Enabling Tomorrow's Road Vehicles by Service-Oriented Platform Patterns

Rolf Johansson, Rikard Andersson, and Markus Dernevik

***Abstract*— This paper addresses how a service-oriented pattern can enable to resolve a number of paradoxes that traditionally are seen as hard to solve in the area of E/E design in the automotive domain. The presented pattern is based on a service-oriented paradigm. This is then extended with explicit requirements on design-time analysis and on run-time capabilities required by the services.**

The services are needed to on the one hand being able to be described in a hierarchical way, and on the other hand being able to be implemented in a service-oriented communication paradigm. These two dimensions are essential to understand to both get the capability to resolve the shown paradoxes and to relate possible solutions to existing automotive standards like adaptive AUTOSAR and SOME/IP. A migration strategy is presented based on this analysis.

The concept of vagrant services is introduced and forms an essential part of both solving paradoxes of today and enabling efficient design patterns for tomorrow, including autonomous vehicles and intelligent traffic systems (ITS).

***Index Terms*— Service-oriented E/E architecture, Autonomous vehicles, Cooperative vehicles, Safety, Continuous deployment, Communication patterns, Adaptive AUTOSAR, Vagrant services, Remote sensors.**

I. Introduction

In the area of automotive electrical electronic (E/E) design, there has for a long time been a tradition of statically scheduled communication; especially this is the case for the safety-related part of E/E implemented functionality. The branch is currently working in the direction of a standardization including more of service-oriented patterns, manifested in the case with Adaptive AUTOSAR [1]. However, there are a number of challenges when designing the vehicles of tomorrow, and it is far from evident that Adaptive AUTOSAR alone (or even together with classical AUTOSAR) would automatically solve all wishes of what to achieve. In this paper we use as a starting point a number of the properties that the area of E/E feature development is aiming at, and then we are pointing out some directions how to address these. Today many of these properties are seen as problematic to combine, and these challenges will become much harder to solve when we add what we today expect about the automotive branch of tomorrow. This paper argues that a service-oriented pattern is a general enabler, but in order to get all the advantages, we need to add a number of properties regarding both design-time analysis and run-time capabilities. We also argue that we need to standardise some properties in several communities to enable the full potential to be exploited from service-oriented strategies. Finally, we discuss a possible migration strategy.

A main part of the work presented in this paper has been performed within the scope of the Swedish nationally funded research project called NGEA (Next Generation Electrical Architecture). The project is a cooperation between car manufacturers, engineering companies and research institutions in western Sweden. What we discuss in the paper is based on the business drivers identified in this project: Shared Mobility, Green - Fuel Economy, Safe, Connected services & infrastructures, Time-to-market, Multi brand and segments, Product evolution after original sales, Increase OEM control over OEM concerns and Maintenance.

The paper is organized as follows. In Section II some business driver implications are described, and some paradoxes when combining them are identified. In section III the area of service-oriented E/E patterns is elaborated, and especially the concept of vagrant services is introduced. A general solution to the paradoxes is presented in section IV. Section V presents possible migration strategies. Finally, section VI contains a general conclusion.

II. Paradoxes to Master

When designing the E/E-implemented functionality in road vehicles of today and of tomorrow, there are a number of different properties to fulfil and/or to optimize towards. Unfortunately they are often hard to combine, that is when optimizing for one property this may lead to a decrease with respect to other properties. This problem of contradicting goals will be even more prominent as we will aim for autonomous road vehicles and intelligent traffic systems (ITS), and going in the direction of faster release cycles. Below are listed some high level properties that have been in focus for the analysis in the paper:

- Continuous Deployment of new features in existing running vehicles
- Fast/Cheap to introduce new functionality enabling a

This work has been financed by The Swedish government agency for innovation systems (VINNOVA) in the NGEA project (ref 2015-04881) and in the ESPLANADE project (2016-04268).

