INVITED: Dynamic Platforms for Uncertainty Management in Future Automotive E/E Architectures

Philipp Mundhenk
Audi Electronics Venture GmbH
Gaimersheim, Germany
philipp.mundhenk@audi.de

Ghizlane Tibba
Robert Bosch GmbH
Stuttgart, Germany
ghizlane.tibba@de.bosch.com

Licong Zhang
Technische Universität München
München, Germany
licong.zhang@tum.de

Felix Reimann
Audi Electronics Venture GmbH
Gaimersheim, Germany
felix.reimann@audi.de

Debayan Roy
Technische Universität München
München, Germany
debayan.roy@tum.de

Samarjit Chakraborty
Technische Universität München
München, Germany
samarjit@tum.de

ABSTRACT

Current automotive E/E architectures are comprised of hardware and software and are mostly designed in a monolithic approach, static over the lifetime of the vehicle. Design, implementation and updates are mostly performed on a per-component-basis, exchanging complete Electronic Control Units (ECUs) or their software image as a whole. With an increasing amount of functionality being realized in software, the benefits of software can be used increasingly. This includes modularization of components, which forms the basis for updates and addition of functions. Additionally, this modularization allows the consolidation of ECUs and supports a higher level of integration. Such modularization and dynamic behavior over the lifetime of a vehicle fleet, as well as a single vehicle does, however, hold a lot of challenges for safety-critical systems. Safety-critical systems, such as cars, require their behavior to be deterministic. The design of such modular systems needs to consider and cope with uncertainties in modular architectures. This paper highlights some of the dimensions of uncertainty, which will exist in future E/E architectures and presents initial approaches on how to manage these.

ACM Reference format:
Philipp Mundhenk, Ghizlane Tibba, Licong Zhang, Felix Reimann, Debayan Roy, and Samarjit Chakraborty. 2017. INVITED: Dynamic Platforms for Uncertainty Management in Future Automotive E/E Architectures. In *Proceedings of DAC '17, Austin, TX, USA, June 18-22. 2017,* 6 pages.
DOI: http://dx.doi.org/10.1145/3061639.3072950

1 INTRODUCTION

Over the last decades, innovation in the automotive domain has shifted considerably to the area of electronics and software. The software-based functionalities are mapped on Electronic Control Units (ECUs), where each ECU fulfills a defined number of specific functions. These ECUs are connected by bus systems like Controller Area Network (CAN), FlexRay and, quite recently, also higher bandwidth networks such as Ethernet. The rapid growth of software functionalities has led to the significant increase in both the size and complexity of the automotive Electric/Electronic (E/E) systems. As a result, the number of ECUs has increased significantly. However, this increase would not be sustainable due to complexity, cost and space reasons.

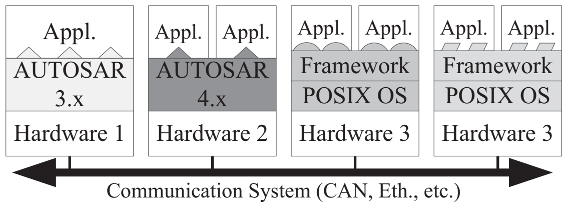

Figure 1: Current automotive architectures are highly diverse and exhibit a high degree of vertical integration. Here, we identify challenges and approaches for dynamic platforms, which are required for vertical decoupling and horizontal integration of applications.

Furthermore, ECUs are in many cases the smallest unit of electronics and software in the vehicle (see Figure 1). If a functionality is to be exchanged or updated, this happens on level of ECUs. This could either be a change of the ECU hardware or the firmware image. For most of the ECUs, there is no smaller unit than the complete firmware image and concepts like software containers do not exist in current automotive software architectures. This paradigm is mainly rooted in the safety requirements of the automotive domain. Since many of the functionalities in a vehicle are safety-critical, they need to have predictable behaviours which are rigorously tested and certified. Therefore, the dynamics of such systems are reduced as much as possible. Consequently, the E/E architecture and the software functions remain largely static during the life cycle of a car and important innovations in terms of architecture, software and functionality can only be introduced in future series.

