Electrical/electronic architectures, a survey

Jean-Baptiste LAURENT
Forvia, ENSTA
Paris, France
jeanbaptiste.laurent.pro@protonmail.com

Julien RICHEFEU
Forvia
Paris, France
julien.richefeu@forvia.com

Bruno MONSUEZ
ENSTA
Paris, France
bruno.monsuez@ensta-paris.fr

Abstract: With the rapidity at which technologies are evolving, it comes without surprises that the embedded automotive system design would tend to become more and more unfit at fulfilling modern expectations. Previous researches have already highlighted the different limitations of the existing system, while proposing solutions for one that would perform well and support features such as SoA [8], [4] [14], [6]. However, even though they share most of the same high-level innovative traits (such as sharing ressources and increasing modularity), properly defined steps to build, or assist the design of such systems is still sparse [15]. This survey intends to provide a comprehensive compilation of the E/E architecture subject, with a focus on the limitation implied by manual design.

Keywords: E/E architectures, survey, system design, tooling, design space exploration

INTRODUCTION

Automotive E/E architectures are shifting toward new designs models to respond to the market increasing needs.

As a result, several clearly defined "megatrends" emerged to shape the ongoing trends in addition to a wide range of resources. This is allowing the involved actors to take safer development decisions based on previously made research.

Moreover, even though the need to switch architecture is clear (i.e. glass ceiling reached due to a deprecated design), the means to reach it remains unclear.

The aims of this paper is to provide a bird compilation of the evolution of E/E automotive architectures, and highlight a complementary view regarding a side, lesser seen subject regarding the development of tools to assist such a transition.

In order to do that, the current article is organized as follow: section 1 will present the subject with the intention of giving a newcomer a fair view of the relevant parts linked to automotive E/E architectures, and the motivations behind the transition. Then, within section 2, the main trends will be separately discussed such as zonal and domain oriented architectures. This will be followed in section 3 by a use-case where three sub-systems are designed independently. This will introduce the potential for tools presented in section 4 as an alternate research path to smooth the challenging task of choosing a relevant between competing designs, and committing to it in the long run. At last, section 5 will concludes this review article and list research opportunities opportunities worth considering.

1. AUTOMOTIVE E/E ARCHITECTURES

> `A system architecture is the conceptual model that defines the structure, behavior, and more views of a system [...] it may include information such as a detailed inventory of current hardware, software and networking capabilities` [2]

In our automotive case, this translates to a network of diverse and self-sufficient components linked to each others, such as: sensors and processing units, along with the intelligence to orchestrate the whole. However, even though it shall be summarized that way, modern cars require more than hundreds of different components and several times that amount on wire.

To save a bit on the latter and for simplicity, components are often connected to a shared link (such as a Controller Area Network (CAN) bus) where exchanged messages are broadcasted and transmitted to every recipients.

Yet, with the increasing demand placed on car makers by both regulations and customers, the complexity embedded into a single car is bound to exponentially explode. At least, if the current distributed architecture is kept.

For example, it will be soon legally required to host a Driver monitoring system (DMS) system. In case drowsiness is detected, its messages will have to be exchanged to one or several other components which in turn could produce a clue or transmit the message further if an action is needed.

This addition is going to increase the stress placed on every shared bus in the path and might require several new wire connections. This is especially true if, for safety reasons, the multiple network hops between the drowsiness detection and the end components takes too much time and can't make up for the hard timing safety requirements.

Subsequently, the cost for manufacturing a car is increasing on every part of the chain, and efforts, from all suppliers, are made to stay competitive. Either from standardization and normalized development processes [12], modular and generic EOM components to increase re-usability, to more specialized or context dependent equipment such as DCUs (Domain-control-units) or zonals [13].

