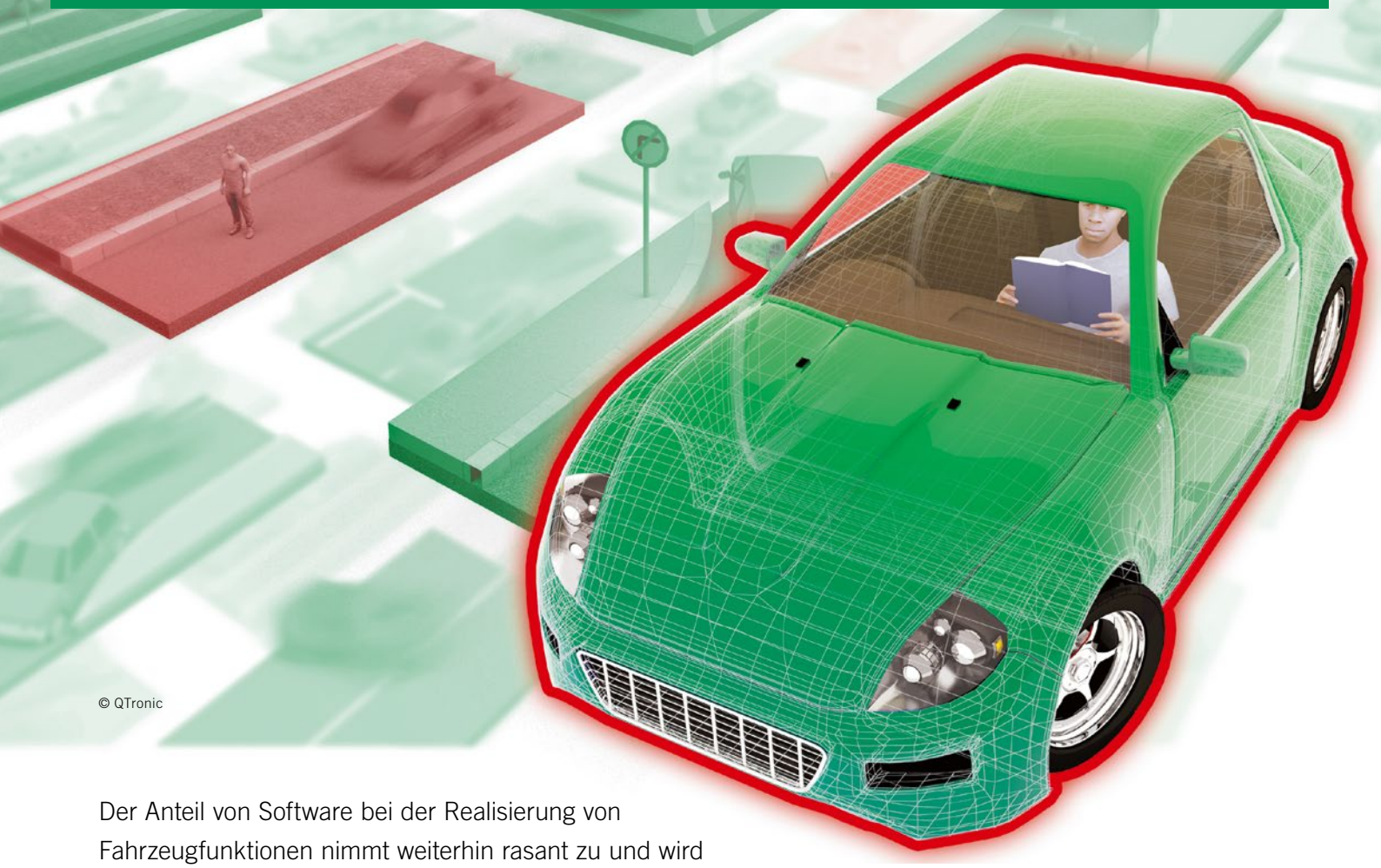
Test von Fahrerassistenzsystemen

Automatisierte Suche nach kritischen Fahrsituationen



© QTronic

Der Anteil von Software bei der Realisierung von Fahrzeugfunktionen nimmt weiterhin rasant zu und wird sich innerhalb der nächsten zehn Jahre noch einmal verdoppeln. Etablierte Test- und Validierungsprozesse halten mit dieser Entwicklung nur mühsam Schritt. In dieser Situation werden hochautomatisierte Testverfahren interessant, die sich in den letzten Jahren in Nischenanwendungen entwickelt haben. Das punktuelle Prüfen der Funktion mit handgeschriebenen Testskripten im Fahrversuch oder auf HiL-Prüfständen wird dabei ergänzt durch eine vollständig autonome Suche nach Schwachstellen und Fehlern auf PC. QTronic beschreibt ein solches Verfahren und gibt einen Überblick über erste Anwendungen im Bereich Fahrerassistenz und automatisiertes Fahren.

