The part of ECU software that performs safety-related functions is growing constantly. This leads to a more frequent interaction between safety-related application software and the AUTOSAR basic software. If the two are located in different partitions, the interaction takes additional computing time because, at the very least, the active task must be changed and, depending on the hardware platform, the memory protection unit (MPU) may need to be reprogrammed as well. However, basic software developed according to the requirements of ASIL D, allows its coexistence with safety-related application software in a single partition. This allows a significant performance gain, since task switching, reprogramming the MPU, and additional copying of data can be avoided. Moreover, additional but frequently needed safety requirements can be implemented in the basic software, making it unnecessary to develop them repeatedly for the individual ECU projects. Figure 1 compares the two approaches graphically.

**Background**
ISO 26262 defines the automotive safety integrity level (ASIL) as an attribute of a function implemented in software. It indicates the quality with which the function is to be implemented. The ASIL is derived from the hazard analysis and risk assessment of the item under development. It is assigned to the safety goals and propagated to the functional and technical safety concept, down to the safety-related software requirements. As soon as a software component implements a safety-related requirement holding a certain ASIL, the corresponding methods for its development, taken from Part 6 of ISO 26262, must be applied (figure 2). Well-known examples of such safety-related functions in the basic software are memory protection by the operating system, temporal monitoring by the watchdog stack, and the safeguarding of communication using end-to-end (E2E) protection as described in the AUTOSAR specification [1].

Safety and Performance with ASIL D AUTOSAR Basic Software

In electronic control units (ECUs) that are developed according to ISO 26262, safety-related and non-safety-related software is often used in parallel. To prevent them from interfering with each other, partitioning measures have previously been used. However, this partitioning often leads to overhead in runtime and higher complexity. The AUTOSAR basic software, developed entirely according to ISO 26262, reduces the number of partitions to a minimum. A report from Vector Informatik.
The use of AUTOSAR E2E protection is the state of the art for safeguarding safety-related signals. Since it allows all relevant faults to be detected, no safety requirements are imposed on the basic software’s communications components. In order to avoid partitioning between E2E protection and communications components, the methods from ISO 26262 Part 6 must be applied for the communication components as well. This is because it is assumed that software with a lower ASIL contains more systematic faults than software developed according to a higher ASIL. Without partitioning, these faults can affect safety-related software as cascading faults, thus violating the designated safety requirements. The potential hazards caused by interference are to be prevented. The solution ISO 26262 offers is developing non-safety-related software components using the same development processes used for safety-related functions. Those software components are then considered to provide sufficient freedom from interference.

**Consequences for Development**

An implication of this argument is that, among other things, an extensive semi-formal design must be developed for the software in question. In general this implies additional development efforts to achieve the requirements for ASIL D. However, the increased effort during design and implementation is low, compared to the high demands on verification. This is especially true for testing.

The requirement of diverse implementation of algorithms as the result of the software safety analysis can be derived from ISO 26262 Part 6 Table 4. The algorithm is implemented in two different ways in order to prevent systematic faults. To avoid this diverse implementation, Vector builds on the excellent testability of its components. This is demonstrated by the low cyclomatic complexity of the software components under test. Cyclomatic complexity is a measure of the paths in the control flow graphs of a function. In order to meet the strict requirements of component testability, many components had to be refactored, i.e. their source code structure had to be improved. This process was also used to improve defensive programming, meaning the increased robustness resulting from additional checks, and the modularization within the components.

Another challenge is the extensive configurability of the AUTOSAR basic software. ISO 26262 only deals with this configurability partially. In particular in the area of unit tests, ISO 26262 implicitly assumes only one possible code variant. For the basic software to be used in a wide variety of configurations as a safety element out of context (SEooC) [3], an approach had to be found to cover this configurability. This configurability is partially reflected in the pre-processor conditions. Vector deduces possible configurations on the basis of use-cases and dependencies between configuration parameters. In its development process, Vector applied the same standards for the exit-criteria to its pre-processor conditions as it did for runtime tests [4]. Moreover, Vector has defined additional exit-criteria for testing in its component development: Complete code coverage for runtime tests in each configuration variant that is necessary for fulfilling pre-processor coverage, must be achieved.