Moving from a distributed design to a more centralized one opens opportunities to share hardware resources, and reduce the amount of equipment in need to be directly connected. It has its drawbacks as well, such as the introduction of single-point-of-failures (SPOF) and the increased cost for changing a merged unit if one of its aggregated part needs replacement. In like manner, centralization and/or segregation will help to mitigate cyber security threats and reduce the attack surface. [10] [9]

Other constraints also have to be taken into account when dealing with E/E architectures, essentially its generic nature and resilience against deprecation (e.g. being able to use a single design on multiple cars, over time). With the emerging need for car connectivity to its surrounding (V2X) or autonomous driving (ADAS), this also implies that cars shall be part of a wider ecosystem. As a result, the architecture should not be only designed with the assumption that the car is its only concern anymore, but as part of a bigger picture.

2. Major related evolutions (Megatrends)

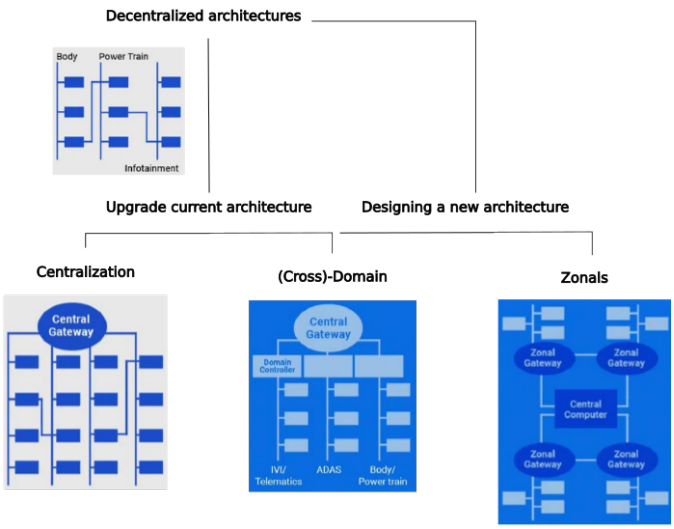


Fig. 1. Architectural alternatives

Fully distributed

In a distributed architecture, components are designed to host a single task. They tend to be small, simple, and auto sufficient but also many, along with a need to get individually connected to the network. However, even though a decentralized architecture makes it simple at the component level, it makes it hard to share commonly needed resources (such as processing power or wiring). It also forces higher level emerging features to be spread among several components. This, in turn, nullify the advantages of such design: developers are now dealing with distributed/concurrent programming issues, and architects have to sort out large amount of dependencies.

Furthermore, the complexity added with every new component is non-linear as each are in need to be linked to many more others or reroute network traffic. As stated in [11], this contributes into making wires the third heaviest component embedded onto a car and makes the integration and validation process no longer straightforward as well.

Additionally, automotive distributed networks historically come with a bus network topology. This works very well for small networks but suffers many scaling issues [1].

Overall, the increasing cost, including time, for keeping up and support higher level features, (at both the components and topological levels) motivated the need to switch to a more effective architecture.

Fully centralized: Merging

Domain and cross-domain: Logical grouping

Since issues raised from the addition of higher level features, it made sense that the first centralization steps were made into logically grouping the components associated with such a feature together onto “domains”. Those domains are parented by a dedicated device bridging the sub-components to the rest of the system.

A “cross”-domain direct link or merge is also viable to group up features that could still be present within different domains, or for domains to efficiently communicate with one another when cooperation is required.

This fixes most of the distributed design issues, such as redundancy, the explosion of network links, and scattered features. Nevertheless, even higher logical features, such as ADAS, are still in need to access a large amount of information located all around the system. In that case, segregation by domain will lead to issues regarding the physical layout of the system: important amount of wire to link physically far components onto the same logical unit, increased average amount of network hops, and finally the remaining need for components of different domains to communicate with each others when dealing with even higher level of functionality.

Zonals: Physical grouping

Grouping up by domains has its pros, but ultimately still suffers the same distributed architecture issues when dealing with system wide functionality, on top of creating a fracture between the logical topology and its corresponding physical topography. In that regard a third architecture design centered around the concept of “zone” is currently being explored.

