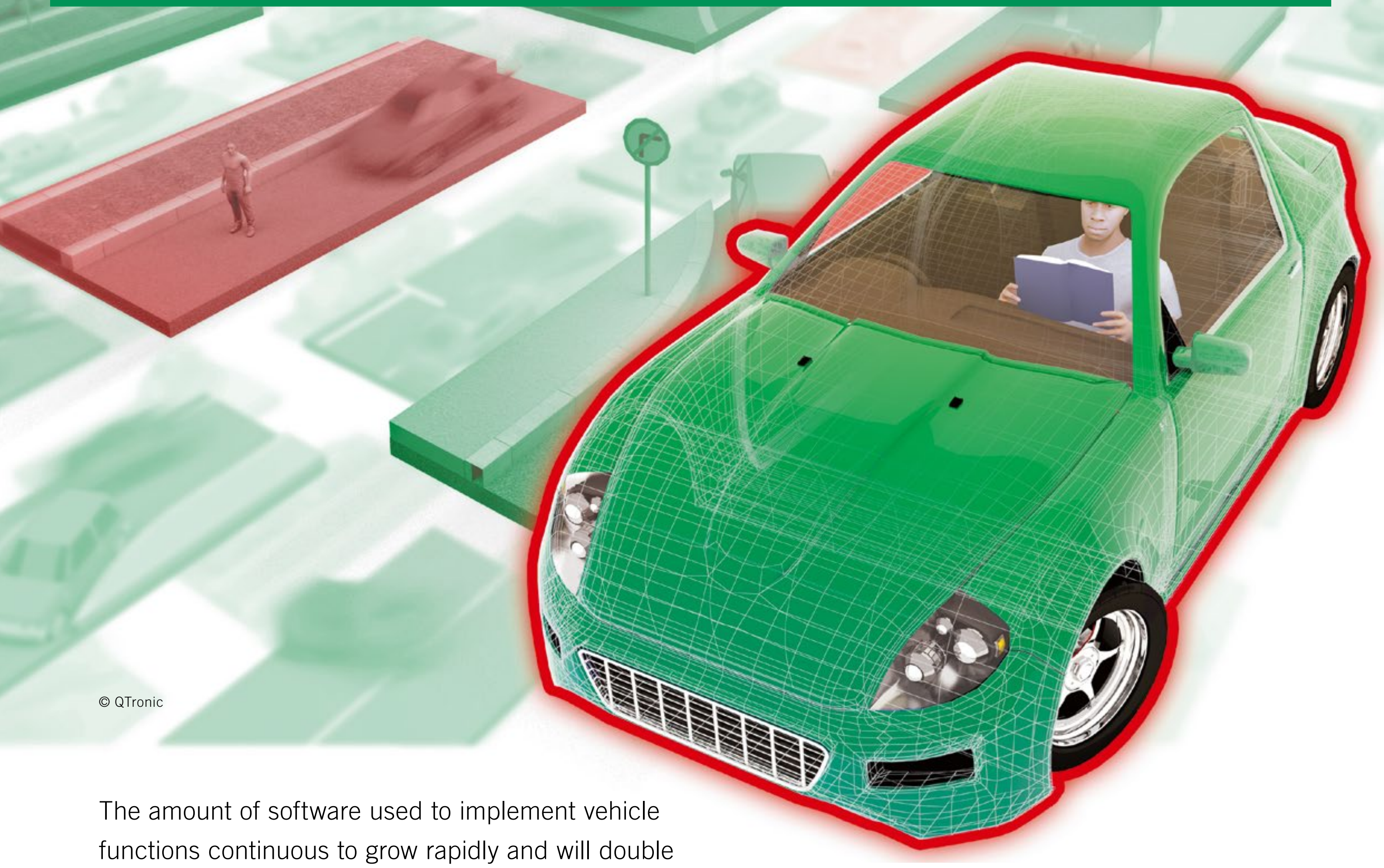
# Test and Validation of Advanced Driver Assistance Systems
## Automated Search for Critical Scenarios


