www.tzm.de

Fachartikel

Dipl.-Ing. (FH) Martin Hein

Einführung in

Autor: Martin Hein, Dipl.-Ing. (FH) Informations- und Medientechnik
Firma: TZM, Robert-Bosch-Straße 6, D-73037 Göppingen
Datum: 12. Dezember 2012

Inhaltsverzeichnis

1 Einführung

1.1 Überblick

Der **AUT**omotive **O**pen **S**ystem **AR**chitecture (AUTOSAR) Standard wird neben FlexRay sicherlich eine immer größere Bedeutung für die Elektrik/Elektronik-Architektur (E/E-Architektur) zukünftiger Automobile gewinnen. Daher sollten sich Fachkräfte in der Automobilindustrie, die sich mit der Entwicklung von Fahrerassistenzsystemen (FAS) und verwandten Themengebieten beschäftigen, zumindest mit den grundlegenden Konzepten dieses Standards vertraut machen.

Der vorliegende Artikel möchte hierfür als Einstiegspunkt dienen. Zunächst wird in einem kurzen Überblick die Entstehungsgeschichte und Zielsetzung von AUTOSAR beleuchtet. Im Anschluss daran werden einige grundlegende Begriffe des Standards, zum Beispiel das Run Time Environment (RTE), vorgestellt. Viele wichtige Aspekte müssen leider unberücksichtigt bleiben, da sonst der Rahmen des Artikels gesprengt würde. Vor allem die AUTOSAR Methodology wird nur ganz am Rand angesprochen, wäre jedoch für sich alleine bereits einen Artikel wert.

Inhaltlich stützt sich der Artikel auf Schulungsunterlagen von Vektor Informatik und Informationen die frei auf der AUTOSAR-Hompage verfügbar sind.

1.2 Entstehungsgeschichte und Ziele von AUTOSAR

Die Anforderungen durch Kunden und Gesetzgebung an die Automobilindustrie (OEM) und deren direkten Zulieferer (Tier 1) sind im Laufe der Zeit immer weiter gewachsen. Vor allem der Wunsch nach mehr Komfort, Sicherheit und Umweltschutz ist enorm gestiegen. Um den gleichermaßen erhöhten Anforderungen an die E/E-Architektur besser begegnen zu können, haben sich im Jahr 2002 eine Reihe von OEMs und Tier 1 zu einer Entwicklungspartnerschaft zusammengeschlossen. In den darauf folgenden Jahren sind weitere Firmen der Gemeinschaft in verschiedenen Zugehörigkeitsgruppen beigetreten. Abbildung 1.1 zeigt eine kleine Auswahl der heute zur AUTOSAR-Gemeinschaft gehörenden Firmen. Eine vollständige Liste ist auf der Internetpräsenz von AUTOSAR (`www.autosar.org`) zu finden.

Durch die gestiegenen Anforderungen sehen sich die Automobilhersteller zum Teil noch heute mit einer ganzen Reihe von Problemen konfrontiert. Im Folgenden sind einige davon aufgeführt:

- Die traditionell eingesetzte ISO/OSEK Softwarearchitektur bietet den Anwendungen keine volle Hardwareunabhängigkeit.
- Anwendungen besitzen keine hinreichende Modularität.
- Die Wiederverwendbarkeit von Codeteilen ist stark eingeschränkt.
- Die Wartung und Erweiterung von Anwendungen gestaltet sich schwierig.

Ziel von AUTOSAR ist es, eine modulare Softwarearchitektur mit standardisierten Schnittstellen zu etablieren, bei der die Anwendung von der Hardware unabhängig ist. Darüber hinaus sollen Tech-

nologien, die als Stand der Technik anzusehen sind, für alle Anwender einheitlich, in Form eines Standard-Stacks, zur Verfügung gestellt werden. Denn es ist nicht sinnvoll, dass jeder Hersteller für Standardkomponenten jeweils eigene Implementierungen vorhält. Ein Beispiel ist der Softwaretreiber für CAN-Controller. Mit einer eigenen Implementierung ließe sich mittlerweile kein Vorteil (im Sinne von Innovation oder Kostenreduktion) mehr erreichen. Mit kundenerlebbaren Funktionen dagegen schon. Daher werden diese im Gegensatz zum einfachen Treiber vom OEM oder dessen Tier 1 eigenständig implementiert. Ein Slogan der AUTOSAR-Gemeinschaft lautet aus diesem Grund: **Cooperate on standards - compete on implementation**.