Rolf Johansson is currently with Zenuity (e-mail: rolf.johansson@zenuity.com), and he has performed part of this work at his previous affiliation: RISE Safety and Transport. Rikard Andersson is with SMSC Sweden Microchip, and Markus Dernevik is with Volvo Cars.

rich product line with short V&V cycles and continuous integration

- Cooperative vehicles taking part in ITS (intelligent traffic systems)
- Autonomous Vehicles including commercially attractive use cases.
- High Quality/Availability by having all functionality always available at its highest performance
- Highly Complex functionality as the base for most customer value
- Safety. All promised functionality always behaving safely.
- Electrical Vehicle. E/E architecture adapted for fully electrical vehicles (FEV).

The above list of preferred targets constitutes intrinsically many paradoxes. For example, reaching all nominal functions at the highest performance (High Quality/Availability) would contradict Safety. This is because the former asks for a minimum of design margins (derating etc), while the latter requires robust margins in order to be able to argue safety.

If we on the other hand regard the property of Continuous Deployment in relation to Safety, there is another obvious paradox in the area of when to perform assessment. Continuous Deployment implies that the vehicle will during its lifetime be changed a number of times in ways not at all known at the time of the decision to start the series production (SOP). On the other hand in the current understanding of Safety, all safety assessment should be available at SOP.

A third example is the ability to enable fast and cheap introduction of new functionality in relation to Safety. The former is pointing to many versions and variants with a high frequency, while safety would be easier to manage if we have fewer versions and variants introduced more seldom.

A fourth paradox is identified when comparing autonomous vehicles with functionally safe vehicles. The functionality of an autonomous vehicle is implicit in the sense that the vehicle has to solve all the problems that occurs. This implies that the ability of being functionally safe would rely on an implicitly defined functionality. On the other hand, from the functional safety domain, we require that functionality to assess should be explicitly defined. We can depict these four relations to safety as in Fig. 1.

In the following sections we discuss what is needed to resolve these paradoxes. We claim that there is a solution emanating from a service-oriented approach. Before discussing the resolution of the paradoxes, we discuss some general service-oriented patterns.

III. Service-Oriented E/E Patterns

In the automotive domain, there is far from an agreement what to consider as a service when designing E/E implemented functionality. Even though how the concept of service-oriented architectures (SOA) would be established in different industry domains, including automotive, has been elaborated since a while [2]. Below, we discuss what the differences are and how these relate to the tradition of the IT industry. Furthermore, we discuss what the differences are between (SOA) and the traditional paradigms in the automotive industry. Some of these differences are mostly a question of how to model a system on a higher abstraction level, not necessary affecting the low-level implementation, while other differences are inherent in the way different parts of a distributed system interact.

A. *Vagrant Services*

The introduction of service-oriented architecture gives the opportunity to move away from the ECU/function driven focus of today. In an SOA, a service is not necessarily

Fig. 1 Some Paradoxes Related to Safety

constrained to one ECU or one vehicle. This concept we have chosen to name *Vagrant Services*. A vagrant service can move in both time and space, meaning that it can exist in multiple ECUs in a vehicle as well as outside the vehicle e.g. in other vehicles or infrastructure devices of an intelligent traffic system (ITS). It also means that the service can be shut down when not needed to save energy and bandwidth in the system. The vagrant service should be able to be transferred to another ECU for emergency purposes or just due to resource management issues. As noted above the service is not necessarily constrained to one vehicle. This should be seen as widened scope of the V2X (communication between ego vehicle and anything) concept.

Because of the definition of a vagrant service, it suits well to a service discovery pattern. A client searching for a certain service might find zero, one, or several instances of this service. If it finds many instances, these might be spread among both the ECUs of the vehicle itself, and in the ITS environment of other vehicles and of infrastructure.

B. Services and Communication Paradigms

As depicted in Fig. 2 it is possible to view the concept of services in two dimensions. The one dimension in this matrix defines the communication paradigm, and the other dimension the system design hierarchy paradigm. The traditional way in the automotive E/E domain is to decompose a function in logical blocks and statically define data flows between them. This is represented by the lower left corner. By the introduction of SOME/IP services in 2015 on the software implementation level, AUTOSAR has changed communication paradigm of the data flow, but nothing else. To get the full benefit of service orientation, the function should also be decomposed in service responsibilities, i.e. going to the right in this Fig. 2.