A zonal oriented architecture is very similar to the domain one (additional parent unit bridging the group, hosting redundant features, etc), but focuses instead on grouping up physically close components together. It is effectively removing the fracture between the topology and topography.

An other major advantage of this layout is the ability to also regroup components and simplify the physical topology. Furthermore it is not conflicting by nature with the domain design allowing the two to be used concurrently.

Yet, there are many open questions regarding details such as the optimal amount of gates, their role(s), network layout,

configuration, and ability to fit when dealing with automotive embedded systems constraints (real time, safety, heat, etc).

3. Designing a system

The most straightforward way to design a system is to start with a simple part, and challenge the result with constraints if relevant to validate the process. This method is then repeated until every component has been included to the overall picture.

Usually there is a processing capable component, paired with a sensor and an actuator to close a given use case. The physical and logical mean for them to communicate and cooperate is then established. Once done, the process can be repeated by designing an other set of those to complete different tasks and so on. Individually, those are fairly simple to visualize, and the easiest way to bind them together is to add a dedicated, central component in charge of providing a unique UI to forward the commands issued by driver.

One could also link them directly, or even keep them completely separated and then add physical buttons for each, but that would be very inconvenient.

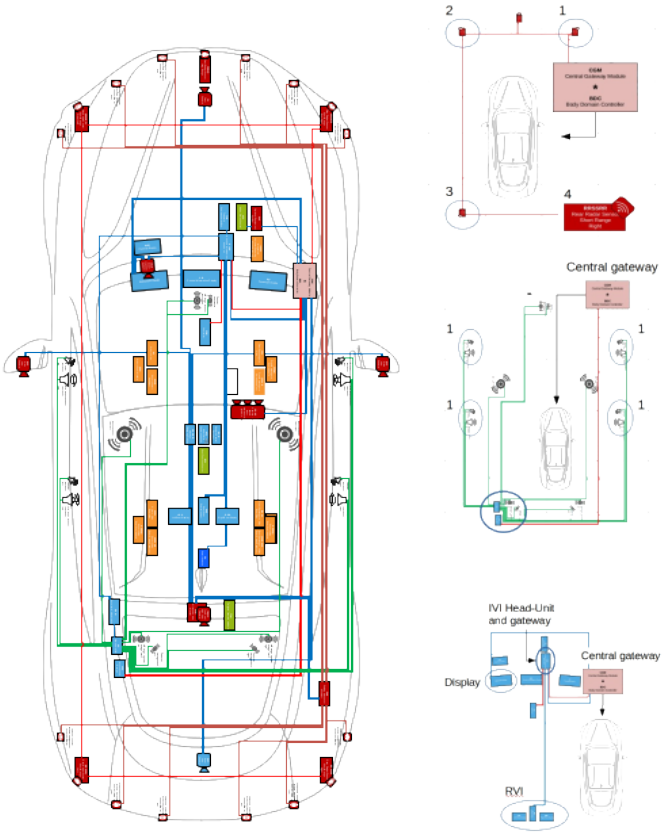


Fig. 2. A set of subsystems: Radar, audio, the IVI against the whole picture

To illustrate, figure 2 contains a bird view example of an automotive system alongside its radar, audio and IVI sub-systems respectively.

Here, the radar sub-system (in red) connects in series 4 radar sensors placed on each corner of the car. While this is a simple solution, it suffers from SPOF (Single point of failure) issues, increased latency for rear radars due to the amount of hops, and an extensive amount of wire dedicated to that feature. The processing is done on the ECU itself. This design is static and distributed.

On the other hand, the audio sub-system (in green) presents a more scattered approach and allocates a physical link for every output device it is attached to. Those are, in turn, using an AVB (Audio Video Bridging) `NOTE: To show that links can be specialized/have a role, not just a wire` link to a dedicated, central gateway that will process the ongoing stream received from a static central gateway (in grey). This is getting closer to a centralized design, yet this iteration can be improved further to address the last issue regarding the duplication of physical resources.

