DIAGNOSE – VON DER ENTWICKLUNG BIS ZUM SERVICE

Fahrzeugdiagnose – vom lästigen Übel zum gewollten Muss (Teil 1)

Die Fahrzeugdiagnose befindet sich bereits seit einigen Jahren in einem rapiden Umbruch. Die Anzahl der Steuergeräte im Fahrzeug steigt nach einer kurzen Phase der Konsolidierung wieder stetig an und bewegt sich nun mit großen Schritten auf 100 ECUs im Fahrzeug zu. Parallel dazu steigt auch die Komplexität sowohl der Funktionen in einer ECU als auch die des Fahrzeugnetzwerks selbst erheblich. Die Diagnose stellt damit eine eigene Klasse von Anforderungen an das Gesamtfahrzeug dar, die parallel zu den eigentlichen Funktionen implementiert werden muss.

In Mittelklassefahrzeugen findet man heute bereits vier Bussysteme, in der Oberklasse kommen noch mal zwei bis drei weitere hinzu, die alle untereinander vernetzt werden müssen. Für die Diagnose ergeben sich daraus mehrere Konsequenzen: Eine reine Beschränkung der Diagnose auf die vom Gesetzgeber vorgeschriebenen Umfänge bezüglich der Einhaltung von Abgasstandards genügt seit Langem nicht mehr. Eine Reparatur von Fahrzeugen ist in einer Werkstatt heute ohne Diagnose nicht mehr möglich. Gleiches gilt für die Fahrzeugproduktion. Und um die Komplexität beherrschen zu können, muss die Diagnose sehr früh im Entwicklungsprozess integriert und als begleitender Prozess durch die gesamte Entwicklung mitgezogen werden – beim Fahrzeughersteller, aber auch unter Einbeziehung der beteiligten Steuergerätelieferanten.

Damit das Zusammenspiel aller Beteiligten funktioniert, sind Standards unabdingbar. Sie stellen das Ineinandergreifen von Daten und Tools an den Schnittstellen von Steuergeräteentwicklung, Prüffeld, Produktion und der Serviceorganisation sicher und ermöglichen dadurch die Konzentration auf die eigentliche Aufgabe. Im Folgenden werden die zugrundeliegenden Standards und die Absicherung der Diagnose im Entwicklungsprozess ausführlich beleuchtet.

Entlang der Wertschöpfungskette des Fahrzeugs spielt die Diagnose heute in allen Bereichen von der Entwicklung über die Produktion bis in den Service eine Rolle. Entscheidend dabei ist, dass die Mechanismen der Diagnose – die Kommunikation eines Testgeräts außerhalb des Fahrzeugs mit einem Steuergerät – heute für eine ganze Reihe von zusätzlichen Aufgaben verwendet werden (**Bild 1**).

Entwicklungsbereich

In der Steuergeräteentwicklung wird die Diagnose für die nachfolgenden Bereiche zur Verfügung gestellt. Die Basis ist die Diagnosekommunikation, bei neuen Steuergeräten in der Regel auf Basis des Standards UDS. Die klassische Diagnose als Funktion im Steuergerät dient der Erkennung von unerwartetem Verhalten innerhalb des Steuergeräts und in seiner Umgebung, einer Bewertung und dem Eintragen dieser Auffälligkeiten im Fehlerspeicher. Von dort können die Einträge dann durch ein externes Testsystem ausgelesen und weiterverarbeitet werden. Neben den klassischen Fehlerspeicherfunktionen – dazu gehören ebenso die Ermittlung von Randbedingungen für den Feh-

lereintrag („Umgebungsbedingungen“) und das Löschen des Fehlerspeichers – kommt dieselbe Basisfunktionalität heute auch für verwandte Anwendungsfälle zum Einsatz. Beispiele sind:

- Flash-Programmierung: Austausch von Code oder Daten im Steuergerät.
- Varianten-Codierung: Anpassung des Verhaltens eines Steuergeräts entsprechend gesetzlicher Vorschriften (z. B. Tagfahrlicht in Skandinavien) oder der kommerziellen Randbedingungen („SW als Produkt“).
- Routinen starten: Funktionen des Steuergerätes werden von außen angestoßen und laufen dann selbstständig ab, beispielsweise ein Selbsttest.
- Messen: Auslesen der durch das Steuergerät mithilfe von Sensoren ermittelten Größen.

Die Diagnosekommunikation wird also immer dann eingesetzt, wenn ein präziser Daten- und Freigabeprozess nötig ist (Flash-Programmierung, Varianten-Codierung) oder sie einen einfachen und preisgünstigen Zugang zur Blackbox Steuergerät erlaubt.

Im Entwicklungsbereich kommt die Diagnose in zahlreichen Anwendungen zum Einsatz, sei es bei der Kommunikationsentwicklung, der Entwicklung von Steuergerätefunktionen, der Entwicklung von Flash- und Variantencodierfunktionen, im HiL, in Systemprüfplätzen oder bei der Steuergeräteintegration. Anschließend wird auch im Fahrzeugtest vielfältig darauf zurückgegriffen.

Produktion

Gerade der einfache Zugang zum Steuergerät ermöglicht in der Produktion eine Vielzahl von Anwendungen, die sonst nur mit komplexen Lösungen kostenintensiv zu realisieren wären. Die heutigen mechatronischen Systeme können ohne Zugang zur Elektronik kaum einer Prüfung unterzogen werden, sollten andererseits aber so früh wie möglich getestet werden. Auch die Variantencodierung des Fahrzeugs, die Schlüsselprogrammierung und die Initialisierung der Wegfahrsperre werden hier durchgeführt. Schließlich erfolgt im „End-of-line“-Test eine finale Kontrolle der Fahrzeugfunktionen und das Löschen des Fehlerspeichers.

Eine besondere Herausforderung stellt in der Produktion die Unterscheidung der „echten“ Fehler von den „systematischen“ Fehlern dar. Letztere ergeben sich aus der Tatsache, dass die Steuergeräte zahlreiche Fehlereinträge führen, die sich aus der zum Verbauzeitpunkt fehlenden Systemumgebung ergeben. Beispielsweise werden Fehler eingetragen, wenn auf dem CAN-Bus ein Signal nicht verfügbar ist, das von einem noch nicht verbauten Steuergerät gesendet werden sollte. Der Eintrag ist selbstverständlich richtig, er verweist aber eben nicht auf ein fehlerhaftes (Teil-)System.