AUTOR

Dr. Mugur Tatar
ist Geschäftsführer und Leiter Testsysteme bei der QTronic GmbH in Berlin.

VIRTUELLER FAHRVERSUCH

Der systematische Test von Funktionen für Fahrerassistenz und automatisiertes Fahren erfordert die Identifikation und Analyse einer großen Zahl von Fahrsituationen: Die Situationen variieren bezüglich Straßenverlauf, Fahrzeugvariante, Richtung und Stärke des Windes, Verhalten des Fahrers und anderer Verkehrsteilnehmer, spontan auftretenden Bauteilfehlern, der zeitliche Abfolge von Ereignissen und vielem mehr. Dieser mehrdimensionale Raum kann im realen Fahrversuch nur sehr unzureichend abgedeckt werden. Nach Einschätzung vieler Entwickler muss daher der reale Fahrversuch künftig verstärkt durch virtuelle Fahrversuche ergänzt werden, zum Beispiel auf Basis eingeführter Werkzeuge für die Fahrzeug- und Verkehrssimulation, wie CarMaker (IPG), PreScan (Tass International) oder VTD (Vires). Nur so lässt sich die Vielzahl der zu betrachtenden Fahrsituationen in praktikabler Weise darstellen und analysieren, **BILD 1**. Ein weiterer Baustein für die Virtualisierung des Fahrversuchs sind virtuelle Steuergeräte. Ein virtuelles Steuergerät („virtual ECU" oder „vECU") ist ein auf PC ausführbares Modell des realen Steuergeräts. Eine vECU führt genau dieselbe Anwendungssoftware aus, wie das reale Steuergerät – falls nötig auch Teile der Basissoftware – und ermöglicht daher die Entdeckung von Softwareproblemen ohne den Einsatz der realen Steuergerätehardware.

Eine vECU liegt idealerweise in einem Format vor, das die einfache Integration mit den Fahrzeugmodellen des eingesetzten Verkehrssimulators ermöglicht. Ein Beispiel für ein eingeführtes Werkzeug zur Erstellung und Ausführung von virtuellen Steuergeräten ist Silver (QTronic). Silver [4, 5] erzeugt vECUs wahlweise als „Simulink SFunktion" oder als FMU (Functional Mock-up Unit) [6]. Eine Silver vECU kann zum Beispiel über die FMU-Schnittstelle von CarMaker in ein Fahrzeugmodell integriert werden, **BILD 2**. Dabei sind auch Busse (CAN, Lin, Flexray) mit modelliert, sodass sich bei Bedarf auch ein ganzes Netzwerk kommunizierender vECUs in ein virtuelles Fahrzeug integrieren lässt.

AUTOMATISIERTE SUCHE NACH KRITISCHEN SITUATIONEN

Eine Schlüsselidee zur Automatisierung der Suche nach Schwachstellen und Fehlern in Fahrzeugsystemen ist folgende Analogie: Testen ähnelt dem Schachspielen. Ein Schachspieler versucht, durch eine Folge von Spielzügen den Gegner in eine Matt-Situation zu treiben. Analog versucht ein Tester, das zu testende System durch eine Folge von „Inputs" in einen Zustand zu manövrieren, in dem es versagt, also beispielsweise seine Spezifikation verletzt oder sich zumindest bezüglich gegebener Qualitätsindikatoren schlecht verhält. Schachcomputer sind bekanntlich sehr spielstark und schlagen heute praktisch jeden menschlichen Spieler. Die Analogie „Test = Schach" weist einen Weg, die enorme Spielstärke von Schachcomputern auf das Testproblem zu übertragen, um so einen sehr „spielstarken" autonomen Testautomaten zu konstruieren, der jeden menschlichen Tester hinsichtlich des Vermögens, Systemfehler aufzuspüren, schlägt, **BILD 3**.