However, the innovations in electronics and software in the automotive domain is accelerating. Especially in the area of Advanced Driver Assistance Systems (ADASs) and autonomous driving, new software functions are developed rapidly in recent years. Compared to the life cycle of a car, the design and development cycle of software functionalities is much shorter. Therefore, the ability of the E/E architecture of a car to enable software updates and accommodate newly developed software would be a promising and possibly essential feature in the future. In the consumer electronics domain, the users can customize their devices by updating Apps and downloading newly developed ones without changing the underlying hardware of the device. In future, this feature will be increasingly expected from cars. This requires that the underlying E/E architecture possesses a certain level of flexibility and dynamics. The introduction of dynamic behaviour into a diverse environment

DAC '17, Austin, TX, USA
© 2017 ACM. 978-1-4503-4927-7/17/06…$15.00
DOI: http://dx.doi.org/10.1145/3061639.3072950

such as current automotive architectures can be a tedious and difficult task. However, ECU consolidation as one important trend in the automotive E/E architectures would potentially be an enabler for dynamic platforms. The ECU consolidation has been discussed in [12], where, in future, an ECU will increasingly be treated as a computing platform and multiple functions can be integrated on one ECU. This trend is expected to continue and gain momentum, as it is currently one of the most promising ways to curb the complexity problem in automotive architectures. However, for the introduction of dynamic platforms in the E/E architecture, some additional challenges need to be addressed. The main challenges can be categorized as the following.

Faster time-to-market. Considering the ever increasing speed of innovation in the consumer electronics domain, the pressure on the automotive industry is increasing to deliver new functionality faster to the customer. Currently, new features are typically introduced with new or updates of current vehicle models. In future, new functionality might also be delivered between model years via software packages. Time-to-market however, cannot be decreased inifinitely for all functions. Functions with safety requirements need to go through rigorous testing to ensure the safety of vehicle, driver and passengers. Following existing norms such as ISO-26262 [8] requires the adherence to strict testing procedures, especially for functions with higher ASIL levels (e.g., ASIL-D).

Increasing computation requirements. With the introduction of autonomous driving and Artificial Intelligence (AI) in the vehicle, the required computation capabilities increase significantly. Image processing, object and pattern recognition, as well as neural networks and machine learning algorithms require large amounts of computation power. With the exception of infotainment, current ECUs typically contain Central Processing Units (CPUs) with 200Mhz or less. This is not sufficient for more advanced applications. Neural networks can additionally be run more efficiently with the support of Graphics Processing Units (GPUs), which typical ECUs do not currently contain.

Increasing bandwidth requirements. The increasing usage of displays, cameras, radar and Light Detection and Ranging (LIDAR) requires communication systems with significantly more bandwidth than the systems currently in use (e.g., CAN). This development is driven through the development of new Human Machine Interface (HMI) technologies, as well as ADASs.

Increasing safety requirements. With the advent of semi-autonomous (piloted) and autonomous driving, the driving tasks are increasingly shifted from the driver to the vehicle software and hardware systems. This also shifts the responsibilities. Thus, the E/E architecture of the vehicle needs to be certified as safe and support the highest level of safety requirements for all related functions. This holds especially when ECU consolidation is applied and mixed-criticality functions are executed on a single hardware system. Additionally, redundancy in software and hardware might need to be available for autonomous driving. Here, the safe state might not necessarily be the shutdown of the vehicle. Thus, continued operation, also in case of an error, needs to be ensured. Furthermore, new certification processes are required when updating single functions dynamically, versus well-understood complete firmware images today.

Increasing security requirements. Existing work has highlighted the importance of security for E/E architectures. When introducing dynamic loading and updating of software, these security requirements will increase significantly. It needs to be ensured that software updates can only be delivered by authenticated authorities.

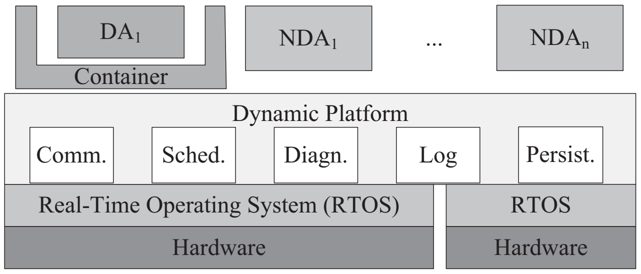

Figure 2: Dynamic platform for mixed-criticality applications. Deterministic applications (DA) and non-deterministic applications (NDA) can be run side-by-side on the same hardware and operating system. Freedom of interference is ensured by the dynamic platform layer.