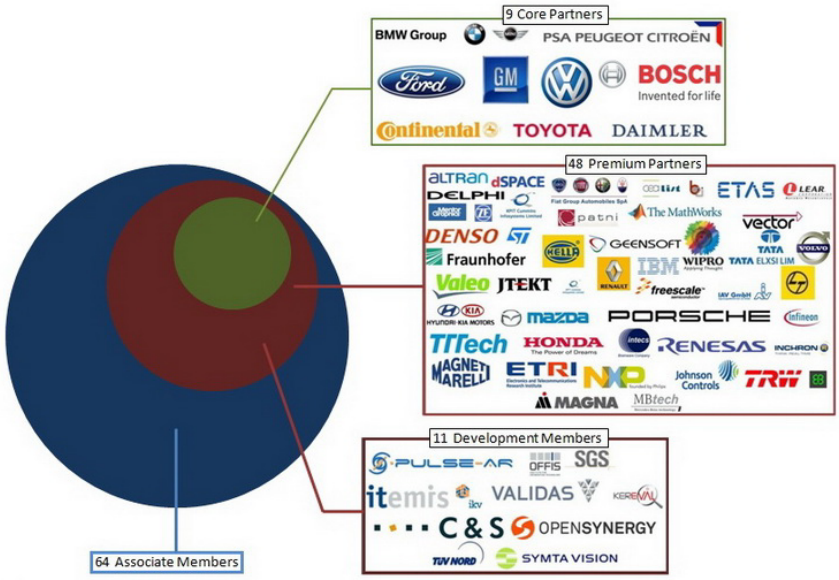

Abbildung 1.1: Firmen der Entwicklungsgemeinschaft AUTOSAR[1]

Durch die Umsetzung der im vorangegangenen Abschnitt vorgestellten Ziele wird erreicht, dass sämtliche Softwaremodule austauschbar und einfach wiederverwendbar sind. Mithilfe dieser beiden Eigenschaften kann das Gesamtsoftwaresystem mit relativ geringem Aufwand für verschiedene Fahrzeugtypen und Plattformvarianten skaliert werden. Zudem lassen Softwaremodule verschiedener Zulieferer kombinieren.

Zum Erreichen dieser Anforderungen wird ein großer Teil eines nach dem AUTOSAR-Standard erstellten Softwaresystems nicht klassisch programmiert, sondern mithilfe von Softwaretools konfiguriert. Für den einheitlichen Austausch zwischen OEM und TIER 1 spezifiziert AUTOSAR für die Konfiguration eine Reihe von Austauschformate auf XML-Basis. Aus dieser einheitlichen Konfigurationsbeschreibung wird ein Software-Grundgerüst (Skeleton) generiert. Ein weiterer Benefit dieses Vorgehens ist die höhere Qualität des generierten Sourcecodes, denn bei den Generatoren handelt es sich in der Regel um vielfach getestete Werkzeuge.

[1] Quelle(abgerufen am: 12. Dezember 2012): `http://www.cvel.clemson.edu/auto/AuE835_Projects_2010/allen_project.html`

2 Grundlagen

2.1 Das Schichtenmodell

Der AUTOSAR-Standard legt eine in Schichten organisierte Softwarearchitektur fest. In Abbildung 2.1 ist eine stark vereinfachte Form des Schichtenmodells dargestellt. Der Vollständigkeit halber ist die unterlagerte Hardware als eigene Schicht eingezeichnet, diese findet jedoch im Rahmen dieses Artikels keine Beachtung.

