AUTOSAR Adaptive can be introduced successfully if the hardware and software that is based on AUTOSAR Adaptive works well with the existing AUTOSAR Classic system components. Full implementation throughout the system – including when implementation takes place on different platforms – will be one of the next major challenges in E/E development. Service-oriented architectures can bridge the gap between two worlds.

Automotive trends such as automated driving or connectivity with a backend system need a lot of computing power and flexible communication concepts. In the same way as in the world of smartphones and consumer electronics, users should be able to install, update and extend applications in their vehicles. The AUTOSAR Adaptive platform responds to this challenge: It provides an environment for executing complex applications and exchanging information with external participants. Automotive Ethernet is used as the central communication technology. Its large bandwidth and IP-based protocols make it easy to integrate external systems and the user’s terminal devices – even while the overall system itself is running. By contrast, the AUTOSAR Classic platform couples hardware and software components to one another in a static way. AUTOSAR Classic ECUs use cost-optimized micro-controllers that directly access sensors and actuators and have to fulfill concrete real-time requirements. ECUs based on the AUTOSAR Classic platform can also communicate via Ethernet and are a reliable choice when developing safety-related applications. The aim of AUTOSAR Adaptive is not to replace the Classic platform but to extend it by adding new, far-reaching capabilities. That is why modern vehicle architectures combine the two platforms and make use of the benefits that each of them offers. The chain of activities that take place between a sensor that controls an actuator via closed-loop control is tightly- and statically-coupled and continues to be a task for AUTOSAR Classic. Loosely- or dynamically-coupled applications, such as communication between the vehicle and its environment or a backend, are performed using the Adaptive platform (Figure 1).

In combined architectures, AUTOSAR Adaptive and AUTOSAR Classic ECUs are usually connected to one another via Ethernet. In many cases, information is exchanged via the service-oriented SOME/IP protocol (Scalable Service-Oriented Middleware over IP). The introduction of Ethernet and SOME/IP into the Classic platform has opened the door to a new design paradigm – the service-oriented architecture: In AUTOSAR Classic, the so-called service instance only appears as a detail in the SOME/IP configuration. In AUTOSAR Adaptive, the service-oriented approach runs from the design stage right through to the provision of the application on the machine, and the service instances are considered as a concrete step in the design process.
The Service-Oriented Architecture

What exactly is a service-oriented architecture? It is a template software architecture in which participants provide or consume functions (services) using a defined protocol. The aim is to distribute the logic over a number of atomic services. They interact with one another to provide more complex functions: Small units with clearly defined functionalities can be combined easily and be re-used for different purposes. Communication between the services takes place over a familiar interface that is defined at an abstract level. The abstract definition focuses on the functionality; the technology that will later be used to implement the communication plays no role. This is a central feature of service-oriented architectures: During the draft design stage, the required technologies, necessary resources and concrete implementation are initially irrelevant.

The most important components of a service-oriented architecture are the services: self-contained functionalities that can be accessed by other services and can be updated and modified separately. Services capture, produce or process data. When acting as a provider, a service makes its capabilities available to consumers. As a consumer, a service uses the data or capabilities of other services in order to provide its own functionalities. A high-value service consists of multiple simple or atomic services. Services have another important characteristic: They treat every request from a consumer as a separate, independent transaction. A service that does not contain or store any state information is referred to as a stateless service. Such services are highly available and easily scalable. And stateless services also have another advantage: Their function is easy to understand based on the interface description; no knowledge of their internal behavior is necessary for this.

A Conceptual Example

The following diagram shows the service architecture for a simple weather service (Figure 2). It was modeled using concepts from the Service Oriented Architecture Modeling Language (SoaML) with Vector’s PREEvision development environment. The main service LocalWeather delivers consolidated information about the weather for a certain city. The service has a provider and a consumer role. The two roles represent service instances in a future implementation. The provider and consumer sides of this service communicate via the interface LocalWeatherInterface. The relationship between the provider and consumer is represented in terms of a usage relation (uses) and is referred to as the service choreography. The consumer requests the weather for given geographical coordinates which are determined by a GPS receiver. The main service bundles the information and makes it available to consumers.
To do this, it needs data from other services: The map service Map supplies a city code on the basis of the geographical coordinates. This makes it possible to query the temperature and humidity in the relevant city from the weather service Weather. The interaction between the services is depicted in the form of a dependency relation. The interaction between services in a service architecture that is designed to implement more complex services is known as service orchestration. In the concrete implementation (Figure 3), the participants are already connected and have a contract with one another. The participants are abstract elements which, in the Automotive context, could be interpreted as ECUs or computers. In any case, all communication is via the service interface.

**The Service Interface**

Services communicate via the so-called service interface, which is defined by AUTOSAR on the basis of methods, events and fields (also referred to as attributes or properties). Methods are operations that can be called by other services – usually with parameters. The consumer expects to receive a result which it can then further process in its own context. Alternatively, consumers can also call fire-and-forget methods: In this case, an operation is started at the provider but no result is returned. Events are triggered in the provider’s context. Consumers register for events and are notified when the event has taken place. A provider variable is stored in a field and can be read or written by consumers. In addition, a field triggers an event if the value of the variable has changed. In this way, the provider informs registered consumers of the change.

**Implementation in AUTOSAR Systems**

In the AUTOSAR context, a defined service interface can be provided or consumed in a Classic or an Adaptive ECU. SOME/IP in combination with Service Discovery is currently the most common transport protocol for service-oriented communication in both AUTOSAR Adaptive and AUTOSAR Classic. However, there is a great difference in the way service interfaces are implemented in the two platforms! For example, software components for Adaptive are programmed on an object-oriented basis in C++. By contrast, Classic components are programmed in C. The runtime environments therefore also differ accordingly and this has to be taken into account when creating and configuring the code. The overall vehicle architecture usually determines whether a software component is implemented as a Classic or as an Adaptive application. The individual functions are distributed across different ECUs and computers in the vehicle network. A service interface can be directly assigned to the port of an Adaptive software component (Figure 4).
Figure 3: Service orchestration of the weather service with connected service participants.

Figure 4: How service interfaces are implemented in AUTOSAR Classic and Adaptive (technology mapping).
Conclusion

Service-oriented architectures describe the dependencies and interactions between services. The service interface contains an exact definition of how these services communicate with one another and the data they exchange – in abstract form. This makes it possible to understand the overall system at an early stage, without having to go deeply into the technical details. The abstract design makes it possible to validate the system in accordance with specific rules and is a good basis for the derivation of a detailed AUTOSAR configuration. This approach can be used for both implementations using both the AUTOSAR Classic and Adaptive platforms. In this way, service-oriented architectures are able to bridge the gap between the two worlds and open the way for vehicle architectures that benefit from the advantages of both platforms.

Author

Marcelino Varas
Vector Informatik GmbH
Product Management Engineer
PREEvision AUTOSAR Design

Originally published in "Elektronik automotive" magazine
Issue 11/2019 – November 2019