Adaptive Software für sicherheitskritische Funktionen in Batterie-elektrischen Fahrzeugen

Safe Adaptive Software for Fully Electric Vehicles

Delphi Deutschland GmbH, Wuppertal: Thorsten Rosenthal thorsten.rosenthal@delphi.com
Timo Feismann, timo.feismann@delphi.com
Fraunhofer ESK, München: Philipp Schleiß philipp.schleiss@esk.fraunhofer.de
Gereon Weiß, gereon.weiss@esk.fraunhofer.de
Siemens AG, München: Cornel Klein, cornel.klein@siemens.com

Kurzfassung

Der Einzug von immer mehr Assistenz- und (teil-) autonomen Systemen in heutige und insbesondere in zukünftige Fahrzeuge verlangt höhere Sicherheitsanforderungen bis hin zu fehlertoleranten Systemen. Eine adaptive Software-Architektur für sicherheitskritische Funktionen hilft diese Fehlertoleranz in einem Fahrzeug robust, kostengünstig und auch energieeffizient umzusetzen. Diese Software bedient sich dabei der ohnehin vorhandenen Steuergeräte im Fahrzeug und ermöglicht dadurch eine Redundanz ohne die Einbringung von zusätzlichen Steuergeräten. Durch die Konformität zu AUTOSAR ist das Konzept universell auf jeder automotive-tauglichen Hardware realisierbar.

Abstract

More and more driver assistance and (partly-) autonomous systems are entering today's and especially tomorrow's vehicles. These systems have heightened safety requirements up to being fail-operational. A safe adaptive software for safety critical functions helps to realize fail-operation ability in a vehicle in a robust, cost and energy efficient way. This software uses the already available electronic control units to create a redundancy without bringing in additional controllers. By using AUTOSAR this concept can be realized on any automotive compliant hardware.

1 Einführung

In Zukunft wird das Führen eines Fahrzeugs für den Fahrer zunehmend durch Assistenzsysteme unterstützt bzw. sogar übernommen. Schon heute sind umfangreiche Funktionen im Fahrzeug integriert, die das Fahren sicherer machen (z.B. das elektronische Stabilitätsprogramm ESP oder etwa ein automatischer Notbremsassistent). Diese Funktionen dürfen im Falle einer Fehlfunktion keinesfalls zu einer kritischen Situation führen und müssen korrekt arbeiten. Im Fehlerfall können solche Zusatzfunktionen aber einfach abgeschaltet werden, d.h. sie werden als fail-silent bezeichnet. Darüber hinaus ist der Fahrer jederzeit in das Verkehrsgeschehen eingebunden und hat die Kontrolle über sein Fahrzeug.

Zukünftig hält eine stärkere Automatisierung Einzug, mit Funktionen, die auch in die Lenkung, Beschleunigung und den Bremsvorgang eingreifen (z.B. der Stauassistent oder das pilotierte Fahren auf der Autobahn). Bei diesem Schritt von einer Teilautomatisierung zur Hochautomatisierung (vgl. [1]), überwacht der Fahrer das System nicht mehr dauerhaft. Sobald der Fahrer aber nicht mehr dauerhaft das Fahrzeug und die Verkehrssituation beobachten muss, kann bei einer Fehlfunktion auch nicht sichergestellt werden, dass der Fahrer unmittelbar die Kontrolle über das Fahrzeug übernehmen kann. Das bedeutet, die elektronischen Systeme, die das Fahrzeug lenken, bremsen und beschleunigen, müssen auch im Fehlerfall noch so lange weiter funktionieren (unter Umständen auch mit Komforteinbußen) bis der Fahrer wieder die Kontrolle übernommen hat und aktiv am Verkehrsgeschehen teilnimmt. Dadurch müssen diese Systeme im Gegensatz zu heutigen Systemen im Fehlerfall weiter funktionieren und dürfen sich nicht einfach abschalten, d.h. sie müssen fail-operational sein.

Hier setzt das EU Projekt „Safe Adaptive Software for Fully Electric Vehicles", kurz SafeAdapt, an und zeigt eine Lösung auf, wie ein solches fehlertolerantes Fahrzeugsystem aufgebaut werden kann, ohne die umfangreichen – und damit teuren – mehrfach redundanten Systeme aus der Luftfahrt zu kopieren [2]. Hierfür wird eine neue adaptive Elektrik/Elektronik (E/E) Fahrzeugarchitektur erarbeitet, die sicherer, zuverlässiger und kostengünstiger ist als heute übliche Architekturen. Zentraler Punkt ist dabei die in SafeAdapt entwickelte Software Komponente Safe Adaptation Platform Core (SAPC). Durch diese wird das System in die Lage versetzt, Funktionen adaptiv während des laufenden Betriebs zu de-/aktivieren oder auf Steuergeräte zu verlagern. So können die Funktionen eines fehlerhaften Steuergeräts von einem anderen übernommen werden, um den fail-operational Betrieb aufrechtzuerhalten. Die Fähigkeit sich zur Laufzeit zu adaptieren ist für solch eine neuartige Architektur eine entscheidende Eigenschaft. Nur so kann beispielsweise die Fahrzeugfunktionalität im Fehlerfall effizient aufrechterhalten werden

