360° Safety

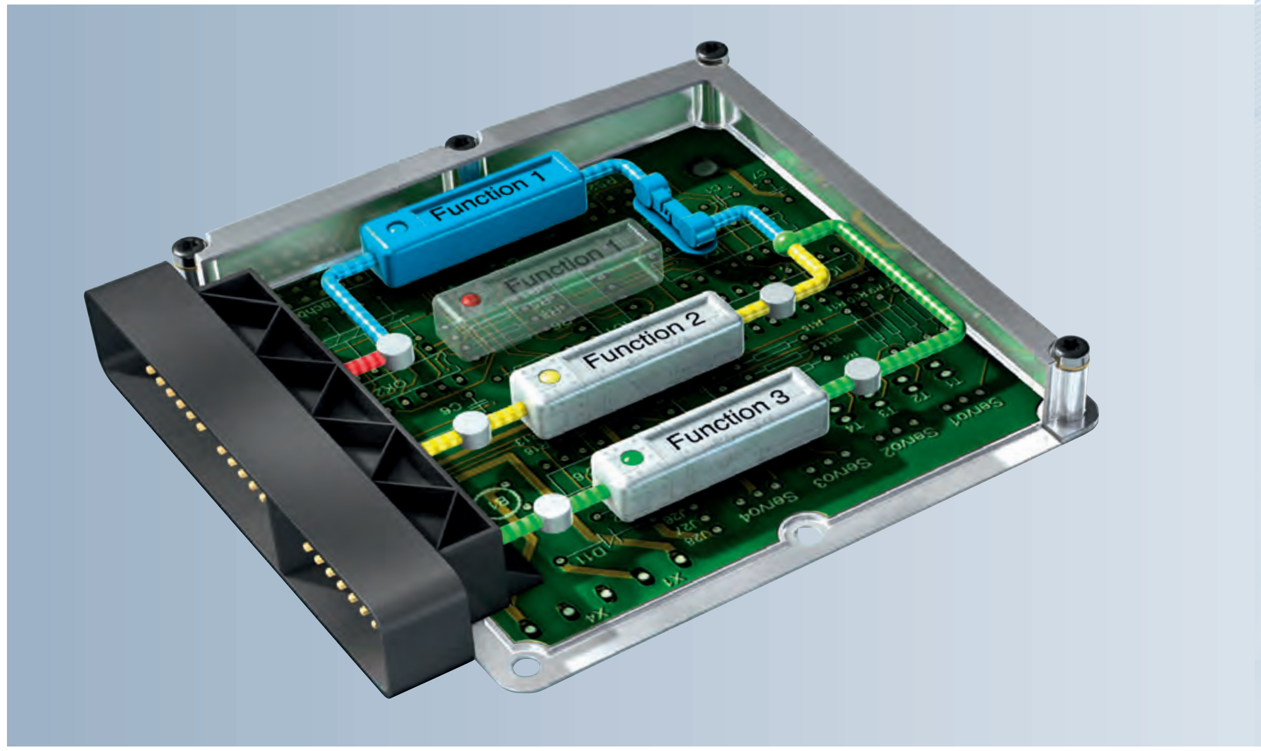
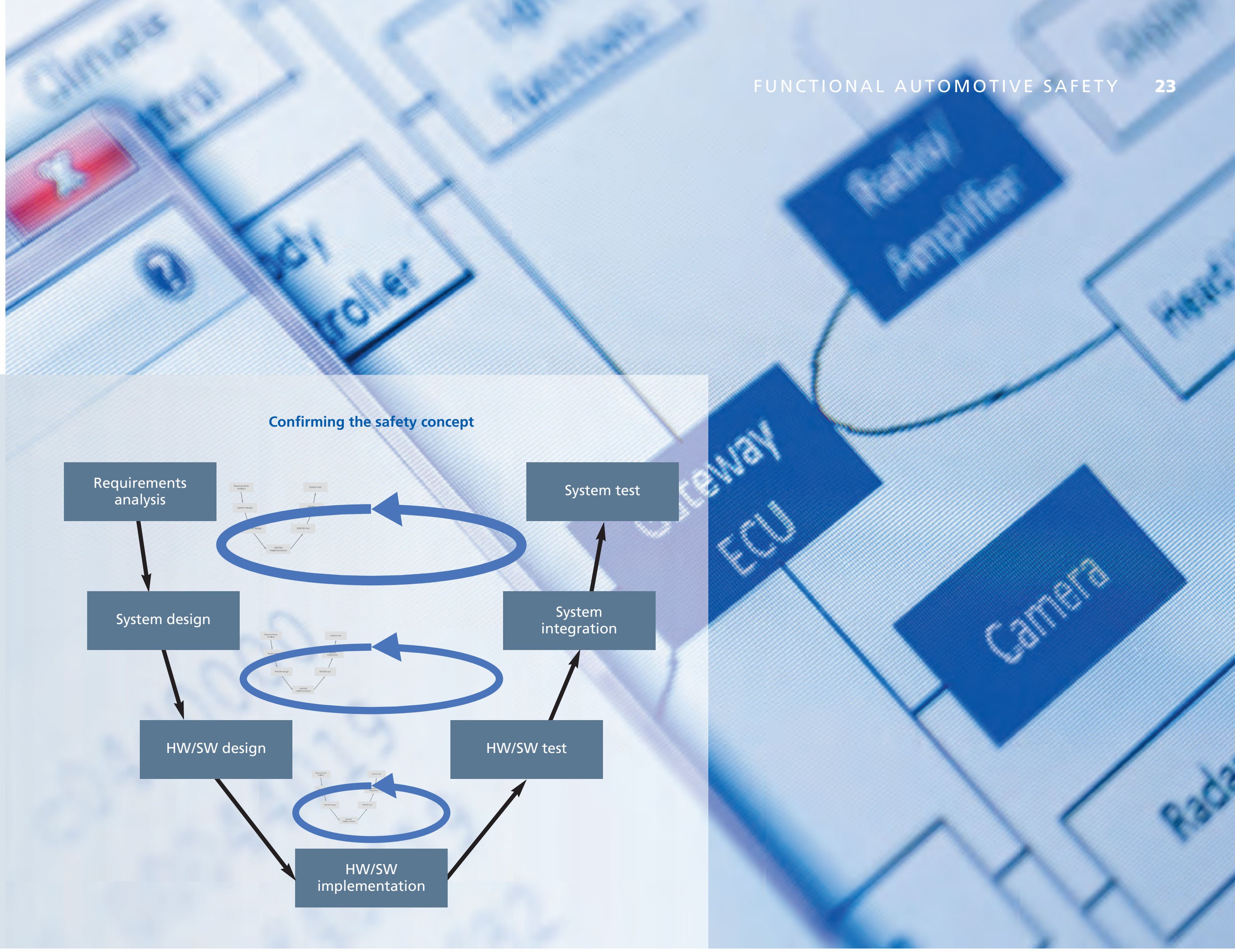