Service

Die Ursprünge der Diagnose finden sich im Service, genauer in der nüchternen Erkenntnis, dass ein Fahrzeug auf der elektronischen Basis von aktuell 30 bis 80 Steuergeräten, bei der eine Vielzahl der Fahrzeugfunktionen in Software realisiert sind, in einer Werkstatt nicht mehr repariert wer-

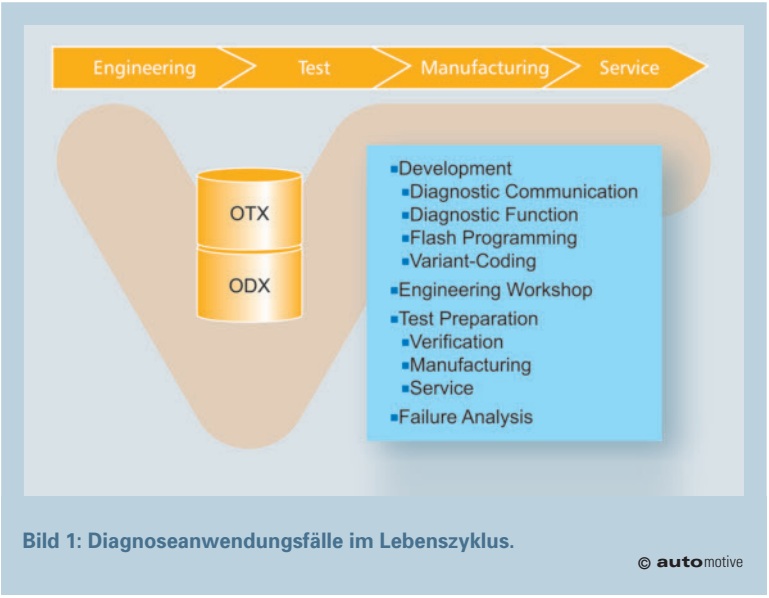

Bild 1: Diagnoseanwendungsfälle im Lebenszyklus.

den kann. Dem Fachmann in der Werkstatt muss für den Fall, dass der Kunde unerwartete Symptome meldet, der passende „Schraubenschlüssel" zur Hand gegeben werden. Das entsprechende Werkzeug ist der Service-Tester. Es verbindet die gemeldeten Symptome mit Fehlereinträgen in den Steuergeräten und Messwerten, die parallel ermittelt werden können. Expertensysteme kombiniert mit der Erfahrung des Fachmanns in der Werkstatt führen dann in aller Regel zu einer Reparaturempfehlung.

Zusätzlich ist es in der Regel möglich, Steuergeräte mit neuen Software-Ständen zu programmieren.

In allen genannten Anwendungsfällen kommen heute ODX-Daten zum Einsatz. Außer in Testsystemen werden diese auch für die Parametrierung von Steuergeräten und für die automatische Erzeugung von Testprozeduren verwendet. Es liegt auf der Hand, dass die Qualität der ODX-Daten – alleine, aber auch in Verbindung mit dem Steuergerät – für die Gesamtqualität eine entscheidende Einflussgröße darstellt. Neben der Durchführung von intensiven Reviews wird dies durch eine aufwendige Datenverifikation erreicht. Die Datenverifikation ist ein wichtiger Anwendungsfall eines modernen Entwicklungstesters. Darüber hinaus kommt dieser im ganzen Entwicklungsprozess zum Einsatz: als Testumgebung für die wichtigen Steuergerätefunktionen Diagnosekommunikation, Diagnosefunktion, Flash-Programmierung und Variantencodierung. Auch als preiswerter „Debugger" wird der Entwicklungstester immer wieder eingesetzt. In späteren Phasen spielt er eine große Rolle bei der Fahrzeugüberprüfung in der Versuchswerkstatt und teilweise sogar im Fahrversuch beim Test einzelner Fahrzeugfunktionen und in der Fehleranalyse, speziell bei Problemen in der Kommunikation. Und auch bei der Prüfvorbereitung spielt der Entwicklungstester als Inbetriebnahmewerkzeug eine große Rolle. Die Kombination aus ODX-Daten und Steuergerät in Verbindung mit dem D-Server stellt für den Testingenieur im Wesentlichen eine Blackbox dar, die für ihn nur schwer in Betrieb zu nehmen ist. Mit dem Entwicklungstester kann hier sehr einfach die Kommunikation in Betrieb genommen werden und das korrekte Verhalten des Testablaufs simuliert werden.

OBD

Die On-board-Diagnose stellt einen besonderen Bereich der Diagnose dar, da sie vom Gesetzgeber vorgeschrieben ist. Er verlangt von den Fahrzeugherstellern – ursprünglich nur von den Pkw-Herstellern, inzwischen aber auch von Lkw-Herstellern –, dass alle abgasrelevanten Systeme kontinuierlich einem Selbsttest unterzogen werden. Im Falle einer Auffälligkeit muss dies dem Fahrer durch eine Warnleuchte unverzüglich mitgeteilt werden, damit er das Problem beheben lassen kann. Für den Fall Kalifornien ist dieses „kann" insofern eine Verniedlichung, da der Fahrer empfindliche Strafen zahlen muss, wenn er das Abgassystem nicht reparieren lässt. Dieser Selbsttest wird durch die Forderung unterstützt, dass jederzeit mit einem einfachen Diagnosetester, dem sogenannten Scantool, die Erkenntnisse der On-board-Diagnose ausgelesen werden können. Sowohl eine funktionierende On-board-Diagnose als auch eine funktionierende Kommunikation mit den Scantools stellen für die Hersteller eine Marktzugangsvoraussetzung dar. Entsprechend genau müssen die Tests dieser Funktionalitäten sein. Diese sind mit der Fahrzeugzulassung auch nicht beendet, da eine Häufung von Meldungen bei bereits verkauften Fahrzeugen schnell auch zu Fahrzeugrückrufen führen kann.

Anforderungen an Diagnosesysteme

Wenn man sich moderne Testsysteme im Automotive-Umfeld anschaut, erkennt man generelle Blöcke, die immer wiederkehren. Zunächst ist entscheidend, dass man in den meisten Fällen ein Erstellsystem und eine Laufzeitumgebung vorfindet. Das Erstellsystem dient der Konfiguration des Laufzeitsystems, in ihm konfiguriert der Testingenieur Testabläufe und die Visualisierung für die spätere Verwendung. In vielen Fällen wird auch die Dokumentation des Tests bereits hier definiert. Das Erstellsystem arbeitet oft in einem Administratormodus. In der Laufzeitumgebung sind die Eingriffsmöglichkeiten des Testers häufig stark eingeschränkt. Der Tester hat nur noch den Anwendungsmodus im Zugriff. Das Ziel ist, dass er sich nur noch auf die Testumgebung konzentriert und das Tool mit wenigen Handgriffen funktioniert.

Die Testsysteme haben für die Konfiguration und Testdurchführung Eingangsdaten, die Konfigurationen und Randbedingungen parametrieren. Standardisierte Eingangsgrößen sind die heute geforderten ODX-Daten und zunehmend auch OTX-Daten (**Bild 2**). Daneben kommen häufig proprietäre Daten zum Einsatz, zum Beispiel Logistikinformationen wie Freigabestände und Verbaulisten. Diese Größen sind stark prozessabhängig, teilweise auch infrastrukturabhängig und können schlecht standardisiert werden.

Daneben werden auch entsprechende Ausgangsdaten erzeugt. Diese können ebenfalls standardisiert oder proprietär sein. In der Diagnose gibt es heute, anders als in der Messtechnik, (noch) kein standardisiertes Ausgabeformat. Ein typisches Ausgabeformat ist die Testdokumentation, in

der erfolgreiche und fehlgeschlagene Testfälle aufgezeichnet sind. Diese können entweder durch Spezialisten analysiert oder später für Regressionstests wieder ins System zurückgespielt werden. Auch Fehlerstatistiken werden vielfach mit den Testergebnissen erzeugt.