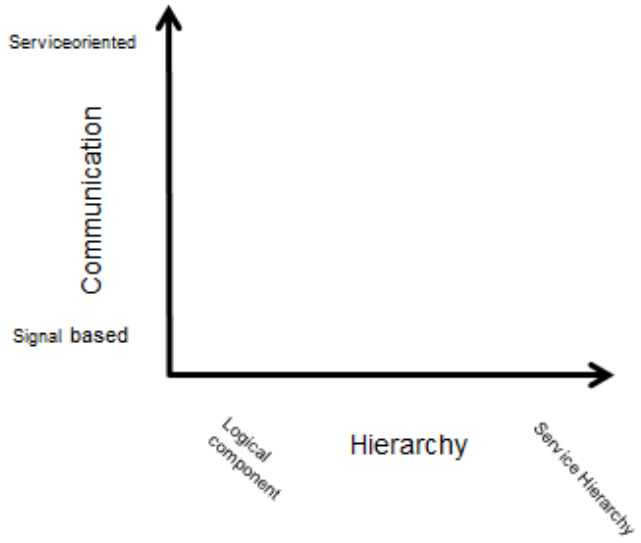

Fig. 2. Services in Two Dimensions

IV. A General Solution

The resolution of the paradoxes is like for any paradox, which means to show that they are not real contradictions. We claim that by using a services-oriented approach where we have some general requirements on the services and on the design methodology, we can enable safety even when the other business drivers can be seen as hard to combine with safety. The general idea is that all the services should be defined as providing both the primary service and selected meta data (quality information) enabling both run-time reasoning and design-time analysis and assessment.

A general enabler to the solving of the paradoxes is to divide the task of arguing for safety between a general design-time argumentation and specific run-time adaptations. This means that the design-time argumentation can be expressed as conditionally valid, given that the run-time choices are within the stated assumptions. We can then decompose the problem by on the one hand assessing that each client asking for a certain service always have a valid alternative for each possible meta-data value, and on the other hand assessing that each service always can produce valid meta data for the primary data.

An example is a service telling the distance to the vehicle in front of the ego vehicle. This service might be asked for in the realisation of several different vehicle features like adaptive cruise control (ACC) or automatic emergency brake (AEB). The primary service is then the one that offers the value of this distance. Meta data can then be everything what is of importance for the possible clients. Some examples are: precision (tolerance margin, ‘how exact’), age (‘how old’), integrity (probability of being correct in stated value, ‘how sure’), etc. For safety, meta data specifically suited for determining ASIL (automotive safety integrity level) attributes for specific failure modes of the service, would be of particular interest.

For the service telling the distance to the vehicle in front, such a set of ASIL specific meta data could be a quadruple of tolerance margins for each ASIL value, for example < (1 m; ASIL A) , (5 m; ASIL B) , (∞; ASIL C) , (∞; ASIL D) >. The infinity as the tolerance margin would denote that there is no tolerance margin that may be guaranteeing such a high level of confidence/integrity. The important thing here is that whatever we identify as important for doing the decoupling between design-time analysis and run-time decisions, are supported by the meta data definition.

In the example paradoxes above we highlight the relations to safety, and these can be resolved by defining meta data either directly having ASIL attributes, or bringing information that can be used to derive ASIL attributes. For other paradoxes, other pieces of information are relevant to enable to perform this separation of concerns.

Below we elaborate this general solution as we look closer on the four chosen paradoxes. After this we look specifically on the implication on the ITS use case and especially on sensing as an ITS service. Finally, in this section we discuss the relation between our general solution and Adaptive AUTOSAR.

