(R)Evolution der Automotive-Software-Architekturen

Wie neue SW-Technologien die Automobilindustrie verändern

Dr.-Ing. Detlef Zerfowski und Niranjan SK
Robert Bosch GmbH Automotive System Integration (C/AIS)

1. Auslöser für Änderungen der Automotive-SW-Architekturen

Die gesamte Automobilindustrie wird aktuell mit dramatischen technologischen Änderung konfrontiert. Diese Veränderungen ergeben sich durch den Einzug völlig neuer elektrischer und elektronischer Architekturen (E/E-Architekturen) und damit einhergehend völlig neuen SW-Architekturen. Schlüsseltreiber dieser Entwicklung sind insbesondere die Folgenden:

- Übergang von µ-Controller-basierten, klassischen embedded Steuergeräten hin zu µ-Prozessor-basierten bzw. Cloud-basierten Lösungen.
- Übergang von regelungsbasierten Algorithmen hin zu datengetriebenen Algorithmen (Artificial Intelligence, Bildverarbeitung, Datenfusion, Connectivity)
- Loslösung der SW im Fahrzeug von der darunterliegenden Hardware.

Früher konnten Fahrzeuge softwareseitig als „geschlossene Systeme“ betrachtet werden, die nach Auslieferung in die Serie nur unter besonderen Bedingungen (d.h. in den Werkstätten) geändert werden konnten. Heute sind moderne Fahrzeuge eher Teile des Internet der Dinge. Diese Entwicklung führt zu gänzlich anderen Architekturtreibern als in der Vergangenheit und hat unmittelbare Auswirkungen auf die SW-Architekturen in der Automobilindustrie.

2. µ-Controller-basierte ECUs und ihre Architekturtreiber

Seit der Einführung des ABS für PKW im Jahr 1978 sind in den vergangenen Jahrzehnten sukzessive eine große Anzahl elektronischer Steuergeräte (Electronic Control Units, ECU) ins Fahrzeug eingeführt worden. In heutigen Premiumfahrzeugen werden mittlerweile um die 100 Steuergeräte der unterschiedlichen Tier 1 Lieferanten verbaut.
Über die vergangenen Jahrzehnte haben sich diese Komponenten iterativ im Rahmen der OEM-spezifischen E/E-Architekturen im Fahrzeug weiterentwickelt.

Aus der Sicht eines Tier 1 Zulieferers wie Bosch werden die ECU-spezifischen Architekturen in einem Umfeld wie in Abb *1* entwickelt. Der OEM besitzt die Hoheit über das Fahrzeug und dessen E/E-Architektur. Entsprechend der E/E-Architektur werden die Funktionen den jeweiligen ECUs zugeordnet (Function Deployment). Hieraus werden die ECU-Spezifikationen und Schnittstellen abgeleitet, die dann dem Tier 1 als Lastenheft zur Verfügung gestellt werden.
Der Tier 1 bewegt sich pro Produktsegment in einem ähnlichen Kontext mit mehreren OEMs und optimiert sich über entsprechende Hardware-Plattformlösungen. In dieser Situation legen die Hardwarekomponenten die wesentlichen Architekturtreiber für die Software fest.

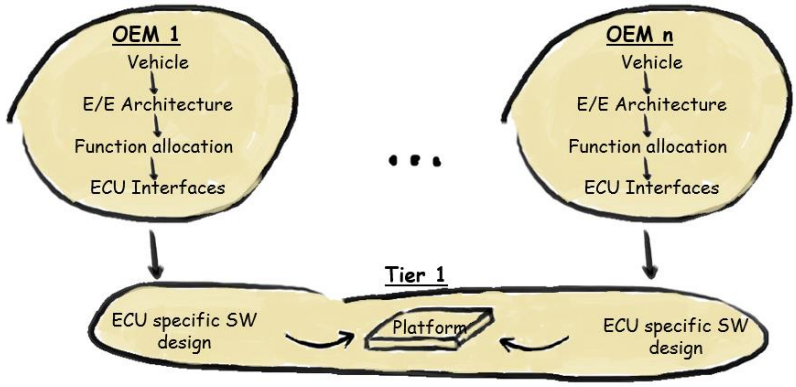

Abb 1: Cross-OEM Plattformentwicklung beim Tier 1.

Dieses Zusammenarbeitsmuster ist in den unterschiedlichen Anwendungsdomänen (Body Electronic, Powertrain, Chassis und Infotainment) wiederzufinden (siehe Abb. *2*). Es gibt jedoch erhebliche Unterschiede (Prozesse, Methoden und Tools) darin, wie die Produkte in den einzelnen Bereichen entwickelt werden. Somit unterscheiden sich die Architekturansätze in den Bereichen erheblich. Während es beispielsweise bei Motorsteuergeräten mit einer überschaubaren Anzahl von Hardwareplattformen eine extrem hohe Software-Variantenanzahl gibt, variiert bei Body Computern die Hardware deutlich mehr, was dazu führt, dass SW-Komponenten für eine breitere Verwendung hardwareunabhängiger entwickelt werden müssen. Hieraus entwickelten sich AUTOSAR-Ansätze für µ-Controller-basierte Basissoftware-Stacks.

Die µ-Controller-Technologie gibt restriktive Vorgaben an verfügbarem Speicher und Rechenlaufzeit vor. Eine moderne Motorsteuergerät-Software muss sich mit bescheidenen 8 MB Speicher begnügen. Für die Software resultiert das in einer speicheroptimierten Architektur, was zwangsläufig zu Lasten der Wartbarkeit geht.
Dynamische Speicherallokierung ist aus Gründen der knappen Ressourcen und aus Sicherheitsgründen ebenfalls zu vermeiden.
Um die deterministischen Laufzeitanforderungen (auch im Hinblick auf die Funktionale Sicherheit) einhalten zu können, kommen Realzeitbetriebssysteme zum Einsatz, die über deterministisch vergebene Zeitscheiben die Rechnerleistung auf die unterschiedlichen Tasks aufteilen.
Außerdem kommen statische Kommunikationsstrukturen (auf Fahrzeugebene definierte CAN-Kommunikationsmatrizen, die zur Entwicklungszeit feststehen) zum Einsatz, die einerseits die Systeme weniger anfällig gegenüber Angriffen von Hackern machen, andererseits jedoch flexible Service-orientierte Anwendungen nicht unterstützen.
Des Weiteren wird den beschränkten Rechner-Ressourcen dadurch Rechnung getragen, dass implementierte Algorithmen häufig mit quantisierten Zahlen in Integer-Arithmetik gerechnet werden.