Die Firma QTronic hat seit 2006 mit TestWeaver eine Testautomatisierungslösung entwickelt, die genau diesen Ansatz implementiert. Für das Aufsetzen eines autonomen Testprozesses mit TestWeaver wird vor allem ein ausführbares Modell des zu testenden Systems benötigt. TestWeaver betrachtet dieses Modell als „Black box", kann das Modell also nur ausführen, aber nicht den Quelltext des Modells einsehen und analysieren, zum Beispiel mittels statischer Code-Analyse. Die Annahme dahinter ist, dass die Quellen des Modells in vielen Anwendungsfällen nicht oder nicht vollständig vorliegen. TestWeaver kennt jedoch die Inputs des Modells und ausgewählte beobachtbare „Outputs". Einige der Outputs fungieren dabei als Qualitätsindikatoren („Alarm"), **BILD 3**. Während eines Testlaufs startet TestWeaver das ausführbare Modell vom stets selben initialen Zustand aus viele Male und treibt die Simulation mit unterschiedlichen Inputfolgen bis zu einem konfigurierbaren Zeithorizont.

TestWeaver verfolgt dabei das Ziel, innerhalb dieses Zeithorizonts Fahrszenarien zu entdecken, bei denen die gegebenen Qualitätsindikatoren möglichst schlecht werden. Dabei versucht TestWeaver zugleich, die Testabdeckung im Zustandsraum zu maximieren. Testfälle werden so generiert, dass sie möglichst unterschiedliche Regionen des Zustandsraums abdecken. Der Zustandsraum enthält allerdings wegen des Auftretens reelwertiger Achsen (wie Fahrzeuggeschwindigkeit) in der Regel unendlich viele Zustände. TestWeaver verwendet hier folgenden Kunstgriff: Beim Konfigurieren eines TestWeaver-Experiments werden alle Inputs und Outputs diskretisiert, das heißt in endlich viele Intervalle partitioniert. Dadurch enthält der Zustandsraum insgesamt nur endlich viele diskrete Zustände. Diese diskrete Sicht auf den Zustandsraum (endlich viele Quadranten, siehe „Zustandsraum" in **BILD 3**) ist die Basis für den TestWeaver-Ansatz zur Messung und Optimierung der Testabdeckung. Ein Grund für die Spielstärke von TestWeaver ist seine Reaktivität. TestWeaver generiert Fahrszenarien nicht im Voraus und führt sie erst dann aus, sondern verzahnt die Generierung und Ausführung von Szenarien zeitlich. Dadurch hängt die Pla-

BILD 1 Werkzeuge für den virtuellen Fahrversuch (© QTronic)

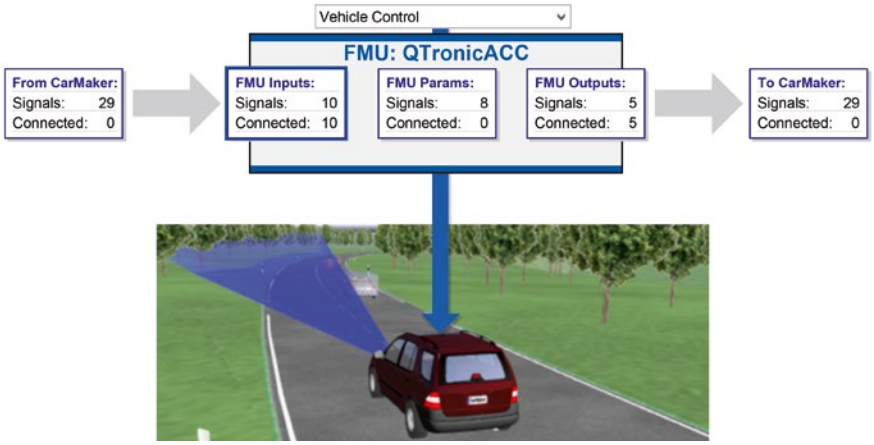


BILD 2 Integration eines virtuellen Steuergeräts für ACC, hier als Silver FMU in CarMaker (© QTronic)

nung des jeweils nächsten Fahrmanövers von den Ergebnissen aller vorangegangenen Fahrmanöver ab. Dies ist der Schlüssel für für die Fähigkeit von TestWeaver, lokale „Worst case"-Szenarien iterativ zu konstruieren. Solche Szenarien sind für Entwickler interessant, weil sie das Versagen einer Funktion besonders klar vor Augen führen.