Need for architecture modeling, simulation & testing. An additional challenge towards new dynamic platforms is the modeling and testing of functions. This challenge is secondary in the way that it already presents solutions to the above challenges. Nevertheless, it opens up new challenges on its own. As architectures are becoming larger, more dynamic and more complex, one key factor to success is the control of complexity in such dynamic architectures. Considering the dependencies and influences between applications in terms of execution and messaging latencies, response times, etc., is non-trivial. An integrated modeling, simulation and testing approach can help to curb this complexity.

1.1 System Model

When defining future dynamic platforms, the terms that are known from existing E/E architectures change slightly. The key elements of the new architecture are outlined in Figure 2. While in current E/E architectures, functions are typically tightly coupled into the system, in dynamic platforms, functions are integrated in **applications**. In analogy to the consumer electronics world, an application (app) is the smallest unit of addition and update. The different types of applications are defined in Section 3.1. These applications are hosted on the dynamic platform, which forms the core of the new E/E architecture. This **dynamic platform** can logically be located across multiple hardware elements and operating systems. This allows to also dynamically exchange the hardware. The dynamic platform integrates functionality common to multiple applications. Such core functionality may include communication services and scheduling of deterministic and non-deterministic tasks. Additional functions can be logging, persistence services (e.g., for configurations), and diagnosis, which is especially important to the automotive industry. To achieve its tasks, the dynamic platform typically requires an **Operating System (OS)**. The OS takes care of the connections to hardware, via drivers, the scheduling of processes and threads, etc. A general abstraction of the hardware and operating system can be provided through Portable Operating System Interface (POSIX). Typically, when running mixed-criticality applications, a Real-Time Operating System (RTOS) is required. However, if only non-deterministic applications are running on a certain hardware, it might be sufficient to run a non-real-time OS.

1.2 Overview

In this paper, we will outline the requirements for modeling and simulation in Section 2. Then, we will address the influences of dynamic platforms on the safety of the overall vehicle in Section 3. Furthermore, the security challenges are evaluated in Section 4. Dynamic architectures, as well as the topics of modeling, safety and security have received coverage in literature, evaluating these from different angles. This related work is presented in Section 5, before concluding in Section 6.

2 MODELING, SIMULATION & TESTING

In today's automotive architectures, functions typically are communicating via signals. These signals are either passed via a communication system, such as CAN or the AUTOSAR Runtime Environment (RTE). Processes have been established to coordinate these signals between different ECUs. There is, however, no unambiguous definition of signals between applications on one ECU. Different ECUs describe signals in different fashions. Some signals are not documented at all. Thus, finding emitting, consuming and controlling entities to a signal can be a tedious task.

2.1 Communication Paradigms

In current architectures, the term signal typically refers to time-continuous data. This originates from the understanding of functions as control applications, which is still a valid approach for many functions in vehicles today. However, new architectures might use different communication paradigms, depending on the needs of the application to be installed. To achieve a more flexible communication, service-oriented or data-centric communication might be used. Potential candidates for this are Scalable service-oriented Middleware over IP (SOME/IP) [18] and Data Distribution Service (DDS) [13], among many others. Via these communication mechanisms, different communication paradigms might be implemented. These are listed in the following:

Event. This communication paradigm describes a "notification". A consumer can subscribe to a certain data (or "topic"). Whenever a producer application publishes new data to this topic, this data is transfered to the consumer. While this requires an initial subscription message, it is typically refered to as one-way communication. In our modeling approach, we assume an owner for every interface, who controls interface description, version, etc. In this paradigm, the owner of the interface is the producer. Futhermore, the currently existing signals can be mapped to this communication paradigm.

Message. In this communication paradigm, a two-way communication is established. Here, a message is sent from a producer to a consumer, which in turn responds to this message with an answer. This paradigm allows the implementation of Remote Procedure Call (RPC), which is essential to command and control functions in vehicles. In our modeling, the owner of the interface is the message consumer, offering the service.

Stream. This communication paradigm describes a one-way communication with continuous data. Typically, it is used for Audio/Video applications, where the single packages (or "frames"), have dependencies on the previous frame(s).