© QTronic

The amount of software used to implement vehicle functions continuous to grow rapidly and will double once more within the next decade. Established test and validation processes can hardly keep pace with this growth rate. In this context, autonomous testing, a technique that evolved substantially in the recent years, gains importance as an enhancement to the traditional testing process. Testing on a real test drive or with handwritten test scripts on a HiL or test bench is complemented by a fully automated search for design flaws and faults using thousands of scenarios simulated on PC. QTronic describes such a test process in detail and surveys first applications in the domain of advanced driver assistance systems (ADAS) and autonomous driving.

AUTHOR

**Dr. Mugur Tatar**
is Managing Director and Head of Test Systems at QTronic GmbH in Berlin (Germany).

## VIRTUAL TEST DRIVES

The systematic test of functions for driver assistance systems and autonomous driving requires the identification and analysis of a huge number of traffic scenarios. The space of possible scenarios is spanned by many dimensions: the road geometry, the behaviour of the driver and of other traffic participants, weather conditions, vehicle characteristics and vehicle variants, spontaneous component faults, the timing of the events and others. This multidimensional space can hardly be covered comprehensively using real test drives. It is therefore widely believed that real test drives must be extensively complemented by virtual test drives, for example using established tools for vehicle and traffic simulation such as CarMaker (IPG), PreScan (Tass International) or VTD (Vires). Simulation seems to be the only option to synthesise and analyse the enormous amount of required traffic scenarios with reasonable effort, **FIGURE 1**.

Another building block for the virtualisation of the test drives are the virtual ECUs. A virtual ECU (vECU for short) is an executable model of the real ECU that runs on PC. A vECU executes the same application software as the real ECU and, if required, even parts of the basic software. A vECU enables therefore the detection of software faults on PC, without using the real ECU hardware.

Ideally, a vECU is available in a binary format that allows an easy integration with the vehicle model used by the vehicle and traffic simulator. An example for an established tool for building and simulating virtual ECUs is Silver (QTronic). Silver [1, 2] can export a vECU as Simulink SFunction of as an FMU [3]. A Silver vECU can be integrated with CarMaker using CarMaker‘s build-in FMU plug-in interface, **FIGURE 2**. Interfaces to the vehicle buses, e.g. CAN, Lin, Flexray, are included in the FMU plug-in when necessary. This way, an entire network of communicating vECUs can be integrated into an accurate vehicle model that can be simulated on a PC.

## AUTOMATED SEARCH FOR CRITICAL SCENARIOS

A key idea for automating the search for weaknesses and faults of vehicle systems using simulation is the following analogy: Testing is similar to playing chess. A chess player attempts to find a sequence of moves over time that drives his opponent into a mate position. Likewise, a tester attempts to find a sequence of inputs over time that drives the system under test into a state where it fails, i.e. it violates its specification or, at least, performs badly with respect to given quality indicators. Chess computers are incredibly strong and can beat virtually every human player today. The analogy “testing is like chess” shows a way how to transfer the enormous power of chess computers to the testing domain in order to create a powerful, autonomous testing machine, that is maybe even able to outperform the human testers, or at least to complement the testing teams.

