Applications*. Cham, Switzerland: Springer, 2009.

[285] P. Obergfell, S. Kugele, and E. Sax, "Model-based resource analysis and synthesis of service-oriented automotive software architectures," in *Proc. ACM/IEEE 22nd Int. Conf. Model Driven Eng. Lang. Syst. (MODELS)*, Sep. 2019, pp. 128–138.

[286] R. Rotermund, T. Häckel, P. Meyer, F. Korf, and T. C. Schmidt, "Requirements analysis and performance evaluation of SDN controllers for automotive use cases," in *Proc. IEEE Veh. Netw. Conf. (VNC)*, Dec. 2020, pp. 1–8.

[287] L. Baresi, C. Ghezzi, A. Miele, M. Miraz, A. Naggi, and F. Pacifici, "Hybrid service-oriented architectures: A case-study in the automotive domain," in *Proc. 5th Int. Workshop Softw. Eng. Middleware*, Sep. 2005, pp. 1–9.

[288] F. Oszwald, P. Obergfell, B. Liu, V. Pazmino Betancourt, and J. Becker, "Model-based design of service-oriented architectures for reliable dynamic reconfiguration," *SAE Int. J. Adv. Current Practices Mobility*, vol. 2, no. 5, pp. 2938–2947, Apr. 2020.

[289] T. Hackel, P. Meyer, F. Korf, and T. C. Schmidt, "Software-defined networks supporting time-sensitive in-vehicular communication," in *Proc. IEEE 89th Veh. Technol. Conf. (VTC-Spring)*, Apr. 2019, pp. 1–5.

[290] T. K. Kuppusamy, L. A. DeLong, and J. Cappos, "Uptane: Security and customizability of software updates for vehicles," *IEEE Veh. Technol. Mag.*, vol. 13, no. 1, pp. 66–73, Mar. 2018.

[291] G. Falco and J. E. Siegel, "Assuring automotive data and software integrity employing distributed hash tables and blockchain," 2020, *arXiv:2002.02780*.

[292] A. Kampmann, B. Alrifaee, M. Kohout, A. Wüstenberg, T. Woopen, M. Nolte, L. Eckstein, and S. Kowalewski, "A dynamic service-oriented software architecture for highly automated vehicles," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 2101–2108.
[293] K. Mansour, W. Farag, and M. ElHelw, "AiroDiag: A sophisticated tool that diagnoses and updates vehicles software over air," in *Proc. IEEE Int. Electric Vehicle Conf.*, Mar. 2012, pp. 1–7.
[294] M. Steger, M. Karner, J. Hillebrand, W. Rom, C. Boano, and K. Römer, "Generic framework enabling secure and efficient automotive wireless SW updates," in *Proc. IEEE 21st Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2016, pp. 1–8.
[295] D. K. Nilsson and U. E. Larson, "Secure firmware updates over the air in intelligent vehicles," in *Proc. ICC Workshops IEEE Int. Conf. Commun. Workshops*, May 2008, pp. 1–14.
[296] D. K. Nilsson, L. Sun, and T. Nakajima, "A framework for self-verification of firmware updates over the air in vehicle ECUs," in *Proc. IEEE Globecom Workshops*, Nov. 2008, pp. 1–8.
[297] K. Mayilsamy, N. Ramachandran, and V. S. Raj, "An integrated approach for data security in vehicle diagnostics over internet protocol and software update over the air," *Comput. Electr. Eng.*, vol. 71, pp. 578–593, Oct. 2018.
[298] M. Steger, A. Dorri, S. S. Kanhere, K. Römer, R. Jurdak, and M. Karner, "Secure wireless automotive software updates using blockchains: A proof of concept," in *Advanced Microsystems for Automotive Applications*. Cham, Switzerland: Springer, 2018.
[299] B. Liu, V. P. Betancourt, Y. Zhu, and J. Becker, "Towards an on-demand redundancy concept for autonomous vehicle functions using microservice architecture," in *Proc. IEEE Int. Symp. Syst. Eng. (ISSE)*, Jul. 2020, pp. 1–19.
[300] S. M. Mahmud, S. Shanker, and I. Hossain, "Secure software upload in an intelligent vehicle via wireless communication links," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2005, pp. 1–10.
[301] V. Cebotari and S. Kugele, "Playground for early automotive service architecture design and evaluation," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Oct. 2020, pp. 1349–1356.
[302] B. Schegolev, H. Wijekoon, and V. Merunka, "Cost perspective of a service oriented architecture in vehicle design," in *Proc. HAICTA*, 2020, pp. 1–5.
[303] J. Leguay, M. Lopez-Ramos, K. Jean-Marie, and V. Conan, "An efficient service oriented architecture for heterogeneous and dynamic wireless sensor networks," in *Proc. 33rd IEEE Conf. Local Comput. Netw. (LCN)*, Oct. 2008, pp. 1–9.
[304] IBM. (2005). *IBM WebSphere Architecture: An Architectural Blueprint for Autonomic Computing*. [Online]. Available: https://www-03.ibm.com
[305] J. Ganci and A. Acharya. (2006). *Patterns: SOA Foundation Service Creation Scenario*. [Online]. Available: http://www.redbooks.ibm.com/redbooks/pdfs/sg247240.pdf
[306] W.-T. Tsai et al., "Architecture classification for SOA-based applications," in *Proc. 9th IEEE Int. Symp. Object Compon.-Oriented Real-Time Distrib. Comput. (ISORC)*. IEEE, 2006, p. 8 and p. 330.
[307] G. Brown and R. Carpenter, "Successful application of service-oriented architecture across the enterprise and beyond," *Intel Technol. J.*, vol. 8, no. 4, pp. 345–359, 2004.
[308] W. T. Tsai, "Service-oriented system engineering: A new paradigm," in *Proc. IEEE Int. Workshop Service-Oriented Syst. Eng. (SOSE)*, 2005, pp. 1–14.
[309] C. Wang, Li, Xi, Y. Chen, Y. Zhang, O. Diessel, and X. Zhou, "Service-oriented architecture on FPGA-based MPSoC," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 10, pp. 2993–3006, Oct. 2017.
[310] L. Voelker. (2022). *Scalable Service-Oriented MiddlewarE Over IP (SOME/IP)*. [Online]. Available: https://some-ip.com/
[311] InfoQ. *A Brief Introduction to REST*. (2021). [Online]. Available: https://www.infoq.com/articles/rest-introduction/
[312] P. Adamczyk, P. H. Smith, R. E. Johnson, and M. Hafiz, "REST and web services: In theory and in practice," in *REST: From Research to Practice*. New York, NY, USA: Springer, 2011.
[313] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. Nielsen, S. Thatte, and D. Winer. (2000). *Simple Object Access Protocol (SOAP) 1.1*. [Online]. Available: http://www.w3.org/TR/SOAP
[314] S. Mumbaikar and P. Padiya, "Web services based on SOAP and REST principles," *Int. J. Sci. Res. Publications*, vol. 3, no. 5, pp. 1–4, 2013.
[315] I. Fette and A. Melnikov. (2011). *The WebSocket Protocol*. [Online]. Available: https://www.hjp.at/doc/rfc/rfc6455.html
[316] Eclipse. (2014). *CoAP RFC 7252 Documentation*. [Online]. Available: https://iottestware.readthedocs.io/en/master/coap_rfc.html
[317] A. Stanford-Clark and A. Nipper. (2011). *MQTT Specification*. [Online]. Available: https://mqtt.org/mqtt-specification/
[318] OASIS. (2021). *Advanced Message Queuing Protocol (AMQP) Overview*. [Online]. Available: http://docs.oasis-open.org/amqp/
[319] Internet Engineering Task Force. (2021). *XMPP Specifications*. [Online]. Available: https://xmpp.org/extensions/
[320] OMG Group. (2015). *Data Distribution Service (DDS)*. [Online]. Available: https://www.dds-foundation.org/dds-resources/
[321] A. Corsaro. (2009). *The DDS Tutorial*. [Online]. Available: http://www.laas.fr/files/SLides-A_Corsaro.pdf
[322] OPC Foundation. (2008). *The OPC Unified Architecture*. [Online]. Available: https://opcfoundation.org/about/opc-technologies/opc-ua/
[323] J. Pfrommer, A. Ebner, S. Ravikumar, and B. Karunakaran, "Open source OPC UA PubSub over TSN for realtime industrial communication," in *Proc. IEEE 23rd Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2018, pp. 1087–1090.
[324] AUTOSAR. (2017). *SOME/IP Service Discovery Protocol Specification*. [Online]. Available: https://www.autosar.org/fileadmin/user_upload/standards
[325] AUTOSAR TM. (2016). *SOME/IP Protocol Specification*. Standard proposition available at https://www.autosar.org/fileadmin/user_upload/standards
[326] K. Gemlau, J. Peeck, N. Sperling, P. Hertha, and R. Ernst, "A new design for data-centric Ethernet communication with tight synchronization requirements for automated vehicles," in *Proc. 45th Annu. Conf. IEEE Ind. Electron. Soc.*, Oct. 2019, pp. 4489–4494.
[327] M. Iorio, M. Reineri, F. Risso, R. Sisto, and F. Valenza, "Securing SOME/IP for in-vehicle service protection," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 13450–13466, Nov. 2020.
[328] S. Kugele, D. Hettler, and J. Peter, "Data-centric communication and containerization for future automotive software architectures," in *Proc. IEEE Int. Conf. Softw. Archit. (ICSA)*, Apr. 2018, pp. 65–6509.
[329] N. Parmar, V. Ranga, and B. Simhachalam Naidu, "Syntactic interoperability in real-time systems, ros 2, and adaptive autosar using data distribution services: An approach," in *Inventive Communication and Computational Technologies*. Cham, Switzerland: Springer, 2020.
[330] A. Elhadeedy and J. Daily, "ECU over named data networking (ECUoNDN): Data management and integration with heterogenous automotive networks," TechRxiv, Tech. Rep., 2022.
[331] V. Tejashree, N. Vidhyashree, S. Anusha, K. Anu, P. Akshatha, and S. Kumar, "MQTT-SN based architecture for estimating delay and throughput in IoT," in *Proc. Int. Conf. Inf. Process.* Cham, Switzerland: Springer, 2021, pp. 481–490.
[332] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A survey of information-centric networking research," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 2, pp. 1024–1049, 2nd Quart., 2014.
[333] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Commun. Mag.*, vol. 50, no. 7, pp. 26–36, Jul. 2012.
[334] F. Fontes, B. Rocha, A. Mota, P. Pedreiras, and V. Silva, "Extending MQTT-SN with real-time communication services," in *Proc. 25th IEEE Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, vol. 1, Sep. 2020, pp. 1–4.
[335] D. Yoshino, Y. Watanobe, and K. Naruse, "A highly reliable communication system for Internet of Robotic things and implementation in RT-middleware with AMQP communication interfaces," *IEEE Access*, vol. 9, pp. 167229–167241, 2021.
[336] T. Zhang, S. Shen, S. Mao, and G. Chang, "Delay-aware cellular traffic scheduling with deep reinforcement learning," in *Proc. IEEE Global Commun. Conf.*, Dec. 2020, pp. 1–6.
[337] I. Benacer, F. Boyer, N. Bélanger, and Y. Savaria, "A fast systolic priority queue architecture for a flow-based traffic manager," in *Proc. 14th IEEE Int. New Circuits Syst. Conf. (NEWCAS)*, Jun. 2016, pp. 1–4.
[338] T. Park and K. Lee, "Supporting ultra-low latency mixed-criticality communication using hardware-based data plane architecture," *J. Netw. Comput. Appl.*, vol. 204, Aug. 2022, Art. no. 103401.
[339] S. B. H. Said, Q. H. Truong, and M. Boc, "SDN-based configuration solution for IEEE 802.1 time sensitive networking (TSN)," in *Proc. ACM SIGBED Rev.*, 2019, pp. 1–15.

[340] A. Sabry, A. Omar, M. Hammad, and N. Abdelbaki, "AVB/TSN protocols in automotive networking," in *Proc. 15th Int. Conf. Comput. Eng. Syst. (ICCES)*, Dec. 2020, pp. 1–7.

[341] Z. Zhou, J. Lee, M. S. Berger, S. Park, and Y. Yan, "Simulating TSN traffic scheduling and shaping for future automotive Ethernet," *J. Commun. Netw.*, vol. 23, no. 1, pp. 53–62, Feb. 2021.

[342] M. Thi, S. Ben Hadj Said, and M. Boc, "SDN-based management solution for time synchronization in TSN networks," in *Proc. 25th IEEE Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2020, pp. 361–368.