Figure 1:
With bypass hook tools, such as ETAS EHOOKS, errors can be deliberately introduced to test a system's robustness.



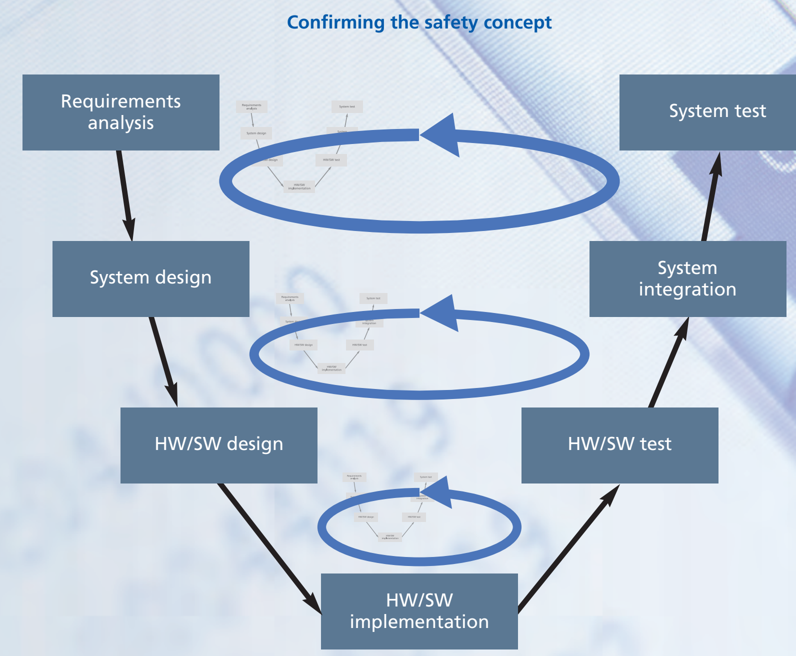


Figure 2 (diagram):
The safety concept must be examined throughout the entire development cycle – testing alone is not enough.

Rigorous development process for functional, safe ECUs

When distributed developer teams from several companies work on the same control unit, then functional safety also becomes a question of organization. This requires merging methodological know-how, proven development tools, and technical expertise into one rigorous process.

AUTHOR

Dr. Simon Burton is Director Global Embedded Software Services at **ETAS GmbH**.

More and more functions in cars are being executed by software in electronic control units (ECUs). Software functions that are connected across ECUs are on the rise, too. Some high-end models feature more than one hundred ECUs that are connected over data buses and communicate with sensors distributed throughout the vehicle. Connectivity between cars and their surroundings is also rapidly expanding, meaning that cars are becoming network nodes in the complex, connected world of mobility.