A. Continuous Deployment vs. Safety

The paradox coming from optimizing for both continuous deployment (CD) and for safety, lies in the assumption that the former asks for the ability to introduce new functionality in existing vehicles while the latter asks for a definite safety case at the time of starting the series production of a vehicle (SOP). By introducing a service oriented approach, and specifically service discovery we apply the proposed division between design-time analysis and run-time decision based on conditional assessment. By the time of SOP the general safety argumentation tells how each vehicle level feature is enabled by the service structure. Each service is then assessed by on the one hand its capability to provide the primary services as assumed in the design concepts, and on the other hand its ability to generate the agreed meta data. Each higher order service acting as the client asking for these services is assed with respect to its capability to always have a safe strategy for the possible combinations of primary service values and values of corresponding meta data. In the continuous deployment phase, each new deployed vehicle-level feature should before being sent out to the vehicles in the field, show that the new higher order services also have safe strategies for the combination of data and meta data it may receive. All lower order services already deployed in the vehicle are already assessed, and as long as the new high-level features respect the existing service structure, the already existing services do not have to know the new vehicle level features and corresponding new hazard analysis to stay safe. They are deemed safe as long as they are capable to fulfil the design-time given task of providing the primary service together with corresponding meta data.

B. Fast/Cheap vs. Safety

In the continuous deployment paradox, the challenge was to master the binding time for the safety argumentation. In this paradox, the challenge is rather to master the validity for a given safety assessment. The paradox is related to the ability to introduce new functionality in a fast and cheap way, enabling a rich product line with short verification and validation (V&V) cycles and continuous integration. Strictly speaking each variant and each version of a vehicle-level feature is requiring a complete safety argumentation of its own. Generating a very large number of versions and variants, could imply a request for a prohibiting large amount of safety argumentation. The trick to resolve this paradox is to find patterns enabling general safety arguments for much of the product structure, valid at the time for a large number of the versions and variants.

Once again, the pattern of enabling conditional safety argumentation assuming appropriate use of meta data for each higher order service (acting client) using lower order service (acting server). The lower order service does not need to know for what purpose the service is used, and it enables to build a flexible safety argumentation structure, where existing safety argumentation easily can be extended as the number of version and variants grow.

C. High Quality vs. Safety

The third paradox highlighted in this paper is the one telling that we aim for high Quality/Availability by having all functionality always available at its highest performance, and this implies as small design margins as possible. On the other hand, managing safety implies that conservative assumptions are needed in order to guarantee that the safety requirements are always met. Traditionally we use a conservative approach implying that we at design time need to take into consideration all worst cases in order to argue that all safety requirements are always met. By the pattern of services providing their own meta data, we do not need to be that pessimistic in the general safety argumentation. Instead we can say that as long as we know that the meta data has high integrity, we only need to use the worst case when this is indicated by the meta data of each service. In all other cases, we can make trade-off in favour for higher performance, still being safe.

D. Autonomous vs. Safety

The most interesting paradox to resolve from our four examples, might be the one related to autonomous driving. The stated conflict relates to how explicit is the vehicle feature definition, and thus how well-defined it is what failure modes of this that may result in an unsafe behaviour of the vehicle.

On the other hand, what is introduced as part of this complex vehicle-level feature of driving autonomously, is the freedom to perform tactical decisions. Example of tactical decisions are what distance to aim for to the vehicle in front, what speed to aim for, when/if to perform an overtake or a lane change, etc. This kind of decisions fit very well to the pattern that the consumer of a service should be able to adjust its decisions on the combination of the service primary values and the corresponding meta data.

Even if we at a first glance would say that it is impossible to have a general safety argument that is valid for all possible situations that an autonomous vehicle could run into, this can be mastered by saying that the task is rather to adapt the tactical decision so that any situation can be deemed safe. If we for example enter an environment where it is very hard to estimate how much free road we have in front of the ego vehicle, we rather say that the service is providing us with the distance to a specific kind of object ahead together with ASIL-related meta for the tolerance margins. Even if we have very short sight and/or some sensors are not operating under their best conditions, we could still expect a tuple from the service of the guaranteed free distance for each ASIL. Based on this there is always a set of tactical decision putting the vehicle in a situation where this guaranteed distance is deemed safe.