[343] D. Casini, T. Blaß, I. Lütkebohle, and B. Brandenburg, "Response-time analysis of ROS 2 processing chains under reservation-based scheduling," in *Proc. Euromicro Conf. Real-Time Syst.* Wadern, Germany: Schloss Dagstuhl, 2019, pp. 1–23.

[344] S. Sudhakaran, V. Mageshkumar, A. Baxi, and D. Cavalcanti, "Enabling QoS for collaborative robotics applications with wireless TSN," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Jun. 2021, pp. 1–6.

[345] S. Schneider, "The future of IIoT software in manufacturing: A guide to understanding and using data distribution service (DDS), time-sensitive networking (TSN), and OPC unified architecture (OPC UA) for advanced manufacturing applications," *Control Eng.*, vol. 66, no. 1, pp. 16–19, 2019.

[346] M. Johansson and O. Pap, "Implementing and comparing static and machine-learning scheduling approaches using DPDK on an integrated CPU/GPU," 2019.

[347] J. Shi, D. Pesavento, and L. Benmohamed, "NDN-DPDK: NDN forwarding at 100 gbps on commodity hardware," in *Proc. 7th ACM Conf. Information-Centric Netw.*, Sep. 2020, pp. 1–11.

[348] W. Kim, "Cloud computing: Today and tomorrow," *J. Object Technol.*, vol. 8, no. 1, p. 65, 2009.

[349] Y. Wei and M. B. Blake, "Service-oriented computing and cloud computing: Challenges and opportunities," *IEEE Internet Comput.*, vol. 14, no. 6, pp. 72–75, Nov. 2010.

[350] L. Qian, Z. Luo, Y. Du, and L. Guo, "Cloud computing: An overview," in *Proc. IEEE Int. Conf. Cloud Comput.*, 2009, pp. 1–6.

[351] L. Bagherzadeh, H. Shahinzadeh, H. Shayeghi, A. Dejamkhooy, R. Bayindir, and M. Iranpour, "Integration of cloud computing and IoT (CloudIoT) in smart grids: Benefits, challenges, and solutions," in *Proc. Int. Conf. Comput. Intell. Smart Power Syst. Sustain. Energy (CISPSSE)*, Jul. 2020, pp. 1–8.

[352] P. Mell et al., "The NIST definition of cloud computing," 2011.

[353] P. Mohanty, L. Kumar, M. Malakar, S. K. Vishwakarma, and M. Reza, "Dynamic resource allocation in vehicular cloud computing systems using game theoretic based algorithm," in *Proc. 5th Int. Conf. Parallel, Distrib. Grid Comput. (PDGC)*, Dec. 2018, pp. 476–481.

[354] M. Whaiduzzaman, M. Sookhak, A. Gani, and R. Buyya, "A survey on vehicular cloud computing," *J. Netw. Comput. Appl.*, vol. 40, pp. 325–344, Apr. 2014.

[355] Y. Agarwal, K. Jain, and O. Karabasoglu, "Smart vehicle monitoring and assistance using cloud computing in vehicular ad hoc networks," *Int. J. Transp. Sci. Technol.*, vol. 7, no. 1, pp. 60–73, Mar. 2018.

[356] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 1, pp. 2322–2358, 4th Quart., 2017.

[357] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.

[358] Y. Ai, M. Peng, and K. Zhang, "Edge computing technologies for Internet of Things: A primer," *Digit. Commun. Netw.*, vol. 4, no. 2, pp. 77–86, 2018.

[359] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," in *Proc. ACM SIGCOMM Comput. Commun. Rev.*, 2014, pp. 1–9.

[360] H. Sabireen and V. Neelanarayanan, "A review on fog computing: Architecture, fog with IoT, algorithms and research challenges," *ICT Exp.*, vol. 7, no2, pp. 162–176, 2021.

[361] G. L. Stavrinides and H. D. Karatza, "Security, cost and energy aware scheduling of real-time IoT workflows in a mist computing environment," in *Information Systems Frontiers*. Cham, Switzerland: Springer, 2022.

[362] E. Rubio-Drosdov, D. D. Sánchez, F. Almenárez, and A. Marín, "A framework for efficient and scalable service offloading in the mist," in *Proc. IEEE World Forum on Internet Things (WF-IoT)*, Apr. 2019, pp. 460–463.

[363] L. Bréhon-Grataloup, R. Kacimi, and A.-L. Beylot, "Mobile edge computing for V2X architectures and applications: A survey," *Comput. Netw.*, vol. 206, Apr. 2022, Art. no. 108797.

[364] R. Meneguette, R. De Grande, J. Ueyama, G. P. R. Filho, and E. Madeira, "Vehicular edge computing: Architecture, resource management, security, and challenges," *ACM Comput. Surveys*, vol. 55, no. 1, pp. 1–46, Jan. 2023.

[365] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Trans. Veh. Technol.*, vol. 65, no. 6, pp. 3860–3873, Jun. 2016.

[366] C. Huang, R. Lu, and K. R. Choo, "Vehicular fog computing: Architecture, use case, and security and forensic challenges," *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 105–111, Nov. 2017.

[367] Z. Ning, J. Huang, and X. Wang, "Vehicular fog computing: Enabling real-time traffic management for smart cities," *IEEE Wireless Commun.*, vol. 26, no. 1, pp. 87–93, Feb. 2019.

[368] Y. Xiao and C. Zhu, "Vehicular fog computing: Vision and challenges," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops)*, Mar. 2017, pp. 6–9.

[369] M. Bonanni, F. Chiti, and R. Fantacci, "Mobile mist computing for the Internet of Vehicles," *Internet Technol. Lett.*, vol. 3, no. 6, p. e176, 2020.

[370] P. K. Sharma and J. H. Park, "Blockchain-based secure mist computing network architecture for intelligent transportation systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 8, pp. 5168–5177, Aug. 2021.

[371] R. El Sibai, T. Atéchian, J. B. Abdo, J. Demerjian, and R. Tawil, "A new software-based service provision approach for vehicular cloud," in *Proc. IEEE Global Summit Comput. Inf. Technol. (GSCIT)*, Jun. 2015, pp. 1–6.

[372] R. Lovas, A. C. Marosi, M. Emödi, Á. Kisari, E. Simonyi, and P. Gáspár, "PaaS-oriented IoT platform with connected cars use cases," in *Proc. Int. Conf. Sensor Netw. Signal Process. (SNSP)*, Oct. 2018, pp. 409–420.

[373] C. Puliafito, C. Cicconetti, M. Conti, E. Mingozzi, and A. Passarella, "Stateful function as a service at the edge," *Computer*, vol. 55, no. 9, pp. 54–64, Sep. 2022.

[374] J. Zhan, Y. Sha, and J. Yan, "Design and implementation of logistics vehicle monitoring system based on the SaaS model," in *Proc. IEEE Int. Conf. Bus. Intell. Financial Eng.*, Aug. 2012, pp. 524–526.

[375] A. Musaddiq, R. Ali, R. Bajracharya, Y. A. Qadri, F. Al-Turjman, and S. W. Kim, "Trends, issues, and challenges in the domain of iot-based vehicular cloud network," in *Unmanned Aerial Vehicles in Smart Cities*. Cham, Switzerland: Springer, 2020.

[376] M. Hamad, E. Regnath, J. Lauinger, V. Prevelakis, and S. Steinhorst, "SPPS: Secure policy-based publish/subscribe system for V2C communication," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Feb. 2021, pp. 529–534.

[377] T. Nishio, Y. Koda, J. Park, M. Bennis, and K. Doppler, "When wireless communications meet computer vision in beyond 5G," *IEEE Commun. Standards Mag.*, vol. 5, no. 2, pp. 76–83, Jun. 2021.

[378] B. P. Kraemer. (2010). *IEEE Standard for Information Technology—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments*. [Online]. Available: https://standards.ieee.org/ieee/802.11p/3953/

[379] S. A. Abdel Hakeem, A. A. Hady, and H. Kim, "5G-V2X: Standardization, architecture, use cases, network-slicing, and edge-computing," *Wireless Netw.*, vol. 26, pp. 6015–6041, Jul. 2020.

[380] N. Liu, M. Liu, J. Cao, G. Chen, and W. Lou, "When transportation meets communication: V2P over VANETs," in *Proc. IEEE 30th Int. Conf. Distrib. Comput. Syst.*, 2010, pp. 1–12.

[381] I. Ahmad, R. M. Noor, I. Ahmedy, S. A. A. Shah, I. Yaqoob, E. Ahmed, and M. Imran, "VANET–LTE based heterogeneous vehicular clustering for driving assistance and route planning applications," *Comput. Netw.*, vol. 145, pp. 128–140, Nov. 2018.

[382] A. Jalooli, E. Shaghaghi, M. R. Jabbarpour, R. Md Noor, H. Yeo, and J. J. Jung, "Intelligent advisory speed limit dedication in highway using VANET," *Sci. World J.*, vol. 2014, pp. 1–20, Jan. 2014.

[383] B. Djamel, G. Nacira, and T. Cherif, "Forecasting approach in VANET based on vehicle collision alert," in *Proc. Int. Conf. Multimedia Comput. Syst.*, May 2012, pp. 573–577.