Die aufgeführten Architekturtreiber schränken den Lösungsraum für SW-Architekturen erheblich ein.

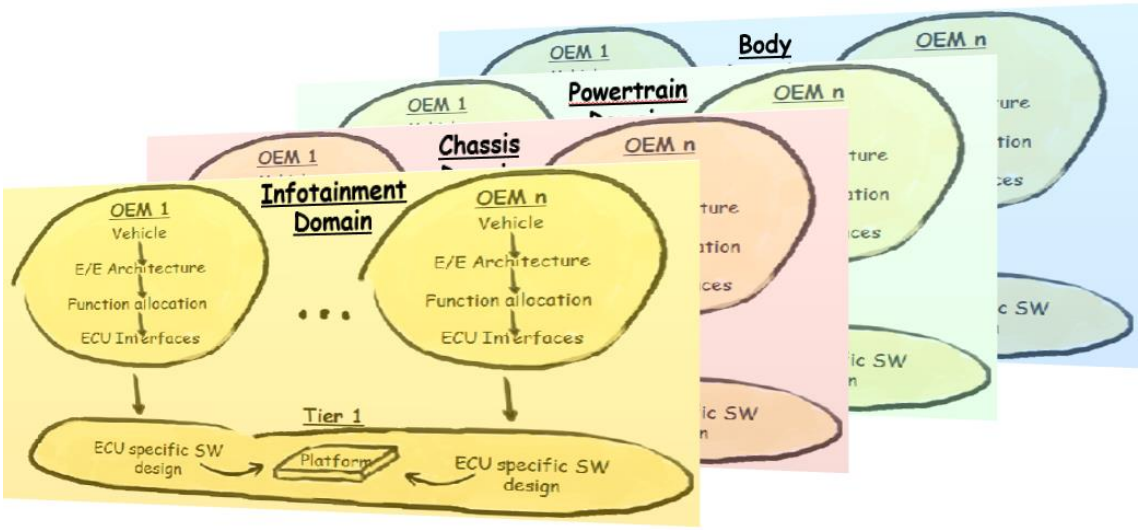

Abb. 2: Anwendungsdomänen-spezifische Ausprägung

Auch die in Abb. *2* gezeigte, sich über mehrere Jahre bei OEMs und Tier 1 entwickelte Domänenaufteilung stellt implizit einen wesentlichen Architekturtreiber dar – die Verteilung der ECU-Komponenten (und damit der in ihnen liegenden Software) auf die unterschiedlichen Domänen!
Wir haben es hier mit einem Fall von „Conway's Law" [1] zu tun:
„…organizations which design systems (in the broad sense used here) are constrained to produce designs which are copies of the communication structures of these organizations."

Mit anderen Worten: Architekturen folgen den Organisationsstrukturen. Solange Aufgabenstellungen nur innerhalb der existierenden Organisationsstrukturen bearbeitet werden müssen, funktioniert dies gut. Wenn jedoch Problemstellungen auftauchen, die quer zu oder außerhalb von vorhandenen Strukturen liegen, lassen sich häufig keine geeigneten Architekturen etablieren. Organisatorische Änderungen sind die notwendige Folge.

In den nachfolgenden Abschnitten werden wir darauf eingehen, dass diese Entwicklung in der Automobilindustrie aktuell stattfindet.

Zusammengefasst ergeben sich für die µ-Controller-basierten Systeme folgende (nicht vollständigen) SW-Architekturtreiber, die durch neue, in die Automotive-Welt eindringende Technologien gleichzeitig in Frage gestellt werden.

Architekturtreiber	Auswirkung auf SW-Architektur

µ-Controller-Hardware: begrenzter Speicher	Speicheroptimierte Software
µ-Controller-Hardware: begrenzte Laufzeit	Fokus auf regelungsbasierte Algorithmen (nahe an der Physik des Fahrzeugs) Deterministische Laufzeitbedingungen.
Organisationstrukturen nach Anwendungsdomänen	Lösungsraum für SW wird auf domänenspezifische Hardware-Lösungen eingeschränkt. ECUs werden zusammen mit der Software vertrieben. Domänenübergreifende Lösung nur schwer umsetzbar.
Statische Kommunikationsstruktur (CAN-Matrix)	Datenflüsse können nicht bzw. nur mit hohem Aufwand und hohen Kosten geändert werden. Datentransferraten liegen im unteren MBit/Sekunde-Bereich.
Eingeschränkte Updatefähigkeit	Software-Updates in erster Linie zur Fehlerbehebung und nicht vorrangig zum Hinzufügen neuer Funktionen

3. µ-Prozessor-basierte Computer erobern das Fahrzeug

Im vorherigen Abschnitt haben wir das Umfeld beschrieben, in dem sich die Automobilsoftware in den vergangenen Jahrzehnten iterativ weiterentwickelt hat. Die zunehmende Vernetzung des Automobils inklusive seiner Komponenten sowie die Einführung der µ-Prozessor-Technologie im Fahrzeug führen aktuell zu einer disruptiven Änderung der Automobilsoftware.

Dabei ist zu bemerken, dass die Infotainment-Anwendungen bereits seit Jahren auf µ-Prozessor-basierten Systemen mit Linux-Betriebssystemen arbeiten. Damit sind einige der Aussagen in den nachfolgenden Abschnitten nur in Teilen für die Infotainment-Domäne zutreffend. Aufgrund der geringeren Anforderungen bezüglich Funktionaler Sicherheit und deutlich geringeren Realzeitanforderungen entwickelten sich die Infotainment-Anwendungen separat von den Embedded-Anwendungen. Mit der Integration von Anwendungsfunktionen unterschiedlicher Domänen auf gemeinsamen Fahrzeugrechnern laufen die bisherigen Entwicklungswege wieder verstärkt aufeinander zu.

