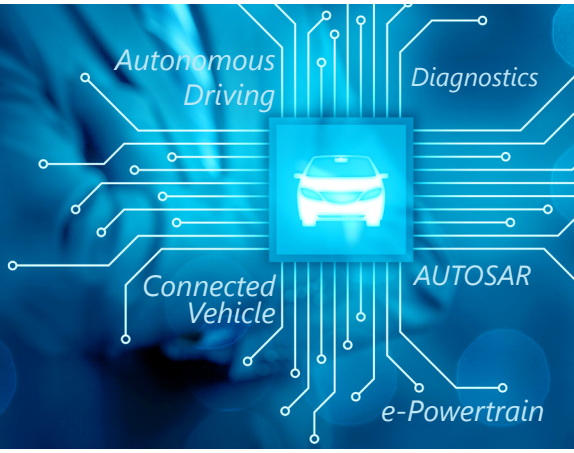
# Adaptive AUTOSAR - A new way of working





## Abstract

The software platforms we are dominantly using currently in the Automotive Industry are firmware based. The firmware is a binary running build for a bare-metal on a given microcontroller. A set of requirements are created and realized as a firmware implementation or configuration. It is downloaded at production, a periodic service in a workshop or in worst-case in a callback because it contains faults. But with the original aim to download it and hopefully, the producer will never need to update the software anymore in the life of the car.

As you know this does not fit to software very well. Not that there are bad software engineers in the Automotive industry. But the features and the tasks that the ECUs must fulfil today are such complex that continuous feature updates and software faults are unavoidable. The more complex a project is the more errors per line of code increases [Code2], i.e. the number of errors is not just a linear function of the lines of code. We can, of course, invest more in verification and validation, but often feature growth is prioritized over the module and integration testing. In addition, typical verification needs same expert level as the implementers. The system test is expensive and hard to coordinate. In addition, requirements are not in all cases clear from the beginning, and this will make validation of the system hard.

The new generation owners of cars are expecting up-to-date and dynamic features in the car. The majority of these features are connected to the software. Smart-phones today are much more efficient in adapting to the user needs. Being all-the-time connected to the Internet, solving many tasks that are considered normal in today's way of living.

The Infotainment and Telematics projects have been early adopters of introducing complex feature implementation and agile work process. They are typically updatable systems with a complex Operating System supporting file-systems, such as Linux. Unfortunately, these projects are in many cases an integration of lots of modules creating a software platform integration and leads little or no reuse. Each generation of a project is basically a start from scratch. In addition, Linux based platforms are not well-fulfilling safety and hard real-time requirements.

With this background, Adaptive AUTOSAR is a new standard that is solving above issues and creates a partly Service Oriented Architecture (SOA).

Classic AUTOSAR is a self-contained firmware platform, whilst Adaptive AUTOSAR is not trying to be complete in the sense to create a full SOA. An SOA is a software platform where Services is central. A service is provided by a Service Provider and consumed by a Service Consumer. The connection between a Service Provider and a Service Consumer is set up by a Service Broker dynamically in run-time. The Service Consumer does not necessarily know where the Service Provider is located. And hence, the software platform is dynamic and can be partially updated without affecting the system.

Adaptive AUTOSAR, as mentioned above, does not claim to create a full SOA. The Operating System is only interfaced and not specified [AUTOSA OS]. Here an Adaptive AUTOSAR application is interfacing an Operating System that provides with the PSE51 API subset of the POSIX standard [POSIX]. This does not mean that the OS can only be PSE51, it means that the Adaptive AUTOSAR application can only use this API. The OS can still be a full PSE54 OS (e.g. Linux which is PSE54-ish). The benefit of this approach is that all POSIX compliant applications and libraries can be modelled. And hence, reused and deployed in any Adaptive AUTOSAR software platform. In other words, AUTOSAR is not trying to redefine POSIX. Just for clarification, POSIX is a standard interface for a programming interface. It is neither an implementation or operating system.