[384] M. Taneja and N. Garg, "Smart traffic monitoring and alert system using VANET and deep learning," in *Proc. Int. Conf. Innov. Comput. Commun.* Cham, Switzerland: Springer, 2022, pp. 1–15.
[385] S. Kwatirayo, J. Almhana, and Z. Liu, "Adaptive traffic light control using VANET: A case study," in *Proc. 9th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jul. 2013, pp. 752–757.
[386] A. Baiocchi and F. Cuomo, "Infotainment services based on push-mode dissemination in an integrated VANET and 3G architecture," *J. Commun. Netw.*, vol. 15, no. 2, pp. 179–190, Apr. 2013.
[387] K. Abboud, H. A. Omar, and W. Zhuang, "Interworking of DSRC and cellular network technologies for V2X communications: A survey," *IEEE Trans. Veh. Technol.*, vol. 65, no. 12, pp. 9457–9470, Dec. 2016.
[388] Y. J. Li, "An overview of the DSRC/WAVE technology," in *Proc. Int. Conf. Heterogeneous Netw. Qual., Rel., Secur. Robustness*. Cham, Switzerland: Springer, 2010, pp. 544–558.
[389] A. H. Sakr, G. Bansal, V. Vladimerou, and M. Johnson, "Lane change detection using V2V safety messages," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 3967–3973.
[390] S. K. Flanagan, J. He, and X.-H. Peng, "Improving emergency collision avoidance with vehicle to vehicle communications," in *Proc. IEEE Int. Conf. High Perform. Comput. Commun., IEEE Int. Conf. Smart City IEEE Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, Jun. 2018, pp. 1322–1329.
[391] A. Cailean, B. Cagneau, L. Chassagne, S. Topsu, Y. Alayli, and J.-M. Blosseville, "Visible light communications: Application to cooperation between vehicles and road infrastructures," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2012, pp. 1055–1059.
[392] D. Kim, S. Yang, H. Kim, Y. Son, and S. Han, "Outdoor visible light communication for inter-vehicle communication using controller area network," in *Proc. 4th Int. Conf. Commun. Electron. (ICCE)*, Aug. 2012, pp. 31–34.
[393] C.-O. Ivascu, D. Ursutiu, and C. Samoila, "Improve VLC LiFi performance for V2V communication," in *Proc. Int. Conf. Remote Eng. Virtual Instrum.* Cham, Switzerland: Springer, 2019, pp. 315–329.
[394] M. Muhammad and G. A. Safdar, "5G-based v2v broadcast communications: A security perspective," *Array*, vol. 11, Sep. 2021, Art. no. 100084.
[395] L. Montero, C. Ballesteros, C. De Marco, and L. Jofre, "Beam management for vehicle-to-vehicle (V2V) communications in millimeter wave 5G," *Veh. Commun.*, vol. 34, Apr. 2022, Art. no. 100424.
[396] H. Townsend et al., "Summary report on request for information (RFI): Enhancing the safety of vulnerable road users at intersections," United States. Dept. Transp., Intell. Transp. Syst. Joint Program Office, 2023, p. 436.
[397] P. Sewalkar and J. Seitz, "Vehicle-to-pedestrian communication for vulnerable road users: Survey, design considerations, and challenges," *Sensors*, vol. 19, no. 2, p. 358, Jan. 2019.
[398] Y. Tian, J. Zhang, J. Ma, B. Du, J. Zhu, and A. J. Khattak, "Understanding scenarios for cooperative V2P safety applications using connected vehicle datasets," in *Proc. CICTP*, Dec. 2021, pp. 523–532.
[399] A. Kabil, K. Rabieh, F. Kaleem, and M. A. Azer, "Vehicle to pedestrian systems: Survey, challenges and recent trends," *IEEE Access*, vol. 10, pp. 123981–123994, 2022.
[400] Y. Yang, K. Lee, Y. Kim, and K. Fawaz, "PEDRO: Secure pedestrian mobility verification in V2P communication using commercial Off-the-shelf mobile devices," in *Proc. 12nd Workshop CPS IoT Secur. Privacy*, Nov. 2021, pp. 1–17.
[401] J. Meijers, E. Au, Y. Cai, H. Jacobsen, S. Motepalli, R. Sun, A. Veneris, G. Zhang, and S. Zhang, "Blockchain for V2X: A taxonomy of design use cases and system requirements," in *Proc. 3rd Conf. Blockchain Res. Appl. Innov. Netw. Services (BRAINS)*, Sep. 2021, pp. 113–120.
[402] M. Suwa, M. Nishimura, and R. Sakata, "LED projection module enables a vehicle to communicate with pedestrians and other vehicles," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2017, pp. 37–38.
[403] M. Soni, B. S. Rajput, T. Patel, and N. Parmar, "Lightweight vehicle-to-infrastructure message verification method for VANET," in *Data Science and Intelligent Applications*. Cham, Switzerland: Springer, 2021.
[404] J. Miller, "Vehicle-to-vehicle-to-infrastructure (V2V2I) intelligent transportation system architecture," in *Proc. IEEE Intell. Vehicles Symp.*, 2008, pp. 715–720.
[405] R. Q. Malik, H. A. Alsattar, K. N. Ramli, B. B. Zaidan, A. A. Zaidan, Z. H. Kareem, H. A. Ameen, S. Garfan, A. Mohammed, and R. A. Zaidan, "Mapping and deep analysis of vehicle-to-infrastructure communication systems: Coherent taxonomy, datasets, evaluation and performance measurements, motivations, open challenges, recommendations, and methodological aspects," *IEEE Access*, vol. 7, pp. 126753–126772, 2019.
[406] E. Eso, Z. Ghassemlooy, S. Zvanovec, A. Gholami, A. Burton, N. B. Hassan, and O. I. Younus, "Experimental demonstration of vehicle to road side infrastructure visible light communications," in *Proc. 2nd West Asian Colloq. Opt. Wireless Commun. (WACOWC)*, Apr. 2019, pp. 85–89.
[407] Z. Ma and S. Sun, "Research on vehicle-to-road collaboration and end-to-end collaboration for multimedia services in the Internet of Vehicles," *IEEE Access*, vol. 10, pp. 18146–18155, 2022.
[408] E. Normanyo and C. L. Ainoo, "Use of traffic signal lights in vehicle–to–vehicle communication on underground mine ramps," *Ghana Mining J.*, vol. 22, no. 1, pp. 50–55, Jun. 2022.
[409] C. Liu, K. T. Chau, D. Wu, and S. Gao, "Opportunities and challenges of vehicle-to-home, vehicle-to-vehicle, and vehicle-to-grid technologies," *Proc. IEEE*, vol. 101, no. 11, pp. 2409–2427, Nov. 2013.
[410] N. Z. Xu and C. Y. Chung, "Reliability evaluation of distribution systems including vehicle-to-home and vehicle-to-grid," *IEEE Trans. Power Syst.*, vol. 31, no. 1, pp. 759–768, Jan. 2016.
[411] H. Turker and S. Bacha, "Optimal minimization of plug-in electric vehicle charging cost with vehicle-to-home and vehicle-to-grid concepts," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 10281–10292, Nov. 2018.
[412] D. Borge-Diez, D. Icaza, E. Açıkkalp, and H. Amaris, "Combined vehicle to building (V2B) and vehicle to home (V2H) strategy to increase electric vehicle market share," *Energy*, vol. 237, Dec. 2021, Art. no. 121608.
[413] R. Stanojevic, S. Abbar, S. Thirumuruganathan, G. D. F. Morales, S. Chawla, F. Filali, and A. Aleimat, "Road network fusion for incremental map updates," in *Proc. Int. Conf. Location Based Services (LBS)*. Cham, Switzerland: Springer, 2018, pp. 1–19.
[414] C. Chen, C. Lu, Q. Huang, Q. Yang, D. Gunopulos, and L. Guibas, "City-scale map creation and updating using GPS collections," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 1–17.
[415] Q. Zou and M. Sester, "Incremental map refinement of building information using LiDAR point clouds," in *Proc. SPIE*, 2021, pp. 277–282.
[416] A. Pretschner, M. Broy, I. H. Kruger, and T. Stauner, "Software engineering for automotive systems: A roadmap," in *Proc. Future Softw. Eng. (FOSE)*, May 2007, pp. 1–9.
[417] S. Aust, "Vehicle update management in software defined vehicles," in *Proc. IEEE 47th Conf. Local Comput. Netw. (LCN)*, Sep. 2022, pp. 261–263.
[418] S. Halder, A. Ghosal, and M. Conti, "Secure OTA software updates in connected vehicles: A survey," 2019, *arXiv:1904.00685*.
[419] A. X. A. Sim and B. Sitohang, "OBD-II standard car engine diagnostic software development," in *Proc. Int. Conf. Data Softw. Eng. (ICODSE)*, Nov. 2014, pp. 1–5.
[420] R. Malekian, N. R. Moloisane, L. Nair, B. T. Maharaj, and U. A. K. Chude-Okonkwo, "Design and implementation of a wireless OBD II fleet management system," *IEEE Sensors J.*, vol. 17, no. 4, pp. 1154–1164, Feb. 2017.
[421] M. Ammar, H. Janjua, A. S. Thangarajan, B. Crispo, and D. Hughes, "Securing the on-board diagnostics port (OBD-II) in vehicles," *SAE Int. J. Transp. Cybersecurity Privacy*, vol. 2, no. 2, pp. 83–106, Aug. 2020.
[422] K. Fizza, N. Auluck, A. Azim, M. A. Maruf, and A. Singh, "Faster OTA updates in smart vehicles using fog computing," in *Proc. 12th IEEE/ACM Int. Conf. Utility Cloud Comput. Companion*, Dec. 2019, pp. 1–12.
[423] P. Efstathiadis, A. Karanika, N. Chouliaras, L. Maglaras, and I. Kantzavelou, "Smart cars and over-the-air updates," in *Cybersecurity Issues in Emerging Technologies*. Boca Raton, FL, USA: CRC Press, 2021.
[424] S. Halder, A. Ghosal, and M. Conti, "Secure over-the-air software updates in connected vehicles: A survey," *Comput. Netw.*, vol. 178, Sep. 2020, Art. no. 107343.
[425] N. Ayres, L. Deka, and D. Paluszczyszyn, "Continuous automotive software updates through container image layers," in *Electronics*, vol. 10, no. 6, p. 739, 2021.
[426] N. Huq, C. Gibson, and R. Vosseler, "Driving security into connected cars: Threat model and recommendations," in *Proc. Trend Micro*, 2020, pp. 1–37.
[427] R. Elsaraf, "Chrysler UConnect hack and automotive computer and cyber security," 2021.
[428] E. N. Witanto, Y. E. Oktian, S.-G. Lee, and J.-H. Lee, "A blockchain-based OCF firmware update for IoT devices," *Appl. Sci.*, vol. 10, no. 19, p. 6744, Sep. 2020.
[429] N. S. Mtetwa, N. Sibeko, P. Tarwireyi, and A. M. Abu-Mahfouz, "OTA firmware updates for LoRaWAN using blockchain," in *Proc. 2nd Int. Multidisciplinary Inf. Technol. Eng. Conf. (IMITEC)*, Nov. 2020, pp. 1–8.

[430] D. F. Blanco, F. L. Mouël, and T. Lin, "Fenrir: Blockchain-based inter-company app-store for the automotive industry," *IEEE Access*, vol. 10, pp. 122933–122953, 2022.
[431] H. Mansor, K. Markantonakis, R. N. Akram, and K. Mayes, "Don't brick your car: Firmware confidentiality and rollback for vehicles," in *Proc. 10th Int. Conf. Availability, Rel. Secur.*, Aug. 2015, pp. 139–148.
[432] R. S. Devi, B. V. Kumar, P. Sivakumar, A. N. Lakshmi, and R. Tripathy, "Bootloader design for advanced driver assistance system," in *Software Engineering for Automotive Systems*. Boca Raton, FL, USA: CRC Press, 2022.
[433] T. Mirfakhraie, G. Vitor, and K. Grogan, "Applicable protocol for updating firmware of automotive HVAC electronic control units (ECUs) over the air," in *Proc. IEEE Int. Conf. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Social Comput. (CPSCom) IEEE Smart Data (SmartData)*, Jul. 2018, pp. 21–26.
[434] A. S. Gedeon, L. Buttyán, and D. F. Papp, "Secure boot and firmware update on a microcontroller-based embedded board," Faculty Elect. Eng. Inform., Dept. Networked Syst. Services, Budapest Univ. Technol. Econ., Budapest, Hungary, Tech. Rep., 2020.
[435] C. Profentzas, M. Günes, Y. Nikolakopoulos, O. Landsiedel, and M. Almgren, "Performance of secure boot in embedded systems," in *Proc. 15th Int. Conf. Distrib. Comput. Sensor Syst. (DCOSS)*, May 2019, pp. 198–204.
[436] E. Westerberg, "Efficient delta based updates for read-only filesystem images: An applied study in how to efficiently update the software of an ECU," Tech. Rep., 2021.
[437] S. Nv, "On-the-fly firmware update for dual bank STM32 microcontrollers," STMicrocontrollers, Tech. Rep., 2019.
[438] X. Wu, T. Gao, and G. Lee, "Method of differential-based remote upgrade for ECUs on vehicles," *J. Phys., Conf.*, vol. 2216, no. 1, Mar. 2022, Art. no. 012004.
[439] A. Prasad and P. Shanthi, "Automotive electronic control unit reprogramming using delta method—A review," in *Proc. Asian Conf. Innov. Technol. (ASIANCON)*, Aug. 2021, pp. 1–6.
[440] Y. Jia, X. Shao, S. Wang, R. Zhai, Q. Li, and Y. Wang, "Research on vehicle OTA upgrade technology based on BSDIFF difference algorithm," in *Proc. IEEE Int. Conf. Adv. Electr. Eng. Comput. Appl. (AEECA)*, Aug. 2021, pp. 1113–1117.
[441] G. Ni, Z. Chen, J. Jiang, J. Luo, and Y. Ma, "Incremental updates based on graph theory for consumer electronic devices," *IEEE Trans. Consum. Electron.*, vol. 61, no. 1, pp. 128–136, Feb. 2015.
[442] B. Patel, V. Nalwade, G. Pagare, and K. Navale, "Miniupdate: A smart update method for smartphones," in *Proc. Int. J. Emerg. Technol. Innov. Res. (JETIR)*, 2015, pp. 1–19.
[443] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "TinyOS: An operating system for sensor networks," in *Ambient Intelligence*, W. Weber, J. M. Rabaey, and E. Aarts, Eds. Cham, Switzerland: Springer, 2005.
[444] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki—A lightweight and flexible operating system for tiny networked sensors," in *Proc. 29th Annu. IEEE Int. Conf. Local Comput. Netw.*, 2004, pp. 1–10.
[445] H. Abrach, S. Bhatti, J. Carlson, H. Dai, J. Rose, A. Sheth, B. Shucker, J. Deng, and R. Han, "MANTIS: System support for multimodAl networks of in-situ sensors," in *Proc. ACM Int. Conf. Wireless Sensor Netw. Appl.*, 2003, pp. 1–9.
[446] S. Jinfan. (2017). *LiteOS: A Heavyweight in IoT Connectivity*. [Online]. Available: https://www.huawei.com/fr/technology-insights/publications/huawei-tech/84/lite-os-smart-iot
[447] BlackBerry. (2023). *QNX Neutrino Whitepapers*. [Online]. Available: http://www.qnx.com/
[448] Windriver. (2023). *VxWorks Safety Platforms*. [Online]. Available: https://www.windriver.com/products/vxworks/safety-platforms
[449] (2023). *VxWorks RTOS*. [Online]. Available: https://www.windriver.com/products/vxworks
[450] D. Kyle and J. C. Brustoloni, "UCLinux: A Linux security module for trusted-computing-based usage controls enforcement," in *Proc. ACM Workshop Scalable Trusted Comput.*, 2007, pp. 1–8.
[451] A Developers. (2022). *Android Things*. [Online]. Available: https://developer.android.com/things
[452] W. Harrington, *Learning Raspbian get up and Running With Raspbian and Make the Most out of Your Raspberry PI*. Birmingham, U.K.: Packt Publishing, 2015.
[453] AUTOSAR. (2021). *Autosar Classic Platform*. [Online]. Available: https://www.autosar.org/standards/classic-platform/
[454] AUTOSAR TM. (2021). *Autosar Adaptive Platform*. [Online]. Available: https://www.autosar.org/standards/adaptive-platform/
[455] eeNews Automotive. (2019). *Autosar Adaptive Software Ready for ASIL-D Certification*. [Online]. Available: https://www.eenewsautomotive.com/news/autosar-adaptive-software-ready-asil-d-certification
[456] WEA Staff. (2010). *A Technical Companion to Windows Embedded Automotive 7*. [Online]. Available: https://manualzz.com/
[457] Google Developers. (2022). *Android Automotive OS Documentation*. [Online]. Available: https://developers.google.com/cars/design/automotive-os
[458] D. Déharbe, S. Galvão, and A. Moreira, "Formalizing FreeRTOS: First Steps," in *Proc. Brazilian Symp. Formal Methods (BSFM)*, 2009, pp. 101–117.
[459] E. Baccelli, O. Hahm, M. Günes, M. Wählisch, and T. C. Schmidt, "RIOT OS: Towards an OS for the Internet of Things," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2013, pp. 79–80.
[460] Apache. (2021). *Mynewt Documentation*. [Online]. Available: https://mynewt.apache.org/about/
[461] A World. (2022). *Mentor Nucleus SafetyCert RTOS Extends Support to ARM Cortex-M4 Processors for Safety-Critical Systems*. [Online]. Available: https://www.plm.automation.siemens.com
[462] GHS Staff. (2023). *Green Hills Platforms for Automotive Documentation*. [Online]. Available: https://www.ghs.com/products/auto_solutions.html
[463] S. Hahm, J. Kim, A. Jeong, H. Yi, S. Chang, S. N. Kishore, A. Chauhan, and S. P. Cherian, "Reliable real-time operating system for IoT devices," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3705–3716, Mar. 2021.
[464] K. P. Lawton, "BOCHS: A portable PC emulator for Unix/X," *Linux J.*, Sep. 1996.
[465] D. Laird. (2000). *Crusoe Processor Products and Technology*. [Online]. Available: https://www.ele.uva.es/~jesman/BigSeti/ftp/Microprocesadores/Transmeta/laird.pdf
[466] F. Bellard, "QEMU, a fast and portable dynamic translator," Tech. Rep., 2005.
[467] S. Nanda, W. Li, L.-C. Lam, and T.-C. Chiueh, "BIRD: Binary interpretation using runtime disassembly," in *Proc. Int. Symp. Code Gener. Optim. (CGO)*, 2006, pp. 1–7.
[468] K. Avi, K. Yaniv, and D. Laor, "KVM: The Linux virtual machine monitor," in *Proc. Linux Symp.*, 2007, pp. 1–9.
[469] *V Staff*, 2023.
[470] P. Barham, B. Dragovic, and K. Fraser, "Xen and the art of virtualization," in *Proc. ACM SIGOPS Operating Syst. Rev.*, 2003, pp. 1–9.
[471] Microsoft. (2021). *Introduction à Hyper-V Sur Windows 10*. [Online]. Available: https://docs.microsoft.com/en-uk/virtualization/hyper-v-on-windows/about/
[472] R. Pandey, "Comparing VMware fusion, oracle virtualbox, parallels desktop implemented as type-2 hypervisors," Nat. College Ireland, 2020.
[473] A. Patel, M. Daftedar, M. Shalan, and M. W. El-Kharashi, "Embedded hypervisor Xvisor: A comparative analysis," in *Proc. 23rd Euromicro Int. Conf. Parallel, Distrib., Netw.-Based Process.*, Mar. 2015, pp. 682–691.
[474] B. D. Ligneris, "Virtualization of Linux based computers: The Linux-VServer project," in *Proc. 19th Int. Symp. High Perform. Comput. Syst. Appl. (HPCS)*, 2005, pp. 1–16.
[475] E. Khen, N. J. Zaidenberg, A. Averbuch, and E. Fraimovitch, "LgDb 2.0: Using lguest for kernel profiling, code coverage and simulation," in *Proc. Int. Symp. Perform. Eval. Comput. Telecommun. Syst. (SPECTS)*, Jul. 2013, pp. 78–85.
[476] Docker. (2021). *Empowering App Development for Developers*. [Online]. Available: https://www.docker.com
[477] Á. Kovács, "Comparison of different Linux containers," in *Proc. 40th Int. Conf. Telecommun. Signal Process. (TSP)*, Jul. 2017, pp. 47–51.
[478] S. S. Kumaran, *Practical LXC and LXD: Linux Containers for Virtualization and Orchestration*. New York, NY, USA: Apress, 2017.
[479] R. Mocevicius, *CoreOS Essentials*. Packt Publishing, 2015.
[480] J. P. Martin, A. Kandasamy, and K. Chandrasekaran, "Exploring the support for high performance applications in the container runtime environment," *Hum.-centric Comput. Inf. Sci.*, vol. 8, no. 1, p. 61, Dec. 2018.
[481] Microsoft. (2022). *Containers on Windows Documentation*. [Online]. Available: https://docs.microsoft.com/en-us/virtualization/windowscontainers/