4. Anstieg der Rechner-Ressourcen

Der offensichtlichste Effekt bei der Einführung der µ-Prozessoren liegt in den verfügbaren höheren Rechner-Ressourcen. Während einem modernen Motorsteuergerät ein Speicher von bis zu 8 MB zur Verfügung steht (in dem neben der Anwendungssoftware auch das Realzeitbetriebssystem und Basis-SW realisiert sind), besitzen die ersten µ-Prozessor-basierten Systeme 8 GB Speicher (SOP 2020) und Nachfolgesysteme 128 GB (SOP 2023). Obwohl dies aus Sicht der Consumer-Elektronik immer noch gering erscheint, hat dieser Zuwachs um einen Faktor von 1000 (SOP 2020) bzw. 16000 (SOP 2023) Auswirkungen, die über die reine Verfügbarkeit des Speichers weit hinausgehen.

Die früher vorhandene Trennung der domänenspezifischen Software auf dedizierte Steuergeräte entfällt, da zukünftige Fahrzeugrechner unterschiedlichste Fahrzeugfunktionen auf einem Rechner darstellen können. Speicheroptimierte SW-

Designs und Implementierungen werden durch SW-Architekturen abgelöst, die insbesondere die Unabhängigkeit von der Hardware und somit Portabilität im Fokus haben. Hieraus folgt, dass sich die Anwendungssoftware von der darunterliegenden Hardware löst und somit auf Rechnern einer anderen Anwendungsdomäne ablaufen kann. Dies hat unmittelbare Auswirkungen auf die bisher etablierten Geschäftsmodelle und Entwicklungsorganisationen. (siehe Abb. *2*). Bereiche, die bisher isoliert voneinander komponentenbasierte Lösungen mit integrierter Software entwickelt und vertrieben haben, müssen fortan domänenübergreifende SW-Lösungen entwickeln. Die große Herausforderung ist dabei Conway's Law. Die bisher vorhandenen domänenspezifischen und hardwareorientierten SW-Architekturen müssen domübergreifenden SW-Architekturen weichen. Diesen stehen jedoch die bisherigen organisatorischen Strukturen entgegen. Als Folge bilden sich aktuell in der Automobilindustrie neue, stärker softwareorientierte Organisationsformen heraus. Software-Architekturen und damit auch Software-Architekten gewinnen massiv an Bedeutung und Einfluss auf die Produktentwicklungen.

Neben der Verfügbarkeit hoher Speichergrößen ermöglicht der Anstieg der Rechnerleistung von ~4 kDMIPS (heutige Motorsteuergeräte) auf ~17 kDMIPS (SOP 2020) bzw. ~300 kDMIPS (SOP 2023) eine Vielzahl neuer Anwendungen, die auf bisherigen Systemen nicht realisierbar sind. Die für µ-Controller häufig notwendige Einschränkung auf Ganzzahlarithmetik entfällt, und es halten zunehmend datenorientierte Algorithmen Einzug ins Fahrzeug, bis hin zu Artificial Intelligence (AI) Lösungen auf Graphical Processor Units (GPU).
Es werden dabei Funktionen realisiert, die sich von Teilfunktionalitäten unterschiedlicher Domänen (inklusive Connectivity und Cloud-Diensten) bedienen und sich somit nicht mehr den klassischen Automotive-Domänen zuordnen lassen (siehe Abb. *3*).

5. Beispiel Reibwertkarte

Ein Beispiel hierfür stellt die sogenannte Reibwertkarte dar, die dem Autofahrer über eine Internetverbindung in seiner Anzeige Informationen über gefährliche Straßensituationen (wie Glatteis, Aquaplaning etc.) nahezu in Realzeit zur Verfügung stellt.
Das Datenmaterial für diese Reibwertkarte liefern vorausfahrende Fahrzeuge, die mittels der ESP-Funktionalität (Anwendungsdomäne Chassis) Informationen über den aktuellen Reibwert der Straße erfassen. Diese Information wird über ein Gateway (Body Computer Domäne) an den Dienstanbieter in der Cloud geschickt (bisher nicht durch entsprechende Domänen abgedeckt). Die von unterschiedlichen Fahrzeugen bereitgestellten Daten werden vom Dienstanbieter konsolidiert und umgehend an die Fahrzeuge verteilt, die den Reibwertkartendienst abonniert haben. Die Information über die unmittelbar vor ihm liegende Gefahrensituation wird dem Fahrer als Warnmeldung und in der Navigationskarte dargestellt (Infotainment-Domäne). Zusätzlich kann der Stauassistent aufgrund der Information bereits die Geschwindigkeit des Fahrzeuges verringern (Powertrain-Domäne). Beim Einsatz im Hochautomatisierten Fahren wird die Information das System und damit das Fahrverhalten ohne Einwirkung des Fahrers direkt beeinflussen.

Anhand dieses einfachen Beispiels wird deutlich, dass die funktionale Architektur und damit auch die SW-Architekturen unterschiedliche Rechnerknoten innerhalb der E/E-Architektur abdecken müssen. Dabei sind die E/E-Architekturen von OEM zu OEM unterschiedlich. Beim SW-Design muss dies berücksichtigt und eine starke Abstraktion von der Hardware sichergestellt werden.