As published in HANSERAutomoitve 7-8 / 2018

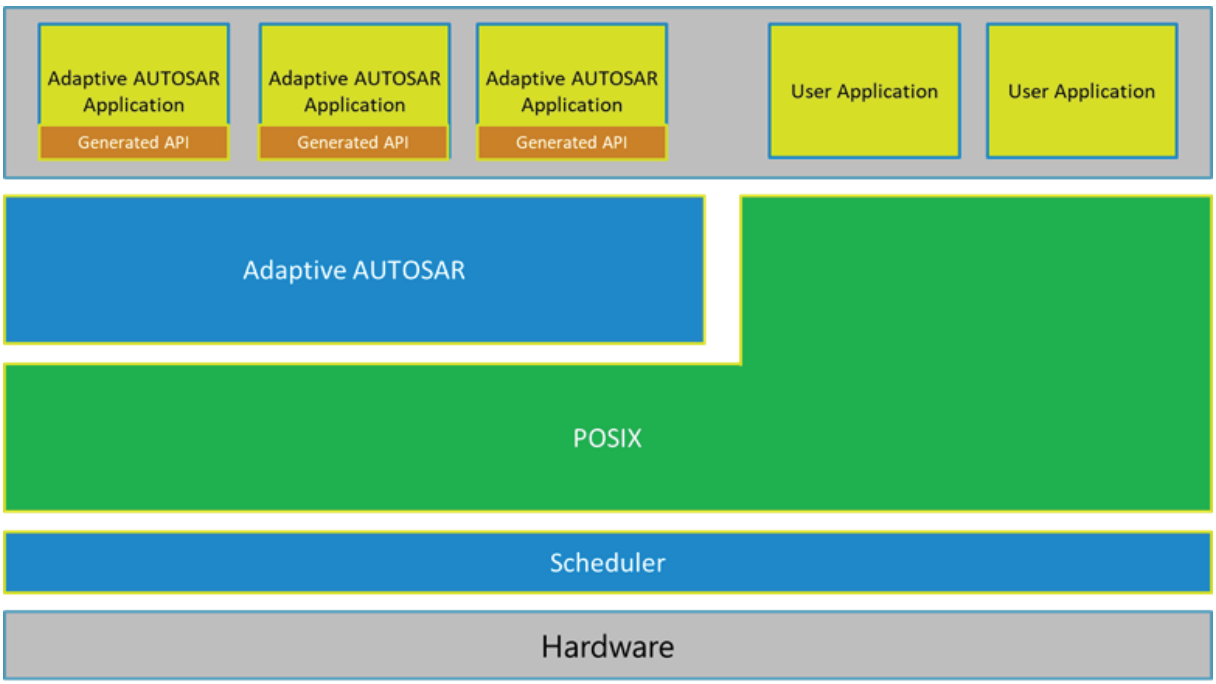



*Figure 1 - SOA using Adaptive AUTOSAR*

I often get the question to use older reliable software that has been in production for a long time. And in the same time arguments that a new software always contains defects, why not simply use the old code and make some small extensions? The evolution of features required in software today is huge. It is very hard to create a good architecture that enables adding all kind of future features not known from the beginning. Refactoring of the software is needed from time to time to be able to maintain and verify the software efficiently. In many cases, the argumentation on classic AUTOSAR is like this. The benefit of Adaptive AUTOSAR is that POSIX compliant software can be reused, with the philosophy "write once, adopt everywhere". We are not forced to rewrite, we can benefit from existing open software or a software company may reuse applications for different customers.

Going for a Service Oriented Architecture (using Adaptive AUTOSAR) approach will not clear all obstacles in the way. Going from a firmware based (i.e. Classic AUTOSAR) approach to an SOA means that we have new challenges that we previously did not have, or mainly previously solved by hand. Some of them are taken up here.

Having an SOA that is updatable implies that projects work-process need to change. The classical waterfall approach where a requirement specification is released to a supplier, and they will implement and test it will not really work anymore. Instead, an agile way of working where Continuous Integration (CI) approach is the foundation, is required. A CI that continuously verifies and releases new software. In addition, the CI will be interfaced by the OEM and suppliers, and the suppliers will have own CI for their applications or software platforms. In order to fulfil such CI eco-system the OEM and suppliers CI:s must interact automatically. CI:s automatically creating releases for other companies (OEM:s) CI will not be an easy nut to crack for IT and legal departments. We need to understand the ownership and how such interactions can be done in a safe and secure way.