The IVI (In-Vehicle Infotainment) (in blue) exports its display component for others to use directly, and allow the use of its physical links by the connectivity domain located near the RSE (Rear Seat Entertainment).

As seen above, even though designing the sub-systems one by one would yield good results individually, sharing the engaged resources from one another is a non-trivial problem, effectively increasing further the complexity of the entire system. Hence the trend to move toward a more domain or zonal oriented architecture type, and the need for tools to allow the involved parties to work with the entire system as a whole.

4. Tools for a guided design process

With the correct tools, it would be possible to automate, or at the very least greatly enhance the exploring phase of the design. This will, in turn, allow the latter to be compiled on-demand, in contrast of the finely tuned, modular yet set in stone architecture.

Candidates prone to this chain of design will be, for example, the deduction of the topology, topographies and scaling dimensions. For a fully centralized architecture, it would then be able to answer questions regarding the wiring to put in place [17].

It would also be capable of finding if it is more efficient to directly connect high bandwidth equipment (e.g. cameras and radars) directly via Ethernet or flex-ray, or via aggregation nodes. Moreover, if adding more network or routing equipment is done dynamically, the amount of processing done there is also on the line. I am highlighting here the possibility of un-intuitive, hybrid equipment performing tasks out of their original duty so that the overall picture is balanced.

To add to that, while optimizing a current design is possible (and will be the first step to be done), the true potential would be to apply that exploration system wide. One could then handle a specialized architecture composed of several of the mega trends traits: as much as designing an entire system with one paradigm is non-realistic (Classic Autosar and Adaptative both have their own pros and cons [7]), using them based on the client constraints would be best.