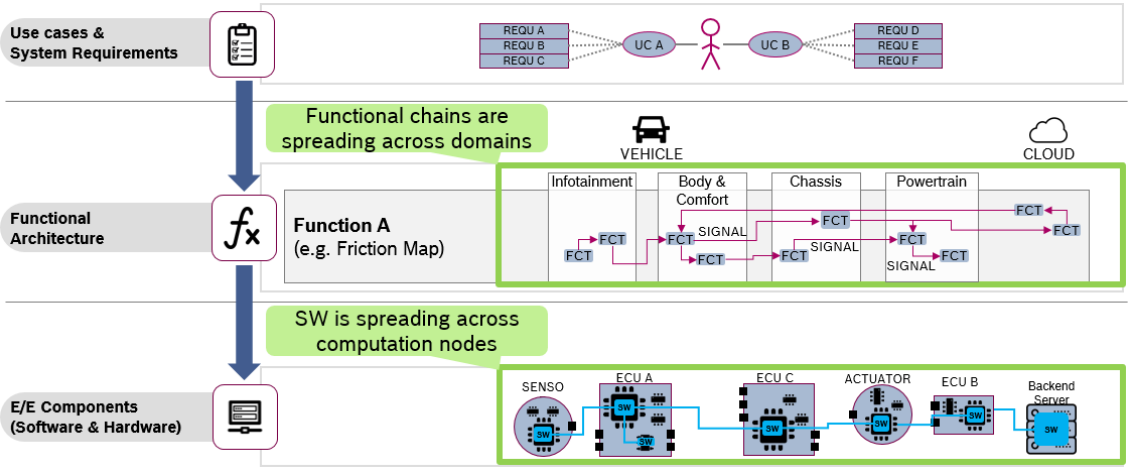

Abb. 3: Trend zu cross-Domänen SW-Architektur für hoch performante Fahrzeugrechner.

6. Einflüsse der Konnektivität

Im obigen Beispiel wurde ein zusätzlicher Aspekt bereits angesprochen. Durch die Konnektivität mit der Cloud entstehen Datenströme ins Fahrzeug und aus ihm heraus, die im klassischen µ-Controller-Umfeld nicht zu bearbeiten waren. Die Bandbreiten der bisherigen Kommunikationsnetzwerke im Fahrzeug (CAN, LIN, FlexRay) sind für den entstehenden Kommunikationsbedarf nicht mehr ausreichend. Aus diesem Grund wird das Ethernet (aktuell 1 GBit/s) in Fahrzeugen eingeführt. Die Erweiterung auf 10 GBit/s Ethernet ist ebenfalls in Planung.

Diese Bandbreitenerhöhung ermöglicht eine enorme Vielfalt neuer Funktionalitäten im und um das Fahrzeug herum. Aber auch hier gilt, dass sich die SW-Architekturen hierfür grundlegend ändern werden. Die bisherige Kommunikation über die zuvor genannten statischen und damit deterministischen Kommunikationsprotokolle ist vorteilhaft bei Betrachtungen zur Funktionalen Sicherheit gemäß ISO26262. Mit der Einführung des Ethernets im Fahrzeug werden Service-orientierte Kommunikationslösungen realisiert werden. Funktionen, die bisher eng an die jeweiligen Kommunikationsschnittstellen angebunden waren, werden zukünftig Daten über im Fahrzeug vorhandene Dienste empfangen bzw. Informationen über bereitgestellte Dienste zur Verfügung stellen. Die verfügbaren Dienste können sich zur Laufzeit ändern bzw. mit unterschiedlichen Qualitäten ausgestattet sein. Damit ergeben sich völlig neue Ansätze für Fallback-Lösungen.

7. Beispiel Verkehrsschildererkennung

Als Beispiel für eine Service-orientierte Kommunikation nehmen wir die automatische Schildererkennung.
Wir gehen davon aus, dass eine im Fahrzeug vorhandene Kamera die Verkehrssituation und mit ihr auch Verkehrszeichen aufnimmt. Die Software im Fahrzeug detektiert und erkennt die Verkehrsschilder und stellt diese im Navigationsdisplay dar. Möglicherweise wird der Fahrer informiert, falls seine aktuelle Geschwindigkeit zu hoch ist.
Gehen wir nun davon aus, dass aus irgendeinem Grund (verschmutze Kamera, technischer Defekt der Kamera etc.) die Schilderinformation nicht mehr zur Verfügung steht. Anstatt nun keine Information in der Anzeige mehr bereitzustellen, können über eine Service-orientierte Architektur anstelle der fehlenden Kamerainformation von einem Cloud-Dienst die Daten bzw. die lokalen Schilderinformationen bereitgestellt werden. Diese sind sicherlich von geringerer Qualität, da die Schilderinformationen nicht überall tagesaktuell vorliegen.
Selbst wenn die Verbindung zur Cloud nun zusammenbricht, besteht die Möglichkeit, statische Schilderinformation von dem im Fahrzeug lokal vorhandenen Navigationssystem zu verwenden.
SW-Architekturen, die auf Quality-of-Service-Ansätzen aufbauen, sind für solche Lösungen wesentlich flexibler aufgestellt, da die Funktionen nicht mehr berücksichtigen müssen, aus welcher Quelle (auch nicht von welchem Rechner im Fahrzeug) die Daten stammen. Lediglich die Qualität der Daten muss berücksichtigt werden. Damit werden diese Funktionen auch leichter auf andere Rechner im Fahrzeug verschiebbar.
Eine wesentliche Rolle spielt in diesem Zusammenhang der sich aktuell in der Entwicklung befindliche Standard AUTOSAR Adaptive, der Standards für Schnittstellen für µ-Prozessor-basierte Basissoftware festlegt, die in Anwendungen mit besonderen Anforderungen bzgl. Funktionaler Sicherheit (ISO26262 mit zugehörigen ASIL-Level) zum Einsatz kommen.