[482] Podman. (2020). *Podman Documentation*. [Online]. Available: https://podman.io/blogs/2020/05/13/podman-v2-update.html
[483] *W Staff*, 2023.
[484] S. Oracle. (1995). *Wabi Documentation*. [Online]. Available: https://docs.oracle.com/cd/E19957-01/802-6743/802-6743.pdf
[485] Ronald Joe Record. (1998). *Linux Emulation for SCO (LxRun)*. [Online]. Available: https://www.usenix.org/legacy/publications/library/proceedings/usenix98/freenix/record_html/lxrun.html
[486] B. Venners, "The Java virtual machine," in *Java and the Java Virtual Machine: Definition, Verification, Validation*. New York, NY, USA: McGraw-Hill, 1998.
[487] J. Richter, *Applied Microsoft. NET Framework Programming*. Redmond, WA, USA: Microsoft Press, 2002.
[488] F. Fagerholm, "Perl 6 and the Parrot virtual machine," 2005.
[489] S. Pinto, J. Pereira, T. Gomes, M. Ekpanyapong, and A. Tavares, "Towards a TrustZone-assisted hypervisor for real-time embedded systems," *IEEE Comput. Archit. Lett.*, vol. 16, no. 2, pp. 158–161, Jul. 2017.
[490] S. Pinto and N. Santos, "Demystifying arm TrustZone: A comprehensive survey," *ACM Comput. Surveys*, vol. 51, no. 6, pp. 1–36, Nov. 2019.
[491] C. Moratelli, S. Zampiva, and F. Hessel, "Full-virtualization on MIPS-based MPSOCs embedded platforms with real-time support," in *Proc. 27th Symp. Integr. Circuits Syst. Design (SBCCI)*, Sep. 2014, pp. 1–7.
[492] K. Zandberg and E. Baccelli, "Femto-containers: DevOps on microcontrollers with lightweight virtualization & isolation for IoT software modules," 2021, *arXiv:2106.12553*.
[493] E. Bringmann and A. Krämer, "Model-based testing of automotive systems," in *Proc. Int. Conf. Softw. Test., Verification, Validation*, Apr. 2008, pp. 1–9.
[494] P. Sivakumar, R. S. Sandhya Devi, A. D. Buvanesswaran, B. V. Kumar, R. Raguram, and M. Ranjithkumar, "Model-based testing of car engine start/stop button debouncer model," in *Proc. 2nd Int. Conf. Inventive Res. Comput. Appl. (ICIRCA)*, Jul. 2020, pp. 1077–1082.
[495] H. Huang, B. Pin-Hsuan Chang, C. Zhou-Peng Liao, and D.-Y. Chen, "A matter of risk management: The effects of the innovation sandboxes on citizens' risk perceptions," in *Proc. 22nd Annu. Int. Conf. Digit. Government Res.*, Jun. 2021, pp. 1–12.
[496] M. de Reuver, A. van Wynsberghe, M. Janssen, and I. van de Poel, "Digital platforms and responsible innovation: Expanding value sensitive design to overcome ontological uncertainty," *Ethics Inf. Technol.*, vol. 22, pp. 257–267, May 2020.
[497] T. Bécsi, S. Aradi, and P. Gáspár, "Security issues and vulnerabilities in connected car systems," in *Proc. Int. Conf. Models Technol. Intell. Transp. Syst. (MT-ITS)*, Jun. 2015, pp. 477–482.
[498] J. T. Burd, *Regulatry Sandboxes for Safety Assurance of Autonomous Vehicles*. University of Pennsylvania Journal of Law and Public Affairs, 2021.
[499] K. Czarnecki, "Software engineering for automated vehicles: Addressing the needs of cars that run on software and data," in *Proc. IEEE/ACM 41st Int. Conf. Softw. Eng., Companion (ICSE-Companion)*. IEEE, 2019, pp. 6–8.
[500] C. Lemieux and K. Sen, "FairFuzz: A targeted mutation strategy for increasing greybox fuzz testing coverage," in *Proc. 33rd IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Sep. 2018, pp. 475–485.
[501] S. Satyal, I. Weber, H.-Y. Paik, C. D. Ciccio, and J. Mendling, "Shadow testing for business process improvement," in *Proc. OTM Confederated Int. Conf. Move Meaningful Internet Syst.* Cham, Switzerland: Springer, 2018, pp. 1–10.
[502] A. Gambi, M. Mueller, and G. Fraser, "Automatically testing self-driving cars with search-based procedural content generation," in *Proc. 28th ACM SIGSOFT Int. Symp. Softw. Test. Anal.*, Jul. 2019, pp. 1–12.
[503] P. McMinn, "Search-based software testing: Past, present and future," in *Proc. IEEE 4th Int. Conf. Softw. Test., Verification Validation Workshops*, Mar. 2011, pp. 153–163.
[504] A. Gambi, T. Huynh, and G. Fraser, "Generating effective test cases for self-driving cars from police reports," in *Proc. 27th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, Aug. 2019, pp. 1–17.
[505] R. Ben Abdessalem, A. Panichella, S. Nejati, L. C. Briand, and T. Stifter, "Testing autonomous cars for feature interaction failures using many-objective search," in *Proc. 33rd IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Sep. 2018, pp. 143–154.
[506] V. Garousi, M. Felderer, C. C. M. Karapıçak, and U. Yılmaz, "Testing embedded software: A survey of the literature," *Inf. Softw. Technol.*, vol. 104, pp. 14–45, Dec. 2018.
[507] L. M. Cysneiros, M. Raffi, and J. C. S. do Prado Leite, "Software transparency as a key requirement for self-driving cars," in *Proc. IEEE Int. Requirements Eng. Conf. (RE)*, 2018, pp. 382–387.
[508] Y. Chen, S. Chen, T. Zhang, S. Zhang, and N. Zheng, "Autonomous vehicle testing and validation platform: Integrated simulation system with hardware in the loop," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2018, pp. 949–956.
[509] J. Koo, C. Saumya, M. Kulkarni, and S. Bagchi, "PySE: Automatic worst-case test generation by reinforcement learning," in *Proc. 12th IEEE Conf. Softw. Test., Validation Verification (ICST)*, Apr. 2019, pp. 136–147.
[510] W. He, R. Zhao, and Q. Zhu, "Integrating evolutionary testing with reinforcement learning for automated test generation of object-oriented software," *Chin. J. Electron.*, vol. 24, no. 1, pp. 38–45, Jan. 2015.
[511] T. Rangnau, R. V. Buijtenen, F. Fransen, and F. Turkmen, "Continuous security testing: A case study on integrating dynamic security testing tools in CI/CD pipelines," in *Proc. IEEE Int. Enterprise Distrib. Object Comput. Conf. (EDOC)*, 2020, pp. 1–9.
[512] J. Mahboob and J. Coffman, "A kubernetes CI/CD pipeline with asylo as a trusted execution environment abstraction framework," in *Proc. IEEE 11th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2021, pp. 0529–0535.
[513] M. Kromer, "Basics of CI/CD and pipeline scheduling," in *Mapping Data Flows in Azure Data Factory*. Cham, Switzerland: Springer, 2022.
[514] S. Liu and L. F. Capretz, "An analysis of testing scenarios for automated driving systems," in *Proc. IEEE Int. Conf. Softw. Anal., Evol. Reengineering (SANER)*, Mar. 2021, pp. 622–629.
[515] A. Gholami, K. Rao, W. Hsiung, O. Po, M. Sankaradas, and S. Chakradhar, "ROMA: Resource orchestration for microservices-based 5G applications," in *Proc. NOMS IEEE/IFIP Netw. Operations Manage. Symp.*, Apr. 2022, pp. 1–9.
[516] M. I. Khan, S. Sesia, and J. Harri, "In vehicle resource orchestration for multi-V2X services," in *Proc. IEEE 90th Veh. Technol. Conf. (VTC-Fall)*, Sep. 2019, pp. 1–5.
[517] S. Mittal, R. K. Dudeja, R. S. Bali, and G. S. Aujla, "A distributed task orchestration scheme in collaborative vehicular cloud edge networks," in *Computing*. Cham, Switzerland: Springer, 2022.
[518] F. Giannone, P. A. Frangoudis, A. Ksentini, and L. Valcarenghi, "Orchestrating heterogeneous MEC-based applications for connected vehicles," *Comput. Netw.*, vol. 180, Oct. 2020, Art. no. 107402.
[519] O. Rana, M. Shaikh, M. Ali, A. Anjum, and L. Bittencourt, "Vertical workflows: Service orchestration across cloud & edge resources," in *Proc. IEEE 6th Int. Conf. Future Internet Things Cloud (FiCloud)*, Aug. 2018, pp. 355–362.
[520] A. Khalili, S. Zarandi, and M. Rasti, "Joint resource allocation and offloading decision in mobile edge computing," *IEEE Commun. Lett.*, vol. 23, no. 4, pp. 684–687, Apr. 2019.
[521] Z. Kuang, L. Li, J. Gao, L. Zhao, and A. Liu, "Partial offloading scheduling and power allocation for mobile edge computing systems," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6774–6785, Aug. 2019.
[522] L. Li, X. Zhang, K. Liu, F. Jiang, and J. Peng, "An energy-aware task offloading mechanism in multiuser mobile-edge cloud computing," *Mobile Inf. Syst.*, vol. 2018, pp. 1–12, Jun. 2018.
[523] M. Hu, J. Luo, Y. Wang, M. Lukasiewycz, and Z. Zeng, "Holistic scheduling of real-time applications in time-triggered in-vehicle networks," *IEEE Trans. Ind. Informat.*, vol. 10, no. 3, pp. 1817–1828, Aug. 2014.
[524] X. Xu, Q. Liu, Y. Luo, K. Peng, X. Zhang, S. Meng, and L. Qi, "A computation offloading method over big data for IoT-enabled cloud-edge computing," *Future Gener. Comput. Syst.*, vol. 95, Jun. 2019, pp. 522–533.
[525] S. Yu, R. Langar, X. Fu, L. Wang, and Z. Han, "Computation offloading with data caching enhancement for mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11098–11112, Nov. 2018.
[526] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2018, pp. 207–215.
[527] D. Vu, N. Dao, W. Na, and S. Cho, "Dynamic resource orchestration for service capability maximization in fog-enabled connected vehicle networks," *IEEE Trans. Cloud Comput.*, vol. 10, no. 3, pp. 1726–1737, Jul. 2022.
[528] F. Guo, F. R. Yu, H. Zhang, H. Ji, M. Liu, and V. C. M. Leung, "Adaptive resource allocation in future wireless networks with blockchain and mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 1689–1703, Mar. 2020.