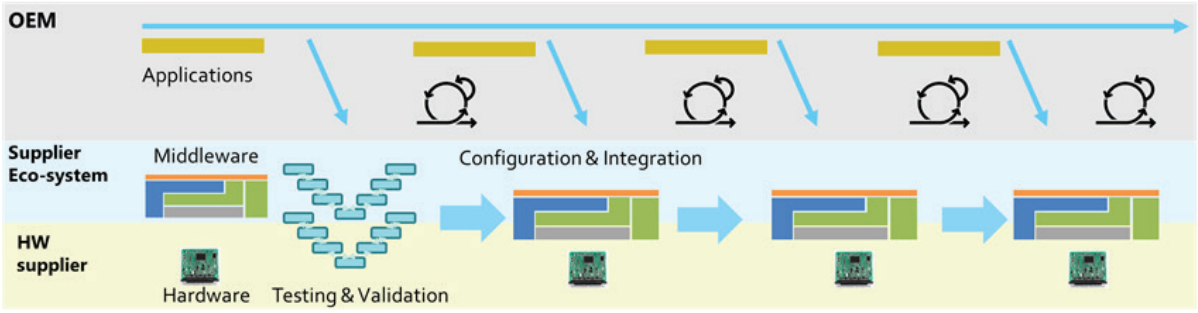



*Figure 2 - Continous Integration*

Fig. 1- Ecosystem Visualization

Another important topic is to make a software platform reusable among different projects, without starting each time from scratch. Here Adaptive AUTOSAR standard is very helpful. On one side, there are well specified AUTOSAR basic services, such as Diagnostic Manager, that handles specific functionality required in the car, and on the other, a clear methodology and exchange format that can be used in the work-process of OEMs and suppliers. Having such standard will make it possible to develop and reuse applications between different projects and customers.

The benefit with a firmware approach is that updates are one-time activities. With an updatable system comes the fact that the system will be partly updated spatial and temporal. The initial image will be deployed, and different organizations (and companies) will make updates to the system at different points in times. This puts strong requirements on the offline tooling. The update must be checked for compatibility before deployed. The update must be verified such that it will not break anything in the vehicle. And also, updates must be orchestrated, e.g. if two ECUs needs to be updated and one fails, the software in both ECUs must be rolled back to their previous version.

The POSIX standard defines several features; the feature level is controlled by the PSE51-54 profiles. In each profile, most of the features are optional [POSIX]. In classic AUTOSAR the feature content is well specified and mostly controlled by the scalability classes. Two different POSIX Operating Systems can differ very much in feature content. It is important to specify the needed features of a POSIX Operating System early in the project, in addition, make sure that the selected POSIX Operating System can be extendable with further features when the project grows in required features.

SOA is often affiliated with dynamic behaviour. But to be able to control and manage the next generation software in vehicles the work process must be stringent, and automated. In addition, the software in the vehicle must be controlled so that verification and safety can be achieved. Suppliers and car OEMs will work closer together using automated continuous integration systems. Static verification must be extended with dynamic verification is the vehicle (e.g. by adding metadata). And different suppliers in the eco-system of the software platform must work together in an efficient way. A contract based approach in development is needed.

These are some topics that are needed to handle in the new SOA paradigm. An SOA based on Adaptive AUTOSAR is not replacing classical AUTOSAR platform 1:1. The aim of this text was to show that we are facing new challenges both in technical and work-process areas. But the advantage of the approach Adaptive AUTOSAR is taking is to align to an existing standard such as POSIX. And therefore, it is possible to reuse software and experiences from various projects (also non-Automotive).

[Code2] Code Complete, 2nd Edition, Steve McConnell

[AUTOSA OS] Specification of Operating System Interface

[POSIX] http://pubs.opengroup.org and IEEE 1003

**Author**
**Anders Kallerdahl,**
*Subject Matter Expert Automotive Software,*
*KPIT Technologies*

KPIT

## About KPIT

KPIT (BSE:532400; NSE: KPIT) is a global technology company focused on providing technology solutions and expertise to the Automotive & Transportation Industry. KPIT is at the forefront of automotive engineering globally with solutions in the areas of Autonomous Driving, Connected Mobility, Vehicle Electrification, AUTOSAR & In-Vehicle Networks and Vehicle Diagnostics. Together with its customers and partners, KPIT creates and delivers technologies to enable creating a cleaner, greener and more intelligent world that is sustainable and efficient.

## Marketing Contact:

*Stefanie Köhler, Head of Marketing Germany, Phone: +49 89 322 99 66 140, stefanie.koehler@kpit.com*