8. Einflüsse von Fremdsoftware auf die Basis-SW

In den vorhergehenden Abschnitten sind wir auf einen wesentlichen Aspekt der Software für zukünftige Fahrzeugrechner noch nicht eingegangen. Wer erstellt eigentlich die Software für die µ-Prozessor-basierten Fahrzeugrechner?
Wie bereits oben erwähnt steigt der Umfang der in den Fahrzeugrechnern realisierten Software um mehrere Zehnerpotenzen den Umfang der µ-Controller-Software.
Bei letzterer entwickelt in der Regel der Tier-1-Zulieferer die Software für die entsprechende Hardware. Dabei kommen auch zugekaufte Teile, wie Automotive-spezifische Realzeit-Betriebssysteme und Basis-SW-Stacks (z.B. AUTOSAR für µ-Controller), zum Einsatz. Auch OEMs stellen eigenen Funktionalitäten bei, die in die finale Software integriert werden. Im Prinzip gilt hierbei, dass alle Softwareanteile direkt aus der Automobilindustrie und ihren Zulieferern stammen und somit mit den besonderen Anforderungen bzgl. Funktionaler Sicherheit vertraut sind.
Große Zulieferer wie Bosch sind zudem in der Lage, Komplettsysteme aus einer Hand zu entwickeln und zu liefern, also mit einer Entwicklungstiefe von 100%: das Realzeitbetriebssystem von der Bosch-Tochter ETAS, die Basis-SW als Inhouse-

Lösung einer Querschnittsabteilung für SW (auch am Markt über ETAS vertrieben) und die Anwendungssoftware durch den beauftragten Geschäftsbereich.
Diese Situation ändert sich grundlegend beim Einsatz von µ-Prozessoren. Zum einen werden POSIX-konforme Betriebssysteme eingesetzt, die nicht durch deterministische Realzeitbetriebssysteme realisiert werden. Des Weiteren werden auf Fahrzeugrechnern Hypervisoren verwendet, um Virtualisierungsansätze zur Separierung von Anwendungen mit unterschiedlichen ASIL-Levels auf einem Rechner zu ermöglichen. Sowohl die POSIX-Betriebssysteme als auch die Hypervisoren kommen nicht von klassischen Automobilzulieferern. Das Marksegment dieser Softwarearten befindet sich aktuell in rasanter Entwicklung. Umso wichtiger ist es, dass die auf diesen Systemen aufbauenden Automotive-spezifischen Lösungen ausreichend gekapselt werden. Eine durchgängige Schichtenarchitektur für Fahrzeugrechner-Software wird zwingend erforderlich.

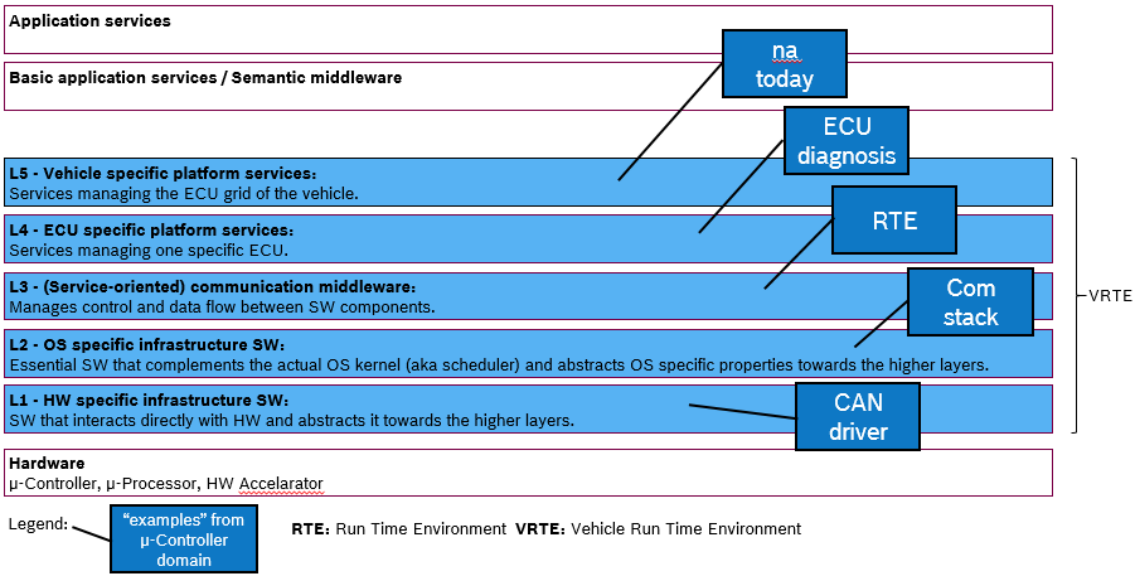

Abb. 4: Schichtenarchitektur des Bosch Vehicle Run-Time Environment.

Abb. *4* zeigt die Schichtenarchitektur des sogenannten Vehicle Run-Time Environment (VRTE), welches in zukünftigen Fahrzeugrechnern verwendet wird. Die fünf Ebenen des VRTE stellen sukzessive Kapselungen

1. der HW-abhängigen Software,
2. der Betriebssystem-spezifischen Software
3. der verwendeten Kommunikationskanäle (Abstraktion von CAN, FlexRay, Ethernet etc.)
4. Computer-spezifische Dienste (z.B. Rechnerdiagnose) und
5. Fahrzeug-spezifische Dienste, die es auf heutigen µ-Controller ECUs nicht gibt (z.B. Life-Cycle Management).

Es ist hierbei anzumerken, dass die abgebildete Software keine monolithische Struktur darstellt, sondern ein System aus unterschiedlichen Softwarekomponenten

unterschiedlichster Hersteller ist, die in einem gemeinsamen Framework integriert und für die Automobilindustrie seriengehärtet wird.
Besondere Herausforderungen ergeben sich dabei für die Software-Integration und deren Absicherung, da neben nicht quelloffenen „3rd Party“ Komponenten auch Open Source Software (z.B. in verwendeten Betriebssystemen) eingesetzt werden müssen.
Auch diese Aspekte müssen im Entwurf der SW-Architekturen berücksichtigt werden.

9. Einflüsse unterschiedlicher Arten von Software

In dem bisher Gesagten wurde stillschweigend davon ausgegangen, dass sich bei allem Komplexitätszuwachs, den die neuen Fahrzeugrechner mit sich bringen, an der eigentlichen Entwicklungsmethodik der Software-Ingenieure und Informatiker nichts ändern wird. Die Realität sieht jedoch völlig anders aus.
Die Software für „Deeply Embedded ECUs“ ist durch die in Abb. *5* unten links angegebenen Elemente charakterisiert. Insbesondere ist zu erwähnen, dass die Entwicklungsprozesse stark durch das in der Automobilindustrie etablierte V-Model und Automotive SPICE (ASPICE) geprägt sind.