[529] L. P. Qian, B. Shi, Y. Wu, B. Sun, and D. H. K. Tsang, "NOMA-enabled mobile edge computing for Internet of Things via joint communication and computation resource allocations," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 718–733, Jan. 2020.

[530] Y. Wang, X. Tao, Y. T. Hou, and P. Zhang, "Effective capacity-based resource allocation in mobile edge computing with two-stage tandem queues," *IEEE Trans. Commun.*, vol. 67, no. 9, pp. 6221–6233, Sep. 2019.

[531] H. Xing, L. Liu, J. Xu, and A. Nallanathan, "Joint task assignment and resource allocation for D2D-enabled mobile-edge computing," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4193–4207, Jun. 2019.

[532] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Netw.*, vol. 33, no. 5, pp. 156–165, Sep. 2019.

[533] Y. Zhou, F. R. Yu, J. Chen, and Y. Kuo, "Resource allocation for information-centric virtualized heterogeneous networks with in-network caching and mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 66, no. 12, pp. 11339–11351, Dec. 2017.

[534] C. Alexakos, C. Katsini, K. Votis, A. Lalas, D. Tzovaras, and D. Serpanos, "Enabling digital forensics readiness for Internet of Vehicles," *Transp. Res. Proc.*, vol. 52, pp. 339–346, Jan. 2021.

[535] M. Sheeny, E. De Pellegrin, S. Mukherjee, A. Ahrabian, S. Wang, and A. Wallace, "RADIATE: A radar dataset for automotive perception in bad weather," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 1–7.

[536] Y. Xun, J. Liu, N. Kato, Y. Fang, and Y. Zhang, "Automobile driver fingerprinting: A new machine learning based authentication scheme," *IEEE Trans. Ind. Informat.*, vol. 16, no. 2, pp. 1417–1426, Feb. 2020.

[537] B. Major, D. Fontijne, A. Ansari, R. T. Sukhavasi, R. Gowaikar, M. Hamilton, S. Lee, S. Grzechnik, and S. Subramanian, "Vehicle detection with automotive radar using deep learning on range-azimuth-Doppler tensors," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 924–932.

[538] J. Kocić, N. Jovičić, and Drndarević, "An end-to-end deep neural network for autonomous driving designed for embedded automotive platforms," *Sensors*, vol. 19, no. 9, p. 2064, May 2019.

[539] D. F. Llorca, S. Sánchez, M. Ocaña, and M. A. Sotelo, "Vision-based traffic data collection sensor for automotive applications," *Sensors*, vol. 10, no. 1, pp. 860–875, 2010.

[540] F. Giobergia, E. Baralis, M. Camuglia, T. Cerquitelli, M. Mellia, A. Neri, D. Tricarico, and A. Tuninetti, "Mining sensor data for predictive maintenance in the automotive industry," in *Proc. IEEE 5th Int. Conf. Data Sci. Adv. Analytics (DSAA)*, Oct. 2018, pp. 351–360.

[541] T. Tiedemann, C. Backe, T. Vögele, and P. Conradi, "An automotive distributed mobile sensor data collection with machine learning based data fusion and analysis on a central backend system," *Proc. Technol.*, vol. 26, pp. 570–579, Jan. 2016.

[542] M. A. Rahim, M. A. Rahman, M. M. Rahman, A. T. Asyhari, M. Z. A. Bhuiyan, and D. Ramasamy, "Evolution of IoT-enabled connectivity and applications in automotive industry: A review," *Veh. Commun.*, vol. 27, Jan. 2021, Art. no. 100285.

[543] Y. Liu, A. Liu, T. Wang, X. Liu, and N. N. Xiong, "An intelligent incentive mechanism for coverage of data collection in cognitive Internet of Things," *Future Gener. Comput. Syst.*, vol. 100, pp. 701–714, Nov. 2019.

[544] T. Li, W. Liu, Z. Zeng, and N. N. Xiong, "DRLR: A deep-reinforcement-learning-based recruitment scheme for massive data collections in 6G-based IoT networks," *IEEE Internet Things J.*, vol. 9, no. 16, pp. 14595–14609, Aug. 2022.

[545] W. Mo, W. Liu, G. Huang, N. N. Xiong, A. Liu, and S. Zhang, "A cloud-assisted reliable trust computing scheme for data collection in Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 18, no. 7, pp. 4969–4980, Jul. 2022.

[546] M. T. Nguyen, "Distributed compressive and collaborative sensing data collection in mobile sensor networks," *Internet Things*, vol. 9, Mar. 2020, Art. no. 100156.

[547] T. Tuor, S. Wang, K. K. Leung, and B. J. Ko, "Online collection and forecasting of resource utilization in large-scale distributed systems," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 133–143.

[548] K. A. Khaliq, O. Chughtai, A. Shahwani, A. Qayyum, and J. Pannek, "Road accidents detection, data collection and data analysis using V2X communication and edge/cloud computing," *Electronics*, vol. 8, no. 8, p. 896, Aug. 2019.

[549] T. Wang, J. Zeng, Y. Lai, Y. Cai, H. Tian, Y. Chen, and B. Wang, "Data collection from WSNs to the cloud based on mobile fog elements," *Future Gener. Comput. Syst.*, vol. 105, pp. 864–872, Apr. 2020.

[550] Y. Jiang, N. Zhang, and Z. Ren, "Research on intelligent monitoring scheme for microservice application systems," in *Proc. Int. Conf. Intell. Transp., Big Data Smart City (ICITBS)*, Jan. 2020, pp. 791–794.

[551] M. Waltereit, M. Uphoff, P. Zdankin, V. Matkovic, and T. Weis, "A digital forensic approach for optimizing the investigation of hit-and-run accidents," in *Proc. Int. Conf. Digital Forensics Cyber Crime*. Cham, Switzerland: Springer, 2021, pp. 1–12.

[552] X. Feng, E. S. Dawam, and D. Li, "Autonomous vehicles' forensics in smart cities," in *Proc. IEEE SmartWorld, Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Cloud Big Data Comput., Internet People Smart City Innov. (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, Aug. 2019, pp. 1688–1694.

[553] A. D. Sathe and V. D. Deshmukh, "Advance vehicle-road interaction and vehicle monitoring system using smart phone applications," in *Proc. Online Int. Conf. Green Eng. Technol. (IC-GET)*, Nov. 2016, pp. 1–6.

[554] J. Xiong, R. Bi, M. Zhao, J. Guo, and Q. Yang, "Edge-assisted privacy-preserving raw data sharing framework for connected autonomous vehicles," *IEEE Wireless Commun.*, vol. 27, no. 3, pp. 24–30, Jun. 2020.

[555] G. Rathee, A. Sharma, R. Iqbal, M. Aloqaily, N. Jaglan, and R. Kumar, "A blockchain framework for securing connected and autonomous vehicles," *Sensors*, vol. 19, no. 14, p. 3165, 2019.

[556] Z. Su, Y. Wang, Q. Xu, and N. Zhang, "LVBS: Lightweight vehicular blockchain for secure data sharing in disaster rescue," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 1, pp. 19–32, Jan. 2022.

[557] Q. Chen, X. Ma, S. Tang, J. Guo, Q. Yang, and S. Fu, "F-cooper: Feature based cooperative perception for autonomous vehicle edge computing system using 3D point clouds," in *Proc. 4th ACM/IEEE Symp. Edge Comput.*, Nov. 2019, pp. 1–17.

[558] Q. Chen, S. Tang, Q. Yang, and S. Fu, "Cooper: Cooperative perception for connected autonomous vehicles based on 3D point clouds," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 514–524.

[559] Y. Fu, F. R. Yu, C. Li, T. H. Luan, and Y. Zhang, "Vehicular blockchain-based collective learning for connected and autonomous vehicles," *IEEE Wireless Commun.*, vol. 27, no. 2, pp. 197–203, Apr. 2020.

[560] T. Hidayat and R. Mahardiko, "A systematic literature review method on AES algorithm for data sharing encryption on cloud computing," *Int. J. Artif. Intell. Res.*, vol. 4, no. 1, pp. 49–57, Apr. 2020.

[561] N. Eltayieb, R. Elhabob, A. Hassan, and F. Li, "A blockchain-based attribute-based signcryption scheme to secure data sharing in the cloud," *J. Syst. Archit.*, vol. 102, Jan. 2020, Art. no. 101653.

[562] M. Shen, J. Duan, L. Zhu, J. Zhang, X. Du, and M. Guizani, "Blockchain-based incentives for secure and collaborative data sharing in multiple clouds," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1229–1241, Jun. 2020.

[563] C. Ge, W. Susilo, Z. Liu, J. Xia, P. Szalachowski, and L. Fang, "Secure keyword search and data sharing mechanism for cloud computing," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 6, pp. 2787–2800, Nov. 2021.

[564] H. Jin, Y. Luo, P. Li, and J. Mathew, "A review of secure and privacy-preserving medical data sharing," *IEEE Access*, vol. 7, pp. 61656–61669, 2019.

[565] X. Cheng, F. Chen, D. Xie, H. Sun, and C. Huang, "Design of a secure medical data sharing scheme based on blockchain," *J. Med. Syst.*, vol. 44, no. 2, p. 52, Feb. 2020.

[566] X. Du, S. Tang, Z. Lu, K. Gai, J. Wu, and P. C. K. Hung, "Scientific workflows in IoT environments: A data placement strategy based on heterogeneous edge-cloud computing," *ACM Trans. Manage. Inf. Syst.*, vol. 13, no. 4, pp. 1–26, Dec. 2022.

[567] B. Shen, J. Guo, and Y. Yang, "MedChain: Efficient healthcare data sharing via blockchain," *Appl. Sci.*, vol. 9, no. 6, p. 1207, Mar. 2019.