Für den Test von Fahrerassistenzsystemen mit TestWeaver hat sich folgendes Vorgehen bewährt, **BILD 3**: Man entwickle ein parametriertes Szenario, das beispielsweise als Skript kodiert ist und den zeitlichen Ablauf einer betrachteten Verkehrssituation festlegt. Die Parameter dienen als Inputs für TestWeaver und legen Zeitpunkte für bestimmte Ereignisse, Geschwindigkeiten von Verkehrsteilnehmern und dergleichen fest. Die Spezifikation des zu testenden Systems wird durch Outputs kodiert, die die Qualität der zu testenden Funktion messen. TestWeaver variiert während eines Experiments die Input-Parameter mit dem Ziel, die durch die Outputs definierte Funktionsqualität zu minimieren. Ein typisches TestWeaver-Experiment läuft auf einem oder mehreren PCs für höchstens einige Tage.

TestWeaver berichtet während des Experiments kontinuierlich bereits gefundene Probleme sowie die erzielte Testabdeckung. Die Simulation läuft dabei unabhängig von der Realzeit, also zum Beispiel in mehrfacher Echtzeit, um eine möglichst hohe Testabdeckung zu erzielen. Die so gefundenen proble-

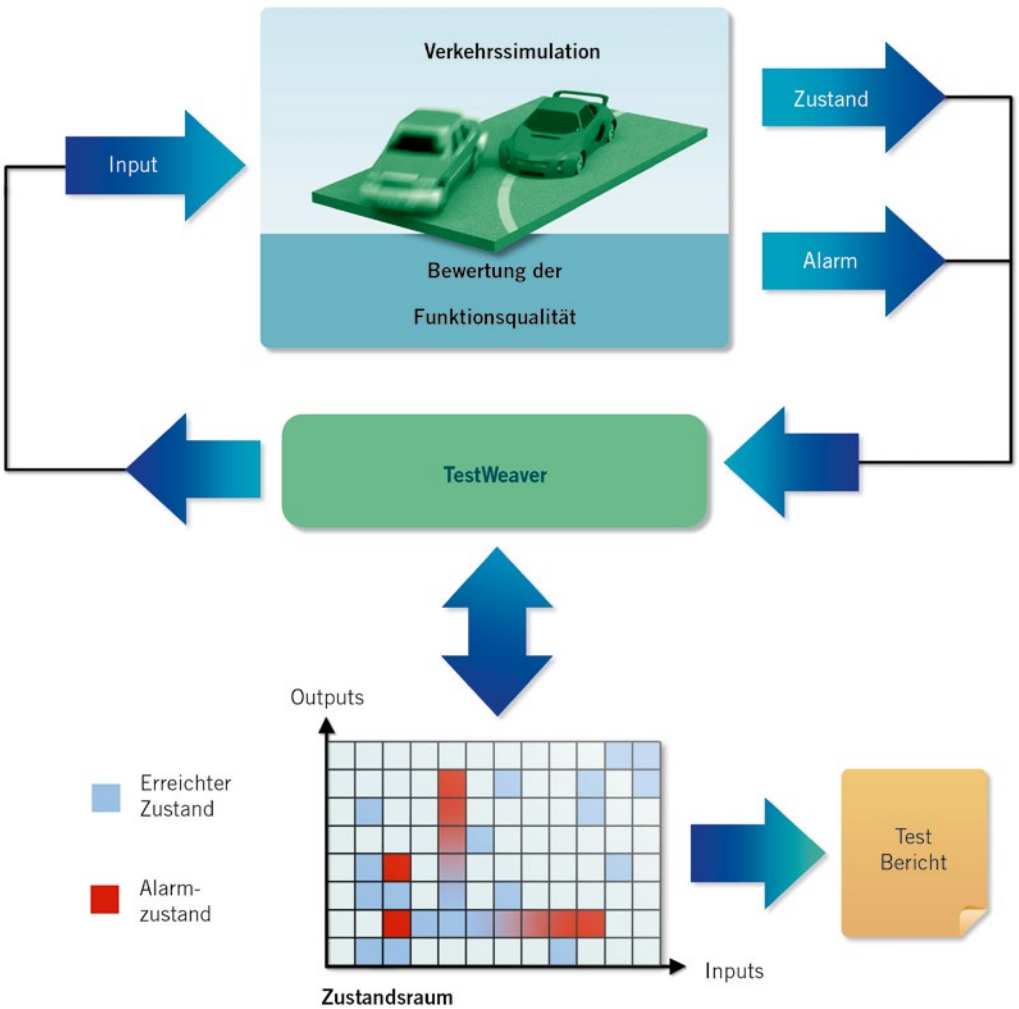


BILD 3 Automatisierte Fehlersuche mit TestWeaver (© QTronic)

matischen Szenarien können anschließend per Simulation so oft wie nötig wiederholt werden. Dies liefert den Funktionsentwicklern Hinweise für die gezielte Optimierung der zu entwickelnden Funktion.

