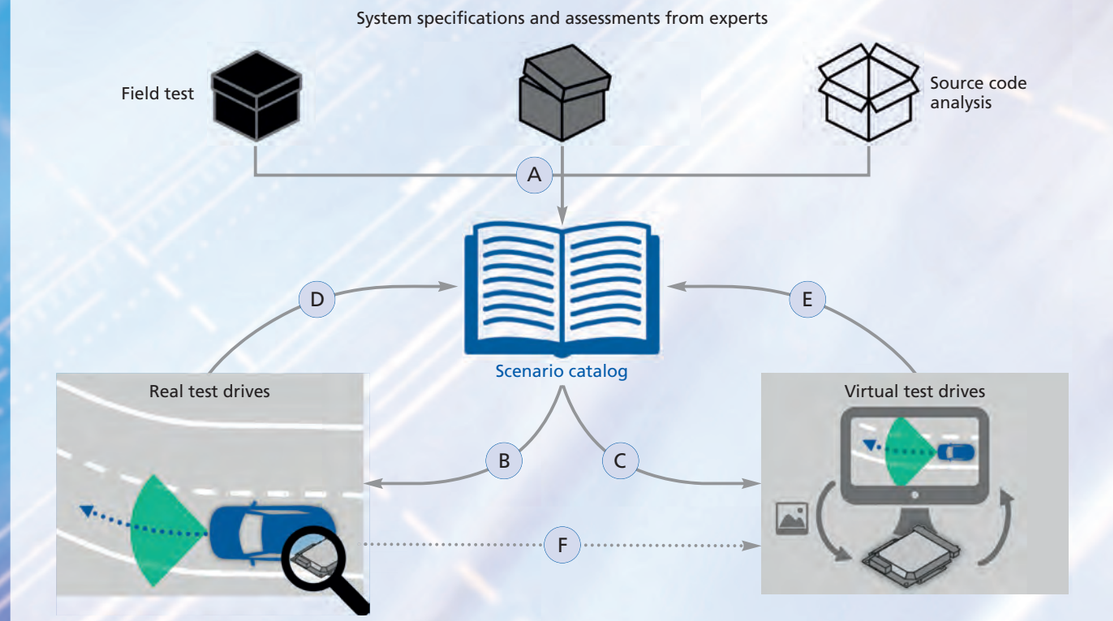
Robust Assistance Systems

Iterative validation strategy

Software-controlled autonomous vehicles must be validated. But how can this be done efficiently? Because it simply isn't possible to address all eventualities of actual traffic situations in a system specification that is drawn up in advance, robust software systems are required. Software robustness is achieved by performing tests in a virtual environment based on a catalogue of scenario descriptions that can be expanded on an iterative basis.



The validation approach. A-F = process steps (see text).

How will drivers use the time that self-driving cars free up for them? It's a personal choice. Some might spend it doing paperwork, shopping online, or relaxing – the possibilities are endless. For drivers to feel comfortable taking their hands off the wheel naturally, they must trust the technology in all aspects.

It is no longer enough to prepare active assistance systems in accordance with ISO 26262 for the event of system failure at the functional level. Autonomous driving requires safe-guards to be built in to ensure that situations aren't misinterpreted. The problem is that there are too many different influencing factors such as traffic, weather, and light conditions to address all eventualities in the system specifications.

In the world of advanced driver assistance systems, this problem has a name: functional insufficiency.

Its solution is referred to as robustness. Robust software must work as defined and take appropriate action even in unusual situations. In the context of autonomous driving, this means bringing the robustness of the software-controlled assistance systems up to a socially acceptable level despite the inevitably incomplete specifications. In other words, taking the most pragmatic approach possible to create the safest systems possible.

Equivalence-class-based scenario descriptions

Recently published technical studies demonstrate the magnitude of this task. Darmstadt-based Professor Hermann Winner, for example, presents a probability-based validation approach for a self-driving car. His method shows that, even in this relatively straight- forward use case, a test vehicle would have to drive $2.4 \cdot 10^8$ **kilometers** to establish the learning required to prevent half as many accidents resulting in injuries as do vehicles without such systems. The aim of such a validation process is to demonstrate the probability **P** that a system meets metric **M**. Based on a test site d_E **= highway**, probability is determined by **P(M|highway)**.

If the system behavior is to be observed somewhere other than the highway, then the test site is d_E **= non-highway**. The overall probability of meeting the metric, therefore, is: $P_{tot}(M) = P(M|highway) \cdot P(highway) + P(M|non\text{-}highway) \cdot P(non\text{-}highway)$. Test drives may reveal that the non-highway sites must be split into urban and rural areas. If the system to be tested behaves in an equivalent fashion within a defined area, then the test site can be split into three equivalence classes: d_E **= {highway, urban, rural areas}**.

During a test campaign, it could be that the system works perfectly on dry roads, but frequently fails on wet surfaces. This finding can be incorporated into the description by introducing a further dimension, road conditions: d_S **= {dry, wet}**. As a result, there are already six scenarios in which the metric must be tested.

Generally speaking, a scenario **S** can be defined as the combination of each equivalence class with each dimension: $S = [d_1, d_2, \ldots d_n]$. The overall probability

$P_{tot}(M) = \sum P(M|S_i) \cdot P(S_i)$

of meeting the metric is generally calculated for n dimensions d_n, where the number of scenarios i is the result of the cardinality of the equivalence classes of all dimensions, i.e., $i = \prod |d_n|$.

In the experiment, specific test cases are needed to determine $P(M|S_i)$. Each test case represents one scenario. The overall sum of the various scenarios serves as a reference value for the number of different test cases and is essential for reliable system validation.

Iterative expansion of scenario descriptions

Scenario descriptions, which are based on equivalence classes and refined with an iterative method, can now be applied in an application-based process for validating autonomous systems. This process is depicted in the figure left.

At the core of the iterative validation strategy is a scenario catalog based on information drawn from three sources (A): findings from endurance and field tests in which the system did not behave as expected, scenarios created on the basis of system specifications and assessments from experts, and scenarios generated from statistic source code analyses.

This scenario catalog is used for systematizing real test drives (B) and also serves to parameterize virtual test drives (C). The latter offer the advantage that they are not dependent on the availability of expensive test vehicles and can be run simultaneously on any number of computers. A further advantage of virtual testing is that engineers can take critical situations that occur during real test drives and reproduce and modify them as needed. Developers can subsequently use these variations to derive new scenarios, which they can then analyze and add to the catalogue (D, E). This ensures that test coverage is continuously improved.

A prerequisite for the validation of the overall cross-domain simulation of assistance systems is the validation of the underlying models (F) in a comparison of real and virtual test drives. Only by making this comparison is it possible to make reliable statements regarding the accuracy of the overall simulation as well as the scopes of the models. Moreover, this process gradually produces an increasingly precise and comprehensive basis for the virtual testing of assistance systems. As a result, virtual testing becomes a key means of increasing quality while reducing cost, time, and administrative effort.

AUTHORS

Marius Feilhauer works in Test and Validation at **ETAS GmbH**, where he is responsible for the development of simulation models for driver assistance systems.

Dr. Jürgen Häring is head of product management in Test and Validation at **ETAS GmbH**.