[568] H. Guo, W. Li, M. Nejad, and C. Shen, "Proof-of-event recording system for autonomous vehicles: A blockchain-based solution," *IEEE Access*, vol. 8, pp. 182776–182786, 2020.

[569] M. Li, J. Weng, J. Liu, X. Lin, and C. Obimbo, "Toward vehicular digital forensics from decentralized trust: An accountable, privacy-preserving, and secure realization," *IEEE Internet Things J.*, vol. 9, no. 9, pp. 7009–7024, May 2022.

[570] N. Vinzenz and T. Eggendorfer, "Proposal for a secure forensic data storage," *J. Cyber Secur. Mobility*, vol. 9, pp. 1–20, Nov. 2020.
[571] A. Mohammad, S. Vargas, and P. Čermák, "Using blockchain for data collection in the automotive industry sector: A literature review," *J. Cybersecurity Privacy*, vol. 2, no. 2, pp. 257–275, Apr. 2022.
[572] R. Jabbar, M. Kharbeche, K. Al-Khalifa, M. Krichen, and K. Barkaoui, "Blockchain for the Internet of Vehicles: A decentralized IoT solution for vehicles communication using ethereum," *Sensors*, vol. 20, no. 14, p. 3928, 2020.
[573] F. Morano, C. Ferretti, A. Leporati, P. Napoletano, and R. Schettini, "A blockchain technology for protection and probative value preservation of vehicle driver data," in *Proc. IEEE 23rd Int. Symp. Consum. Technol. (ISCT)*, Jun. 2019, pp. 167–172.
[574] S. Jain, N. J. Ahuja, P. Srikanth, K. V. Bhadane, B. Nagaiah, A. Kumar, and C. Konstantinou, "Blockchain and autonomous vehicles: Recent advances and future directions," *IEEE Access*, vol. 9, pp. 130264–130328, 2021.
[575] H. Tabrizchi and M. Kuchaki Rafsanjani, "A survey on security challenges in cloud computing: Issues, threats, and solutions," *J. Supercomput.*, vol. 76, no. 12, pp. 9493–9532, Dec. 2020.
[576] J. Wu, L. Ping, X. Ge, Y. Wang, and J. Fu, "Cloud storage as the infrastructure of cloud computing," in *Proc. IEEE Int. Conf. Intell. Comput. Cogn. Inform.*, Sep. 2010, pp. 380–383.
[577] I. Odun-Ayo, O. Ajayi, B. Akanle, and R. Ahuja, "An overview of data storage in cloud computing," in *Proc. Int. Conf. Next Gener. Comput. Inf. Syst. (ICNGCIS)*, Dec. 2017, pp. 29–34.
[578] M. Wang and Q. Zhang, "Optimized data storage algorithm of IoT based on cloud computing in distributed system," *Comput. Commun.*, vol. 157, pp. 124–131, May 2020.
[579] O. I. Khalaf and G. M. Abdulsahib, "Optimized dynamic storage of data (ODSD) in IoT based on blockchain for wireless sensor networks," *Peer-to-Peer Netw. Appl.*, vol. 14, pp. 2858–2873, 2021.
[580] R. Li, T. Song, B. Mei, H. Li, X. Cheng, and L. Sun, "Blockchain for large-scale Internet of Things data storage and protection," *IEEE Trans. Services Comput.*, vol. 12, no. 5, pp. 762–771, Sep. 2019.
[581] W. Liang, Y. Fan, K. Li, D. Zhang, and J. Gaudiot, "Secure data storage and recovery in industrial blockchain network environments," *IEEE Trans. Ind. Informat.*, vol. 16, no. 10, pp. 6543–6552, Oct. 2020.
[582] C. A. Györödi, D. V. Dumşe-Burescu, D. R. Zmaranda, R. C. S. Györödi, G. A. Gabor, and G. D. Pecherle, "Performance analysis of NoSQL and relational databases with CouchDB and MySQL for application's data storage," *Appl. Sci.*, vol. 10, no. 23, p. 8524, 2020.
[583] Z. Lv, X. Li, H. Lv, and W. Xiu, "BIM big data storage in WebVRGIS," *IEEE Trans. Ind. Informat.*, vol. 16, no. 4, pp. 2566–2573, Apr. 2020.
[584] S. Bradshaw, E. Brazil, and K. Chodorow, *MongoDB: The Definitive Guide: Powerful and Scalable Data Storage*. Sebastopol, CA, USA: O'Reilly Media, 2019.
[585] J. Yang, Y. Li, Q. Liu, L. Li, A. Feng, T. Wang, S. Zheng, A. Xu, and J. Lyu, "Brief introduction of medical database and data mining technology in big data era," *J. Evidence-Based Med.*, vol. 13, no. 1, pp. 57–69, Feb. 2020.
[586] V. Kumar, J. Petit, and W. Whyte, "Binary hash tree based certificate access management for connected vehicles," in *Proc. ACM Conf. Secur. Privacy Wireless Mobile Netw.*, 2017, pp. 1–9.
[587] D.-K. Kim, E. Song, and H. Yu, "Introducing attribute-based access control to AUTOSAR," SAE, Warrendale, PA, USA, Tech. Rep. 2016-01-0069, 2016.
[588] M. Rumez, A. Duda, P. Gründer, R. Kriesten, and E. Sax, "Integration of attribute-based access control into automotive architectures," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2019, pp. 1916–1922.
[589] U. Farooq, M. ul Hasan, M. Amar, A. Hanif, and M. Usman Asad, "RFID based security and access control system," *Int. J. Eng. Technol.*, pp. 309–314, 2014.
[590] M. Gupta, J. Benson, F. Patwa, and R. Sandhu, "Dynamic groups and attribute-based access control for next-generation smart cars," in *Proc. ACM Conf. Data Appl. Secur. Privacy*, 2019, pp. 1–12.
[591] A. B. Asif, M. Imran, N. Shah, M. Afzal, and H. Khurshid, "ROCA: Auto-resolving overlapping and conflicts in access control list policies for software defined networking," *Int. J. Commun. Syst.*, vol. 34, no. 9, Jun. 2021.
[592] S. Lachmund and G. Hengst, *Auto-Generation of least Privileges Access Control Policies for Applications Supported by User Input Recognition*. Cham, Switzerland: Springer, 2010.
[593] V. Kabir Veitas and S. Delaere, "In-vehicle data recording, storage and access management in autonomous vehicles," 2018, *arXiv:1806.03243*.
[594] Y. Ren, F. Zhu, J. Qi, J. Wang, and A. K. Sangaiah, "Identity management and access control based on blockchain under edge computing for the industrial Internet of Things," *Appl. Sci.*, vol. 9, no. 10, p. 2058, May 2019.
[595] J. Qiu, Z. Tian, C. Du, Q. Zuo, S. Su, and B. Fang, "A survey on access control in the age of Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 4682–4696, Jun. 2020.
[596] V. C. Hu, D. R. Kuhn, D. F. Ferraiolo, and J. Voas, "Attribute-based access control," *Computer*, vol. 48, no. 2, pp. 85–88, Feb. 2015.
[597] Y. Zhu, D. Huang, C. Hu, and X. Wang, "From RBAC to ABAC: Constructing flexible data access control for cloud storage services," *IEEE Trans. Services Comput.*, vol. 8, no. 4, pp. 601–616, Jul. 2015.
[598] S. Long and L. Yan, "RACAC: An approach toward RBAC and ABAC combining access control," in *Proc. IEEE 5th Int. Conf. Comput. Commun. (ICCC)*, Dec. 2019, pp. 1609–1616.
[599] A. Al-Alaj, R. Krishnan, and R. Sandhu, "SDN-RBAC: An access control model for SDN controller applications," in *Proc. 4th Int. Conf. Comput., Commun. Secur. (ICCCS)*, Oct. 2019, pp. 1–8.
[600] S. Terzi, C. Savvaidis, A. Sersemis, K. Votis, and D. Tzovaras, "Decentralizing identity management and vehicle rights delegation through self-sovereign identities and blockchain," in *Proc. IEEE 45th Annu. Comput., Softw., Appl. Conf. (COMPSAC)*, Jul. 2021, pp. 1217–1223.
[601] N. Kaur, "A study of biometric identification and verification system," in *Proc. IEEE Int. Conf. Advance Comput. Innov. Technol. Eng. (ICACITE)*, Mar. 2021, pp. 60–64.
[602] C. Lupu and V. Lupu, "Multimodal biometrics for access control in an intelligent car," in *Proc. Int. Symp. Comput. Intell. Intell. Informat.*, Mar. 2007, pp. 1–14.
[603] Y. Chen and J. Yin, "Design of electroencephalogram authentication access control to smart car," *Healthcare Technol. Lett.*, vol. 7, no. 4, pp. 109–113, Aug. 2020.
[604] G. Zhang and W. Gong, "The research of access control based on UCON in the Internet of Things," *J. Softw.*, vol. 6, no. 4, Apr. 2011.
[605] D. D. F. Maesa, P. Mori, and L. Ricci, "A blockchain based approach for the definition of auditable access control systems," in *Comput. Secur.*, vol. 84, pp. 93–119, Jul. 2019.
[606] G. Ali, N. Ahmad, Y. Cao, M. Asif, H. Cruickshank, and Q. E. Ali, "Blockchain based permission delegation and access control in Internet of Things (BACI)," *Comput. Secur.*, vol. 86, pp. 318–334, Sep. 2019.
[607] B. Tang, H. Kang, J. Fan, Q. Li, and R. Sandhu, "IoT passport: A blockchain-based trust framework for collaborative internet-of-things," in *Proc. ACM Symp. Access Control Models Technol.*, 2019, pp. 1–17.
[608] S. Jajodia, P. Samarati, M. L. Sapino, and V. S. Subrahmanian, "Flexible support for multiple access control policies," *ACM Trans. Database Syst.*, vol. 26, no. 2, pp. 214–260, Jun. 2001.
[609] Yingcheng. Wang and Daniel. T. Gladwin, "Power management of EV car parks," in *Proc. 45th Annu. Conf. IEEE Ind. Electron. Soc.*, vol. 1, Oct. 2019, pp. 4316–4322.
[610] G. Zhong, J. Wen, X. Li, J. Hu, and M. Yang, "Pioneering low cost solution for power amplifiers managing car engine start-stop feature," in *Proc. 10th IEEE Int. Conf. Solid-State Integr. Circuit Technol.*, Nov. 2010, pp. 141–143.
[611] Z. Al-Saadi, D. Phan Van, A. Moradi Amani, M. Fayyazi, S. Sadat Sajjadi, D. Ba Pham, R. Jazar, and H. Khayyam, "Intelligent driver assistance and energy management systems of hybrid electric autonomous vehicles," *Sustainability*, vol. 14, no. 15, p. 9378, 2022.
[612] S. A. Sajadi-Alamdari, H. Voos, and M. Darouach, "Ecological advanced driver assistance system for optimal energy management in electric vehicles," *IEEE Intell. Transp. Syst. Mag.*, vol. 12, no. 4, pp. 92–109, Winter. 2020.
[613] I. Iglesias, L. Isasi, M. Larburu, V. Martinez, and B. Molinete, "I2V communication driving assistance system: On-board traffic light assistant," in *Proc. IEEE 68th Veh. Technol. Conf.*, Sep. 2008, pp. 1–14.
[614] A. Joshi, P. Gaonkar, and J. Bapat, "A reliable and secure approach for efficient car-to-car communication in intelligent transportation systems," in *Proc. Int. Conf. Wireless Commun., Signal Process. Netw. (WiSPNET)*, Mar. 2017, pp. 1617–1620.
[615] H. Hontani and Y. Higuchi, "Vehicle positioning method with car-to-car communications in consideration of communication delay," in *Proc. 5th Int. Conf. Networked Sens. Syst.*, Jun. 2008, pp. 1–9.

[616] M. Braun, F. Weber, and F. Alt, "Affective automotive user interfaces–reviewing the state of driver affect research and emotion regulation in the car," in *Proc. ACM Comput. Surveys (CSUR)*, 2021, pp. 1–7.

[617] M. Braun, J. Li, F. Weber, B. Pfleging, A. Butz, and F. Alt, "What if your car would care? Exploring use cases for affective automotive user interfaces," in *Proc. 22nd Int. Conf. Human-Computer Interact. with Mobile Devices Services*, Oct. 2020, pp. 1–8.

[618] S. Urooj, I. Feroz, and N. Ahmad, "Systematic literature review on user interfaces of autonomous cars: Liabilities and responsibilities," in *Proc. Int. Conf. Advancements Comput. Sci. (ICACS)*, Feb. 2018, pp. 1–10.