Im Folgenden werden die Projektziele und adressierten Szenarien vorgestellt. Ferner werden die Herausforderungen für die Soft- und Hardware, die mit der Lösung verbunden sind, erläutert. Anschließend werden die Fallstudien mit verschiedenen Demonstratoren vorgestellt, anhand derer die Konzepte evaluiert und verifiziert werden.

1.1 Ziele

Das SafeAdapt Projekt mit 9 europäischen Konsortialpartnern ist im Juli 2013 gestartet und wird im Juni 2016 enden [2]. Im Wesentlichen werden mit dem SafeAdapt Projekt vier Ziele verfolgt:

1. Verringerung der Entwicklungskosten durch die Bereitstellung eines generischen Fehler- und Erweiterungsmechanismus basierend auf dem SAPC. Dadurch werden die Entwicklungszeiten und Testkosten spürbar reduziert.
2. Erhöhung der (funktionalen) Sicherheit durch die Fähigkeit, komplexe Fehlerszenarien durch den SAPC zu behandeln und dem System eine Fehlertoleranz zu geben (siehe Bild 1). In der genannten Abbildung wird eine Brake-by-Wire Funktion (BBW) von einem Steuergerät auf ein anderes verlagert.
3. Die Materialkosten einer mehrfach redundanten Auslegung (wie im Flugzeugbau) sind verhältnismäßig hoch. Durch Nutzung der ohnehin vorhandenen Steuergeräte können diese Kosten spürbar reduziert werden. Beispielsweise können im Fehlerfall Funktionen eines defekten Steuergeräts auf ein anderes Steuergerät verlagert werden, indem dort beispielsweise Komfort-Funktionen abgeschaltet werden.
4. Durch geschickte Verlagerung der Funktionen auf die vorhandenen Steuergeräte kann unter bestimmten Bedingungen ein Steuergerät abgeschaltet werden. Das ermöglicht dann eine höhere Energieeffizienz.

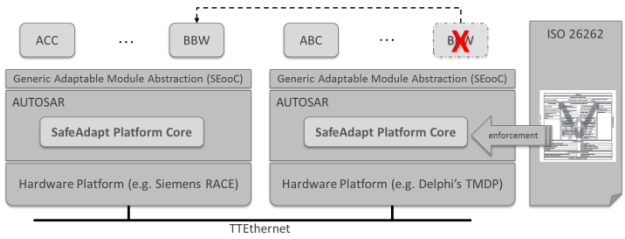

Bild 1 Verlagerung der Brake-by-Wire Funktion von einem Steuergerät auf ein anderes

2 Herausforderung Adaptivität

Innerhalb eines Steuergerätes gibt es eine Reihe von Möglichkeiten, um in Abhängigkeit von der verbauten Hardware auf Fehler zu reagieren. Ein Beispiel hierfür ist ein fehlerhafter Speicherbereich. Im Falle eines Hypervisors kann man auf eine Partition ausweichen, die einem anderen Speicherbereich zugewiesen ist. In einem einfacheren System ohne einen Hypervisor (dafür aber mit einer Speicherüberwachung, MPU) kann man die Bereichsschranken für den Speicher so neu konfigurieren, dass der defekte Bereich ebenfalls nicht mehr genutzt wird. Diese Mechanismen sind heute bekannte und bereits verwendete Verfahren, so dass hier nicht weiter darauf eingegangen wird. Interessanter wird der Fall, wenn ein Fehlerbild angenommen wird, das ein Verschieben der Software Komponente auf eine andere ECU (Electronic Control Unit) notwendig macht. Für die sichere Adaption ist dabei eine Reihe von Randbedingungen einzuhalten. Es ergeben sich folgende Adaptionsszenarien.