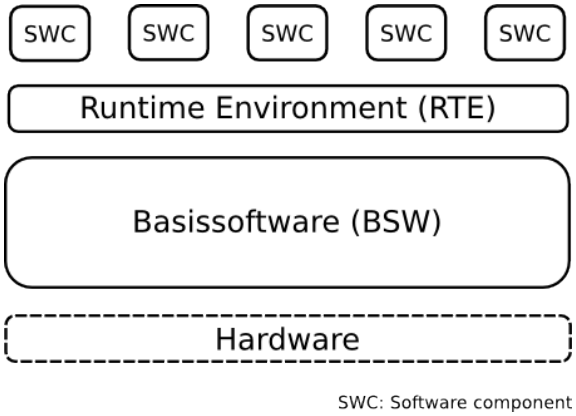

Abbildung 2.1: Schichtenmodell des AUTOSAR-Standards

Wie bereits im Abschnitt „Entstehungsgeschichte und Ziele von AUTOSAR“ erwähnt, besitzt jede Schicht standardisierte Schnittstellen zu der jeweils darüber bzw. darunter gelegenen Schicht. Ähnlich dem ISO/OSI-Schichtenmodell ist auch hier nicht erlaubt bzw. möglich, dass eine Schicht auf die Schnittstellen einer ihr nicht direkt benachbarten Schicht zugreift.

AUTOSAR kennt drei Arten von Schnittstellen zwischen den einzelnen Schichten und deren Komponenten. Dabei ist durch den Standard festgelegt, welche Schnittstellen eine bestimmte Komponente anbieten muss.

AUTOSAR Interface:

Der Aufbau dieser Schnittstellen ist im AUTOSAR-Standard durch die Vorgabe von Mustern für den Aufbau der Funktionsprototypen festgelegt. Ein Beispiel hierfür ist das folgende Muster eines Funktionsprototypen für Sender/Receiver-Ports: Rte_Write_<Port>_<Data>(...)

Standardized Interface:

Standardized Interfaces sind einfache C-APIs. Der Aufbau der Funktionsprototypen ist durch den AUTOSAR-Standard fest vorgegeben. Ein Beispiel hierfür ist die Funktion Schedule().

Standardized AUTOSAR Interface:

Der Aufbau von Standardized AUTOSAR Interfaces ist mit denen von AUTOSAR Interfaces identisch. Der Unterschied liegt in der Verwendung dieser Schnittstellen. Sie werden ausschließlich von den Servicekomponenten der Basissoftware angeboten. Daher spricht man bei ihnen auch von Service-Ports.

2.2 Planung und Virtual Function Bus

Bei der Erstellung eines AUTOSAR-konformen Softwaresystems für ein Steuergerätecluster werden in einem ersten Planungsschritt unter anderem alle benötigten Softwarekomponenten (SW-C) und die zwischen ihnen notwendigen Kommunikationskanäle festgelegt. Dabei ergibt sich eine Struktur wie sie beispielhaft in Abbildung 2.2 dargestellt ist.

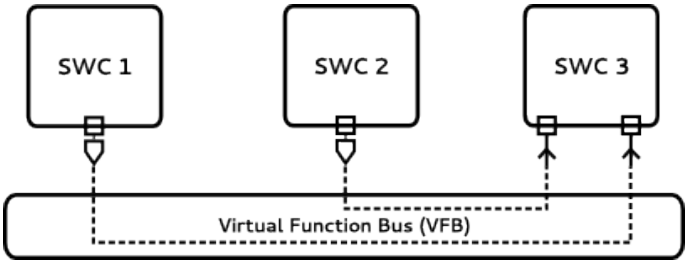

Abbildung 2.2: Der Virtual Function Bus mit Softwarekomponenten

Zu diesem Zeitpunkt ist die Verteilung der SW-Cs auf die einzelnen Steuergeräte (ECU) noch nicht festgelegt. Der unter den SW-Cs eingezeichnete Virtual Function Bus (VFB) symbolisiert folglich sowohl die Kommunikation zwischen SW-Cs, die auf derselben als auch auf unterschiedlichen ECUs laufen. Im Anschluss an diesen Planungsschritt werden die SW-Cs auf die einzelnen ECUs verteilt.