[619] D. Kern and A. Schmidt, "Design space for driver-based automotive user interfaces," in *Proc. 1st Int. Conf. Automot. User Interfaces Interact. Veh. Appl.*, Sep. 2009, pp. 1–10.

[620] M. Nabi, M. Toeroe, and F. Khendek, "Availability in the cloud: State of the art," *J. Netw. Comput. Appl.*, vol. 60, pp. 54–67, Jan. 2016.

[621] V. S. Sharma and A. Santharam, "Implementing a resilient application architecture for state management on a PaaS cloud," in *Proc. IEEE 5th Int. Conf. Cloud Comput. Technol. Sci.*, vol. 1, Dec. 2013, pp. 142–147.

[622] E. Ahmed, A. Naveed, A. Gani, S. H. Ab Hamid, M. Imran, and M. Guizani, "Process state synchronization-based application execution management for mobile edge/cloud computing," *Future Gener. Comput. Syst.*, vol. 91, pp. 579–589, Feb. 2019.

[623] M. Teichmann, M. Weber, M. Zöllner, R. Cipolla, and R. Urtasun, "MultiNet: real-time joint semantic reasoning for autonomous driving," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2018, pp. 1013–1020.

[624] J. Cui, L. S. Liew, G. Sabaliauskaite, and F. Zhou, "A review on safety failures, security attacks, and available countermeasures for autonomous vehicles," *Ad Hoc Netw.*, vol. 90, Jul. 2019, Art. no. 101823.

[625] S. Parkinson, P. Ward, K. Wilson, and J. Miller, "Cyber threats facing autonomous and connected vehicles: Future challenges," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 11, pp. 2898–2915, Nov. 2017.

[626] A. Humayed, F. Li, J. Lin, and B. Luo, "Cansentry: Securing can-based cyber-physical systems against denial and spoofing attacks," in *Proc. Eur. Symp. Res. Comput. Secur.* Cham, Switzerland: Springer, 2020, pp. 1–12.

[627] M. Buinevich and A. Vladyko, "Forecasting issues of wireless communication networks' cyber resilience for an intelligent transportation system: An overview of cyber attacks," *Information*, vol. 10, no. 1, p. 27, Jan. 2019.

[628] R. Gupta, S. Tanwar, N. Kumar, and S. Tyagi, "Blockchain-based security attack resilience schemes for autonomous vehicles in industry 4.0: A systematic review," *Comput. Electr. Eng.*, vol. 86, Sep. 2020, Art. no. 106717.

[629] F. Sommer, J. Dürrwang, and R. Kriesten, "Survey and classification of automotive security attacks," *Information*, vol. 10, no. 4, p. 148, Apr. 2019.

[630] A. Bazzi, B. M. Masini, A. Zanella, and I. Thibault, "On the performance of IEEE 802.11p and LTE-V2V for the cooperative awareness of connected vehicles," *IEEE Trans. Veh. Technol.*, vol. 66, no. 11, pp. 10419–10432, Nov. 2017.

[631] A. Chowdhury, G. Karmakar, J. Kamruzzaman, A. Jolfaei, and R. Das, "Attacks on self-driving cars and their countermeasures: A survey," *IEEE Access*, vol. 8, pp. 207308–207342, 2020.

[632] J. L. Pisarov and G. Mester, "Self-driving robotic cars: Cyber security developments," in *Research Anthology on Cross-Disciplinary Designs and Applications of Automation*. Hershey, PA, USA: IGI Global, 2022.

[633] E.-J. Jang and S.-J. Shin, "Proposal of new data processing function to improve the security of self-driving cars' systems," *J. Inst. Internet, Broadcast. Commun.*, vol. 20, no. 4, pp. 81–86, 2020.

[634] F. Pascale, E. A. Adinolfi, S. Coppola, and E. Santonicola, "Cybersecurity in automotive: An intrusion detection system in connected vehicles," *Electronics*, vol. 10, p. 1765, Jun. 2021.

[635] P. Zhuang, T. Zamir, and H. Liang, "Blockchain for cybersecurity in smart grid: A comprehensive survey," *IEEE Trans. Ind. Informat.*, vol. 17, no. 1, pp. 3–19, Jan. 2021.

[636] K. Shaukat, S. Luo, V. Varadharajan, I. A. Hameed, and M. Xu, "A survey on machine learning techniques for cyber security in the last decade," *IEEE Access*, vol. 8, pp. 222310–222354, 2020.

[637] J. Nikander, O. Manninen, and M. Laajalahti, "Requirements for cybersecurity in agricultural communication networks," *Comput. Electron. Agricult.*, vol. 179, Dec. 2020, Art. no. 105776.

[638] R. O. Andrade, S. G. Yoo, L. Tello-Oquendo, and I. Ortiz-Garcés, "A comprehensive study of the IoT cybersecurity in smart cities," *IEEE Access*, vol. 8, pp. 228922–228941, 2020.

[639] C. Cheng, G. Srivastava, J. C. Lin, and Y. Lin, "Fault-tolerance mechanisms for software-defined Internet of vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 6, pp. 3859–3868, Jun. 2021.

[640] A. Bhat, "Practical solutions for fault-tolerance in connected and autonomous vehicles (CAVS)," Ph.D. dissertation, Elect. Comput. Eng., Carnegie Mellon Univ., Pittsburgh, PA, USA, 2019.

[641] J. Lex, U. Margull, D. Fey, and R. Mader, "Fault tolerance in heterogeneous automotive real-time systems," in *Echtzeit*. Cham, Switzerland: Springer, 2022.

[642] E. Malekzadeh, "A study on fault-tolerance of deep neural networks for embedded systems," 2021.

[643] I. Gräßler, E. Bodden, J. Pottebaum, J. Geismann, and D. Roesmann, "Security-oriented fault-tolerance in systems engineering: A conceptual threat modelling approach for cyber-physical production systems," in *Advanced, Contemporary Control*. Cham, Switzerland: Springer, 2020.

[644] A. Bhat, S. Samii, and R. R. Rajkumar, "Fault-tolerance support for adaptive AUTOSAR platforms using SOME/IP," in *Proc. IEEE 26th Int. Conf. Embedded Real-Time Comput. Syst. Appl. (RTCSA)*, Aug. 2020, pp. 1–6.

[645] P. Kumari and P. Kaur, "A survey of fault tolerance in cloud computing," *J. Comput. Inf. Sci.*, vol. 33, no. 10, pp. 1159–1176, Dec. 2021.

[646] S. Bharany, S. Badotra, S. Sharma, S. Rani, M. Alazab, R. H. Jhaveri, and T. R. Gadekallu, "Energy efficient fault tolerance techniques in green cloud computing: A systematic survey and taxonomy," *Sustain. Energy Technol. Assessments*, vol. 53, Oct. 2022, Art. no. 102613.

[647] N. Gupta and N. H. Vaidya, "Fault-tolerance in distributed optimization: The case of redundancy," in *Proc. 39th Symp. Princ. Distrib. Comput.*, Jul. 2020, pp. 365–374.

[648] Y. Xiao, N. Zhang, J. Li, W. Lou, and Y. T. Hou, "Distributed consensus protocols and algorithms," in *Blockchain for Distributed Systems Security*. Hoboken, NJ, USA: Wiley, 2019.

[649] R. Kaur and P. Luthra, "Load balancing in cloud system using max min and min min algorithm," *Int. J. Comput. Appl.*, vol. 975, p. 8887, 2014.

[650] M. Mesbahi and A. Masoud Rahmani, "Load balancing in cloud computing: A state of the art survey," *Int. J. Modern Educ. Comput. Sci.*, vol. 8, no. 3, pp. 64–78, Mar. 2016.

[651] N. Thapliyal and P. Dimri, "Load balancing in cloud computing based on honey bee foraging behavior and load balance min-min scheduling algorithm," *Int. J. Electr. Electron. Res.*, vol. 10, no. 1, pp. 1–6, Mar. 2022.

[652] D. Patel and A. S. Rajawat, "Efficient throttled load balancing algorithm in cloud environment," *Int. J. Modern Trends Eng. Res.*, vol. 34, p. e7208, Oct. 2015.

[653] K. Nishant, P. Sharma, V. Krishna, C. Gupta, K. P. Singh, Nitin, and R. Rastogi, "Load balancing of nodes in cloud using ant colony optimization," in *Proc. UKSim 14th Int. Conf. Comput. Model. Simul.*, Mar. 2012, pp. 3–8.

[654] Q. Liu, T. Xia, L. Cheng, M. van Eijk, T. Ozcelebi, and Y. Mao, "Deep reinforcement learning for load-balancing aware network control in IoT edge systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 6, pp. 1491–1502, Jun. 2022.

[655] D. Majumder, S. Mohan Kumar, D. V. Ashoka, and A. S. Naragunam, "Edge computer-enabled Internet of Vehicle applications with secure computing and load balancing," *J. Phys., Conf.*, vol. 1964, no. 4, Jul. 2021, Art. no. 042015.

[656] Y. Wu, J. Wu, L. Chen, J. Yan, and Y. Han, "Load balance guaranteed vehicle-to-vehicle computation offloading for min-max fairness in VANETs," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 11994–12013, Aug. 2022.

[657] U. Ahmed, J. C. Lin, G. Srivastava, U. Yun, and A. K. Singh, "Deep active learning intrusion detection and load balancing in software-defined vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 1, pp. 953–961, Jan. 2023.

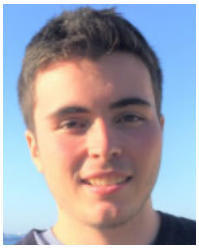
[658] A. A. Syed, S. Ayaz, T. Leinmüller, and M. Chandra, "MIP-based joint scheduling and routing with load balancing for TSN based in-vehicle networks," in *Proc. IEEE Veh. Netw. Conf. (VNC)*, Dec. 2020, pp. 1–7.

[659] V. Yemelyanov, T. Tochilkina, A. Nedelkin, and E. Shved, "Automation of monitoring and diagnosing the technical condition of torpedo ladle cars," in *Proc. MATEC Web Conf.*, 2018, pp. 1–11.

[660] R. N. Charette, "This car runs on code," *IEEE Spectr.*, vol. 46, no. 3, pp. 1–6, Feb. 2009.