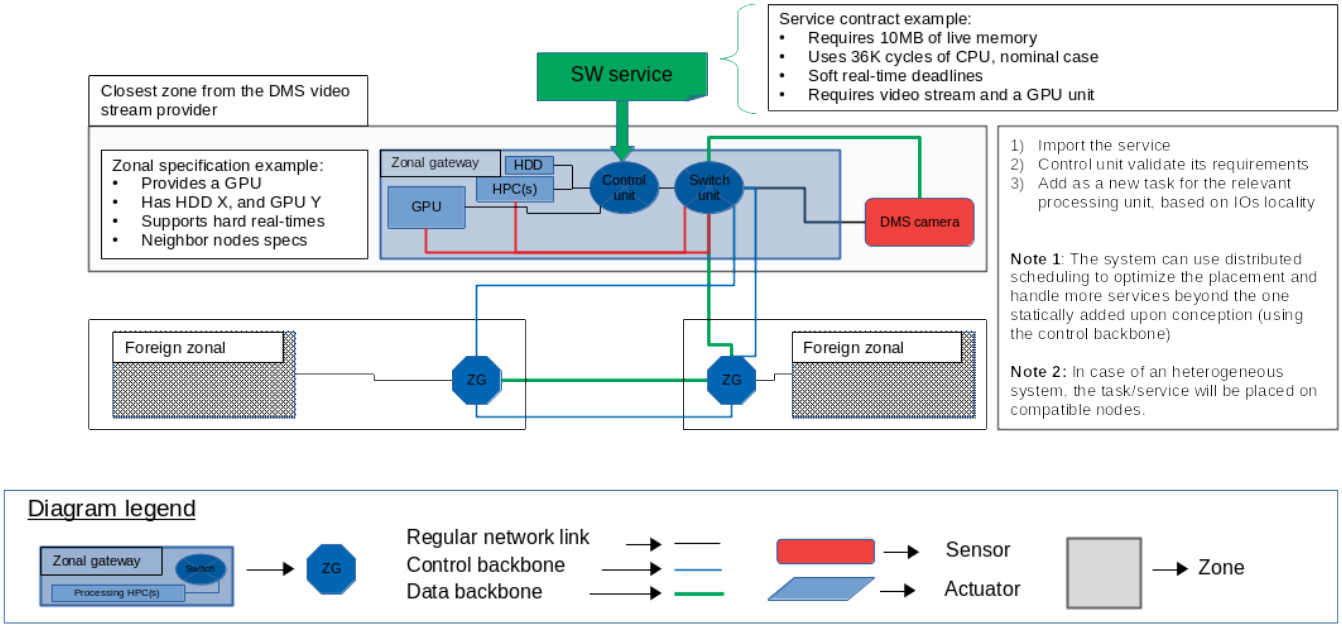


Fig. 3. Dynamic system reconfiguration use-case

Figure 3 illustrates a use case where a given, modular system is challenged to host a new service post production. In that scenario, services provide a set of benchmark and requirements that will get analyzed by a dedicated control unit upon addition. Depending on the current load, state of the system, and freedom to move existing tasks around while meeting real-time deadlines, it will decide if the current system can run the new service.

Although, while this kind of dynamic flexibility is attainable using SoA like oriented systems, over scaling processing resources and maximizing topological options would be required to stay flexible in the long run, and allow good-enough configurations to be found with a fixed HW design.

5. Conclusion and Future Work

The design of automotive E/E architecture is a rich subject with no one-fits-all solution. Due to its high complexity, several designs can be relevant and discriminated based on criteria such as carried-over legacy, expertise, or technological constraints. This is especially true in a time where the final design hosts components manufactures by different providers, which forces compatibility and could lead to license contagion (e.g. having to embed an CAutosar layer to communicate with a legacy component that can't be redeveloped).

This heterogeneous factor makes it harder to knit together all the pieces manually in an optimal way. Also, due to the iterative nature of design space exploration and the volatile tendency of new technologies, late modification on the architecture incurs a high fixup cost. This is something that will likely tend to happen more and more.

We are in a phase where past architecture design choices are showing limitation. Existing and new actors are probing different approaches between 1. maintaining the design at hand, 2. reworking it to some extends as to support new disruptive features such as V2X, or 3. redoing it from scratch and gaining a well needed liberty. Horizon for such changes are showing an approximated release date for 2030+ [4].

To finalize, even though it is highly unlikely that a complete breakthrough capable of working with the whole system and its constraints will happen in that research area [15], significant breaches are in reach in the short term.

Glossary

E/E Architecture Electrical/Electronic architectures
CAN Controller Area Network
DMS Driver monitoring system
DCU Domain control units
SPOF Single point of failure
ADAS Advanced driver-assistance system
V2X Vehicle-to-Everything
UX/UI User Interface
IVI In-vehicle infotainment
RSE Rear Seat Entertainment
ECU Electronic control unit
AVB Audio/Video Bridging
UML Unified Modeling Language
SW/HW Software/Hardware
SoA Software Oriented Architecture
HDD Hard disk drive,
CPU Central processing unit
GPU Graphics processing unit

References

[1] 2022. Bus network. *Wikipedia*. Retrieved November 14, 2022 from https://en.wikipedia.org/w/index.php?title=Bus_network&oldid=1097084205

[2] 2022. Systems architecture. *Wikipedia*. Retrieved November 14, 2022 from https://en.wikipedia.org/w/index.php?title=Systems_architecture&oldid=1121039273

[3] The evolution of E/E architecture and Software platform for R-Car/RH850 | Renesas. Retrieved October 7, 2022 from https://www.renesas.com/us/en/blogs/evolution-ee-architecture-and-software-platform-r-carrh850

[4] Victor Bandur, Gehan Selim, Vera Pantelic, and Mark Lawford. 2021. Making the Case for Centralized Automotive E/E Architectures. *IEEE Transactions on Vehicular Technology* 70, 2 (February 2021), 1230–1245. DOI:https://doi.org/10.1109/TVT.2021.3054934

[5] Tania Banerjee, Yashaswi Karnati, Ke Chen, Yury Lebedev, Aotian Wu, Rahul Sengupta, Anand Rangarajan, and Sanjay Ranka. 2022. A Multi-Sensor Edge-Based System for Intelligent Traffic Intersections.