Die folgenden Abschnitte widmen sich der Erläuterung von Fachtermini, die im vorangegangenen Text und in den Abbildungen erwähnt, aber nicht näher betrachtet worden sind.

2.3 Softwarecomponent, Runnables und Ports

Bei einer SW-C handelt es sich um ein atomares[2] Softwaremodul. Es sind die Puzzleteile aus denen sich die eigentliche Applikation zusammensetzt. AUTOSAR kennt unterschiedliche Arten von SWCs:

Application SW-C:

Eine Application SW-C implementiert einen bestimmten Algorithmus. Somit handelt es sich um einen Baustein, der durch Verknüpfung mit anderen Application SW-Cs zur Lösung komplexer Ausgabenverwendet wird. Diese Art von SW-C ist völlig hardwareunabhängig und kann fast beliebig auf ECUs verteilt werden.

Sensor/Actuator SW-C:

Bei den Sensor/Actuator SW-Cs handelt es sich um Softwarebausteine, die einen Sensor oder Aktuator auf der SWC-Ebene abbilden. Sie haben ein Verständnis über den Ihnen zugeordneten Sensor bzw. Aktuator. Zwar sind sie nicht an die ECU gebunden an dem der zugehörige Sensor angeschlossen ist, werden aber aus Performancegründen meistens auf ihr platziert.

[2] Atomar bedeutet in diesem Zusammenhang, dass eine SWC nicht auf unterschiedliche Steuergeräte aufgeteilt werden kann

Compositions SW-C:

Composition SW-Cs dienen lediglich der Strukturierung auf Toolebene. Mit ihnen können logisch zusammengehörende Application- und Sensor/Actuator-SWCs zusammengefasst werden. Eine Composition ist nicht atomar. In ihr enthaltene SW-Cs können also beliebig auf unterschiedliche Steuergeräte verteilt werden.

SW-Cs bestehen aus einer oder mehreren Runnables. Dabei handelt es sich um C-Funktionen, die die eigentliche Implementierung der Algorithmen enthalten. Folglich sind sie die ausführbaren Einheiten einer SW-C. Die Runnables werden durch die im nächsten Abschnitt beschriebene RTE aufgerufen. Dies geschieht entweder periodisch oder event-getriggert. Zudem werden die Runnables einer Task des in der Basissoftware (BSW) angesiedelten Betriebsystems zugeordnet.

Ports dienen den Softwarekomponenten als Kommunikationsschnittstelle. Es sind die Endpunkte der im vorherigen Abschnitt angesprochenen Kommunikationskanäle. AUTOSAR unterscheidet zwischen Ports für das Sender/Receiver und das Client/Server Kommunikationmuster:

Sender/Receiver-Port:

Sender/Receiver-Ports repräsentieren Datenelemente. Falls die Übertragung der Daten über einen physikalischen Bus erfolgt, müssen die einzelnen Datenelemente einem Netzwerksignal zugeordnet werden. Mit den Sender/Receiver-Ports wird eine asynchrone, unidirektionale Eins-zu-N- oder N-zu-Eins-Kommunikation realisiert. Die RTE verbirgt den oder die Empfänger vor dem Sender, sodass sich dieser nicht um das Management der Empfänger kümmern muss.

Durch diese Art der Kommunikation wird eine einfache Übertragung von unabhängig in der Sender-SWC generierten Daten an den oder die Empfänger realisiert.

Bei den Datenelementen kann es sich sowohl um einfache (int, double etc.) als auch um komplexe Datentypen (struct, array etc.) handeln.