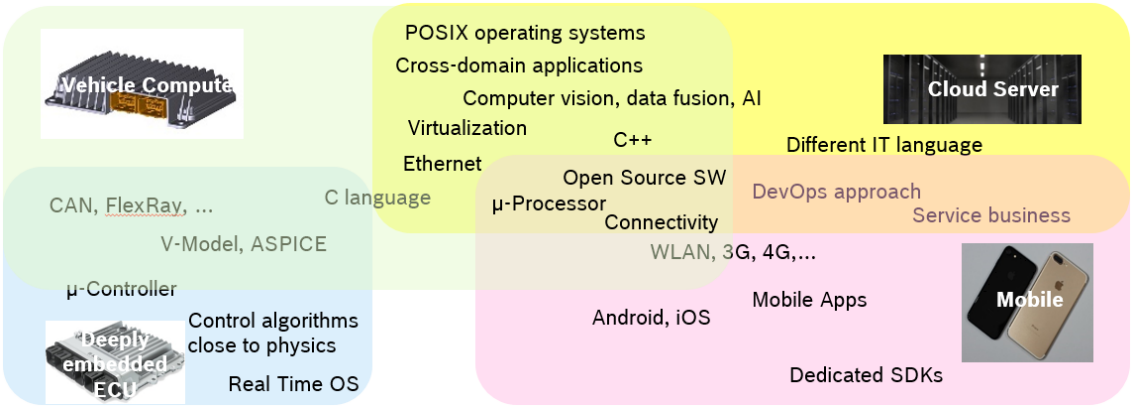

Abb. 5: SW ≠ SW - Unterschiedliche Arten von Software dringen in die Automotive-Domäne ein.

Mit den Fahrzeugrechnern werden neue Softwarearten eingeführt. Neben Betriebssystemen, die aus der IT-Welt stammen, kommen andere Programmiersprachen und Programmierparadigmen in die Automobilindustrie. Die klassische Lastenheft-getriebene, V-Model-basierte Entwicklungsmethodik trifft nun auf die agile, durch DevOps getriebene Entwicklungsansätze, die aus dem Umfeld der Internetentwicklungsplattformen kommen. In Abb. *5* wird deutlich, dass die Fahrzeugrechner softwareseitig eine deutlich stärkere Überlappung mit den IT-Softwaremethoden besitzen. Zusätzlich drängen zunehmend Anwendungen aus den Smartphone-Eco-Systemen ins Fahrzeug (Abb. *5* unten rechts).
Zusammen mit dem in Abb. *3* dargestelltem Trend zu Cross-Domänen-Architekturen ergibt sich die Notwendigkeit, SW-Architekturen übergreifend über unterschiedliche SW-Plattformen und Zielsysteme konsistent zu beschreiben und zu pflegen. Hier liegt eine besondere Herausforderung für die Automobilindustrie, deren Produkte im

Vergleich zu den schnelllebigen Consumerprodukten der IT-Welt deutlich längere Lebenszyklen durchlaufen und in der Regel deutlich höhere Anforderungen hinsichtlich Sicherheit erfüllen müssen.
Als Konsequenz des Gesagten ergibt sich, dass die Automobilindustrie eine deutlich größere Anzahl an Informatikern mit einer Vielzahl unterschiedlicher Softwarefähigkeiten benötigt, und das zusätzlich zu den bereits vorhandenen Ingenieuren der regelungsbasierten embedded Welt.

10. Bedeutung der Funktionalen Sicherheit und IT-Sicherheit

In den vorherigen Abschnitten wurde ein wichtiger Themenkomplex und Architekturtreiber nur am Rande berührt - die Kombination aus Funktionaler Sicherheit und IT-Sicherheit. (Die englische Sprache unterscheidet den Sicherheitsbegriff deutlich besser durch die Worte Safety und Security.) Die Kombination dieser Anforderungen ist nach unserem Wissen in keinem anderen Industriezweig so ausgeprägt. Selbst in der Luftfahrt sowie auch im Schienenverkehr stellt sich eine andere Situation dar. In beiden genannten Industriezweigen besitzt nur geschultes Personal (z.B. Piloten, Lokführer, etc.) Zugang zu den zentralen Systemen der Fortbewegungsmittel. Im privaten PKW-Umfeld hat jedoch jeder Autobesitzer im Prinzip den physikalischen Zugang zu den Systemen. Des Weiteren ergibt sich durch die zunehmenden Connectivity-Lösungen und Cloud-basierten Dienste eine größer werdende Angriffsfläche für Hacker.
Hiermit erhöht sich deutlich die Gefahr für einen unberechtigten Zugang in die Computersysteme des Fahrzeuges, die es durch geeignete Security-Maßnahmen abzuwehren gilt.
Auch die Lösungsansätze zur Funktionalen Sicherheit sehen in der Luftfahrt und im Schienenverkehr anders aus als im PKW-Sektor. Aufgrund der vergleichsweise geringen Volumina an Flugzeugen und Schienenfahrzeugen und einem deutlich geringeren Commodity-Druck kommen in den erstgenannten Bereichen teilweise komplett redundante Systeme zum Einsatz. Außerdem werden Flugzeuge und Schienenfahrzeuge mit einer deutlich höheren Frequenz auf mögliche Fehler überprüft als PKW. Diesem Aspekt müssen SW-Architekturen durch entsprechende Safety- und Sicherheits-Konzepte Rechnung tragen.
Bei Fahrzeugrechnern mit Applikationen unterschiedlicher ASIL-Level ist eine Separierung bzw. Virtualisierung der Anwendungsdomänen unter Verwendung von Hypervisor-Technologien erforderlich. Dabei nimmt der Hypervisor eine zentrale Rolle im SW-System ein, da er die Hoheit über Rechnerressourcen besitzt, deren Allokation zu den unterschiedlichen Anwendungsfunktionen sicherstellt und diese überwacht.
Mit dieser zentralen Position des Hypervisors im System ergibt sich eine weitere zentrale Angriffsfläche für SW-Hacker. Aus diesem Grund müssen eingesetzte Hypervisoren in besonderem Maße auf potentielle Security-Lücken überprüft werden.