Since 2006 the German company QTronic has developed TestWeaver, a tool for test automation that implements exactly this approach. To autonomously test a system, TeatWeaver requires an executable model of the system under test. TestWeaver uses this model as a “black box”, that is, it can run the model, but has no access to the model‘s source code. In practical applications complete model sources are often not available anyway. However, TestWeaver can stimulate the inputs and can observe key outputs of the simulated model. Some of these outputs (called “alarms” in **FIGURE 3**) act as function quality indicators. During a test, TestWeaver runs many simulations and drives the model with many differing input sequences within a configured time horizon. The outputs of the system are continuously monitored and classified according to multiple criteria. TestWeaver attempts to find (actually synthesise) driving scenarios within the time horizon that, on the one side, minimise the given quality indicators and, on the other side, maximise test coverage in the system state space. To support the second goal, TestWeaver generates scenarios such that they cover different regions of the state space. The system state space contains in general an infinite number of states, due to real-valued dimensions, such a vehicle’s speed. To deal with this, a TestWeaver setup discretises the real-valued dimensions of the state space, i.e. partitions each axis into a number of intervals. This discrete view on the state space, depicted as a grid in **FIGURE 3**, is the foundation for TestWeaver‘s approach to measure and optimise the test coverage. One reason for TestWeaver‘s power to find “bad” scenarios is its reactivity. TestWeaver does not generate scenarios in advance for later execution, but interleaves generation, simulation, classification and planning of the next scenarios. Therefore, the planing of the next scenarios depends on the information gained during the execution of all previous scenarios. This is the key to TestWeaver‘s ability to synthesise local worst case scenarios and to improve the test coverage.

For testing a driver assistance system, the following procedure proved to be useful: Develop a parametrised scenario, implemented for example as a script that drives a traffic simulator. The parameters are used as inputs for TestWeaver and determine the temporal order of events, speeds and accelerations of the traffic participants, etc. The functions

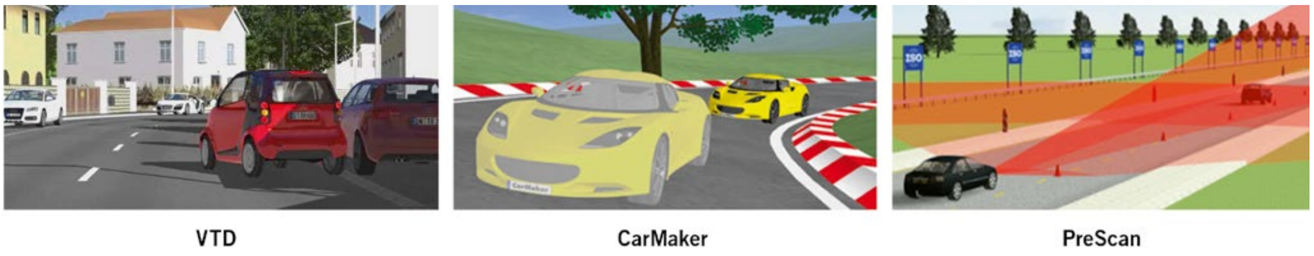


**FIGURE 1** Tools for virtual test drives (© QTronic)

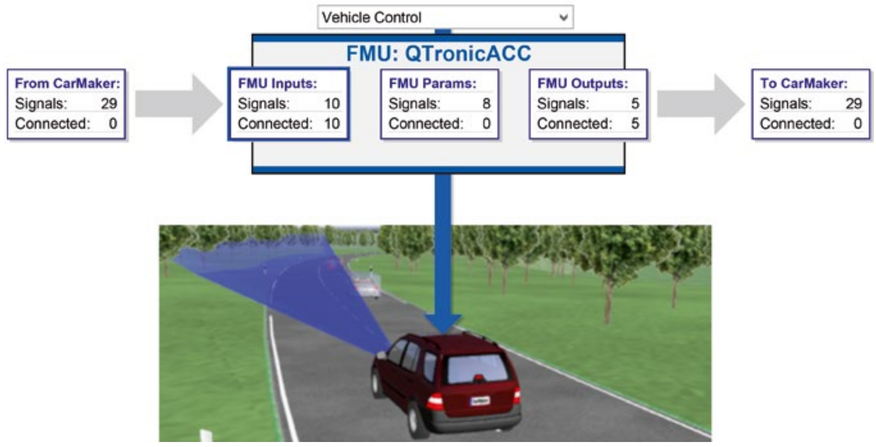

Vehicle Control
FMU: QTronicACC
From CarMaker: Signals: 29 Connected: 0
FMU Inputs: Signals: 10 Connected: 10
FMU Params: Signals: 8 Connected: 0
FMU Outputs: Signals: 5 Connected: 5
To CarMaker: Signals: 29 Connected: 0