2.1 Adaptionsszenarien

2.1.1 Hot Standby

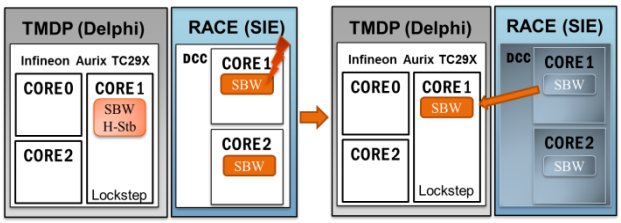

Bild 2: SafeAdapt Fallbeispiel für eine Hot Standby Funktion

Als Beispiel dient hier die Funktion „Steer-by-Wire“ (SBW), also eine elektrische Lenkung ohne mechanische Verbindung zwischen Lenkrad und Rädern (vgl. Bild 2). Für eine solche Funktion ist es selbsterklärend, dass das System fehlertolerant sein muss, also im Fehlerfall weiter funktionieren muss. Ferner sind die Fehlertoleranzzeiten für eine solche Funktion recht kurz und werden durch den maximalen Zeitraum bestimmt, in dem ein Fehler auftritt und das Auto unter Kontrolle bleibt, z.B. das Fahrzeug bei Ausfall der Lenkung beim Fahren auf der Autobahn stabil in der Fahrspur bleibt. Für SafeAdapt ist diese Zeit mit 50ms festgelegt worden, d.h. innerhalb dieser Zeit muss der Fehler erkannt und die Funktionalität durch eine sekundäre Kontrolleinheit wieder hergestellt worden sein. Um diese Anforderung an das Zeitverhalten zu erfüllen, wird eine sogenannte „Hot Standby“ Funktion genutzt, d.h. eine Funktion, die auf dem Backup System in vollem Umfang ausgeführt wird, deren Ausgangstreiber jedoch inaktiv sind. Damit nimmt diese Funktion im normalen Betrieb keinen Einfluss auf die Aktuatoren. Sollte ein Fehler im System ein Aktivieren der Funktion auf dem Backup Gerät notwendig machen, so kann dieses Gerät die betroffene Funktion ohne Zeitverlust durch Aktivieren der Ausgänge übernehmen. Das ursprünglich verantwortliche Steuergerät wird zeitgleich seine Ausgänge deaktivieren, indem die direkte physikalische Ansteuerung der Aktuatoren unterbunden und der Datenaustausch mit Aktuatoren verhindert wird, die über ein Netzwerk erreichbar sind.

Solche Mechanismen sollen dabei auch für heterogene Systeme funktionieren, d.h. es ist zunächst einmal gleichgültig auf welcher Architektur die Applikation ausgeführt wird. Es kann allerdings im Falle einer heterogenen Architektur aufgrund der zur Verfügung stehenden Rechenleistung notwendig werden, die Applikation auf einem Backup-Rechner in einem funktional reduzierten Modus zu aktivieren, um im Fehlerfall die Verfügbarkeit der essentiellen Funktionen zu garantieren. Komfortmerkmale, wie zum Beispiel Assistenzfunktionen im Falle der Lenkung, könnten wegfallen.

2.1.2 Cold Standby

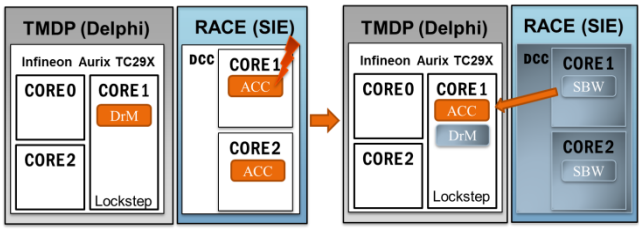

Bild 3: SafeAdapt Fallbeispiel für eine Cold Standby Funktion

Auf den ersten Blick (s. Bild 3) sieht diese Lösung nicht gravierend anders aus als das Szenario in Kapitel 2.1.1, der entscheidende Unterschied soll jedoch im Folgenden gezeigt werden.
Zunächst aber zu dem Beispiel selbst. Wir haben hier wieder ein heterogenes System mit einer ECU mit einer Fahrer-Überwachung (Driver Monitoring, DrM) und eine ECU, auf der sich eine adaptive Geschwindigkeitsregelanlage (Adaptive Cruise Control, ACC) befindet. Im Normalbetrieb sind diese ECUs also durchaus verschiedenen Domänen zugeordnet. Da das ACC-System für das autonome Fahren als sicherheitskritisch einzustufen ist, wird auch in diesem Fall eine Fehlertoleranz gefordert. Zu diesem Zwecke ist die Funktion auf dem ersten Steuergerät kompiliert und im Speicher hinterlegt. Jedoch wird die Funktion im Normalbetrieb nicht ausgeführt, nimmt also weder Rechenzeit noch Speicher in Anspruch. Diese Ressourcen stehen also für andere Funktionen zur Verfügung. Eine solche Verwendung wird „Cold Standby“ genannt, da im Normalbetrieb keinerlei Aktivitäten mit ihr verbunden sind.
Nach einem Fehlerfall muss der zeitliche Ablauf der Softwareausführung auf dem Backup-Steuergerät neu geplant werden. Dazu können Funktionen deaktiviert werden, die als nicht sicherheitskritisch eigestuft sind, um Speicherraum und Rechenkapazität für kritische Funktionen zu schaffen, die dann wiederum zeitgleich mit der Deaktivierung auf gestartet werden. Dies wird durch das Betriebssystem gesteuert, wobei die zu aktivierenden Funktionen zunächst geladen und initialisiert werden müssen. Dabei geht Zeit verloren, weshalb dieses Verfahren nur für weniger zeitkritische Anwendungen in Frage kommt.
Ein Beispiel für dieses Verfahren ist es, ein Entertainment System, welches bereits über hohe Rechenleistung verfügt, als Backup-Rechner zu verwenden. Sollte dann ein kritischer Teil des autonomen Fahrens ausfallen, würde das Radio die Funktion übernehmen und das Fahrzeug sicher bis zum Stillstand weiterführen. Der Fahrer würde also neben einer Fehlermeldung im Anzeigeinstrument nur bemerken, dass er keine Musik mehr hört.