2.2 Modeling approach

Due to the complexity of new architectures, an integrated modeling approach is required. Such an approach ensures that no information is lost when designing and building the system. Yet, the chosen approach needs to offer sufficient amount of flexibility, such that applications can be designed as independently from each other as possible. A set of Domain-Specific Languages (DSLs) can be a good approach to describe the system in a formal way, which can be checked for correctness. Such a set of DSLs requires separate approaches to describe the hardware architecture, the interfaces between applications and a deployment to different hardware architectures and communication technologies. The hardware architecture needs to define all required ECUs, including all attributes to be checked (e.g., computational and storage resources, hardware support for encryption, etc.) and the communication network interconnecting them. The interface definition needs to ensure interdependencies and requirements of applications are clearly defined. These requirements might consists of multiple attributes, such as latency and jitter for real-time applications or bandwidth for streaming applications. The communication is no longer based on signals defined by bit offsets, but on complex objects, defined by complex data types.

An attached verification engine should ensure that the interconnections and deployment mappings fulfill the defined requirements.

Integration is key for a modeling approach. It can, e.g., be used to generate code stubs, configurations for communication stacks and a middleware on devices, or input for simulation environments.

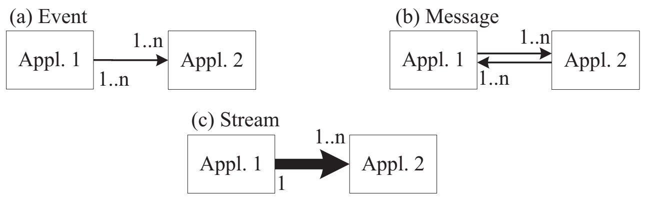

Figure 3: Communication Paradigms in future automotive architectures. Management connections such as subscribe messages have been omitted.

2.3 Design Space Exploration & Simulation

As a dynamic platform is being modeled, the variability needs to be encounted for in the model. Thus, it can be necessary to include variances in the model and not define every mapping and interconnection uniquely. The final mapping might only be applied in the vehicle on the road. E.g., the deployment of a function to a hardware can depend on the installed applications and current load of every hardware component in the vehicle. However, it needs to be ensured that every possible mapping is functional, safe, and secure. Design space exploration and simulation can be used to achieve this assurance. The design space exploration can operate on the output of the model and use simulation or verification approaches to guarantee parameters in all possible combinations, as well as define the optimal approach for every combination of functions, parameters and hardware.

2.4 Testing approach

The complexity of ECUs software leads to particularly long development and testing cycles. The long time span between the availability of the first software components and the first system prototype highlights the need and the importance of testing earlier on in the development process. Several test levels can be leveraged to shift a big amount of testing activities to an earlier stage enabling incremental testing and intermediate validation, verification and correction. we refer to these levels as XiL, with X representing any control model (M), software (S), or hardware (H) under test. For the development of ECUs software, the hardware independence provided via especially MiL and SiL environments offers a major advantage. Long before target hardware or prototypes are available, functions, individual software components, as even the complete software can be tested and validated when integrated on a virtual control unit. Using the full potential of computing power of a PC, debugging and error reproduction in MiL and SiL can be performed much faster than on ECUs. Time consuming procedures such as flash programming can be reduced [17].

3 SAFETY

Future dynamic automotive architectures are often compared to architectures in the consumer electronics domain, however, the safety requirements of automotive architectures are significantly different. Currently, safety is ensured through tight integration and control of the developed ECUs. This tight integration, however, goes contrary to the new dynamics as outlined in Section 1. The tight integration of all components, in conjunction with extensive tests, ensure the predictability of the system. However, extensive tests are contrary to the requirement of shorter time-to-market. Thus, new approaches to ensuring safety in dynamic platforms are required.

Additionally, when modularizing software, it is important to ensure the correct safety ratings for all dependencies of a software module. Only with correct safe dependencies can a software module be considered safe.

3.1 Application Model

Some applications might have more stringent requirements on safety, interference and execution time than others. We thus divide applications in two categories: **Deterministic applications** have strict schedule requirements, with fixed execution times and jitters. These can be typical control loops or more complex ADAS functions. Deterministic applications are well-defined in the model and do not significantly change their behavior at runtime. A typical contributor are the motor and suspension domains. **Non-deterministic applications** are typically more complex, potentially requiring threading or performing longer term, potentially asynchronous communications. On the other hand, non-deterministic applications have relaxed scheduling requirements. Typically, infotainment contributes to this category.