Although this connectivity promises safer, more efficient driving, it is not without risks. This is partly due to an increasingly complex development process for connected ECUs, often involving internationally distributed teams. Not only are an ECU manufacturer's developers and calibration engineers involved in developing ECU software and pre-calibration, but so are its suppliers and the automaker's developers. Different teams working on engine ECUs deal with fuel injection, air supply, ignition, and many other parameters. Furthermore, the entire process could take place several times, because OEMs often order identical ECUs from several suppliers to assure delivery, although of course car buyers must not notice any difference. Independent of their source, these ECUs and their software must function reliably, functional safety must be guaranteed.

Safety in the collaborative development process

How can functional safety be ensured under these conditions, especially when specific software functions are distributed over multiple ECUs and only a portion of the software is to be replaced? One key to success is to use innovative methods and tools, such as virtualization, which make it possible to validate software in realistic plant, environment, and driver models at an early stage of development. A tool such as ETAS' ISOLAR-EVE software, combined with PC simulation, allows developers to start testing like this long before an ECU prototype exists. Errors or faulty assumptions are discovered before any damage occurs. Developers can also sound out borderline areas in a virtual environment without any danger, which is particularly helpful when it comes to designing safety-critical assistance systems.

But virtualization is just one building block in a more complete, total architecture. The entire development process must be structured in advance (Figure 2) and include clearly defined software architecture and how each team will contribute. Furthermore, close process monitoring is crucial. Regular assessments and audits should be performed to confirm that teams have internalized the agreed upon safety philosophy, and that all participants share the same interpretation. Standards such as ISO 26262 provide the basis for this.

Established rules for software and development processes

The first step in preparing software development is to define the scope of the item under development. A top-down approach begins by considering the overall system and context and setting the limits for the functional scope of the interfaces and of interactions with other systems. Once this has been done, a structured hazard and risk analysis is conducted in which the likelihood and controllability of possible errors are weighted equally against the threatened extent of damage. The aim of the analysis is to define binding safety objectives. From this point on, these objectives are the guiding principle for development – and must first be broken down into specific work packages for the work groups.

Programming must be supported by clear rules and tried-and-tested methods, such as a focus on best practice, compliance with style guides, structured documentation, as well as dedicated regular reviews and code analyses. A testing plan is necessary as well. When and how will the software be tested, and in which contexts? Fault injection tests, which check the software's response against deliberately introduced errors, are indispensable. To conduct these, developers feed in faulty data using a bypass hook tool such as ETAS' EHOOKS. In contrast to earlier tests with ISOLAR-EVE, this is performed on real hardware (Figure 1). Using EHOOKS, test engineers can manipulate ECUs by replacing their internal signals with the planted errors or faulty data, with the ECU's calibration and diagnostic interface functioning as an access point. The faulty data is then generated with the ETAS INCA measurement, calibration, and diagnostics software.

Safely limit functions with AUTOSAR

It is by provoking these exceptional circumstances that developers can tell whether safety objectives and concepts really hold up. Robust software must recognize faults and either keep the ECU operational or bring it into a safe state. Precautions can also be taken to ensure that errors don't spread. This is where the AUTOSAR standard with memory protection helps: Developers can use hardware support to prevent software from accessing the memory of other software features. This ensures that errors remain locally contained and cannot compromise any safety-relevant software applications.

However, to integrate the AUTOSAR memory protection or timing protection mechanisms, all project participants must make their software components' source or object code public. Since this degree of transparency is often lacking or not desired, ETAS developed the RTA-HVR Hypervisor. This partitions the ECU into several strictly divided virtual ECUs and thus completely shields safety-relevant functions from each other. Communication between the partitions works in the same way as between various ECUs using defined interfaces and predetermined rules.

Technical and organizational partitioning

Partitioning offers another significant advantage: Teams from different companies can develop software independently from each other that will run within their own isolated, shielded partitions later. This means reciprocal access to the code is not necessary in the early stages of software development. Even so, parallel development requires the ECU manufacturer to coordinate and guide the process. Out of self-interest, all participants must commit to the same safety objectives and a binding schedule. This outlines when development partners must provide proof of functional safety, and in what form. The ECU manufacturer remains responsible for integration – merging the tested software components and their documentation from its suppliers and furnishing complete proof of the ECU's safety.

Prudent project management

Constantly validating, justifying, and – where necessary – readjusting assumptions are part and parcel of project management. Partners also need to be kept up to date. But the effort pays off: in instances in which a collective safety philosophy guides development, testing effort is decreased and expensive, nerve-racking corrections are eliminated in the final development stage. Less strict project management is often plagued by problems. Audits and assessments often bring to light different interpretations of safety objectives, presuppositions, and norms; without targeted countermeasures, these could soon give rise to expensive error chains. That is precisely why these checks are firmly anchored in the ISO 26262 safety standard.

Summary

It is possible to ensure the functional safety of ECU software, even in complex collaborative development processes. This requires more than just expertise in ECU development, however. It has far more to do with thoroughly embedding standardized development and modern methods, such as validation in a virtual environment, into a carefully controlled development process. With tools, services, and consulting, ETAS offers the right solution for every project phase.