Today’s E/E architectures differ from previous architectures in many ways. Inside the vehicle, available tools – including high-performance computers with very significant computing power. At the same time, the E/E architecture extends beyond vehicle limits all the way to the IT backend and to mobile end devices. Along with the practice-proven AUTOSAR Classic software platform, AUTOSAR Adaptive and other software platforms will also be used in the future. Unlike in the past, system and functional designers have access to a design space that has been expanded to an extreme level. The challenge is to exploit new degrees of freedom to develop vehicle functions for more customer value. At the same time, growing complexity must also be overcome.

On the one hand, long-existing constraints are being eliminated – such as limited computing capacities in the vehicle. On the other hand, AUTOSAR Adaptive and Automotive-Over-The-Air connectivity are enabling entirely new functions and services such as software upgrades, software updates and remote diagnostic functions. Therefore, this article focuses primarily on roles in system and functional development and on the development process.

What roles will be involved in the future? Which model-based methods and notations are best suited for specifying the systems and functions? How can a development process for system development, functional development and software development be created that is viable for the future? What will change and what will stay the same? And what are the implications for the requirements of the necessary development tools?

Modern E/E Architectures
The typical layout of a current E/E architecture distinguishes between four layers (Figure 1):

> The lowermost layer contains the sensor-actuator ECUs as well as the sensors and actuators. In the future, AUTOSAR Classic will also be used here. These ECUs manage the sensors and actuators whose geometric layout is widely distributed throughout the vehicle. They handle simple functions for open and closed loop control, monitoring and diagnostics.

> The second layer contains the integration ECUs. Both AUTOSAR Classic and AUTOSAR Adaptive are used here. These ECUs provide higher-order, integrative functions for the underlying sensor-actuator ECUs. They are networked with proven bus technologies such as CAN, CAN FD and LIN. The scope of integration is either defined in a function-oriented way, as previously, or if the goal is to minimize wiring harness expense, it may also focus on the installation space.
The third layer is formed by a few high-performance computers. AUTOSAR Adaptive is used here, which is supported by AUTOSAR Classic for fail-operational scenarios. But Linux or Android might also be used. The integration ECUs are interconnected over Ethernet.

The high-performance computers also connect to the fourth level via a Connectivity Unit: to the servers in the IT backend and to the mobile end devices.

The new high-performance computers often consist of a group of microprocessors and microcontrollers. Figure 2 shows an example. On the microprocessor, a Hypervisor is used as a host for various virtual machines and as a virtual switch. A different operating system may be used on each virtual machine (AUTOSAR Adaptive and Android in this example). AUTOSAR Classic is used on the microcontroller.
Feature Model for Vehicle Functions of a Product Line

The E/E architecture creates the framework for designing vehicle functions. As previously, development follows a product line approach. The development object is a system or a function, but it is not for a specific vehicle, rather it is for an entire vehicle family with many different powertrain, body and equipment options. The starting point for developing the system and functions is to define the vehicle features to be developed for this product line and to capture the necessary variants. This can be done in a list, or better yet in a feature model (Figure 3).

In AUTOSAR, four different logistical decomposition relationships are defined between a composition feature and its sub-features:
> mandatory (Must Use)
> alternative (Requires Xor)
> multiple (Requires Or) and
> optional (Optional).

Furthermore, the excludes and requires logical relations may also be used between any features in the composition tree. Strategic product planning is responsible for creating the feature model which sets key requirements for the E/E architecture.

Functional Development for Distributed and Networked Systems

System and functional development are performed in the prescribed framework of the feature model and E/E architecture for a vehicle product line. In recent years, acceptance has been very positive for a model-based specification of functional relations which supplementing the requirements-based system specification. This approach focuses on customer functionality from start to finish, i.e., from the participating sensors to the actual functions and necessary actuators.

The description is made with block diagrams on the abstract logical level – largely independent of the later implementation in software and hardware. While the implementation in hardware and software is subject to change cycles from one vehicle generation to the next, for many functions this logical architecture remains nearly stable over multiple generations. Many of these functions are distributed throughout the ECU network in the implementation. There are various reasons for this, and it offers many advantages. To name just a few beneficial aspects:
> The same sensors and/or actuators can be used for multiple functions
> Functional safety is implemented by hardware redundancy
> Geometric layouts can cover sensors and actuators located far from one another in the vehicle, and wiring harness expense can be reduced by a distributed implementation

Let us consider a simple example of a logical architecture involving two vehicle features X and Y (Figure 4): Feature X is implemented by the causal chain of the three blocks Sensor function 1, Function 1 and Actuator function 1. Feature Y is implemented by 9 blocks. There are overlaps between the two features. Organizationally, the blocks are uniquely assigned to the two systems A and B.