Client/Server-Port:

Bei der Client-Server-Kommunikation handelt es sich um eine Eins-zu-Eins- oder N-zu-Eins-Kommunikation. Der Client ruft eine Operation (runnable) in der Server-SW-C auf. Dieser Aufruf kann aus Sicht des Client blockierend (synchron) oder nicht blockiernd (asynchron) erfolgen. Beim blockierenden Aufruf wartet die Client-SW-C auf die Antwort der Server-SW-C. Beim nicht blockierenden Aufruf läuft die Client-SW-C weiter, während sie auf die Antwort wartet.

Ein Client-Server-Port kann mehrere von einander unabhängig aufrufbare Operationen anbieten. Wie bei den Sender/Receiver-Ports ist neben der ECU-internen auch die Kommunikation zwischen SW-Cs auf unterschiedlichen ECUs möglich.

Unabhängig vom verwendeten Kommunikationsmuster unterscheidet man zudem zwischen dem Provide-Port (PPort), über den eine SW-C eine Operation oder ein Datenelement anbietet, und dem Require-Port (RPort), mit dessen Hilfe eine SW-C eine Operation aufrufen oder auf ein Datenelement zugreifen kann.

Sämtliche Aspekte einer SW-C werden formal in der SW-C Description zusammengefasst. Mithilfe dieser Beschreibung wird später unter Verwendung weiterer Beschreibungsdateien ein AUTOSAR-konformes Software-Grundgerüst generiert. Dafür enthält die SW-C Description die folgenden Informationen:

- Attribute von Operationen und Datenelementen (zum Beispiel blockierend/nicht-blockierend oder die Datenlänge) die von der SW-C zur Verfügung gestellt oder benötigt werden.
- Die Anforderungen der SW-C an die unterlagerte Infrastruktur (RTE und BSW).
- Die von der SW-C benötigten Ressourcen. Zum Bespiel CPU-Zeit und Speicherbedarf.
- Informationen zur Implementierung der SW-C

Der grundsätzliche Aufbau einer SW-C-Description Datei wird durch das Software Component Template vorgegeben.

2.4 Runtime Environment

Das Runtime Environment (RTE) ist unter anderem die Implementierung des VFBs. Dazu müssen (wie im vorletzten Abschnitt beschrieben) zunächst die SW-Cs auf die einzelnen ECUs verteilt werden, denn das logische Konstrukt des VFBs zerfällt bei der Implementierung in einzelne, steuergerätespezifische RTEs (siehe Abbildung 2.3).

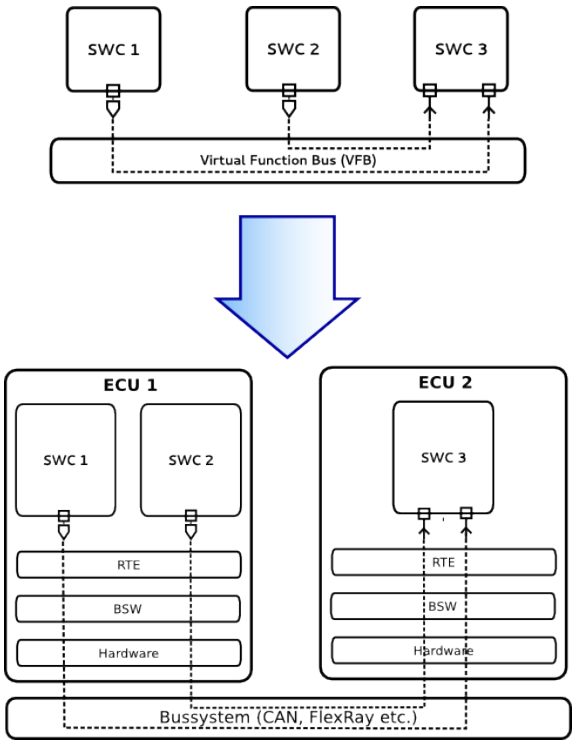

Abbildung 2.3: Implementierung des VFB durch das RunTime Environment

In diesem Zusammenhang übernimmt das RTE die Rolle einer Middleware. Bei einer Middleware handelt es sich um eine fortgeschrittene Art der Interprozesskommunikation. Sie ermöglicht unter-