Anforderungen an Entwicklungstester

Speziell an Entwicklungstester, die ja im gesamten Prozess zum Einsatz kommen, müssen noch eine Reihe zusätzliche Anforderungen gestellt werden. Dies liegt in erster Linie an den Einsatzfällen, aber auch an den zugrundeliegenden Verarbeitungsschichten. Insbesondere werden ODX-Daten durch einen sogenannten D-Server verarbeitet. Dieser stellt die Daten aufbereitet zur Verfügung. Es ist aber für das Testsystem kaum möglich festzustellen, welchem Zweck die Daten dienen. D. h., Diagnosedienste für die Kommunikationskontrolle unterscheiden sich in der Praxis an der Schnittstelle meist nicht von Diagnosediensten zum Messen oder zum Parametrieren eines Steuergerätes. Der Diagnosetester muss den Anwender aber durch anwendungsspezifische Darstellungen unterstützen, die entsprechend konfiguriert werden müssen. Erst dadurch wird die Darstellung mit den passenden Diagnosediensten verbunden.

Die Arbeit am Fehlerspeicher stellt vielleicht die wichtigste Diagnosefunktion dar. Er muss gelesen, aber auch gelöscht werden, um den korrekten Wiedereintrag eines Fehlers prüfen zu können. Je nach Anwender interessieren die Fehler des ganzen Fahrzeugs oder eines Steuergerätes, nur die Liste der Fehler oder die Fehler mit den abgespeicherten Umgebungsbedingungen und den Statusinformationen. Gerade die Statusinformationen sind für eine tiefer gehende Analyse von Bedeutung, da sie den Fehler näher beschreiben. Man erfährt beispielsweise, ob ein Fehler dauerhaft eingetragen wurde oder nur temporär, ob er sporadisch auftritt oder dauerhaft. Eine weitere wichtige Information ist das Readiness-Flag. Es gibt an, ob die zur Erkennung des Fehlers im Steuergerät implementierte Routine bereits vollständig durchlaufen wurde. Für eine ganze Reihe von Fehlern müssen dazu Vorbedingungen eingehalten werden, wie „Motortemperatur muss Mindestwert erreicht haben“, „vorgegebene Drehzahl muss Wert überschritten haben“, usw. Die Anforderungen an die Visualisierung der Flash-Programmierung sind gänzlich anderer Natur. Diese beschränkt sich im Wesentlichen auf die Auswahl der Programmierdaten (z. B. Programmcode, Kennfelder,…) einen Start-Button und möglichst eine Fortschrittsanzeige, da bei Datengrößen von mehreren MByte durchaus nennenswerte Programmierzeiten entstehen. Die eigentliche Herausforderung steckt bei der Flash-Programmierung im Ablauf. Hier wird zunächst eine Authentifizierung durchgeführt, in die Programmiersession geschaltet, die restlichen Busteilnehmer aufgefordert, im Folgenden keine Busfehler einzutragen und anschließend die Buskommunikation zwischen den Steuergeräten beendet, um ausreichend Bandbreite zur Verfügung zu haben. Danach

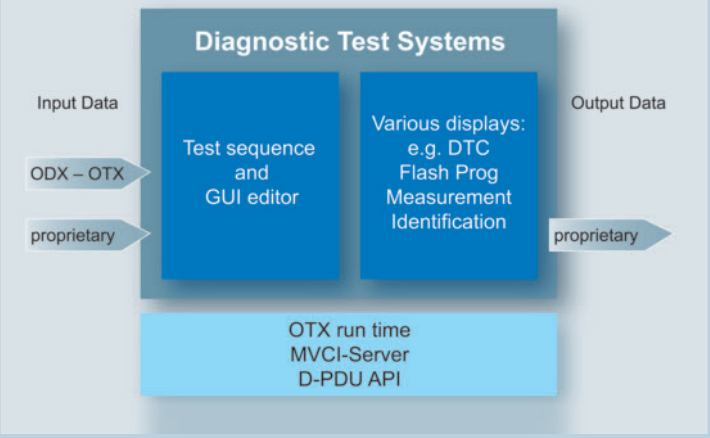

Bild 2: Generelle Anforderungen an Testsysteme.

beginnt die möglichst performante Übertragung der Daten. Am Ende werden Checksummen überprüft, das programmierte Steuergerät reinitialisiert und die Buskommunikation wieder freigeschaltet. Abschließend sind manchmal noch Systemanpassungen durchzuführen.

Beim Messen muss hauptsächlich unterschieden werden zwischen graphischer Darstellung der einzelnen Messwerte und textueller Darstellung. Bei der graphischen Darstellung sind die Grenzen im Wesentlichen durch die Bedienbarkeit gesetzt, von der Art der Darstellung (Zeigerinstrument, Balkendiagramm, x-t-Darstellung, …) über die verwendeten Farben bis zum Darstellungsbereich (Wertebereich, Gültigkeitsbereich mit Farbumschlag, …) ist hier fast alles möglich. Daneben müssen häufig Messwertblöcke dargestellt werden. Diese werden auf einmal vom Steuergerät ausgelesen, um bei zusammenhängenden Größen keine Latenzen durch die Busübertragung in die Messung einfließen zu lassen.

Bei der Ausführung von Steuergeräteroutinen ist wiederum ein sehr breites Anwendungsspektrum abzudecken. Das eine Ende wird durch Steuergeräteroutinen gebildet, bei denen das Feedback allein von Aktuatoren gegeben wird. Beispiele sind der Zeigertest am Kombiinstrument oder Ventile am ABS-Steuergerät, die öffnen und schließen. Für diese Art von Routinen muss der Tester eigentlich nur einen Start-Button anbieten sowie vielleicht ein Beschreibungsfeld für den Test. Das entgegengesetzte Ende stellen Selbsttests dar, die als Ergebnis eine Ergebnisliste über den Bus zurückliefern. Hier muss der Tester die relevanten Teile der Steuergeräteantwort auf dem Bildschirm darstellen. Ähnlich funktioniert auch die wichtige Darstellung der Fahrzeug- oder Steuergeräteidentifikation, die in der Regel aus einer Liste von Informationen wie der VIN (VehicleInformation), HW-Stand, SW-Stand, usw. gebildet wird.

Der Einsatzfall Datenverifikation erfordert einen erheblich weiteren Darstellungsraum. Der Zugriff sowohl auf die Parametrierung von Diagnosediensten als auch auf die Ergebnisse muss so vollständig sein, dass alle Methodiken, die in automatischen Tests angewendet werden, mit dem Entwicklungstester auch manuell überprüft werden können. Im Entwicklungstester müssen auch Werte einstellbar sein, die eigentlich nicht erlaubt sind, um das korrekte Verhalten an den Grenzen verifizieren zu können. Bei der Überprüfung ist besonders wichtig, dass nicht nur die Werte analysiert werden können, sondern auch die Strukturierung der Ergebnisse. Diese ist für den Zugriff durch eine Testautomatisierung eine Fehlerquelle, die nicht unterschätzt werden darf. Es sind übrigens nicht alle Tests automatisierbar, da zwar die Plausibilität automatisch geprüft werden kann, nicht aber die richtige Umsetzung von hexadezimalen (Bus-)Daten in symbolische Daten. Dies ist eine Aufgabe für Spezialisten, die massiven Einfluss auf die Gesamtqualität des Systems hat.