The influences of software modules on every other software module and thus the complete system needs to be quantified. Specifically when pursuing ECU consolidation and thus creating a mixed-cirticiality platform, freedom of interference needs to be ensured. Multiple aspects of freedom of interference are to be considered. These aspects and our approach to address them are listed in the following.

CPU. When deterministic and non-deterministic applications are running on the same system side-by-side, a scheduling scheme is required that ensures all deterministic applications are given their required scheduling time. Specifically, deterministic applications have fixed activation intervals and computation deadlines. Thus, they need to be activated at specific points in time. With the scheduling approaches (time- or priority-based) existent in RTOSs, this can be achieved. On general purpose operating systems, this might not be achievable. There, only non-deterministic applications can be run. When the set of applications is changed at runtime, the schedule needs to be adjusted accordingly encompassing the changed requirements of all applications. Generating a new schedule at runtime is potentially computationally expensive. We propose to generate a schedule from the model and test this schedule in simulations in the backend, also against the current configuration of the installing vehicle. This way, the computation power of the backend can be used and the suitability of the schedule can be ensured.

Memory. Freedom of interference between applications also requires to fully separate their memory. In most OSs, such memory protection mechanisms are already implemented and used to separate processes. Thus, separate applications need to be executed in separate processes. However, OSs with support for memory separation often require a Memory Management Unit (MMU), this is an additional requirement for the hardware underlying the new dynamic platform. Additionally, a large amount of processes might slow dow+n a system. Thus, it is important to define which applications need to run in separate processes and which can be combined in a single process. We propose to define this process separation based on the model defined in the previous section.

Hardware Access & Communication. The third large category to consider regarding freedom of interference is the access to hardware and specifically communications, as the most often required hardware resource. When a deterministic application needs to transmit data, these transmissions typically have an accompanying urgency. If at the same time a non-deterministic application is transmitting a large amount of data (e.g., as a stream, see Section 2.1), it needs to be ensured that this does not delay the urgent transmission of the deterministic application. These conditions and order of priorities holds for all hardware access (e.g., crypto module, persistent memory, etc.) Again, the behavior of applications might change, as the installed applications on the dynamic platform change. Adjusting this behavior at runtime can be challenging, similar to the CPU scheduling above. An initial approach to solve this dilemma is presented in [21].

3.2 Update Safety

Especially critical are updates at runtime. Non-deterministic applications usually do not require precautions when updating, as their impact might be limited to user experience. These applications can be (1) stopped, (2) updated, (3) restarted. However, updating deterministic applications at runtime can have high impact on the safe functionality of the vehicle. A staged way for updates needs to be defined. Support for this is required in the dynamic platform. An updated binary of an application can be (1) started in parallel with the old version. Afterwards, all internal states need to be (2) synchronized with the existing application version. Then, all traffic can be (3) redirected to the new application. After all redirection is completed the older version of the application is (4) stopped.

When updating distributed functions with distributed dependencies, this concept can be used to update step-by-step via defined update paths. By verifying the safety of every intermediate update step, the safety of the complete update can be ensured. The disadvantage of such an update is of course the additional amount of resources required in the update process, as every application to be updated needs to be instantiated twice. On the positive side, it is to mention that such an orchestrated approach is significantly more robust than a simple centrally organized switch from old to new version. In the latter case, high accuracy clock synchronization is required and a single point of failure is created.

3.3 Redundancy

Other than in current vehicles, the fail-safe state of an autonomous vehicle is not necessarily a safe shutdown. When driving on the highway, e.g., and experiencing and ECU failure, a larger amount of coordination is necessary to return the vehicle to a safe state. For such cases, an autonomous vehicle needs to provide fail-operational mechanisms, to ensure that the vehicle can continue to operate safely. To achieve these, the dynamic platform needs to support instantiating applications multiple times. It might be necessary to install multiple ECUs running the dynamic platform and synchronized applications across these ECUs.