2.1.3 Repository Standby

Ein weiteres Szenario ist die Verlagerung einer Softwarefunktion aus einem Repository in ein Steuergerät. Im Wesentlichen unterscheidet sich dieses Szenario nicht von dem Fall in Kapitel 2.1.2 beschriebenen, allerdings ist die redundante Software nicht mehr von Anfang an auf dem Backup-Steuergerät vorhanden. Stattdessen befindet sich die Software auf einem beliebigen Steuergerät im Fahrzeugverbund. Im Fehlerfall muss die Software von diesem Repository heruntergeladen und auf dem ausgewählten Steuergerät gestartet werden. Ein Ziel von SafeAdapt ist es, dieses Szenario in einer TTEthernet-Umgebung zu testen und festzustellen, wo die Grenzen der Machbarkeit liegen. Der besondere Fokus liegt hierbei auf dem Timing, wie lange würde es dauern, bis eine solche Funktion wieder zur Verfügung steht.

3 Neue Flexible E/E-Architektur

Bisher lag der Fokus auf der Software Architektur und wie einzelne Software Komponenten im Fehlerfall verschoben werden. Damit dieser Ansatz lauffähig wird, muss eine Reihe von Voraussetzungen erfüllt sein, auf die im Folgenden eingegangen werden soll.

3.1 Hardware-Architektur

3.1.1 Universelle Rechenplattform

Im SafeAdapt Projekt kommen zwei verschiedene Hardware (HW) Plattformen zum Einsatz: Zum einen die TMDP (Trusted Multi Domain Platform) der Firma Delphi und zum anderen die RACE Plattform der Firma Siemens (siehe Bild 4). Beide Plattformen müssen in der Lage sein, die sicherheitskritische Software selbstständig auszuführen, sollten also nach ASIL D zertifiziert sein. Auf Systemebene wird damit ebenfalls ASIL D erreicht, nicht nur weil die ISO 26262 [3] in ihrer aktuellen Fassung keine höhere Einstufung vorsieht, sondern auch weil die Funktion nur auf einer ECU ausgeführt wird. Für diese ECU reicht es aus, wenn als Fehlerreaktion ein fail-silent Verhalten angenommen wird – insbesondere auch für die Kommunikation. Damit ist jede andere ECU im System in der Lage, den Ausfall zu erkennen und die in SafeAdapt entwickelten Gegenmaßnahmen einzuleiten.

3.1.2 Sensorik & Aktuatorik

Hier kommen wir noch einmal auf die beispielhafte Steer-by-Wire (SBW) Funktion zurück. Damit diese Technologie verwendet werden kann muss, wie gesagt, sichergestellt sein, dass kein einzelner Fehler in einem der beteiligten Elemente zu einem Totalausfall der Lenkung führen kann. Die Adaption der funktionalen Verteilung von Softwareapplikationen auf verschiedene Steuergeräte ist dabei ein wichtiger Teilaspekt. Bezüglich der Sensoren und Aktuatoren, lässt sich die geforderte funktionale Sicherheit allerdings so nicht erreichen. Eine Plausibilitätsprüfung der Sensordaten ist hier ein erster Schritt. Eine sichere Fehlererkennung aber lässt sich sowohl bei der Sensorik, als auch bei der Aktuatorik nur durch eine mehrfache Redundanz erreichen. Verständlich wird dieser Sachverhalt durch das folgende Beispiel:
In einem SBW System braucht es im einfachsten Falle einen Lenkwinkelsensor und einen Stellmotor für die Lenkmechanik. Nur ein Sensor wäre im Fehlerfall eine unzureichende Lösung, denn ohne Winkelvorgabe könnte nicht mehr gelenkt werden. Das Gleiche gilt für den Stellmotor. Also werden an dieser Stelle wenigstens zwei