Variantencodierung

Mit Hilfe der Variantencdierung werden Funktionen im Steuergerät ein- und ausgeschaltet. Dies hat erhebliche Kostenvorteile, da nicht für jede Variante der Steuergerätecode einzeln frei getestet werden muss. Die Funktion Variantencodierung muss dafür umso genauer getestet werden. Dies ist mit hohem Aufwand verbunden, da die einzelnen Codierbits nicht unabhängig sind. Dies gilt innerhalb eines Steuergeräts, umso mehr aber auch für das Gesamtfahrzeug. Offensichtlich wird dies an der Kombination von Motor und Getriebe. Die Motorleistung wird heute in der Regel für einen Motor codiert, muss aber mit bestimmten Getrieben kombiniert werden, da das eine Getriebe für ein Drehmoment nicht spezifiziert ist, anderseits bei hohen Leistungen häufig nur noch ein Automatikgetriebe angeboten wird. Der Diagnosetester muss für die Funktionsprüfung die Kombination verschiedener Codierungen möglichst einfach erlauben. Gleichzeitig sollte auch hier das Setzen fehlerhafter Kodierkombinationen möglich sein, um das Fehlerverhalten überprüfen zu können.

Restbussimulation

Eigentlich kein Thema der Diagnose ist die Restbussimulation. Dennoch hat sie einen großen Einfluss, da in vielen Fällen – insbesondere, wenn nur ein einzelnes Steuergerät oder ein Teilnetzwerk in Betrieb genommen werden sollen – die Diagnose nicht funktioniert. Oftmals ist die Diagnose in einem Steuergerät nicht aktiv, solange bestimmte Signale nicht auf dem Bus übertragen werden. Ein typisches Beispiel ist die Zündungserkennung, die von einem Steuergerät durchgeführt wird und dann als Signal allen anderen ECUs auf dem CAN-Bus zur Verfügung gestellt wird. Gleiches gilt für Signale, die für eine plausible Diagnose nötig sind (z. B. einem Geschwindigkeitssignal). In jedem Fall muss der Entwicklungstester diese Signale simulieren können, indem er einzelne CAN-Nachrichten auf den Bus zyklisch sendet. Sinnvollerweise sollten die einzelnen Signale auch änderbar sein, da sich beispielsweise das Diagnoseverhalten von niedrigen zu hohen Fahrzeuggeschwindigkeiten ändern kann.

Auch die Anforderung Analysefähigkeit erzeugt Anforderungen an den Entwicklungstester. Zum einen dient er der Datenvisualisierung. Er muss also die Daten geeignet auf-

bereiten und Messwerte auf passenden Instrumenten darstellen, den Fehlerspeicher auflisten und die Daten zur Flashprogrammierung so präsentieren, dass sie fehlerfrei ins Steuergerät programmiert werden können. Zum anderen dient er aber der Inbetriebnahme und soll zu diesem Zweck einen möglichst genauen Zugriff auf die ODX-Informationen und eventuell auftretende Kommunikationsfehler ermöglichen. Falls in der Kommunikation selbst Probleme auftreten, muss aber auch der Busverkehr so dargestellt werden, dass das jeweilige Problem effizient auffindbar und dokumentierbar ist.

Für die Prüfung der OBD-Funktion müssen die OBD-Modes, es handelt sich dabei um spezielle Diagnosedienste, in der jeweils adäquaten Form präsentiert werden. Grob gegliedert dienen sie zum Lesen der Messwerte, zur Fehlerspeicherbearbeitung und zur Auswertung von Testergebnissen. Die Adressierung erfolgt dabei immer auf die Funktion „OBD", es können also mehrere Steuergeräte betroffen sein. Das VCI muss dazu die funktionale Adressierung beherrschen, das Diagnosesystem mit mehreren Antworten von verschiedenen Steuergeräten umgehen können. Welche Teilfunktionen bei den einzelnen Modes unterstützt werden, wird bitcodiert von den Steuergeräten ans Testsystem gemeldet. Hier ist es besonders wichtig, dass der Tester erlaubt, Anfragen an die Steuergeräte zu stellen, die erst einmal falsch erscheinen, da die Scantools die Spezifikation teilweise unterschiedlich interpretieren.

Welche wichtigen Standards muss man beachten?

Heute in der Diagnose verwendete Standards können grundsätzlich in drei Kategorien eingeteilt werden: Protokolle, Datenbeschreibungen und Programmierschnittstellen. Die älteste Kategorie stellt sicher die der Protokolle dar. Eingeführt ursprünglich um die Forderung des Gesetzgebers nach einer Überwachungsmöglichkeit des Abgasverhaltens zu genügen, erfüllen diese Protokolle heute zahllose weitere Aufgaben. Die Vielzahl der herstellerspezifischen Protokolle wurde heute durch standardisierte Protokolle abgelöst, allerdings sollte die Bedeutung der Altprotokolle für Servicetester nicht unterschätzt werden, schließlich sind die Fahrzeuge noch viele Jahre auf der Straße aufzufinden.

Im zweiten Teil dieses Beitrags werden die verschiedenen Protokolle und Programmierschnittstellen aufgezeigt. Schließlich wird am Beispiel des DTS-Monaco von Softing Automotive ein Entwicklungstester vorgestellt, der D-PDU API, ODX, MCD-3D und OTX beherrscht.

Markus Steffelbauer leitet das Produktmanagement und verantwortet Entwicklung und Marketing der Hardware- und Softwareprodukte bei der Softing Automotive Electronics GmbH. Daneben vertritt Steffelbauer seit vielen Jahren Softing in verschiedenen Standardisierungsgremien, seit 2008 als Vertreter im ASAM TSC (Technical Steering Committee).

Softing Automotive Electronics GmbH
www.softing.com

DIAGNOSE – VON DER ENTWICKLUNG BIS ZUM SERVICE

Fahrzeugdiagnose – vom lästigen Übel zum gewollten Muss (Teil 2)

Die Fahrzeugdiagnose befindet sich bereits seit einigen Jahren in einem rapiden Umbruch. Die Diagnose stellt eine eigene Klasse von Anforderungen an das Gesamtfahrzeug dar, die parallel zu den eigentlichen Funktionen implementiert werden muss. Im zweiten Teil dieses Beitrags werden die verschiedenen Protokolle und Programmierschnittstellen aufgezeigt. Schließlich wird am Beispiel des DTS-Monaco von Softing Automotive ein Entwicklungstester vorgestellt, der D-PDU API, ODX, MCD-3D und OTX beherrscht.

Kommunikationsprotokoll UDS

Die heute gängigsten Protokolle sind in Europa ISO 14765 (KWP2000 auf CAN) und das darauf aufbauende ISO 14229 (UDS – Unified Diagnostic Services). Sie teilen sich ein gemeinsames Transportprotokoll und beschreiben im Wesentlichen auch die gleichen Klassen an Diagnosediensten (DiagnosticServices). Allerdings wurden für UDS, dem Namen entsprechend, die Dienste für unterschiedliche Anwendungsfälle und entsprechend der bei den verschiedenen Herstellern verwendeten Altprotokolle so verallgemeinert, dass eine Migration relativ einfach möglich ist. Im Wesentlichen werden folgende Dienstklassen beschrieben:

- Datenlesen
- Flashprogrammierung
- Fehlerspeicher
- Steuergeräteroutinen
- IO Control
- Kontrollfunktionen

Um darüber hinaus in nicht wettbewerbsrelevanten Bereichen Einsparungen zu ermöglichen, ist schon vor einigen Jahren im Rahmen des ASAM e.V. eine Art Diagnosebetriebssystem von OEMs und Toolherstellern gemeinsam entwickelt worden. Es handelt sich dabei um ein datengetriebenes System, die Daten beschreiben die Fähigkeiten des Systems im Zusammenspiel mit dem jeweiligen Steuergerät oder Fahrzeug. Die Notwendigkeit eines solchen Vorgehens ist einfach zu verstehen, wenn man sich vergegenwärtigt, dass ein Türsteuergerät gänzlich andere Funktionalitäten und Größen implementiert als etwa ein Motorsteuergerät.