schiedlichen Softwaremodulen einen Datenaustausch, ohne etwas über den Kommunikationspartner wissen zu müssen. Dazu greifen die einzelnen Softwaremodule auf standardisierte Schnittstellen der Middleware zu. Man kann sich eine Middleware daher als virtuelles Bussystem vorstellen. Folglich interessiert es auch nicht, ob sich der Kommunikationspartner im gleichen oder auf einem anderen ECU befindet.

Darüber hinaus ist die RTE für das Ausführen der einzelnen Runnables verantwortlich. Die SWCs sind durch die RTE vollständig von der BSW und damit vom Betriebsystem getrennt. Die Runnables werden bei ihrer Konfiguration einer Task des Betriebsystems zugeordnet. Die RTE erhält dadurch Codeabschnitte, die durch die Tasks des OS ausgeführt werden und ihrerseits die einzelnen Runnables aufrufen. Dieser Sachverhalt ist in Abbildung 2.4 dargestellt. Zudem können Runnables beim Auftreten von Kommunikationsevents zur Ausführung gebracht werden.

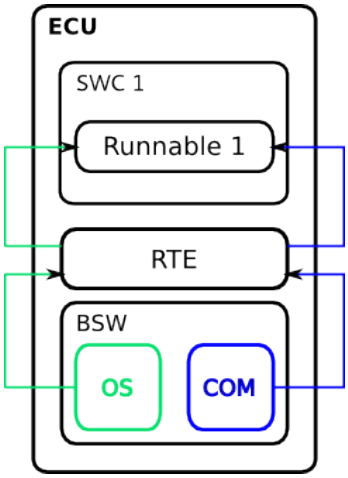

Abbildung 2.4: Aufruf von Runnables aus der Basis Software

Die RTE eines jeden ECU ist zwar aus Entwicklersicht skalierbar, wird jedoch schlussendlich durch Werkzeuge statisch generiert. Das bedeutet auch, dass die RTE bei einer Änderung an der Kommunikation zwischen SWCs neu generiert werden muss.

2.5 Basicsoftware

In klassisch erstellter ECU-Software wird viel Aufwand bei der Implementierung und Optimierung von Software betrieben, die als Basis für kundenerlebbare Funktionen fungieren. Durch die Verwendung einer einheitlichen Basissoftware wird nicht nur der Entwicklungsaufwand reduziert, sondern auch die Softwarequalität erhöht. Der Fokus der Entwicklung kann zu großen Teilen auf kundenerlebbare Funktionen gelenkt werden. Daher sind weite Bereiche der BSW durch den AUTOSAR Standard spezifiziert und werden mittels entsprechender Tools konfiguriert und anschließend generiert.

Die BSW ist in vier Schichten unterteilt, die ihrerseits eine Reihe von Komponenten zur Erfüllung der jeweiligen Aufgaben enthalten. Abbildung 2.5 zeigt, wie die Schichten (blau hervorgehoben) ineinander greifen.

Microcontroller Abstraction Layer:

Das Microcontroller Abstraction Layer (MCAL) sorgt mit seiner standardisierten Schnittstelle zur nächst höheren Softwareschicht für die Unabhängigkeit vom unterlagerten Microcontroller. Ziel ist es, unabhängig von der Verfügbarkeit eines bestimmten Microcontrollers zu werden, denn ein Austausch des Microcontrollers bedeutet in klassischen Systemen einen erheblichen Aufwand bei der Anpassung des gesamten Softwarestacks. Durch die Unabhängigkeit vom unterlagerten Microcontroller kann dieser bei gestiegenen Anforderungen zudem einfacher ausgetauscht werden.