Lenkwinkelsensoren und zwei Stellmotoren gebraucht, um im Fall eines Einfachfehlers operativ bleiben zu können. Für die Sensoren gibt es noch eine weitere Anforderung. Angenommen beide Sensoren liefern einen plausiblen, aber unterschiedlichen Lenkwinkel. In diesem Fall kann nicht erkannt werden, welcher Winkel korrekt oder welcher Sensor defekt ist. Um das zu verhindern, werden sogar drei Sensoren benötigt. Es kann dann durch das Steuergerät mittels einer Entscheidung zwei-aus-drei das korrekte Signal ermittelt werden.
Eine weitere Anforderung, um eine Fehlertoleranz zu realisieren, ist eine Redundanz in den Kommunikationspfaden vorzuhalten. Eine ausführlichere Beschreibung findet sich im folgenden Kapitel 3.1.3. An dieser Stelle sei schon erwähnt, dass natürlich nicht alle Sensoren oder Aktuatoren an nur einem Kommunikationspfad angeschlossen sein dürfen. Beide Aktuatoren müssen jeweils eine eigene Verbindung zu dem Fahrzeugnetzwerk haben. Für die dreifache Sensorik sollten auch zumindest zwei Pfade zu Verfügung stehen.

3.1.3 Netzwerkarchitektur

Der Aufbau des Netzwerks muss natürlich die Anforderungen an ein fehlertolerantes System erfüllen. Im Bild 4 ist der Aufbau im Demonstrationsfahrzeug als Beispiel dargestellt.

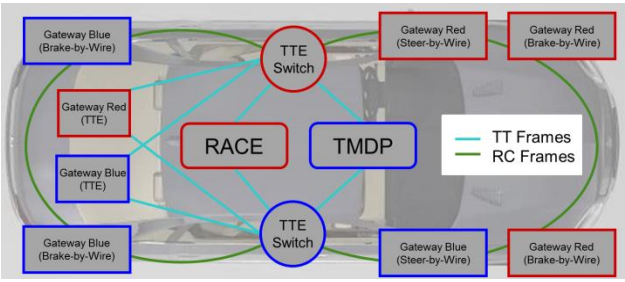

Bild 4: Netzwerkarchitektur im Versuchsfahrzeug

Die Farbumrandungen (rot oder blau) zeigen, dass entsprechende Module an unterschiedlichen Spannungsversorgungspfaden angeschlossen sind. Diese Vorgehensweise ist notwendig, damit beim Verlust einer Spannungsversorgung die Redundanz nicht ebenfalls ausfällt. Grundsätzlich bietet dieser Architekturansatz die Möglichkeit, Verkabelung einzusparen, da die Sensoren grundsätzlich über eine (multi-hop) Linientopologie erreichbar sind. Hier ist allerdings die Linientopologie aus Redundanzgründen wiederum zu einem Ring geschlossen, an welchem die Sensorik und Aktuatorik für das Steer- und Brake-by-Wire angeschlossen sind.
Die TTEthernet (TTE)-Switches verbinden in einer Doppel-Stern-Architektur die ECUs RACE und TMDP mit den TTE Gateways und der äußeren Ringstruktur. Die TTE Gateways sind als sogenannte Sync Master in der Netzwerkstruktur eingebunden (vgl. dazu [5]). Die Doppel-Stern-Architektur ermöglicht in jedem Fall noch eine Kommunikation aller Teilnehmer miteinander, falls eine der Netzwerkverbindungen ausfallen sollte.

3.2 Software-Architektur

3.2.1 SAPC mit AUTOSAR Architektur

Da in der Automotive Industrie viel Wert auf die Wiederverwendbarkeit von Software-Komponenten gelegt wird, hat sich der AUTOSAR Standard etabliert [4]. Im SafeAdapt Projekt wird daher auch der Safe Adaptation Platform Core (SAPC) als AUTOSAR Software-Komponente ausgelegt (siehe dazu Bild 5). Dadurch wird eine Hersteller-Unabhängigkeit ermöglicht, so dass der SAPC prinzipiell ohne Anpassungen in verschiedenen Steuergeräten mit unterschiedlicher Hardware laufen kann. Auch der Test- und damit Kostenaufwand, um den SAPC für sicherheitskritische Systeme zu zertifizieren, fällt durch diese Maßnahme nur einmal an.
Die Funktion des SAPCs lässt sich im Prinzip als eine Art Watchdog für das Gesamtsystem zusammenfassen, d.h. er erfasst zyklisch das Vorhandensein und den Zustand der anderen Steuergeräte im Verbund. Voraussetzung hierfür ist der Austausch einer erweiterten „Heart Beat“ Information, die in SafeAdapt Healthvector genannt wird.
Diese Erweiterung des Heart Beat ist nötig, um eine ausreichende Granularität der Information zu erreichen, sonst wäre der Informationsgehalt auf ein boolsches „lebt“ oder „lebt nicht“ beschränkt. Damit müssten dann aber alle Funktionen, die auf dem betroffenen Steuergerät laufen, adaptiert werden. Ziel ist es jedoch, wie bereits in Kapitel 2.1 dargestellt, dies auf Funktionsebene zu realisieren. Daher beinhaltet der Healthvector einen Teilbereich, der Informationen zum Zustand einzelner Funktionen auf dem Steuergerät enthält.
Eine entsprechende Reaktion des SAPCs auf einen Fehler im System wird durch einen AUTOSAR Complex Device Driver (siehe Kapitel 3.2.3) auf der jeweiligen Plattform umgesetzt.