Innerhalb des ASAM e.V. wurde eine Programmierschnittstelle (ASAM MCD-3D) zum symbolischen Zugriff auf Steuergeräte- und Fahrzeuginformation und die Datenbeschreibung (ASAM MCD-2D ODX – Open Diagnostic data eXchange) als Austauschformat zwischen den Beteiligten im Diagnoseprozess definiert. Beide wurden auch als ISO-Standards übernommen. Innerhalb der ISO wurden noch zwei

weitere Standards ergänzt: D-PDU API als Low-level-Programmierschnittstelle zum einfachen Austausch von Diagnoseinterfaces und OTX (Open Test sequence eXchange format). Letzteres ermöglicht den Austausch von Diagnoseabläufen, z.B. zwischen Entwicklung und Produktion.

D-PDU API – die Integrationsschicht für VCIs

Der Standard beschreibt eine Programmierschnittstelle auf hexadezimaler Ebene. Das Transportprotokoll wird dabei vollständig transparent behandelt, d. h., für die darüberliegende Anwendung spielt es keine Rolle, welches Protokoll im D-PDU API bearbeitet wird. Der Schnittstelle werden die Daten in Form eines Bytestreams zusammen mit den Adressierungsinformationen übergeben und das Protokoll parametriert (z. B. Timings). Die Ausführung entsprechend des eingestellten Protokolls erfolgt dann automatisch. Steuergeräteantworten werden der Anwendung anschließend ebenfalls als Bytestream mit den Adressinformationen der antwortenden Steuergeräte zurückgespiegelt. Weitergehende Fähigkeiten der Interfaces, wie Ein- und Ausgänge, die angesprochen werden können, können über D-PDU API ebenfalls bedient werden. Insgesamt wird die Beschreibung der Fähigkeiten der Interfaces, die je nach Produkt abweichen, in einer XML-Datei abgelegt. Dadurch ist ein Wechsel des Interfaces sehr einfach möglich, es kann beispielsweise ein Testprogramm in der Entwicklung mit einem USB-Interface vom Hersteller A betrieben werden und später im Prüfstand mit einem WLAN-Interface vom Hersteller B. In der Praxis werden – allerdings geringe – Anpassungen nötig sein.

SOFTING UND KVASER

Softing Automotive Electronics und Kvaser vertiefen ihre Partnerschaft. Die Partnerschaft erweitert die bereits bestehende technische Partnerschaft von Softing Automotive Electronics und Kvaser AB. Durch die Ergänzung mit Kvaser-CAN-Interfaces kann das bisherige Interface-Portfolio von Softing weiter abgerundet werden. Softing tritt somit als Vertriebspartner von Kvaser-Hardware am Markt auf und unterstützt die Hardware in ihren Anwendungen. Geplant ist auch ein neues Paket im Bereich der Diagnoselösung. Dieses soll mit DTS-Monaco (Software für Steuergeräte-Kommunikation, -Diagnose und Onboard Analyse) und dem Leaf Light HS von Kvaser kombiniert werden.

Peter Biermann, Geschäftsführer der Softing Automotive Electronics GmbH: „Mit dem Leaf Light Interface von Kvaser können wir dem Kunden eine preisgünstige Diagnose-Lösung anbieten, die noch dazu einfach zu bedienen ist und die Kundenanforderungen an Performance und Handhabbarkeit erfüllt."

ODX – das Austauschformat für Diagnosedaten

Der Standard beschreibt die Dateninhalte der Diagnose-Kommunikation für die im Fahrzeug verbauten Steuergeräte als XML-Datei. Er stellt damit in einer Quelle sowohl die Dokumentation als auch die Testsystemparametrierung zur Verfügung. Die Daten können somit von der Spezifikationsphase bis zur Nutzung in Produktion und Service gleichermaßen verwendet werden. Über die Datei kann eine Umsetzung von hexadezimalen in symbolische Werte erfolgen, d. h., für ein „Motorsteuergerät" existiert die Beschreibung eines Diagnosedienstes „DrehzahlLesen". Die entsprechenden Hex-Werte zum Senden über D-PDU API können aus der ODX-Datei ermittelt werden. Aus der Antwort kann der hexadezimale Wert ermittelt werden, der in „1900 U/min" umgerechnet wird. Neben diesen Kommunikationsinformationen werden in ODX auch die Protokoll-Parametrierung, Verbaulisten eines Fahrzeugs in verallgemeinerter Form, Flashprogrammierdaten und Daten für die Variantencodierung beschrieben. Hauptziele der Standardisierung war die Schaffung eines Standards, der maschinenlesbar und langzeitstabil ist (Verwendung von XML) und den Entwicklungsprozess unterstützt und (möglichst) redundanzfrei ist (Vererbungskonzept).

Das Vererbungskonzept beruht auf der Idee, dass für ein bestimmtes Steuergerät eine Vielzahl von Informationen beschrieben werden kann, die über den gesamten Lebenszyklus konstant bleibt. Diese Information wird in der sogenannten BaseVariant beschrieben. Die einzelnen Varianten eines Steuergerätes, beschrieben z. B. über den Identifikationsstring oder einen Softwarestand, unterscheiden sich nur geringfügig. Der Hauptteil der Informationen kann also für eine Variante aus der Basisvariante geerbt werden, in der ECUvariant muss dann nur noch das Delta beschrieben werden. Dieses kann dazugefügt werden oder überflüssige Information kann ausgeblendet werden. Darüber hinaus können Informationen, die in der Steuergerätevariante anders beschrieben sind als für die BaseVariant, wie in einer objektorientierten Programmiersprache überschrieben werden. Zusätzlich zu diesen beiden Ebenen BaseVariant und ECUvariant existieren noch zwei weitere Ebenen: Die Protokollebene beschreibt die durch das Diagnoseprotokoll vorgegebenen Dienste und Parametrierungen und dient damit als „Blaupause" für die davon abgeleiteten Steuergeräte. Die Ebene FunctionalGroup schließlich ermöglicht die Beschreibung der funktionalen Adressierung, bei der mehrere Steuergeräte einer logischen Gruppierung über eine gemeinsame Adresse angesprochen werden können. Das bekannteste Beispiel ist die OBD-Funktionalität, bei der alle abgasrelevanten Steuergeräte zusammengefasst werden und gemeinsam angefragt werden. Beim OEM werden in einer ODX-Datenbank in der Regel die Daten für eine Baureihe zusammengefasst. Die Gesamtdatenmenge kann dabei relativ groß werden, weil die Datenbank folgende Informationen enthält:

- die im Fahrzeug verwendeten Protokolle,
- die Information über die mit funktionaler Adressierung zugreifbaren Steuergeräte,
- die Gesamtheit der Fahrzeug verbaubaren Steuergeräte, also sowohl optional erhältliche Steuergeräte (Getriebe-

steuergerät für Automatikgetriebe) als auch alternative Steuergeräte (Motorsteuergerät für 4- und 6 Zylinder Diesel- und Benzinmotor),
- die Varianten über die Produktionszeit des Fahrzeugs.