E. Remote Sensors

An important enabler for many foreseen future automotive applications is the concept of remote sensors. This means that one vehicle can use sensors values or calculated understanding from other vehicles or from infrastructure, as if it were coming from a sensor onboard. A challenge is to enable this even if the vehicles or the infrastructure is not part of a cooperating functionality, as would be the case for road trains or for coordinated road crossings. In the case of cooperating functionality, what data to share is part of the definition of how to cooperate and would hence be subject to a design-time agreement when defining what it means to implement the function. The more complicated case from a communication point of view, is when there is no agreement upfront how to use the data. Many advanced driver assisting systems (ADAS) features of today would get higher performance if they could be extended with remote sensor values, and many new ADAS features would be able to introduce then. In the field of autonomous driving, the presence of high-quality remote sensors would in many situations become the difference between high and low driving performance. In any case, it is necessary that there is an agreed ITS protocol to enable the service discovery that such vagrant services as remote sensors would constitute.

Example of what kind of meta data to agree on is in-line with the discussion above, but from a safety argumentation point of view consistent tuples of tolerance margins constituting failure modes and of confidence claims for these would be a minimum. This concept of standardizing meta data for ITS applications was first presented in [3].

F. Relation to Adaptive AUTOSAR

Adaptive AUTOSAR is an on-going initiative in the automotive domain. Different OEMs has different expectations on what Adaptive AUTOSAR shall be the solution for. Some has the view that Adaptive AUTOSAR only shall solve the problems when introducing new SW in a car, i.e. be the solution for problems introduced with Continuous Integration and Deployment. Other OEMs have the view that Adaptive AUTOSAR shall be able to setup or change the Electrical Architecture. The first version of the specification has just gone public [1], and so far there is no explicit support for neither hierarchical services nor for dedicated meta data. On the other hand, there is nothing in the current concept contradicting this. In the following section, we argue for a way to complement Adaptive AUTOSAR in this direction.