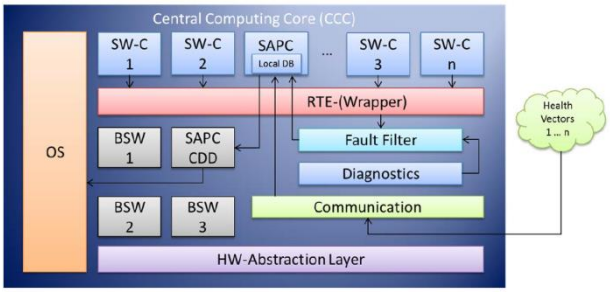

Bild 5 Softwarearchitektur mit SAPC oberhalb der RTE

Natürlich muss auch der eigene Gesundheitszustand an die anderen Teilnehmer im System durch den Healthvector kommuniziert werden.
Damit die universelle SAPC Software-Komponente unabhängig von der HW laufen kann, müssen auch die Fehler in einem Standard Interface an den SAPC weitergegeben werden. Diese Aufgabe übernimmt in der SafeAdapt-Architektur ein sogenannter Fault Filter (siehe Kapitel 3.2.4)
Zu guter Letzt muss der SAPC ein Wissen über das System als Ganzes besitzen, damit sinnvoll adaptiert werden kann. Der neu adaptierte Systemzustand muss unabhängig von allen SAPCs auf den verschiedenen Steuergeräten

berechnet werden. Das errechnete Ergebnis muss auch zwingend auf allen Steuergeräten gleich sein. Damit das gewährleistet ist, wird eine Datenbank im Speicher vorgehalten, die das notwendige Wissen über das System beinhaltet und dem SAPC zur Verfügung stellt (siehe Kapitel 3.2.2).

3.2.2 Datenbank

Um nach einem Fehlerfall das Gesamtsystem in einen sicheren und damit konsistenten Zustand zu bringen, muss der Verbund an SAPC-Instanzen, welche jeweils einem anderen Steuergerät zugewiesen sind, in einer deterministischen Weise agieren. Um diese Art der verteilten Fehlerbehandlung mit minimalem Koordinationsaufwand zu erreichen, verfügt jede SAPC-Instanz über einen Datensatz an Systemkonfigurationen, die jeweils einem oder mehreren Systemzuständen zugewiesen sind. Somit wird sichergestellt, dass für jeden antizipierten Fehlerfall, der sich auch auf den Gesamtsystemzustand auswirkt, eine angemessene Notfallkonfiguration vorhanden ist. Da sowohl antizipierte Systemzustände als auch Notfallkonfigurationen auf allen Steuergeräten identisch sind, ist jedes Steuergerät in der Lage, die nötigen lokalen Anpassungen aus diesem neuen Zielsystemzustand abzuleiten.
Im Detail beinhaltet eine solche Anpassung die Änderung des lokalen Schedules, welche im Falle von AUTOSAR-basierten Systemen durch das Umschalten zwischen synchronisierten Schedule Tables abgebildet werden kann. Somit ist es auf einfache Weise möglich, schon während der Designphase die korrekte Behandlung aller sicherheitskritischen Fehlerfälle zu garantieren, um letztendlich auch den Zertifizierungsaufwand des Gesamtsystems durch eine Reduktion der Laufzeitdynamik zu senken,
Neben diesem einfachen statischen Ansatz, verfügt das Konzept jedoch auch über eine dynamische Planungskomponente, welche zur Laufzeit neue Systemkonfiguration bestimmt. Da das Lösen von Scheduling- und Allokationsproblemen für Systeme mit moderater Größe bereits ein inakzeptables Laufzeitverhalten aufweist, ist es nötig, die Problemgröße dem spezifischen Anwendungsfall anzupassen. Das bedeutet, dass zum Beispiel bei der Adaption im Falle von Energiesparmaßnahmen die Auswahl einer sicheren Systemkonfiguration aus mehreren vorberechneten durch eine Metrik abgebildet werden kann. Im Gegenzug hierzu benötigt die Adaptation für eine Performanzoptimierung einzelner Funktionen ein Verständnis darüber, innerhalb welcher Grenzen sich die Laufzeitanforderungen einzelner Applikationen bewegen. Ein typischer Anwendungsfall ist zum Beispiel die Objekterkennung aus Sensordaten, deren Laufzeitcharakteristiken stark von der Fahrsituation abhängig sind. Dementsprechend ist eine mögliche Erweiterung des Planungsalgorithmus‘, eine Optimierung innerhalb der dynamischen Grenzen einer sicheren und vordefinierten Systemkonfiguration durchzuführen. So kann die Leistungsfähig über mehrere Funktionalitäten hinweg gesteigert werden, ohne dabei Garantien bezüglich der funktionalen Sicherheit zu beeinträchtigen.