[661] S. A. Hameed, O. Khalifa, M. Ershad, F. Zahudi, B. Sheyaa, and W. Asender, "Car monitoring, alerting and tracking model: Enhancement with mobility and database facilities," in *Proc. Int. Conf. Comput. Commun. Eng. (ICCCE)*, May 2010, pp. 1–10.
[662] S. A. Hameed, S. Abdulla, M. Ershad, F. Zahudi, and A. Hassan, "New automobile monitoring and tracking model: Facilitate model with handhelds," in *Proc. 4th Int. Conf. Mechatronics (ICOM)*, May 2011, pp. 1–5.
[663] A. Norouzi, H. Heidarifar, H. Borhan, M. Shahbakhti, and C. R. Koch, "Integrating machine learning and model predictive control for automotive applications: A review and future directions," *Eng. Appl. Artif. Intell.*, vol. 120, Apr. 2023, Art. no. 105878.
[664] S. Kuutti, R. Bowden, Y. Jin, P. Barber, and S. Fallah, "A survey of deep learning applications to autonomous vehicle control," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 2, pp. 712–733, Feb. 2021.
[665] G. Georgakos, U. Schlichtmann, R. Schneider, and S. Chakraborty, "Reliability challenges for electric vehicles: From devices to architecture and systems software," in *Proc. 50th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, May 2013, pp. 1–9.
[666] D. L. Parnas, "Software aging," in *Proc. 16th Int. Conf. Softw. Eng.*, 1994, pp. 1–5.
[667] J. Costa, R. Matos, J. Araujo, J. Li, E. Choi, T. A. Nguyen, J.-W. Lee, and D. Min, "Software aging effects on kubernetes in container orchestration systems for digital twin cloud infrastructures of urban air mobility," *Drones*, vol. 7, no. 1, p. 35, Jan. 2023.
[668] M. S. Aktas, "Hybrid cloud computing monitoring software architecture," *Concurrency Comput., Pract. Exper.*, vol. 33, no. 21, p. e4694, 2018.
[669] J. Spring, "Monitoring cloud computing by layer, part 1," *IEEE Secur. Privacy*, vol. 9, no. 2, pp. 66–68, Mar. 2011.
[670] D. A. Tamburri, M. Miglierina, and E. D. Nitto, "Cloud applications monitoring: An industrial study," *Inf. Softw. Technol.*, vol. 127, Nov. 2020, Art. no. 106376.
[671] J. Shao, H. Wei, Q. Wang, and H. Mei, "A runtime model based monitoring approach for cloud," in *Proc. IEEE 3rd Int. Conf. Cloud Comput.*, Jul. 2010, pp. 1–15.
[672] T. Maksymyuk, S. Dumych, M. Brych, D. Satria, and M. Jo, "An IoT based monitoring framework for software defined 5G mobile networks," in *Proc. 11th Int. Conf. Ubiquitous Inf. Manage. Commun.*, Jan. 2017, pp. 1–12.
[673] A. Razzaq, "A systematic review on software architectures for IoT systems and future direction to the adoption of microservices architecture," *Social Netw. Comput. Sci.*, vol. 1, p. 350, Oct. 2020.
[674] I. Rakhmonov and N. Kurbonov, "Analysis of automated software for monitoring energy consumption and efficiency of industrial enterprises," in *Proc. E3S Web Conf.*, 2020, pp. 1–6.
[675] S. He, W. Ren, T. Zhu, and K. R. Choo, "BoSMoS: A blockchain-based status monitoring system for defending against unauthorized software updating in industrial Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 2, pp. 948–959, Feb. 2020.
[676] K. McCord, *Automotive Diagnostic Systems: Understanding OBD I and OBD II*. Minneapolis, MI, USA: CarTech, 2011.
[677] D. Rimpas, A. Papadakis, and M. Samarakou, "OBD-II sensor diagnostics for monitoring vehicle operation and consumption," *Energy Rep.*, vol. 6, no. 3, pp. 55–63, 2020.
[678] M. Mesgarpour, D. Landa-Silva, and I. Dickinson, "Overview of telematics-based prognostics and health management systems for commercial vehicles," in *Proc. Int. Conf. Transp. Syst. Telematics*, 2013, pp. 1–12.
[679] M. Singh, R. K. Dubey, and S. Kumar, "Vehicle telematics: An Internet of Things and big data approach," in *Artificial Intelligence and Machine Learning for Edge Computing*. Amsterdam, The Netherlands: Elsevier, 2022.
[680] A. Theissler, J. Pérez-Velázquez, M. Kettelgerdes, and G. Elger, "Predictive maintenance enabled by machine learning: Use cases and challenges in the automotive industry," *Rel. Eng. Syst. Saf.*, vol. 215, Nov. 2021, Art. no. 107864.
[681] G. Bhatti, H. Mohan, and R. R. Singh, "Towards the future of smart electric vehicles: Digital twin technology," in *Renew. Sustain. Energy Rev.*, vol. 141, May 2021, Art. no. 110801.
[682] M. Schindewolf, H. Stoll, H. Guissouma, A. Puder, E. Sax, A. Vetter, M. Rumez, and J. Henle, "A comparison of architecture paradigms for dynamic reconfigurable automotive networks," in *Proc. Int. Conf. Connected Vehicle Expo. (ICCVE)*, Mar. 2022, pp. 1–12.
[683] A. Ioana, A. Korodi, and I. Silea, "Automotive IoT Ethernet-based communication technologies applied in a V2X context via a multi-protocol gateway," *Sensors*, vol. 22, no. 17, p. 6382, Aug. 2022.
[684] S. Kim and R. Shrestha, *Automotive Cyber Security*. Singapore: Springer, 2020.
[685] M. O. Metais, O. Jouini, Y. Perez, J. Berrada, and E. Suomalainen, "Too much or not enough? Planning electric vehicle charging infrastructure: A review of modeling options," *Renew. Sustain. Energy Rev.*, vol. 153, Jan. 2022, Art. no. 111719.
[686] T. Chen, X. Zhang, J. Wang, J. Li, C. Wu, M. Hu, and H. Bian, "A review on electric vehicle charging infrastructure development in the UK," *J. Modern Power Syst. Clean Energy*, vol. 8, no. 2, pp. 193–205, Mar. 2020.
[687] M.-K. Tran, A. Bhatti, R. Vrolyk, D. Wong, S. Panchal, M. Fowler, and R. Fraser, "A review of range extenders in battery electric vehicles: Current progress and future perspectives," *World Electric Vehicle J.*, vol. 12, no. 2, p. 54, Apr. 2021.
[688] S. Singh and B. S. Saini, "Autonomous cars: Recent developments, challenges, and possible solutions," *IOP Conf. Mater. Sci. Eng.*, vol. 1022, Oct. 2021, Art. no. 012028.
[689] O. Hagman and J. Lindh, *How Autonomous Cars can Affect the car Industry-Implications for User Experience and Competition*, 2019.
[690] F. Cugurullo, R. A. Acheampong, M. Gueriau, and I. Dusparic, "The transition to autonomous cars, the redesign of cities and the future of urban sustainability," *Urban Geography*, vol. 42, no. 6, pp. 833–859, 2021.
[691] A. Albatayneh, M. N. Assaf, D. Alterman, and M. Jaradat, "Comparison of the overall energy efficiency for internal combustion engine vehicles and electric vehicles," in *Rigas Tehniskas Universitates Zinatniskie Raksti*. Riga, Latvia: Riga Technical University, 2020.
[692] Q. Demlehner, D. Schoemer, and S. Laumer, "How can artificial intelligence enhance car manufacturing? A delphi study-based identification and assessment of general use cases," *Int. J. Inf. Manage.*, vol. 58, Jun. 2021, Art. no. 102317.
[693] E. Staniszewska, D. Klimecka-Tatar, and M. Obrecht, "Eco-design processes in the automotive industry," *Prod. Eng. Arch.*, vol. 26, no. 4, pp. 131–137, Dec. 2020.
[694] O. Kaya, K. D. Alemdar, A. Atalay, M. Y. Çodur, and A. Tortum, "Electric car sharing stations site selection from the perspective of sustainability: A GIS-based multi-criteria decision making approach," *Sustain. Energy Technol. Assessments*, vol. 52, Aug. 2022, Art. no. 102026.
[695] E. de Bellis, C. Hildebrand, K. Ito, A. Herrmann, and B. Schmitt, "Personalizing the customization experience: A matching theory of mass customization interfaces and cultural information processing," *J. Marketing Res.*, vol. 56, no. 6, pp. 1050–1065, Dec. 2019.
[696] M. Hasenjäger, M. Heckmann, and H. Wersing, "A survey of personalization for advanced driver assistance systems," *IEEE Trans. Intell. Vehicles*, vol. 5, no. 2, pp. 335–344, Jun. 2020.
[697] C. Atkinson. (2022). *Is Personalization the key to Successful Automotive Experiences?* [Online]. Available: https://www.sbdautomotive.com/post/is-personalization-the-key-to-successful-automotive-experiences
[698] S. Hussain, U. Mahmud, and S. Yang, "Car e-talk: An IoT-enabled cloud-assisted smart fleet maintenance system," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9484–9494, Jun. 2021.
[699] F. Arena, M. Collotta, L. Luca, M. Ruggieri, and F. G. Termine, "Predictive maintenance in the automotive sector: A literature review," *Math. Comput. Appl.*, vol. 27, no. 1, p. 2, 2022.
[700] M. A. der Mauer, T. Behrens, M. Derakhshanmanesh, C. Hansen, and S. Muderack, "Applying sound-based analysis at porsche production: Towards predictive maintenance of production machines using deep learning and Internet-of-Things technology," in *Digitalization Cases: How Organizations Rethink Their Business for the Digital Age*. Cham, Switzerland: Springer, 2019.
[701] J. Woo, J. Shin, H. Kim, and H. Moon, "Which consumers are willing to pay for smart car healthcare services? A discrete choice experiment approach," *J. Retailing Consum. Services*, vol. 69, Nov. 2022, Art. no. 103084.
[702] W. Jiao, W. Huang, and H. Fan, "Evaluating spatial accessibility to healthcare services from the lens of emergency hospital visits based on floating car data," *Int. J. Digit. Earth*, vol. 15, no. 1, pp. 108–133, Dec. 2022.
[703] S. A. Cohen and D. Hopkins, "Autonomous vehicles and the future of urban tourism," *Ann. Tourism Res.*, vol. 74, pp. 33–42, Jan. 2019.

[704] M. Frackiewicz. (2022). *The Potential of Smart Cities for Smart Agriculture and Food Systems*. [Online]. Available: https://ts2.space/en/the-potential-of-smart-cities-for-smart-agriculture-and-food-systems/
[705] T Group. (2022). *A Mobile Library to Make Culture Accessible to Everyone*. [Online]. Available: https://www.toutenkamion-group.com/en/news/news-reader/mobile-library-herault.html
[706] C. Schartmüller, S. Sarcar, A. Riener, A. L. Kun, O. Shaer, L. N. Boyle, and S. Iqbal, "Automated cars as living rooms and offices: Challenges and opportunities," in *Proc. Extended Abstr. CHI Conf. Human Factors Comput. Syst.*, Apr. 2020, pp. 1–11.
[707] C. P. Janssen, A. L. Kun, S. Brewster, L. N. Boyle, D. P. Brumby, and L. L. Chuang, "Exploring the concept of the (future) mobile office," in *Proc. 11th Int. Conf. Automot. User Interfaces Interact. Veh. Appl., Adjunct*, Sep. 2019, pp. 1–27.
[708] F. Samouh, V. Gluza, S. Djavadian, S. Meshkani, and B. Farooq, "Multimodal autonomous last-mile delivery system design and application," in *Proc. IEEE Int. Smart Cities Conf. (ISC)*, Sep. 2020, pp. 1–7.
[709] C. Wienrich and K. Schindler, "Challenges and requirements of immersive media in autonomous car: Exploring the feasibility of virtual entertainment applications," *I-Com*, vol. 18, no. 2, pp. 105–125 2019.
[710] E. Karanastasis, E. Chondrogiannis, and S. Papasotiriou, "A novel AR application for in-vehicle entertainment and education," in *Proc. 11th Int. Conf. Virtual Worlds Games Serious Appl. (VS-Games)*, Sep. 2019, pp. 1–4.



DAVID FERNÁNDEZ BLANCO received the integrated B.S. and M.Eng. degrees in telecommunications engineering from the National Institute of Applied Sciences of Lyon (INSA Lyon), France, in 2020, and the parallel specialization Diploma degree in research and development from INSA Lyon. He is currently pursuing the industrial Ph.D. degree in computer science with INSA Lyon in collaboration with Stellantis. Prior to the beginning of the Ph.D., he was a Research Engineer with the CITI Laboratory for a year and an Associate Lecturer and a Professor with CPE Lyon and INSA Lyon. His research interests include the development of new-generation embedded software architectures and middleware for the automotive sector, the coordination with the cloud (classic, fog, and edge), and the development and design of complex distributed systems architectures, such as blockchain storage layers.

FRÉDÉRIC LE MOUËL is a Full Professor with the National Institute for Applied Sciences of Lyon (INSA Lyon), France, a leading engineering school in France, part of the University of Lyon. He is heading the Center for Innovation in Telecommunication and Integration of Services (INSA/Inria CITI Lab) and leading the Dynamic Software and Distributed Systems for the Internet of Things Research Group (DynaMid Team).

Since 2022, he has been the SPIE/INSA Research Chair of Edge AI. He is especially animating the research topic on geo-distributing data and resources for federated learning. From 2019 to 2022, he was the SPIE/INSA Research Chair of the Internet of Things (IoT). He was especially animating the research topic on the IoT large-scale deployments. He has published more than 100 publications in international journals and conferences. His main interests are distributed systems, operating systems, component and service-oriented middleware, virtual machines, programming languages, and more specifically dynamic adapting, self-coordinating, and autonomic environments. He is especially studying these topics in the domain of ambient intelligence, the IoT, home automation, and vehicular networks. He is a member of several conference committees.

TRISTA LIN received the B.S. degree in mathematics and communications engineering from National Tsing Hua University, Taiwan, and the Ph.D. degree in computer science from the National Institute of Applied Sciences of Lyon (INSA Lyon), France. She has been an Onboard IT Architect and a Technical Specialist with the E/E Architecture Design Department, Stellantis, since 2018. She is responsible for TCP/IP architecture design and protocol deployment. Prior to Stellantis, she has eight years of industry experience in wireless network simulators and three years of research experience in open-source software development. Her research interests include IT solution adaptation for cars toward software-defined architectures and services.

MARIE-PIERRE ESCUDIÉ is a Full Professor of humanities and social sciences with the National Institute of Applied Sciences of Lyon (INSA Lyon, France), in particular, the ethics and responsibility of engineers. Besides, she carries out her research on the social responsibility of engineers at the Gaston Berger Institute and collaborates with the Center for Innovation in Telecommunication and Integration of Services (INSA/Inria CITI Lab) on frugal computing and emancipatory security, helping ICT go through the ecological transition from an ethic point of view. Her research interests include the links between science, technology, and society; the social responsibility of engineers; the humanism and ethics of the engineer.