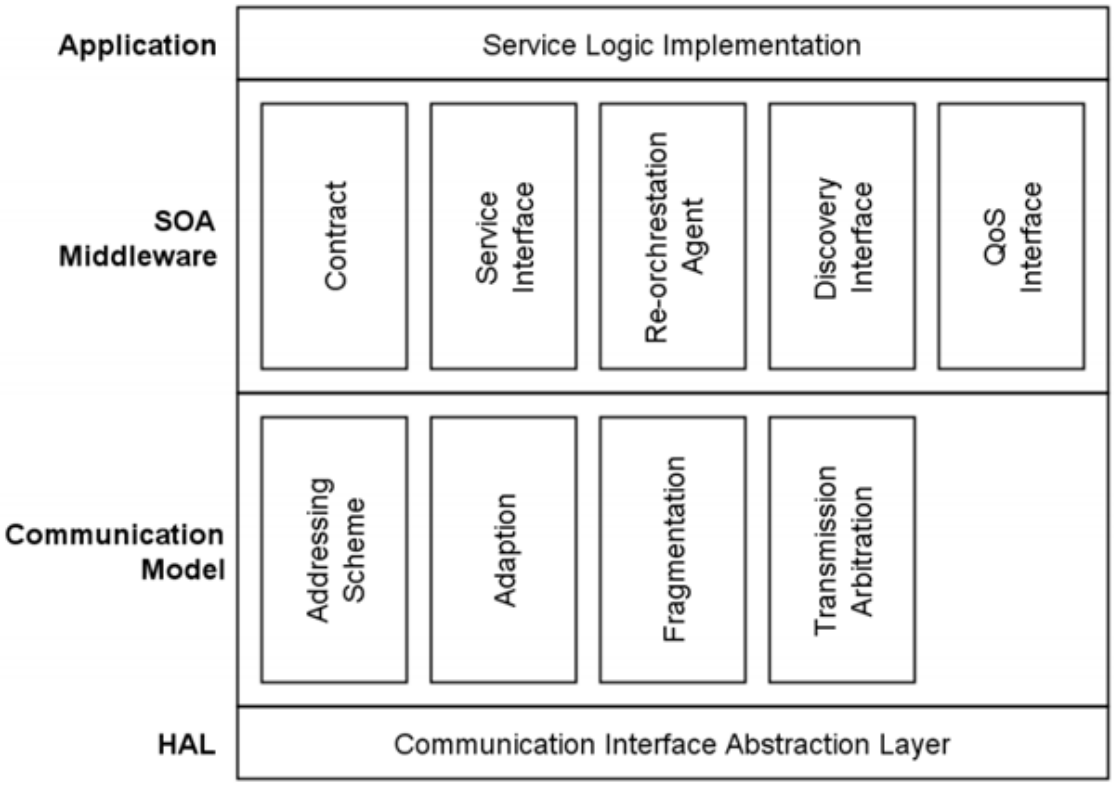

Fig. 3 Overview of SOAcom Architecture

V. Migration Strategies

The general pattern of innovation in the automotive domain is to combine a large portion of reuse with a minor part of new solutions. The question is hence if it makes sense to introduce a service-oriented pattern according to what is discussed in this paper, only to a part of an entire E/E implementation. For example, the ongoing introduction of adaptive AUTOSAR is only considered to be introduced in some ECUs of a car, while many (of the safety-critical) ECUs still are assumed to be classical AUTOSAR. In the following, we elaborate what a mixed strategy could look like, and how it would be possible to get out most of the advantages from a service-oriented architecture, even if some ECUs still are connected in a static signal-based way.

A. Incorporation of Existing Signal-Based Concepts in a Service -Oriented Environment

A requirement for service-oriented communication is the possibility to use dynamic addressing. Networks that support IP-based communication fits to this, such as Ethernet and MOST. More traditional automotive networks, such as CAN and LIN, are created for static addressing and therefore needs to be adapted to support service orientation. One way to do this is to create a middleware to build a bridge between the high-level communication in the service oriented communication layer and the low-level automotive network communication layer. This has been examined by A. Ballesteros, M. Wagner and D. Zöbel in the paper SOAcom: Designing Service communication in adaptive automotive networks [4]. See further Fig. 3.

Another way would be to make the service layer of the electrical system IP only, and reduce CAN and FlexRay nodes to peripheral devices.

In a service-oriented network, the offers are broadcasted on the network and ECUs needing to use services request the specific services. This means a possibility of having a lot of setup communication in some states, such as network startup. Depending on functional requirement, this must be taken in consideration when designing the system. In the signal-based network, the ECU know when to send and what to send and can startup messaging directly at network startup, see Fig.4.

B. Introduction of SOME/IP

The introduction of SOME/IP may lead to the introduction of communication services in the main design toolchain. Services have previously been used in a proprietary ethernet protocol, but having them introduced in the design tools, changes the design pattern for both system architects and subsystem design teams.