INTEGRATION IN DEN ENTWICKLUNGSPROZESS

Die Umstellung des Entwicklungsprozesses auf ISO 26262 ist fast überall in der Automobilindustrie in vollem Gange. Rückverfolgbarkeit von Anforderungen und anforderungsbasiertes Testen sind allgemein akzeptierter Stand der Technik. Die TestWeaver-Methodik unterstützt anforderungsbasiertes Testen wie folgt: Anforderungen werden als Beobachter des Systemverhaltens („Bewertung der Funktionsqualität“ in **BILD 3**) implementiert. Diese Beobachter berechnen Alarmsignale, die die Verletzung von Anforderungen anzeigen. ISO 26262 empfiehlt für die Absicherung sicherheitskritischer Funktionen zusätzliche Prüfungen, zum Beispiel Erreichen bestimmter Codeabdeckungsziele beim Testen, Fehlerinjektion, Nachweis der Abwesenheit unerwünschten Verhaltens etc. TestWeaver wurde entwickelt, um bei derartigen Aufgaben eine hohe Testabdeckung möglichst automatisch zu erreichen, um so die Sicherheit des Entwicklungsprozesses zu erhöhen. Autonomes Testen ist damit in vielen Anwendungsfällen eine nützliche Ergänzung des klassischen Tests mit handgeschriebenen Skripten.

ANWENDUNGEN

Das TestWeaver-Verfahren wurde ursprünglich für den systematischen Test von Steuerungen für Automatikgetriebe entwickelt. Das Verfahren setzen AMG und Mercedes-Benz heute beispielsweise in Verbindung mit Silver für den automatisierten Test von Getriebesteuerungen ein [1]. Seit 2009 wird dieses Verfahren vermehrt auch im Bereich ADAS eingesetzt. [2] berichtet zum Beispiel darüber, wie TestWeaver in Verbindung mit dem Daimler-eignen Simulationswerkzeug CASCaDE eingesetzt wurde, um eine neu zu entwickelnde Seitenwindassistenzfunktion in Tausenden unterschiedlichen Straßen- und Windsituationen zu testen und zu optimieren. [3] beschreibt Anwendungen im Kontext ESP, zum Beispiel für die Anhängerstabilisierung. TestWeaver bietet Schnittstellen zu CarMaker (IPG), PreScan (Tass International) oder VTD (Vires). TestWeaver-Anwendungen auf Basis dieser Werkzeuge befinden sich zurzeit in der Entwicklung.

FAZIT

Virtuelle Fahrversuche sind heute ein nützliches Instrument für die Entwicklung von Funktionen für ADAS und automatisiertes Fahren. Sie bieten beste Voraussetzungen, um Test- und Validierungsprozesse noch weiter zu automatisieren. Das ist dringend nötig, weil der eingeführte skriptbasierte Testprozess mit der rasch wachsenden Zahl der zu testenden Softwarefunktionen kaum Schritt halten kann. Ein Beispiel für ein solches hochautomatisiertes Testverfahren ist TestWeaver, ein Werkzeug, das per virtuellem Fahrversuch selbstständig nach kritischen Fahrsituationen sucht.

LITERATURHINWEISE

[1] Gloss, S.; Slezák, M.; Patzer, A.: Systematic Validation of over 200 Transmission Variants“. In: ATZelektronik 8 (2013), Nr 4

[2] 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

[3] 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, 12th International Symposium on Advanced Vehicle Control (AVEC2014), Tokyo, Japan, 2014

[4] Junghanns, A; Serway, R.; Liebezeit, T.; Bonin, M.: Building Virtual ECUs Quickly and Economically. In: ATZelektronik 7 (2012), Nr. 3

[5] Mauss, J.: Virtuelle Steuergeräte für die Antriebsentwicklung. Tagung VPC – Simulation und Test 2015, Hanau bei Frankfurt am Main.

[6] Functional Mock-up Interface for Model Exchange and Co-Simulation, https://www.fmi-standard.org/