3.2.3 Complex Device Driver

Wie bereits erwähnt, muss die Adaption der Software auf allen Steuergeräten systemweit zwingend gleich sein, damit z.B. nach dem Ausfall einer Komponente ein Aktuator keine widersprüchlichen Anweisungen von verschiedenen ECUs erhält. Aus diesem Grunde ist der SAPC als AUTOSAR Komponente entwickelt worden und enthält somit auch keine Komponenten, die ECU-spezifisch sind.
Trotzdem muss das Ergebnis des SAPCs natürlich in Aktionen für das Betriebssystem umgesetzt werden. So müssen einzelne Applikationen deaktiviert, isoliert oder aber auch Standby Funktionen aktiviert werden.
Genau hierfür wird ein Complex Device Driver (CDD) implementiert, der die Schnittstelle zwischen SAPC und Operationssystem darstellt.
Die Kommunikation zwischen SAPC und CDD lässt dabei zwei verschiedene Arbeitsweisen zu: Zum einen kann der CDD eine Liste mit den Applikationen erhalten, die für den nächsten Zyklus aktiviert oder passiviert werden sollen. Zum anderen kann dem CDD eine ID übermittelt werden, die eine vordefinierte Applikationskonfiguration repräsentiert. Je nach Verfahren kann also mehr oder weniger Intelligenz in der Implementierung des CDDs stecken. Dies ermöglicht es, den CDD für das Betriebssystem zu optimieren, da beispielsweise nicht alle Betriebssysteme mit Tabellen arbeiten.

3.2.4 Fault Filter

Ein sogenannter Fault Filter hat die Aufgabe, alle auftretenden spezifischen Fehler in der ECU zu sammeln, zu aggregieren und zu abstrahieren. Der abstrahierte Zustand wird dann an den SAPC gegeben.
In AUTOSAR ist diese Komponente typischer Weise unterhalb der RTE angesiedelt, da hier eine starke Hardwareabhängigkeit besteht. Nur an dieser Stelle ist ein kompletter Durchgriff auf alle Hardwarefehler möglich. Oberhalb der RTE wäre beispielsweise ansonsten für jeden möglichen Fehler ein Port vorzuhalten. Dies ist schon aufgrund des hohen Konfigurationsaufwands sowie des Overheads in der RTE unerwünscht.
Wenn also z.B. ein Fehler in der Selbstdiagnose oder in der Spannungsversorgung des Controllers auftritt, so ist die Art wie dies entdeckt und gemeldet wird, von Controller zu Controller verschieden. Der SAPC braucht allerding nur die Information, dass ein Fehler aufgetreten ist und wie kritisch dieser Fehler ist. Genau diese Abstraktion leistet der Fault Filter.

4 Demonstratoren

Im Kapitel 2 sind typische Adaptionsszenarien vorgestellt worden, die im Rahmen des Projekts demonstriert werden. Aufgrund der verschiedenen Ziele, die mit diesen Szenarien verfolgt werden – von der Anwendungsdemonstration bis hin zur Machbarkeitsuntersuchung – werden die Szenarien durch verschiedene Demonstratoren umgesetzt. Beispielhaft sei hier noch einmal auf Kapitel 2.1.3 und das Nachladen von Funktionen aus einem Repository verwiesen. In einer aktuellen AUTOSAR-konformen Umgebung ist dieser Fall nicht darstellbar, da

hier grundsätzlich von einer statischen Konfiguration ausgegangen wird [4]. Andere Fälle können nicht praktisch demonstriert werden, da z.B. Aktuatorik oder Sensorik schlicht nicht vorhanden ist oder durch physikalische Randbedingungen nicht verbaut werden kann. Daher werden in SafeAdapt mehrere Demonstratoren umgesetzt.

Bild 6: Physikalischer Demonstrator: Roding Roadster der Firma Siemens.