To assure later universality of the logical architecture in software implementation, the interfaces of logical functions are specified according to AUTOSAR. Along with AUTOSAR Classic, AUTOSAR Adaptive is also used as an implementation platform. Furthermore, consideration must also be given in the system design to functional content that is considered offboard – that is, outside the vehicle in the IT backend. This is the case in example of Function 6: It provides a service to Function 5 in the vehicle and uses an-
other service of Function 5. Along with the actual functionality, it may also be necessary to define the functional safety concept. The functional diagnostic and security concepts must also be defined. Finally, the individual functional blocks must be mapped to their associated sensors, actuators, ECUs and computing nodes. This results in the system component matrix:

<table>
<thead>
<tr>
<th>Component</th>
<th>System A</th>
<th>System B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor 1</td>
<td>Sensor Function 1</td>
<td>Sensor Function 2</td>
</tr>
<tr>
<td>Sensor 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor Actuator ECU 1</td>
<td>Function 1</td>
<td></td>
</tr>
<tr>
<td>Sensor Actuator ECU 3</td>
<td></td>
<td>Function 3</td>
</tr>
<tr>
<td>Integration ECU 2</td>
<td>Function 2</td>
<td>Function 4</td>
</tr>
<tr>
<td>High-Performance Computer 1</td>
<td>Function 5</td>
<td>Backup Function 6</td>
</tr>
<tr>
<td>Backend Server 1</td>
<td></td>
<td>Function 6</td>
</tr>
<tr>
<td>Actuator 1</td>
<td>Actuator Function 1</td>
<td></td>
</tr>
<tr>
<td>Actuator 2</td>
<td></td>
<td>Actuator Function 2</td>
</tr>
</tbody>
</table>

Signal- and Service-Orientation

Along with the previous signal-oriented perspective of AUTOSAR Classic for CAN, CAN FD and LIN (with sender-receiver interfaces, data elements, signals, PDUs, frames, etc.), in the area of Ethernet networking the service-oriented perspective is a second design perspective that may be needed in both AUTOSAR Classic and in AUTOSAR Adaptive. In this case, an important task is to define the service interfaces with methods, properties and events. In AUTOSAR Adaptive, the software architecture contains service provider ports and service consumer ports for this purpose. In AUTOSAR Classic, a service port is modeled as a bundle of sender, receiver, client and server ports. Cross-platform software relations can also be modelled with suitable port adapters (Figure 5).
Figure 5: Software architecture diagram with AUTOSAR Classic and AUTOSAR Adaptive components.

Figure 6: SOA diagram in PREEvision.

The SOA diagram abstracted from the software platform is used to define the interplay of various services of a service-oriented architecture (SOA). Along with the various participants, which correspond to the logical function blocks, it is possible to depict various possible scenarios as results of the Service Discovery as well as alternative service providers and other service consumers. Permissible service dependencies and the classification of services in a layer architecture can also be represented here (Figure 6). In addition, class diagrams according to the Unified Modeling Language (UML) are frequently used for the software specification. The behavior of function blocks and software components can be specified with state diagrams.
Conclusions and Outlook

The system developer is responsible for the functional specification, defining the functional variants and distributing the function blocks to the various ECUs and computing nodes in the vehicle and in the backend. The system developer creates the functional safety, diagnostic and security concepts. Model-based specification of the logical functional architecture offers many benefits here: for tuning the functional components together with software architects and developers and tuning communication requirements with communication experts. The necessary functional variants flow from the higher-level feature model. The design space of E/E architectures is now much larger than before. For one, it comprises the signal-oriented software architecture in the vehicle in the area of sensor-actuator ECUs and integration ECUs. Then there is the service-oriented software architecture in the vehicle’s integration computers and high-performance computers as well as in the IT backend (Figure 7).

Various software platforms are used here. AUTOSAR Classic, AUTOSAR Adaptive and others. The service-oriented software architecture in the vehicle is implemented in part by AUTOSAR Classic and in part by AUTOSAR Adaptive. Abstract modeling on the SOA level is therefore just as necessary as integration of AUTOSAR Classic and AUTOSAR Adaptive software components. In the early phase of system design, it is already necessary to carefully balance the capabilities and limitations of the various software platforms. For example, the new capabilities for over-the-air upgrades and updates of the software by AUTOSAR Adaptive can only be used if they are implemented together with the necessary safety and security concepts. The opportunities arising from integration of functions in the IT backend can only be exploited if onboard fallback scenarios are well thought out and implemented. They assure correct overall functionality should the network connection fail. All of this leads to a situation in which system and functional development is, and will remain, one of the most creative and complex tasks in the overall vehicle creation process. It can only be successful and viable for the future if the system and functional relations are precisely captured and are transparent for all participants.

As can be seen from the examples described above, this can only be achieved by a model-based approach to working and not by a prose-based specification. Driven by these requirements – and inspired by AUTOSAR – the PREEvision E/E development environment from Vector offers a multi-user platform and supports the described modeling concepts: for specifying E/E architectures, feature models, systems, functions, services, software components and hardware components. PREEvision also offers universal consistency in neighboring application areas: for requirements and test specifications, and in communication design and wiring harness design. Finally, from all these vehicle-wide model levels, it enables automatic generation of consistent component specifications at the automotive OEM and derivation of necessary documents as well as relevant exchange formats (such as ReqIF and AUTOSAR) for suppliers.

Originally published in “Elektronik automotive” magazine
Issue 2/2020 – February 2020