Reale ODX-Datenbanken erreichen heute die 100-MByte-Grenze. Um den Datenaustausch zu vereinfachen, wurde zusätzlich zu ODX das PDX-Format standardisiert (packed ODX). Dabei handelt es sich um eine indizierte ZIP-Datei.

ASAM MCD-3D – das Diagnosebetriebssystem

Der Standard beschreibt eine Programmierschnittstelle zum Zugriff auf Steuergeräte mithilfe von ODX-Daten. Das entsprechende Laufzeitsystem wird in der Regel D-Server oder MVCI-Server genannt. Es kann im Diagnoseprozess von der Entwicklung über die Produktion bis in den Service in verschiedensten Anwendungen zum Einsatz kommen und stellt somit eine konsistente Verwendung der ODX-Daten sicher. Dies wird insbesondere dadurch möglich, dass zur Verwendung mit verschiedenen Programmiersprachen Referenzimplementierungen für COM/ DCOM, JAVA und C++ beschrieben und im Markt auch verfügbar sind.
Der Zugriff erfolgt auf symbolischer Ebene, d. h., es wird ein Steuergerät über seinen Namen „Motorsteuergerät" ausgewählt und anschließend können für dieses Steuergerät Diagnosedienste ausgeführt werden. Ein Beispiel ist der vorher beschriebene Dienst „DrehzahlLesen", der von der ODX-Ablaufmaschine im D-Server verarbeitet und über D-PDU API an das Steuergerät gesendet wird. Die Antwort wird ebenfalls über die ODX-Daten prozessiert und das Ergebnis an der API als „1900 U/min" zur Verfügung gestellt. Um insbesondere in der Produktion die Performance zu verbessern, können im Prinzip beliebig viele solcher Anfragen parallel abgearbeitet werden. Die Grenze ergibt sich im Wesentlichen aus der Buslast des (in der Regel) verwendeten CAN-Busses und der Fähigkeit des Gateway-Steuergerätes, die Anfragen auf die verschiedenen Busse hinter dem Gateway sinnvoll zu verteilen.
Die ODX-Daten werden in realen D-Servern heute mithilfe eines Laufzeitformats verarbeitet. Dies hat zum einen Performance-Gründe – in einem Binärformat können die Daten gepackt und für den Zugriff optimiert abgelegt werden – zum anderen Sicherheitsgründe. Gerade im Service liegen die Daten im Prinzip ungeschützt auf den Service-Testern und können von Interessierten verändert und entwendet werden. Binärdaten sind einfach zu verschlüsseln und gegen Zugriff zu sichern, sodass hiermit ein deutlicher Sicherheitsgewinn zu verzeichnen ist.

OTX – das Austauschformat für Diagnoseabläufe

Der Standard beschreibt ein Austauschformat für Diagnose-Testsequenzen. Auch hier waren – ähnlich wie bei ODX – die Maschinenlesbarkeit und Langzeitverfügbarkeit die zentralen Anforderungen. Die Idee hinter dem Standard ist, eine Möglichkeit zu schaffen, bereits in der Spezifikationsphase grundlegende Abläufe zwischen Testssystem und Steuergerät formal zu beschreiben. Im Prozess können diese Abläufe weiterverwendet und spezialisiert werden, man muss also nicht wieder bei Null anfangen und einen Grundablauf aus einer Papierspezifikation abschreiben. Durch die Maschinenlesbarkeit wird darüber hinaus eine graphische Darstellung der Abläufe z. B. als Flussdiagramme unterstützt, die den Ingenieuren in Entwicklung, Verifikation, Produktion und Service eine allgemeingültige Basis für ihre Diskussionen bietet. Der Standard OTX wird in mehrere Bereiche unterteilt. Der Kern (Core) enthält eine Programmiersprache mit den typischen Elementen Variablen, Ausdrücken/Anweisungen und Operationen (Schleifen, Verzweigungen, Vergleiche,…). Er ist nicht diagnosebezogen und kann grundsätzlich für unterschiedliche Aufgaben angewendet werden. Standardisierte Bibliotheken stehen für spezielle Aufgaben zur Verfügung. Neben der Diagnosekommunikation (Anbindung an den D-Server) gehören dazu insbesondere Stringoperationen, Größenhandling, Darstellungsfunktionen (HMI) und die Internationalisierung, um die Ausgaben von Abläufen für verschiedene Märkte in verschiedenen Sprachen anbieten zu können. Die standardisierten Bibliotheken nutzen einen allgemeinen Erweiterungsmechanismus. Dieser kann auch für nicht-standardisierte Erweiterungen verwendet werden, wodurch in den Ablauf beliebige Testsysteme integrierbar sind. Typische Beispiele sind HiL-System, Simulationen oder Messtechnik. OTX unterstützt den Entwicklungsprozess durch verschiedene Mechanismen. Zunächst ermöglicht es für frühe Prozessphasen eine reine Spezifikationssicht. Dabei wird die Idee des Ablaufs beschrieben, ohne schon eine konkrete Implementierung des Ablaufs zu hinterlegen. Es kann somit beispielsweise ohne echte ODX-Datei, in der dann reale Diagnosedienste beschrieben sind, schon die Kommunikation mit einem Steuergerät skizziert werden. Dabei entsteht keine ablauffähige OTX-Sequenz. Dies wird dann später in der Implementierung der Sequenz nachgeholt. Um auch die Variantenvielfalt ähnlich wie in ODX zu unterstützen, ohne eine Sequenz jedes Mal vollständig neu freigeben zu müssen, wurden ebenfalls spezielle Mechanismen realisiert.

Im Unterschied zu ODX existiert für OTX kein standardisiertes API für eine Ablaufmaschine. Als Austauschformat steht es dem Toolhersteller frei, das Format in einem Interpreter zu verarbeiten, es in ein Maschinenformat zu kompilieren oder es in bereits existierende Testsysteme zu importieren und dort weiter zu verarbeiten.

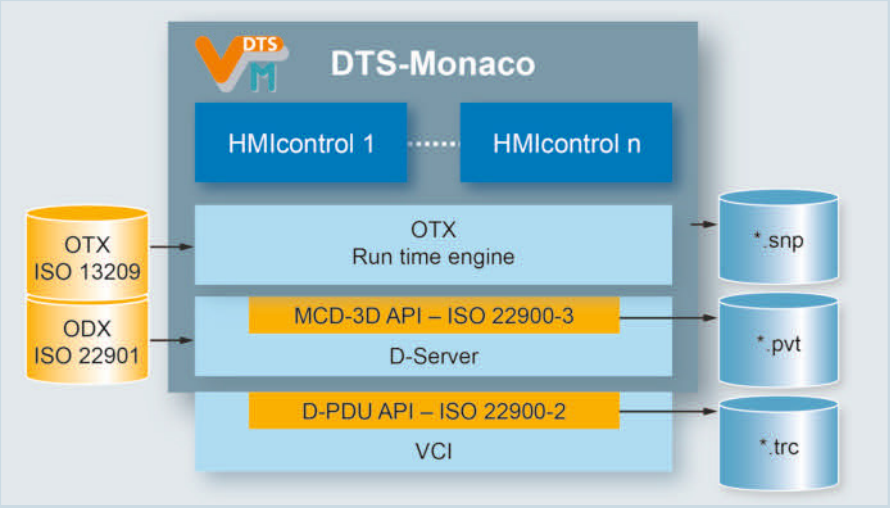

Bild 3: DTS-Monaco von Softing ist ein Beispiel für einen Entwicklungstester. In Abhängigkeit vom verwendeten Interface können grundsätzlich beliebige Protokolle unterstützt werden, solange ein Interface mit D-PDU API und entsprechende ODX-Daten existieren.