[6] Alessio Bucaioni, Patrizio Pelliccione, and Saad Mubeen. 2022. Modeling Centralised Automotive E/E Software Architectures. *SSRN Electronic Journal* (2022). DOI:https://doi.org/10.2139/ssrn.4085480

[7] Youssef Elkharaz, Saad Motahhir, and Abdelaziz Elghzizal. 2021. Comparison between autosar platforms with functional safety for automotive software architectures. (2021). DOI:https://doi.org/10.48550/ARXIV.2109.00099

[8] C. Farcas, E. Farcas, I. H. Krueger, and M. Menarini. 2010. Addressing the Integration Challenge for Avionics and Automotive Systems—From Components to Rich Services. *Proceedings of the IEEE* 98, 4 (April 2010), 562–583. DOI:https://doi.org/10.1109/JPROC.2009.2039630

[9] Paula Fraga-Lamas and Tiago M. Fernandez-Carames. 2019. A Review on Blockchain Technologies for an Advanced and Cyber-Resilient Automotive Industry. *IEEE Access* 7, (2019), 17578–17598. DOI:https://doi.org/10.1109/ACCESS.2019.2895302

[10] Jingjing Hao and Guangsheng Han. 2020. On the Modeling of Automotive Security: A Survey of Methods and Perspectives. *Future Internet* 12, 11 (November 2020), 198. DOI:https://doi.org/10.3390/fi12110198

[11] Jun Huang, Mingli Zhao, Yide Zhou, and Cong-Cong Xing. 2019. In-Vehicle Networking: Protocols, Challenges, and Solutions. *IEEE Network* 33, 1 (January 2019), 92–98. DOI:https://doi.org/10.1109/MNET.2018.1700448

[12] Ahmed M. Khan and Timothy D. Blackburn. 2021. AUTILE Framework: An AUTOSAR Driven Agile Development Methodology to Reduce Automotive Software Defects. *IEEE Systems Journal* 15, 3 (September 2021), 3283–3290. DOI:https://doi.org/10.1109/JSYST.2020.2999587

[13] Jochen Klaus-Wagenbrenner. 2019. Zonal EE Architecture: Towards a Fully Automotive Ethernet–Based Vehicle Infrastructure. (2019), 20.

[14] Sidi Lu, Nejib Ammar, Akila Ganlath, Haoxin Wang, and Weisong Shi. 2022. A Comparison of End-to-End Architectures for Connected Vehicles. In *2022 Fifth International Conference on Connected and Autonomous Driving (MetroCAD)*, IEEE, Detroit, MI, USA, 72–80. DOI:https://doi.org/10.1109/MetroCAD56305.2022.00015

[15] Lucas Mauser, Stefan Wagner, and Peter Ziegler. 2022. Methodical Approach for Centralization Evaluation of Modern Automotive E/E Architectures. (2022). DOI:https://doi.org/10.48550/ARXIV.2209.14118

[16] Jonathan Sprinkle, Bernhard Rumpe, Hans Vangheluwe, and Gabor Karsai. 2010. 3 Metamodelling: State of the Art and Research Challenges. In *Model-Based Engineering of Embedded Real-Time Systems*, Holger Giese, Gabor Karsai, Edward Lee, Bernhard Rumpe and Bernhard Schätz (eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 57–76. DOI:https://doi.org/10.1007/978-3-642-16277-0_3

[17] Kanajan Sri and Zeng Haibo. 2006. Exploring Trade-off's Between Centralized versus Decentralized Automotive Architectures Using a Virtual Integration Environment. Retrieved July 19, 2022 from https://past.date-conference.com/proceedings-archive/2006/DATE06/PDFFILES/05E_1.PDF