Static code analysis is an important part of the process, since it allows certain types of errors, such as memory corruption, to be found significantly more reliably than a functional test. Specifically for the context of configurable software, Vector developed a static code analysis for detection of out-of-bounds access and invalid pointers.
Hazard and risk analysis

Safety goal

Functional safety requirements

Technical safety requirements

Software safety requirements

Partition

Software component A

Software component B

Derived ASIL according to methods of ISO 26262.

By using the same methods in the development of component B, a coexistence with A in the same partition is possible.

Figure 2: Development of software components according to ASIL resulting from safety requirements or a desire for coexistence.

Because of the many variants of the basic software, this involves automated verification of the generated configurations that is performed by the user. This ensures freedom from interference with respect to memory between non-safety-related parts and safety-related parts of the basic software.

Safety Concept

The safety-oriented development process described above allows the definition of new safety requirements for the SEoC basic software because almost all relevant requirements from ISO 26262 have already been met. Vector has formulated additional safety requirements for the basic software: detection of loss, corruption, and masking of data in non-volatile memory – caused by the lower layers of the basic software or the hardware. Restarting of software applications and AUTOSAR timing protection can be used as a safety mechanism in the future, too. Vector has also assumed safety requirements to bring an ECU into a defined condition during initialization. The safety requirements that are necessary for partitioning are usually still required. For example, QM software, which is often present, must be integrated because a cost-benefit analysis has shown that new development or qualification (acc. ISO 26262 Part 8 Clause 12) is not economical.

Assumed safety requirements must be analyzed carefully and the underlying assumptions communicated before these requirements can be used in a safety concept of an ECU project. This means that only functions that are part of the assumed safety requirements may be used in the project-specific safety concept. For example, E2E protection is necessary to transfer safety-related data because Vector has no assumed safety requirements for the communications components, even though they have been developed according to the methods in ISO 26262. A list of assumed safety requirements should be made available in a very early phase of a given project.

Integration

After the user of the basic software has validated the assumed safety requirements with the project context, he uses the safety manual included in the delivery to perform a configuration and integration of the basic software. A unit test for the basic software is normally already performed by its supplier. The integration and verification step, as described in ISO 26262 [5], must be performed by the user, however. The basic software supplier cannot contribute to the safety case at these levels, since he has insufficient information about configuration, application, and the exact target environment during development of the basic software. The user receives confirmation of compliance with an ISO 26262-conformant development process through the safety case for the basic software. The user references this safety case for the creation of his safety case then. The safety case is created on the basis of the configuration submitted by the user, among other things. This check of the configuration by Vector is necessary for validating the component test. Not all possible combinations of configuration parameters can be tested during development. In order to be certain that even exotic configurations are tested, project-specific testing is necessary.

The effectiveness of this approach has been confirmed in the exemplary assessment by the certification company exida in an independent assessment. During the assessment the operating system, the components for CAN, LIN, and FlexRay communication, and the components in the system and memory cluster were evaluated. Figure 3 shows the current overview of the components developed according to ASIL D.
# Outlook

The approach that has been introduced shows how runtime overhead and the number of partitions can be reduced with ASIL D basic software. Additionally, the complexity in an ECU project decreases because safety mechanisms do not have to be implemented by the user again, but are already included in the basic software.

Vector’s goal is to develop more AUTOSAR 4 basic software components according to its new, safety-oriented process, thus offering an optimum solution for as many application cases as possible. Challenges faced in the near future are AUTOSAR components, which undergo many changes. Another challenge is the extensions provided by OEMs and the hardware drivers of the various semiconductor manufacturers, which are not yet available at the necessary quality and can therefore be used with safe basic software only with increased effort and sacrifices to performance. In an additional step, Vector and exida are working on the implementation of an RTE that can be used in the context of safety-related software without costly qualification measures.