Entwicklungstester – ein Beispiel

DTS-Monaco ist ein Beispiel für einen Entwicklungstester. Es basiert auf den Standards D-PDU API, ODX, MCD-3D und OTX und beherrscht damit die relevanten Standards (**Bild 3**). DTS-Monaco besteht aus einigen wenigen Hauptkomponenten und kann anwendungsspezifisch erweitert werden: Das Framework, die HMIcontrols und die Run-times für die Standards OTX und ODX. ODX werden durch den D-Server DTS-COS bearbeitet. Er stellt das standardisierte API entsprechend ASAM MCD-3D zur Verfügung, bietet darüber hinaus aber noch einige Erweiterungen, die den Einsatz in Testautomatisierungen vereinfachen, und im Entwicklungstester über den Standard hinausgehende Funktionalitäten ermöglichen. Für OTX wurde ein Laufzeitinterpreter integriert.

Das Framework stellt die Basisfunktionalitäten zur Verfügung, die für die Diagnose notwendig sind. Neben der Anbindung an die Laufzeitsysteme ist das vor allem die Werkzeugkonfiguration, die abgespeichert und wieder geladen werden kann. Auch die Rollenverwaltung ist im Framework integriert. Monaco unterstützt die Rollen „Adminstrator" und „Anwender". Der Administrator kann Konfigurationen erstellen, abspeichern und sie dem Anwender zur Verfügung stellen. Dies betrifft sowohl Oberflächenkonfigurationen als auch OTX-Abläufe. Diese können durch den Administrator im Tool erstellt und angepasst werden. Der Anwender kann diese Vorkonfigurationen lediglich ins Tool laden und verwenden. Änderungen sind nur im geringen Umfang möglich. Dadurch ist sichergestellt, dass jeder Anwender nur die Funktionen am Fahrzeug verwendet, für die er auch ausgebildet ist. Innerhalb des Frameworks laufen als Komponenten die HMIcontrols. HMIcontrols übersetzen die Kommuni-

kationssicht, die durch den D-Server und die OTX run time zur Verfügung gestellt werden, in die Anwendungssicht, die vom Nutzer für seine spezielle Aufgabe erwartet wird. Dies kann eine Fehlerspeicherliste oder ein Messinstrument sein. Jedes HMIcontrol bringt für den Administrator seinen eigenen Konfigurationsdialog mit. Neben Einstellungsmöglichkeiten, die allen HMIcontrols gemeinsam sind, können dort auch spezifische Einstellungen vorgenommen werden. Beim Fehlerspeicher wäre dies zum Beispiel, ob Statusinformationen angezeigt werden sollen oder nicht, und beim Messinstrument, ob es als Zeigerinstrument dargestellt wird oder als Balkendiagramm.

Die Implementierung als Komponenten ermöglicht es auf einfache Weise zusätzliche Anwendungsfälle mit einem Kunden zu diskutieren, zu implementieren und nachträglich oder in einem spezifischen Setup zu installieren. Der Vorteil ist, dass diese Komponente völlig unabhängig getestet werden kann und die Systemintegrität in keiner Weise beeinflusst. Besonders wichtig werden diese Erweiterungskomponenten immer dann, wenn eine Anbindung an Logistiksysteme gefordert ist. Diese ist natürlich kundenspezifisch, kann also nicht ohne weiteres in ein Produkt implementiert werden, soll aber auch nicht für jeden Anwender sichtbar sein und muss deshalb unabhängig installiert werden.

Datenversorgung

Die Basis für DTS-Monaco sind ODX- und OTX-Laufzeitumgebungen. Die wichtigsten Daten im System sind also OTX- und ODX-Daten. Der Zugriff auf die ODX-Daten erfolgt – unabhängig ob von den HMIcontrols oder aus OTX heraus – über Namen („Drehzahl-Lesen"). Diese ODX-Namen müssen mit den Konfigurationen und den OTX Abläufen zusammen verwaltet werden. Dies erfolgt über Projekte.

Die OTX-Daten werden direkt im XML-Format verarbeitet. Dies erleichtert das Handling, weil hier entweder Daten aus einer Spezifikationsphase übernommen und angepasst werden oder direkt für die Anwendungsfälle des Entwicklungstesters erstellt werden. Datenkonvertierungen wären hier nur sperrig in der Bedienung. Für ODX-Daten gilt dies so nicht. Diese werden in einem kleinen Bereich der Anwendungsfälle erstellt, in der Vielzahl der Anwendungsfälle stehen sie allerdings schon unter Versionsverwaltung und sollen nur unter streng reglementierten Vorgaben verändert werden. Hier ist ein Schutz der Daten also angebracht. In DTS-Monaco erfolgt eine Konvertierung in ein Binärformat, das sowohl verschlüsselt als auch passwortgeschützt ist. Es lässt sich also auch mit einem Datenbankeditor nicht öffnen, solange man das Passwort nicht kennt. Das Datenformat unterstützt verschiedene Verwaltungsprozesse: Es ist sowohl möglich die Daten in einer ODX-Datenbank in einer Datei zusammenzufassen – typischerweise die Daten einer Baureihe –, als auch jedes Steuergerät mit seinen Varianten in einer eigene Datei abzuspeichern. Beide Vorgehensweisen haben Vor- und Nachteile.

Neben diesen Daten können in einem Projekt noch weitere Datentypen verwaltet werden, z. B. Simulationsdateien, die CAN-Matrix oder Filterdateien.

Anwendungsfälle mit HMIcontrols

Eines der zentralen HMIcontrols ist das DiagnosticServices Control. Es deckt einen großen Teil der Anwendungsfälle Datenverifikation und Analyse ab. Dies erfolgt über ein dreigeteiltes Fenster: Im ersten Teil kann man die Diagnosedienste der Datenbank in einer Baumdarstellung „browsen", sortiert nach Steuergeräten und nach Funktionsklassen. Der selektierte Diagnosedienst kann im zweiten Teil parametriert werden. Dies kann symbolisch erfolgen, indem der Request Parameter „Variable" auf „Drehzahl" gestellt wird, oder, indem direkt im Bytestream die Hexwerte geändert werden. Auf diese Art und Weise können zu Testzwecken auch „not OK"-Werte eingestellt werden, die symbolisch nicht erreichbar sind. Im dritten Teil werden die Ergebnisse dargestellt, wobei konfigurierbar ist, ob lediglich die Ergebnisse dargestellt werden oder auch die Struktur der Ergebnisse mit dargestellt wird. Für einen Schnellzugriff können zusätzlich Buttons konfiguriert werden, mit denen ohne lange Suche im Browser auf Diagnosedienste zugegriffen werden kann.