**FIGURE 2** Integration of a virtual ECU for ACC, Silver FMU running CarMaker (© QTronic)

of the ADAS system are monitored by "alarm" outputs that measure the safety and the quality of the tested driver assistance function. During an experiment TestWeaver varies the scenario parameters, attempting to minimise the given quality indicators, **FIGURE 3**, in differing state space regions. A typical TestWeaver experiment runs on one or several PCs for a few days. During an experiment TestWeaver reports continuously the current test coverage as well as all problems found so far. The simulations need not be performed in real time. Without connection to real hardware the simulations can often be performed a few times faster than real time, which helps to achieve a higher test coverage. The critical scenarios found during an experiment can be reproduced later in simulation as often as needed for detailed analysis. This provides the function developers with precise hints on how to best improve the functions of interest.

## INTEGRATION IN THE TEST PROCESS

The adoption of the ISO 26262 safety standard is currently spreading in the automotive industry. Requirement traceability and requirement-based testing are common techniques that are more and more implemented in the development process. It is important to note that TestWeaver supports requirement-based testing as well. Requirements can be imple-

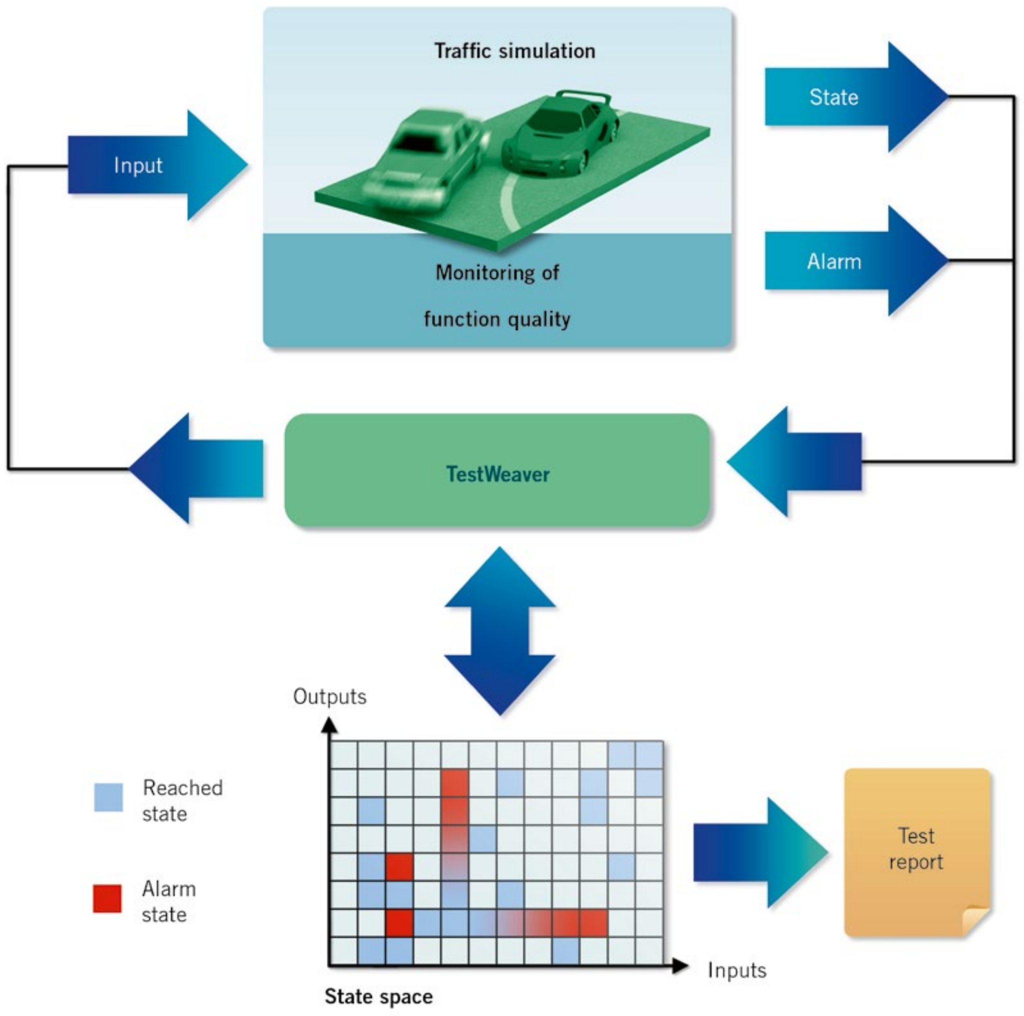


**FIGURE 3** Automated search for critical scenarios with TestWeaver (© QTronic)

mented as observers of the system simulation that provide “alarm” signals when requirement violations are encountered in a simulation. Moreover, in the case of safety-critical functions ISO 26262 recommends further more advanced test methods that go beyond the simple requirement coverage, for instance: consideration of source code coverage goals, fault injection tests, test derived by analysis of equivalence classes and min-max values and others. TestWeaver is a tool that supports specifically the generation of tests with a high coverage, especially at higher levels of functional integration. It is thus increasing the safety of the test process. Nevertheless, we see this method as an enhancement of the current manual / script-based test process, rather than as a replacement of the established test and validation process.