3.4 Runtime Monitoring

While the complete system should be modeled, simulated and thus known at design time, the rising complexity and number of variants might require runtime monitoring. Such monitoring capabilities need to especially target the key parameters of deterministic applications, such as period, deadline, jitter, memory usage, etc. With such monitoring capabilities, faults can easily be detected, the conditions leading to such faults recorded and, if an internet connection is available, be transferred to the manufacturer for further examinations. In turn, an update can be created and rolled out to remedy the detected error. Additionally, runtime monitoring can generate data sets, efficiently supporting the safety certification processes.

4 SECURITY

The security of E/E architectures has come under increasing scrutiny. With the introduction of dynamic and modular software platforms, this will become an increasingly important issue. If software can be added and updated at runtime and over-the-air, it needs to be ensured that the installed updates are legitimate. Furthermore, when creating dynamic bindings of services at runtime, it needs to be ensured that the binding partners are authenticated and that communication is authorized. These aspects are discussed in the following.

4.1 Package Security

Securing packages and updates to be installed is not a new topic. Approaches are known in the consumer electronics domain. Attacks on these approaches, however, show how fragile such system can be [2]. Used in a safety-relevant context, such as a vehicle, this can have devastating consequences. Thus, the implementation needs to be put under increased scrutiny to ensure its correctness. An additional constraint in the automotive domain is that not all ECUs might have sufficient power to perform cryptographic operations at runtime. For such ECUs we propose to use an update master to which a trust relationship can be established. This update master can in turn ensure the security of and administer the update. To avoid a single point of failure, the update master would need to be instantiated in a redundant fashion.

4.2 Authentication and Authorization

Authentication and authorization have been addressed in many contexts. Authentication can be integrated with the communication system and performed in an efficient manner [10]. In the context of service-oriented architectures, it is necessary to find a distributed access control method, which is updatable as well. Such an access control method needs to define which client is allowed to access which service. These definitions should be automatically extracted from the modeling approach described in Section 2. This way, the security model can be checked already at integration time. The implementation of such an access mechanism is highly dependent on the chosen communication mechanism. In the case of DDS, e.g., such mechanisms are integrated in the DDS security specification.

Additional questions remain regarding clients which potentially have access to every service, such as a data logger. Such permissions might need to be loaded and adjusted at runtime. This is synchronous to the discussions of the permission model in the Android operating system, where many apps request all available access rights as default [5]. Ensuring correct access rights at all times without bypasses is crucial to the security of the service-oriented communication.

Chances. While dynamic architectures open up a lot of security issues, they also have the chance to enable a higher level of security throughout the vehicle. When utilizing modeling approaches and consistent design throughout the vehicle, it is possible to better surveil all applications and their behavior than currently possible.

5 RELATED WORK

5.1 Modeling, Simulation & Testing

Approaches to use design space exploration in automotive architectures have been presented in [9] and [14]. There, sets of tasks and messages are mapped to a defined hardware architecture. Variants are accounted for and can be solved through attached simulation and optimization frameworks, delivering optimal deployments for the given set of software modules. An approach for testing XiL system testing has been proposed in [17].

5.2 Modular architectures

Modular architecture and interface standardization is one important trend in modern automotive software architectures to address issues like complexity, scalability and maintainability. Towards this, AUTOSAR (AUTomotive Open System ARchitecture) [4] is developed and adopted by the automotive industry. AUTOSAR standardizes the interface between different software components and proposes a reference architecture, where the software is divided into the basic software layer, the Runtime Environment (RTE) and application software layer. Thus the application software components are abstracted from the basic software layer. However, the original AUTOSAR standard does not provide support for a dynamic platform. To address this problem, the AUTOSAR development partnership is currently developing the AUTOSAR Adaptive Platform [4] that would support dynamic platforms, where the RTE can link services and clients dynamically during runtime.

Another related approach in terms of modular architectures is proposed and developed in the RACE project [15], where a centralized architecture is developed for future electric vehicles. Aligned with the ECU consolidation trend, the hardware architecture consists of a central computing platform of processing units connected to sensors and actuators through Ethernet. The application software is abstracted from the hardware by a RACE RTE based on the Chromosome middleware [3]. The RACE RTE [1] provides scheduling and data transfer services to the applications based on a publish-subscribe mechanism and also integrates transparent safety features. The architecture also supports the Plug-and-Play of functionalities.