Zur Darstellung von Messgrößen stehen zwei unterschiedliche Methoden zur Verfügung: Messwertblöcke können textuell in einer Liste dargestellt werden, was einen sehr kompakten Überblick über die Größen ermöglicht. Die Wiederholrate ist einstellbar, ein Zugriff nur auf manuelle Anforderung ist auch möglich, um die Buslast gering zu halten. Daneben können Instrumente konfiguriert werden, die Vielfalt dieser Darstellungsmöglichkeiten ist erheblich. Übrigens

können solche Instrumente auch verwendet werden, um beispielsweise über Drehknöpfe oder Schalter Werte im Steuergerät zu verändern. Eine Darstellung mehrer Messwerte über der Zeit ist zusätzlich möglich.
Für die Flash-Programmierung steht ein eigenes HMIcontrol zur Verfügung. Dieses deckt zwei Anwendungsfälle ab: es kann sowohl für die Entwicklung der Funktion Flash-Programmierung verwendet werden, als auch für die reine Ausführung, um Programmstände zu aktualisieren oder Datensätze auszutauschen. Im ersten Fall ist es möglich, über ein fünfschrittiges Vorgehen Initialisierung, Kompatibilitätsprüfung, Programmierablauf, Validierung und Abschluss Teilfunktionen einzeln im Steuergerät in Betrieb zu nehmen. Dadurch werden die einzelnen Testläufe deutlich beschleunigt. Im zweiten Fall wird lediglich das zu programmierende Datenpaket ausgewählt und anschließend das Steuergerät auf Knopfdruck programmiert. In beiden Fällen wird der Programmierfortschritt über eine Fortschrittsanzeige dargestellt.
Die Behandlung von Fehlerspeicher und Identifikation ist in einem HMIcontrol „Quicktest" zusammengefasst. Der Schnelltest wird vor allem in der Versuchswerkstatt häufig eingesetzt. Dazu werden die Steuergeräteidentifikationen gelesen und anschließend von allen erkannten Steuergeräten der Fehlerspeicher. Ob zusätzlich die Umgebungsbedingungen abgefragt werden, kann der Anwender selbst konfigurieren. Die Herausforderung besteht darin, dass in der Versuchswerkstatt häufig nicht freigegebene Softwarestände in den Steuergeräten programmiert sind, sodass keine Variante erkannt werden kann. Ebenso häufig sind die gültigen ODX-Daten noch nicht verfügbar, auch in diesem Fall kann keine Variante erkannt werden und die Diagnose muss über die Basisvariante erfolgen. Um den häufigen Anwendungsfall Schnelltest möglichst performant ausführen zu können, werden möglichst alle Steuergeräte parallel angesprochen. Dies ermöglicht in der Praxis Performance-Gewinne um den Faktor 4. Allerdings können nicht alle Steuergeräte parallel angesprochen werden. Zunächst müssen die alternativen Steuergeräte erkannt werden, da beispielsweise nur entweder das Steuergerät für den 4-Zylinder-Dieselmotor oder das für den 6-Zylinder-Benziner vorhanden sein kann. Genauso muss aber beachtet werden, dass entsprechend der Bus-Architektur Steuergeräte hinter Gateways häufig erst nach dem Gateway selbst angesprochen werden können. Sequenziell anzusprechende Steuergeräte werden von einem Spezialisten konfiguriert.
Darüber hinaus sind noch zahlreiche weitere Ansichten auf Diagnoseinformationen verfügbar, die hier jedoch nicht erörtert werden sollen.

Testdokumentation

Es stehen mit DTS-Monaco verschiedene Möglichkeiten zur Verfügung, um die durchgeführten Tests zu dokumentieren. Der symbolische Trace zeichnet alle Sende- und Empfangsinformationen auf, die zwischen Tester und Steuergerät(en) ausgetauscht werden. Dabei werden zum einen die über die ODX-Daten interpretierten Werte gespeichert, zum anderen aber auch die auf hexadezimalem Level gesendeten und empfangenen Daten. Dadurch ist eine vollständige Prüfung auch im Nachgang zu einem Test darstellbar. Gleichzeitig ermöglicht es aber auch die Dokumentation des durchgeführten Tests. Die Ablage erfolgt als XML-Datei, sodass Konvertierungen in andere Formate oder Filterung auf die im jeweiligen Kundenprozess benötigte Darstellung unproblematisch machbar sind.
Parallel dazu kann auf der Ebene Bus-Kommunikation ein Trace mitgeschrieben werden. Dieser Trace, in den meisten Fällen ein CAN-Trace, erlaubt auch Auffälligkeiten in der Protokollverarbeitung sicher zu erkennen und zu dokumentieren.

Zusammenfassung

Standards haben sich im Diagnoseumfeld nicht nur durchgesetzt, es finden sich mit steigender Verwendungstiefe sogar weitergehende Bereiche, die standardisiert werden. Den Anfang haben Diagnoseprotokolle gemacht, bei denen bald kein Hersteller mehr für die Implementierung seiner Sonderlösungen in der steigenden Anzahl von Steuergeräten bezahlen wollte. Datenformate und Programmierschnittstellen zur Integration von Diagnoselösungen in unterschiedlichste Testsysteme waren ein logischer nächster Schritt. Unabhängig davon, dass Single-source grundsätzlich zum Kostensparen geeignet ist, gewinnt man durch den breiten Einsatz der entsprechenden Standards auch an Qualität. Sobald nämlich die Diagnose leicht zu integrieren ist, spricht nichts dagegen, deren durchaus mächtigen Methoden auch in vielen Bereichen einzusetzen und damit zu einer viel größeren Testbreite und -tiefe zu kommen.
Die Qualität ist aber das zentrale Argument für die Kundenbindung. Wenn man sieht, wie nervös die Hersteller bei Rückrufaktionen reagieren, sieht man, dass die Nachricht verstanden wurde. Wenn man sieht, wie penibel die Kunden Pannenstatistiken durcharbeiten und zu ihrer Kaufentscheidung heranziehen, weiß man warum. Wenn aber ein Fahrzeug trotz aller Bemühungen im Vorfeld einen Defekt hat, dann muss der Kunde nach einem möglichst kurzen Aufenthalt wieder zufrieden aus der Werkstatt fahren. Diagnose ist dazu der Schlüssel.
Sehr offensichtlich wird die Ersparnis, wenn man kalkuliert, was spezielle Prüfmechanismen in Entwicklung und Produktion kosten würden. Natürlich kann man zu Prüfzwecken spezielle Sensorik und Aktuatoren aufbauen, um Tests durchzuführen. Natürlich kann man zusätzliche Werkzeuge für diese Prüfungen bereithalten. Viel schneller und preiswerter – und in vielen Fällen ausreichend gut – geht es aber über die eingebauten Mechanismen im Fahrzeug mit ohnehin vorhandenen Diagnosetestern. Deshalb wird es genau so gemacht. (oe)

Markus Steffelbauer leitet das Produktmanagement und verantwortet Entwicklung und Marketing der Hardware- und Softwareprodukte bei der Softing Automotive Electronics GmbH. Daneben vertritt Steffelbauer seit vielen Jahren Softing in verschiedenen Standardisierungsgremien, seit 2008 als Vertreter im ASAM TSC (Technical Steering Committee).

Softing Automotive Electronics GmbH
www.softing.com