11. SW Architekturtreiber und ihre Dokumentation

In den vorhergehenden Abschnitten haben wir eine Reihe von SW-Architekturtreibern beschrieben, die mit der Einführung der Fahrzeugrechner massiv an Bedeutung gewinnen. Mit der steigenden Komplexität und Öffnung der SW-Systeme muss ein viel höheres Gewicht auf wohldurchdachte und mit allen relevanten Stakeholdern abgestimmte SW-Architekturen gelegt werden.

Die Fokussierung auf rein funktionalen SW-Aspekte ist bei weitem nicht ausreichend. Abb. *6* gibt eine Übersicht der wesentlichen domänenunabhängigen Architekturtreiber für µ-Prozessor-basierte Fahrzeugrechner.

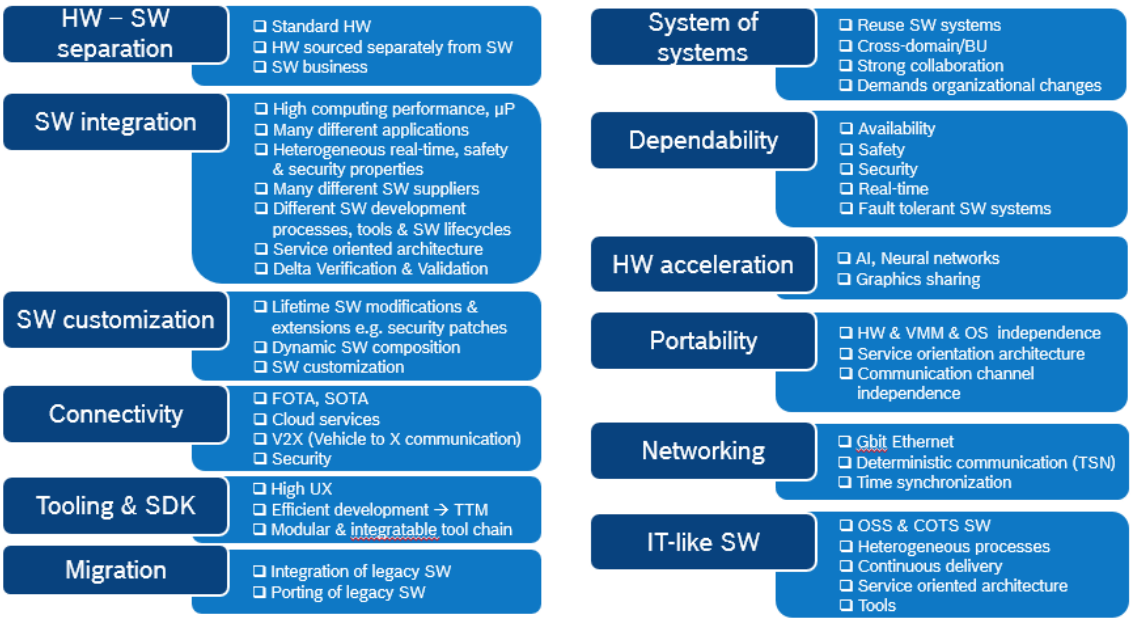

Abb. 6: Architekturtreiber für µ-Prozessor-basierte Fahrzeugcomputer

Im Rahmen des Architekturentwurfes muss sichergestellt werden, dass die nicht-funktionalen Treiber ausreichend berücksichtigt und nachvollziehbar dokumentiert werden. Insbesondere in einem zunehmend heterogenen und komplexer werdenden SW-Umfeld ist ein verstärkter Fokus auf SW-Architekturen von entscheidender Bedeutung. Ein Startpunkt für den Architekturentwurf bietet der als arc42 bekannte Ansatz, welcher eine Strukturierungshilfe für SW-Architekturen darstellt und folgende Kapitel bzw. Architektursichten umfasst:

1. **Einführung und Ziele:**
   Wichtige Einflussfaktoren bzw. Kräfte, die auf die SW-Architektur einwirken. Insbesondere sind die Geschäftsziele zu benennen, die einen wesentlichen Einfluss auf die SW-Architektur haben.
2. **Randbedingungen:**
   Technische und organisatorische Randbedingungen, Konventionen, Tools, Stakeholder etc.
3. **Kontextabgrenzung:**
   Fachlicher und technischer Kontext, externe Schnittstellen etc.
4. **Lösungsstrategie:**
   Überblick über Architektur-relevante, grundlegende Entscheidungen und Lösungsansätze (z.B. Verwendung zugekaufter SW versus eigener Programmierung)

5. **Komponentensicht (Bausteinsicht):**
   Überblicksbild der ersten Ebenen (hierarchisch) des zu entwerfenden Systems, inkl. Zusammenhänge und Abhängigkeiten der Bausteine untereinander
6. **Dynamische Laufzeitsicht:**
   Beschreibung (z.B. als Laufzeitszenarien inklusive Latenzzeiten) der Bausteine, wie sie sich während der Laufzeit verhalten (d.h. Prozesse, Tasks, Aktivitäten, Interruptaufrufe etc.).
7. **Verteilungssicht:**
   Oberste Ebene von Hardware und technischer Infrastruktur mit der Beschreibung, wo welche Teilfunktion abzulaufen hat. Dies ist besonders wichtig für verteilte Anwendungen, z.B. SW-Lösungen, die nicht auf einen Prozessor beschränkt sind (Multi-Prozessorlösungen, Connectivity-Lösungen, Zuordnung zu unterschiedlichen virtuellen Maschinen etc.)
8. **Querschnittliche Konzepte**:
   Umfangreicher Abschnitt, der technische Konzepte beschreibt (z.B. Ablaufsteuerung, Fehlerbehandlung, Logging, Parallelisierung, Bedienoberflächen (UX) und vieles mehr)
9. **Entwurfsentscheidungen:**
   Wesentliche Entwurfsentscheidungen mit Gründen
10. **Qualitätsszenarien:**
    Beschreibung von Grenz- bzw. Stressszenarien und dem erwarteten Verhalten des Systems (z.B. unter definiert hoher Belastung von Kommunikationsbussen)
11. **Risiken und technische Schulden**
    Dokumentiert Risiken aus SW-Architektursicht, die z.B. auch in das Risikomanagement des Projekts übernommen werden müssen
12. **Glossar**