5.3 Safety

Towards guaranteeing the safety in terms of mixed-criticality scenarios, the common way is to partition and isolate resources for applications of different criticalities. [16] considers both spatial and temporal partitions between applications of different Safety-Integration Levels (SIL). In the case of communication, FlexRay offers a combination of time-triggered deterministic communication and priority-based communication, which can be used to partition and isolate deterministic and non-deterministic applications. In addition, in the upcoming Time-Sensitive Networking (TSN) standards [7] for Ethernet, expected to meet requirements in the automotive domain, a mixed-criticality scheme is employed. Highly critical applications requiring deterministic communication can use time-triggered scheme, where non-deterministic applications will use priority based communication and the transmission selection on switches will prevent its interference on deterministic communication.

Another important issue for the safety of the dynamic platform is the online resource management, where computation and communication resources need to be allocated and re-allocated. To ensure the safety of the applications, adequate platform resources are necessary to guarantee the non-functional properties like timing. Towards this, [6] addresses the problem by employing the admission control, where the a compositional analysis approach is used to check whether there is enough resources to satisfy the timing requirements and compute the corresponding configurations. [19] also considers an admission test by online schedulability analysis. In the context of time-triggered scheduling, [21] has proposed a schedule management framework to deal with the problem of online schedule synthesis for time-triggered architectures, where a mixed local and cloud-based framework is used and further incremental design techniques are employed to reduce the disturbance to existing applications.

In addition, redundancy is sometimes necessary to ensure safety and provide fail-operational behaviours of the system, especially for highly critical applications. Redundancy in hardware, software and communication can be employed against fault and failures for each. The RACE architecture [15] implements redundant sensors and actuators for critical applications and uses ring topology in the Ethernet network to provide redundant communication channels. The central computing platform consists of multiple processing units running redundant software for the same functionalities in a master-slave fashion [1].

Towards the update safety issue, [20] has proposed a runtime control to allow runtime adaptation, activating or deactivating specific software components. A synchronization component is implemented to coordinate between different ECUs.

5.4 Security

Securing automotive architectures is not trivial, as the available resources on ECUs are typically relatively low. In [10], an approach to a lighweight authentication framework has been presented. This approach has been specifically tuned to the requirements of the automotive industry and can also be used in a dynamic platform, as it follows the authentication principles on the highly dynamic Internet.

Additionally, in [11], an approach to evaluate automotive architectures for their security has been presented. The presented approach is based on probabilistic model checking and allows to judge the security of the architecture or single components, based on the security evaluations of single components. Such an approach could be extended to also encompass software functions.

6 CONCLUSION

To foster innovations, automotive E/E architectures need to evolve. We outlined the challenges that E/E architectures are facing today. Some of these challenges have been addressed in literature either for other domains or, in some cases, even for the automotive domain. However, a large number of challenges remain to be solved to enable dynamics in automotive E/E architectures. Among them are the aspects introduced in this paper, namely modeling and simulation aspects, as well as safety and security. Where modeling and simulation is required to curb the complexity of new architectures. Additionally, ensuring safety and security in such modular, dynamic mixed-criticality systems is key to their customer acceptance.

REFERENCES

[1] K. Becker, J. Frtunikj, M. Felser, L. Fiege, C. Buckl, S. Rothbauer, L. Zhang, and C. Klein. 2015. RACE RTE: A Runtime Environment for Robust Fault-Tolerant Vehicle Functions. In *CARS 2015 - Critical Automotive applications: Robustness & Safety*. Paris, France.

[2] A. Bellissimo, J. Burgess, and K. Fu. 2006. Secure Software Updates: Disappointments and New Challenges.. In *HotSec*.

[3] C. Buckl, M. Geisinger, D. Gulati, F. J. Ruiz-Bertol, and A. Knoll. 2014. CHROMOSOME: A Run-time Environment for Plug & Play-capable Embedded Real-time Systems. *SIGBED Rev.* 11, 3 (Nov. 2014), 36–39. https://doi.org/10.1145/2692385.2692391

[4] AUTOSAR Consortium. 2017. AUTOSAR. https://www.autosar.org/. (2017). Accessed: 2017-03-16.