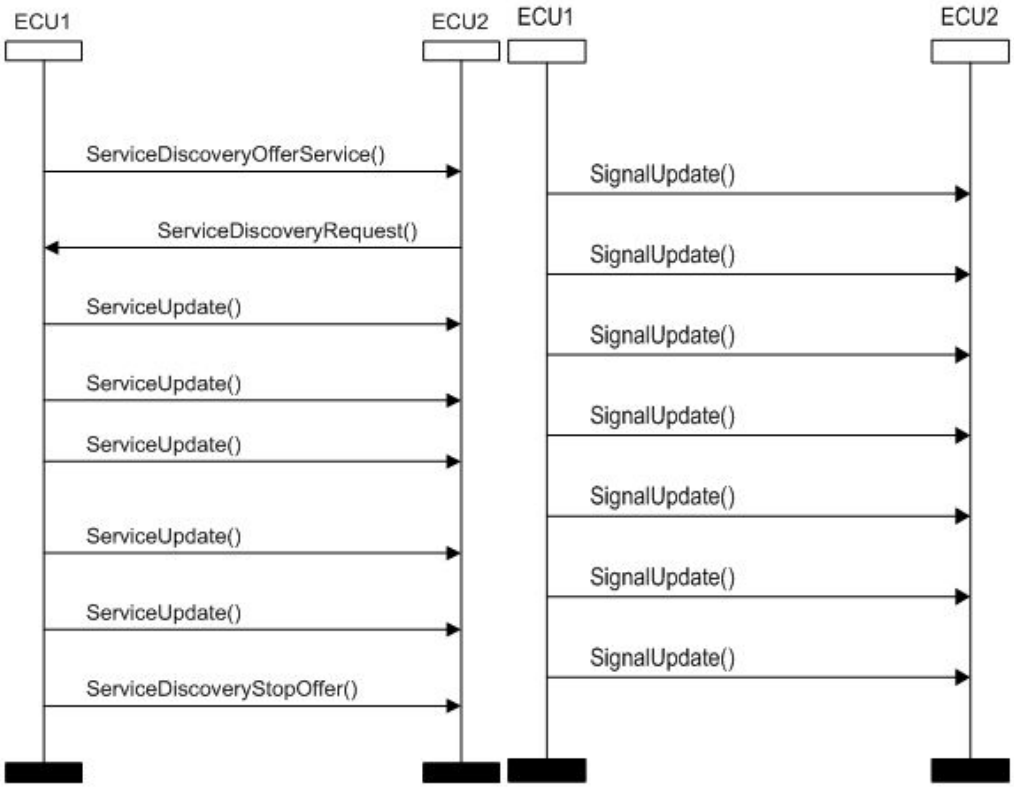

Fig. 4 Difference Between Service-Oriented and Signal-Based Startup Sequence

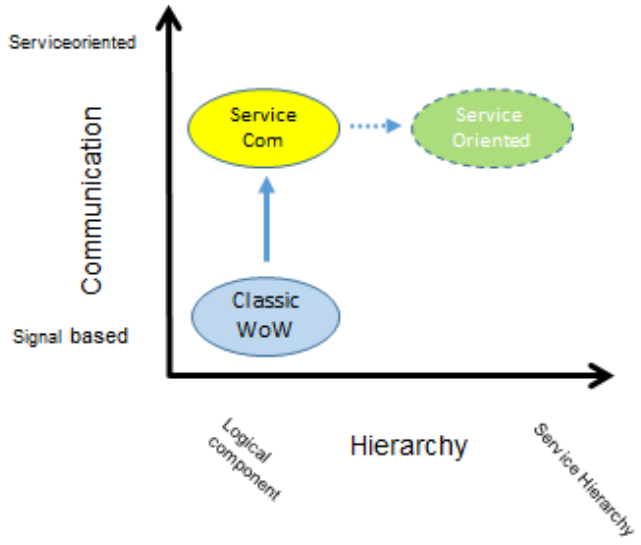

Fig. 5 Migration in the Design Matrix

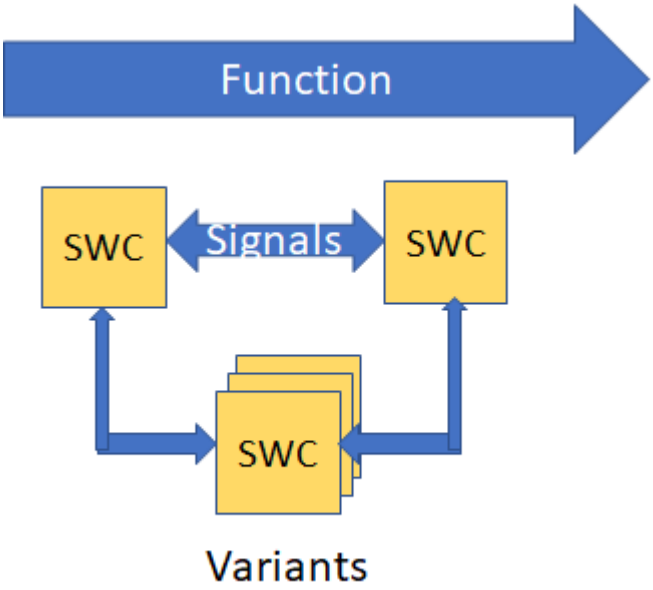

Fig. 6 Function Decomposition

Previously the services were broad functional categories partitioning the communication on Ethernet. With the introduction of SOME/IP, the service architecture and deployment of service instances becomes more important. Initially, this may introduce a lot of discussions about responsibilities between teams, and on methodology. This is a complex task that may need tweaking

Design teams request new services from the architecture team, which determines what services are to be implemented, where service instances shall be deployed, the network endpoints carrying IP and VLAN information, and application endpoints carrying protocol and port number information. More importantly, a service architecture is needed. To solely use an incremental approach where communication needs on ethernet drives a new service leads to a suboptimal design.