Das MCAL als microcontrollerspezifische Schicht wird im Normalfall vom Hersteller der Microcontroller geliefert. Sie kapselt alle controllerspezifischen Eigenheiten. Dazu zählt unter anderem die von Controller zu Controller unterschiedliche Handhabung von Schnittstellen aller Art. Als Beispiel sind hier vor allem die digitalen und analogen Ein- und Ausgänge und die Speicheranbindung zu nennen.

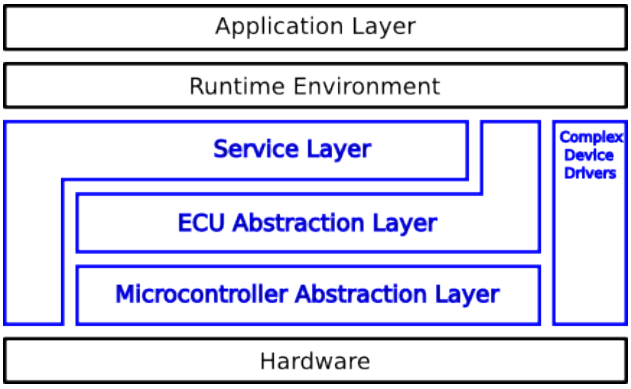

Abbildung 2.5: Die Schichten der Basis Software

ECU Abstraction Layer:

Das ECU Abstraction Layer kapselt die als Peripherie an den Microcontroller des ECU angebundene Hardware. Damit werden die darüber gelegenen Schichten unabhängig von der Lage der Peripherie (auf dem oder außerhalb des Microcontroller-ICs). Es kann sich dabei beispielsweise um Schaltungen (ASICs etc.) handeln, die Sensorsignale für den Microcontroller oder vom Microcontroller generierte Signale für Aktuatoren aufbereiten. Auch ICs für die Anbindung an verschiedene Bussysteme gehören in diese Kategorie. Durch diese Schicht wird in Verbindung mit der MCAL die gesamte Hardware des Steuergeräts abstrahiert.

Service Layer:

Das Service Layer ist die oberste Schicht der BSW und in weiten Teilen hardwareunabhängig. Sie umfasst Servicekomponenten für die einheitliche Kommunikation über Bussysteme, das Speichermanagement und System Services.

Complex Device Driver:

Die Complex Device Driver bieten die Möglichkeit, spezielle Software in den AUTOSAR-Stack zu integrieren. Einzige Anforderung ist, dass standardkonforme Schnittstellen angeboten werden.

Bei dieser Art von Software kann es sich beispielsweise um die Anbindung komplexer Sensoren oder Aktuatoren handeln, die einen direkten Zugriff auf die Microcontroller-Hardware erfordern. Ein anderer Grund für die Implementierung von Software als Complex Device Driver kann die Verwendung eines ASICs sein, für den der AUTOSAR-Stack keine Funktionalität bietet.

Darüber hinaus sind Complex Device Driver als Migrationpfad gedacht, damit OEMs und Zulieferer nicht gezwungen sind, vorhandene Implementierungen sofort komplett nach AUTOSAR zu portieren.

3 Fazit

Wie bereits in der Einführung des Artikels erwähnt, gewinnt AUTOSAR immer mehr Bedeutung bei der Entwicklung von Software für ECUs. Dieser Prozess ist gerechtfertigt, denn nicht zuletzt durch das modulare Konzept, die klare Definition von Schnittstellen und einer einheitlichen, durchgängigen Beschreibung mittels XML-basierter Dateien handelt es sich um einen zukunftsträchtigen Standard. Damit ermöglicht der Standard eine in weiten Teilen toolgestützte Entwicklun. Dies ist der Qualität der erzeugten Software zuträglich.

Auch wenn dieser Artikel sicherlich viele der zu den Grundlagen von AUTOSAR gehörenden Aspekte unbehandelt lässt, hoffe ich, einen Einblick in die Welt von AUTOSAR gegeben zu haben. Kritik, Richtigstellungen und sonstige Anmerkungen nehme ich gern per E-Mail entgegen (martin.hein@tzm.de).