In der praktischen Umsetzung der Architekturbeschreibung ist es eine große Herausforderung, die unterschiedlichen Sichten aktuell und konsistent zu halten, da diese oft über mehrere Dokumente und Tools verteilt erstellt werden. Aus diesem Grund wurde in Bosch eine entsprechendes arc42-Profile für das SW-Architekturtool Rhapsody der Firma IBM erstellt. Das Rhapsody-Profile unterstützt Architekten im Entwurf und Pflege konsistenter Architekturen und ihren Sichten.
Das Profil stellen wir als Beigabe „as it is“ der SW-Architekten-Community auf den arc42 Internetseiten http://arc42.org/download#ibm-rhapsody-format zum Download zur Verfügung.

12. Zusammenfassung

In den vorhergehenden Kapiteln sind wir auf die disruptiven Änderungen in der SW-Entwicklung der Automobilindustrie eingegangen. Mit dem Einzug von µ-Prozessor-basierten Rechnerplattformen im Fahrzeug findet eine Leistungsexplosion hinsichtlich Speicher, Rechenleistung und Konnektivität statt, die völlig neue Lösungsräume für bisher nicht betrachtete Problemstellungen bietet.

Diese Leistungsexplosion geht einher mit

- einem extremen Anstieg der SW-Komplexität hinsichtlich unterschiedlicher Arten von SW,

- unterschiedlicher aufeinandertreffender Entwicklungsmethodiken,
- bei einem gleichzeitigen Anstieg der beteiligten Softwarelieferanten,
- deren SW-Komponenten in Fahrzeugrechnern integriert werden müssen und
- dabei Anforderungen zur Funktionalen Sicherheit und Security abdecken.

Diese Situation kann nur mit einem starken Fokus auf SW-Architekturen und deren relevante Architektursichten beherrscht werden.
Dies ist die Herausforderung, der sich die gesamte Automobilindustrie aktuell stellt.

Abkürzungsverzeichnis

HW	Hardware
SW	Software
ECU	Electronic Control Unit (Elektronisches Steuergerät)
OEM	Original Equipment Manufacturer (in unserem Kontext die Automobilhersteller)
SOP	Start of Production
kDMIPS	„kilo Dhrystone Million Instructions Per Second": auf Ganzzahlarithmetik aufbauender Benchmark zum Vergleich von Rechnerleistungen
AI	Artificial Intelligence oder auch Künstliche Intelligenz
GPU	Graphical Processor Unit, die als Prozessorplattform für AI-Algorithmen genutzt wird
ESP	Electronic Stability Program
ASPICE	Automotive SPICE: Aus dem ISO Standard ISO/IEC 15504 (SPICE) abgeleitetes Automotive-spezifisches Reifegradmodel zur Bewertung der Steuergeräteentwicklung in der Automobilindustrie
POSIX	Portable Operating System Interface: Ein Satz von IEEE Standards der Application Programming Interface (API) für POSIX kompatible Betriesbssysteme definiert
arc42	Frei verfügbarer Strukturierungsansatz für Architekturen

Literaturverzeichnis

[1] Conway, Melvin E. (1968): How do committees invent?
http://www.melconway.com/Home/Conways_Law.html
[2] IEEE Standards Association: POSIX – Austin Joint Working Group
http://standards.ieee.org/develop/wg/POSIX.html
[3] Dr. Gernot Starke, Dr. Peter Hruschka: arc42 Ressourcen für Softwarearchitekten
http://www.arc42.de/template/index.html

Autoren:

Dr.-Ing. Detlef Zerfowski (Vice President, Robert Bosch GmbH)

Nach seiner Promotion in der Informatik (TH Karlsruhe) im Bereich medizinischer Signalverarbeitung wechselte Herr Zerfowski zur Robert Bosch GmbH. Seine 18-jährige Automotive-Erfahrung deckt die Embedded SW-Entwicklung in verschiedenen Domänen (Body Electronics, Bremssysteme, Park Assistenten) ab.
Von 2009-2012 baute Herr Zerfowski für den Geschäftsbereich Automotive Electronic einen Entwicklungsbereich in Indien auf. Anschließend übernahm er als Managing Director die Leitung des 100% Bosch-Tochterunternehmens ETAS Automotive India Prv. Ltd.
Seit März 2015 ist Herr Zerfowski im Corporate Sector Automotive System Integration für die strategische Ausrichtung der Software-Themen im Automotive-Bereich der Firma Bosch verantwortlich.
Email: Detlef.Zerfowski@de.bosch.com

Niranjan SK (Senior SW Architect and Product Manager, Robert Bosch GmbH)

Nach seinem Studium in Instrumentation Technology an der VTU University, India, wechselte Herr Niranjan SK in 2005 zu Robert Bosch Engineering and Business Solution Prv. Ltd. India (RBEI). Für 12 Jahre sammelte Erfahrung in der embedded SW-Entwicklung, insbesondere im Themenkomplex AUTOSAR und dessen Anwendung in den unterschiedlichen Anwendungsdomänen. Von 2013-2015 war Herr Niranjan SK Product Manager für AUTOSAR Produkte aus dem Haus Bosch. Seit März 2016 arbeitet er im Corporate Sector Automotive System Integration und arbeitet an der unternehmensweiten Etablierung von SW Architekturvorgehensweisen.
Email: Niranjan.SK@in.bosch.com