[5] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner. 2011. Android Permissions Demystified. In *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS '11)*. ACM, New York, NY, USA, 627–638. https://doi.org/10.1145/2046707.2046779

[6] D. Gangadharan, J. H. Kim, O. Sokolsky, B.G. Kim, C.-W. Lin, S. Shiraishi, and I. Lee. 2016. Platform-Based Plug and Play of Automotive Safety Features: Challenges and Directions. In *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2016 IEEE 22nd International Conference on*. IEEE, 76–84.

[7] IEEE 802.1 Working Group. 2017. Time-Sensitive Networking Task Group. http://www.ieee802.org/1/pages/tsn.html/. (2017). Accessed: 2017-03-16.

[8] International Standardization Organization (ISO). 2011. Road vehicles – Functional safety. (2011).

[9] M. Lukasiewycz, F. Sagstetter, and S. Steinhorst. 2015. Efficient design space exploration of embedded platforms. In *Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, June 7-11, 2015*. 127:1–127:6. https://doi.org/10.1145/2744769.2744829

[10] P. Mundhenk, A. Paverd, A. Mrowca, S. Steinhorst, M. Lukasiewycz, S. A. Fahmy, and S. Chakraborty. 2017. Security in Automotive Networks: Lightweight Authentication and Authorization. *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 22, 2 (2017), to appear.

[11] P. Mundhenk, S. Steinhorst, M. Lukasiewycz, S. A. Fahmy, and S. Chakraborty. 2015. Security Analysis of Automotive Architectures using Probabilistic Model Checking. In *Proceedings of the 52nd Design Automation Conference (DAC 2015)*. https://doi.org/10.1145/2744769.2744906

[12] M. Di Natale and A. L. Sangiovanni-Vincentelli. 2010. Moving From Federated to Integrated Architectures in Automotive: The Role of Standards, Methods and Tools. *Proc. IEEE* 98, 4 (April 2010), 603–620. https://doi.org/10.1109/JPROC.2009.2039550

[13] G. Pardo-Castellote. 2003. OMG Data-Distribution Service: Architectural overview. In *Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on*. IEEE, 200–206.

[14] F. Reimann. 2015. *Design Space Exploration for Automotive E/E Architectures*. Ph.D. Dissertation. University of Erlangen-Nuremberg.

[15] S. Sommer, A. Camek, K. Becker, C. Buckl, A. Zirkler, L. Fiege, M. Armbruster, G. Spiegelberg, and A. Knoll. 2013. RACE: A Centralized Platform Computer Based Architecture for Automotive Applications. In *2013 IEEE International Electric Vehicle Conference (IEVC)*. 1–6. https://doi.org/10.1109/IEVC.2013.6681152

[16] D. Tamas-Selicean and P. Pop. 2011. Design optimization of mixed-criticality real-time applications on cost-constrained partitioned architectures. In *Real-Time Systems Symposium (RTSS), 2011 IEEE 32nd*. IEEE, 24–33.

[17] G. Tibba, C. Malz, C. Stoermer, N. Nagarajan, L. Zhang, and S. Chakraborty. 2016. Testing automotive embedded systems under X-in-the-loop setups. In *Proceedings of the 35th International Conference on Computer-Aided Design, ICCAD 2016, Austin, TX, USA, November 7-10, 2016*. 35. https://doi.org/10.1145/2966986.2980076

[18] L. Völker. 2013. SOME/IP-Die Middleware für Ethernetbasierte Kommunikation. *Hanser automotive networks* (2013).

[19] J. W. Yoo, Y. Lee, D. Kim, and K. Park. 2012. An Android-based automotive middleware architecture for plug-and-play of applications. In *2012 IEEE Conference on Open Systems*. 1–6. https://doi.org/10.1109/ICOS.2012.6417614

[20] Marc Zeller, Christian Prehofer, Daniel Krefft, and Gereon Weiss. 2013. Towards Runtime Adaptation in AUTOSAR. *SIGBED Rev.* 10, 4 (Dec. 2013), 17–20. https://doi.org/10.1145/2583687.2583691

[21] L. Zhang, D. Roy, P. Mundhenk, and S. Chakraborty. 2016. Schedule Management Framework for Cloud-Based Future Automotive Software Systems. In *22nd IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, RTCSA 2016, Daegu, South Korea, August 17-19, 2016*. 12–21. https://doi.org/10.1109/RTCSA.2016.11