Als Demonstrationsfahrzeug wird ein für die Elektromobilität umgebauter Roding Roadster vom Projektpartner Siemens genutzt. Um einen Teil der Anwendungsfälle auf der Zielhardware darzustellen, wird die vorhandene E/E-Architektur für den SafeAdapt Ansatz angepasst (vgl. Kapitel 3.1.3 und Bild 4) und die zusätzlich benötigten Komponenten werden nachgerüstet. Ziel zum Ende des Projekts ist es, die Basisszenarien mit Hot Standby und Cold Standby anhand dieses Fahrzeugs zu demonstrieren.
Das Nachladen von Funktionen wird durch eine Machbarkeitsanalyse evaluiert. Hierfür wird zusätzlich ein Repository mit vorkompilierten Softwarekomponenten in die SafeAdapt Netzwerkarchitektur installiert. Ziel ist es, für dieses Szenario eine Netzwerkanalyse durchzuführen und Datenlast sowie zeitliches Verhalten zu ermitteln. Anhand dieser Ergebnisse lassen sich später die Randbedingungen wie Latenzzeiten oder Speichervorhalt für ein solches System mit Repository Standby bestimmen.
Zur weiteren Validierung wird die virtuelle Co-Simulationsumgebung „Dynacar“ genutzt. Dies ist ein echtzeitfähiges System, um dynamische Prozesse softwarebasiert zu simulieren. Damit ermöglicht Dynacar existierende Hard- oder Software „in the loop“ gegen das dynamische Modell des Fahrzeuges zu testen. Dieser Demonstrator wird in SafeAdapt genutzt, um die Anwendungsfälle zu testen, bei denen entweder nur ein Softwaremodell der Komponenten zur Verfügung steht oder bei denen die benötigte Hardware nicht im Fahrzeug verbaut werden kann. Darüber hinaus können mit diesem Demonstrator „Driver in the loop“ Evaluierungen sehr kritischer Fahrsituationen virtuell simuliert werden, z.B. die Bestimmung maximaler Fail-Over-Zeiten, die mit einem realen Fahrzeug nicht erprobt werden können.

5 Zusammenfassung und Ausblick

Die Ergebnisse, die im SafeAdapt Projekt erarbeitet wurden können die Grundlage für eine neue Form der Fahrzeugarchitektur sein, da die weiter ansteigende Integration von neuen, automatisierten Fahrfunktionen den Bedarf nach fehlertoleranten, robusten und kostengünstigen Rechenplattformen weiter steigert, wie sie durch SafeAdapt genutzt werden.
Zukünftige Architekturen können sich dieses Ansatzes und der erforschten Mechanismen bedienen und erlauben eine neue E/E-Gesamtarchitektur. Mit der zur Verfügung stehenden Rechenleistung ergibt sich die Möglichkeit, alle Logik und Algorithmen in solchen ECUs als Rechenzentren zusammenzufassen und dafür die Leistungstreiber in separaten ECUs, sogenannten Lastboxen, zu integrieren. Die Vorteile einer solchen Architektur sind vielversprechend:

a. Reduzierung von Verkabelung mit großen Querschnitten, da die Lastboxen in der Nähe der elektrischen Lasten platziert werden können.
b. Erweiterte Redundanz wie sie durch SafeAdapt ermöglicht wird, da die Lastboxen von verschiedenen ECUs der Rechenzentren angesprochen werden können.
c. Bessere Verteilung der Wärmeerzeuger. Da sich mit der steigenden Rechenleistung auch die Abwärme erhöht, wird es unmöglich sein, auch noch Wärme durch Treiberverluste in den Rechenzentren abzuleiten.
d. Reduzierte Kosten durch Gleichteilstrategie sowohl für die Rechenzentren als auch für die Lastboxen.
e. Bessere Skalierbarkeit der Architektur, da bei einer höheren Ausstattung einfach weitere Lastboxen hinzugefügt werden können.

6 Literatur

[1] VDA, Automatisiertes Fahren: https://www.vda.de/de/themen/innovation-und-technik/automatisiertes-fahren.html

[2] Offizielle SafeAdapt Website: http://www.safeadapt.eu/

[3] ISO 26262, First Edition 2011-11-15, Website: http://www.iso.org/iso/home/search.htm?qt=ISO+26262&sort=rel&type=simple&published=on

[4] AUTOSAR (Automotive Open System Architecture), Release 4.2.x, Website: http://www.autosar.org/

[5] SAE AS6802 – Time -Triggered Ethernet, 2011, Website: http://standards.sae.org/as6802/

Förderhinweis

SafeAdapt wurde von der Europäischen Union im 7. Rahmenprogramm unter dem Förderkennzeichen 608945 gefördert.