## APPLICATIONS

The TestWeaver method has originally been developed to test the software of transmission control systems. In conjunction with Silver‘s vECUs, TestWeaver is used in the series development process by AMG, Mercedes-Benz [4] and other OEMs and suppliers to test automatic and hybrid transmission systems. Since 2009, the method is also applied in the ADAS domain. For example, [5] reports how TestWeaver has been used to develop a new side-wind stabilisation function, in conjunction with Daimler‘s in-house tool CASCaDE for vehicle simulation. TestWeaver generated thousands of differing road and wind scenarios that helped to validate and optimise this function. [6] describes other applications of TestWeaver in the context of ESP (electronic stability program), namely for trailer swing mitigation. TestWeaver offers interfaces to CarMaker (IPG), PreScan (Tass International) and VTD (Vires). Applications based on these tools are currently under development.

## CONCLUSION

Today virtual test drives are used to aid the development of ADAS and of functions for autonomous driving. Virtual test drives provide a solid foundation to further automate the test and validation processes. This is urgently needed because the established manual and script-based testing does not scale well with the growing number of complex functions that need to be tested. An example of a highly automated test process using virtual test drives and automated search for critical situations with TestWeaver was presented in this paper.

**REFERENCES**

**[1]** Junghanns, A; Serway, R.; Liebezeit, T.; Bonin, M.: Building Virtual ECUs Quickly and Economically, ATZelektronik 7 (2012) No. 3

**[2]** Mauss, J.: Virtuelle Steuergeräte für die Antriebs-entwicklung. Tagung VPC - Simulation und Test 2015, Hanau bei Frankfurt am Main

**[3]** https://www.fmi-standard.org/

**[4]** Gloss, S.; Slezák, M.; Patzer, A.: Systematic Validation of over 200 Transmission Variants, ATZelektronik 8 (2013) No. 4

**[5]** Hilf, K.-D.; Matheis, I.; Mauss, J.; Rauh, J.: Automated Simulation of Scenarios to Guide Development of a Crosswind Stabilization. IFAC Symposium Advances in Automotive Control, München, 2010

**[6]** Lutz, A.; Baust, B.; Steiner, B.; Vogler, M.: Gain of Efficiency and Robustness in the ESP-Application Process using Vehicle Dynamics Simulation with DoE-Methods, 12. International Symposium on Advanced Vehicle Control (AVEC 2014), Tokyo, Japan, 2014