Using the matrix defined above, we may move from the lower left quadrant, to the higher left. To really get the full benefit of service orientation, a move to upper right quadrant is necessary, see Fig. 5.

Such a move requires breaking the rules of classic automotive design thinking, where end-user functions are decomposed into logical blocks with formation flows between them on the design level, and onto software components with signals between them on the software implementation/AUTOSAR level. This pattern together with a static signal configuration at design time has issues with Variability, Flexibility and Extendability. See Fig. 6

Instead a service architecture is needed. Functions can similarly be broken down into service responsibilities provided by contracts defined by the service interfaces. Just by having a service mindset the design becomes more modular with clearer interfaces. See Fig. 7.

Variability is improved by allowing variant realizations of the constant service interface, and by allowing different deployments of service instances. Deployment of service instances, version of service interfaces, and what subset of service operations is currently available, can be resolved at run time. This in essence is a decoupling in the design of the electrical system. The independence of the exact version or the deployment of the service instance at design time, together with a dynamic signal configuration provided by service discovery, spare the electrical system from a combinatorial explosion of the number of top-level system variants. This design paradigm would belong to the upper right quadrant in the design matrix as depicted in Fig. 5.

As can be seen in Fig. 7, variants of the service realization sharing a constant interface is tolerated. The actual service can itself be seen as realized by logical blocks or SWCs, or by an hierarchical structure of services, or a by a combination thereof.

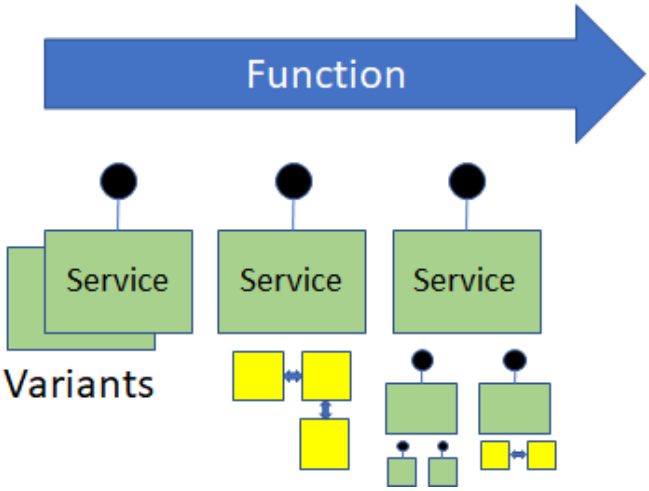

Figure 7 Architecture Pattern with Service Interfaces

VI. Conclusion

In this paper we have argued for how a service-oriented architecture will resolve a number of paradoxes that are likely to identify when aiming for the business drivers many OEMs are facing., We propose a general solution how a service-oriented architecture should be defined. The proposal goes beyond what is today the subject for standardization in the domain, including things like hierarchical notion of services, meta data needed for run-time decisions, and capabilities of design-time analyses. As elaborated in the paper, the proposal has the potential to extend the current initiative of adaptive AUTOSAR. We argue that it would be possible to combine a service-discovery pattern with high demands on functional safety, which is contradicting the established understanding that we need a solution like traditional AUTOSAR for all safety-critical applications. Furthermore, the proposed solution fits into service discovery of vagrant services, not only inside a certain vehicle but in a complex ITS environment.

References

[1] Fürst, S, and Bechter, M, "AUTOSAR for Connected and Autonomous Vehicles: The AUTOSAR Adaptive Platform", In 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop, 2016.

[2] Bohn, H, Bobek, A, and Golatowski, F, "SIRENA - Service Infrastructure for Real-time Embedded Networked Devices: A service oriented framework for different domains", In Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies, 2006.

[3] Eriksson, H. and Söderberg, A. "Remote Sensing and Functional Safety in ITS" In Proceedings of 12th ITS European Congress, 2017.

[4] Ballesteros, A, Wagner, M, and Zöbelz, D, "SOAcom: Designing Service communication in adaptive automotive networks", In 8th IEEE International Symposium on Industrial Embedded Systems (SIES), 2013.