AUTomotive **O**pen **S**ystem **AR**chitecture

Eine Einführung

von Robert Neue

Inhaltsverzeichnis

AUTOSAR

1. Allgemeines

AUTOSAR ist in der Automobilbranche in aller Munde. Doch warum? In der Automobilbranche bietet AUTOSAR die Möglichkeit das immer wichtiger gewordene Konzept der modellgetriebenen Entwicklung von Softwarekomponenten zu verwenden. Da die Anzahl der Mikrocontroller in einem Auto von vor 10 Jahren von einem halben Dutzend Mikrocontroller auf inzwischen mehr als 60 Mikrocontrollern (in nahezu allen gängigen Autos der Luxuskategorie) angestiegen ist, wird die Wiederverwendbarkeit der Software, sowie ihre Wartbarkeit, immer wichtiger.

1. Was ist AUTOSAR?

AUTomotive **O**pen **S**ystem **AR**chitecture ist ein Zusammenschluss verschiedener Automobilhersteller, wie zum Beispiel BMW, DaimlerChrysler und Ford, sowie Zulieferern, wie BOSCH und Siemens VDO.

Ihr Motto:
„Cooperate on Standards – Compete in Implementation“ ist Programm.

Bei AUTOSAR sind selbst viele Teile der Basissoftware in austauschbare Module gepackt worden. Das Ziel von ist die Schaffung eines Standards für die Entwicklung von Softwarekomponenten für den Automobilbereich. Ende 2006 soll der Standard soweit abgeschlossen sein, dass mit der Serienproduktion begonnen werden kann. Die Zulieferer müssen daher bereits ab Anfang 2007 zu AUTOSAR konforme Steuergeräte liefern um Autos mit dem neuen Standard fertigen zu können.

AUTOSAR versteht sich wie seine Vorgängerinitiativen nur als **Standardisierungsgremium,** das Spezifikationen erarbeitet, aber keine verbindliche Implementierung vorschreibt. Aufgrund der Komplexität fördert die Initiative prototypische Referenzimplementierungen, die die Machbarkeit nachweisen. Natürlich erhoffen sich die Zulieferer und Werkzeuganbieter, solcher Referenzimplementierungen, einen späteren Wettbewerbsvorsprung und versuchen verständlicherweise, bestehende Lösungen aus ihrem Hause in den Standardisierungsprozess einfließen zu lassen.

2. Warum ein neues System?

Einige Firmen merkten, dass es bestimmte Problembereiche bei bisherigen Standards gab, und wollten diese in einem neuen Standard lösen. Folgende Eigenschaften sollte der neue Standard aufweisen:

a) Implementierung und Standardisierung von Basisfunktionen
das heißt eine gewisse Basisfunktionalität (die spätere Basissoftware) sollte gegeben sein. Diese sollte jedoch modular und gegebenenfalls durch Module von Drittherstellern austauschbar sein.

b) Skalierbarkeit hinsichtlich verschiedener Fahrzeugtypen
Möglichkeit passend zu den jeweiligen ECUs passend skalierte Lösungen zu implementieren, die trotzdem von gut getesten und qualitativ hochwertigen Standardmodulen profitieren können.

c) Möglichkeiten zur redundanten Auslegung
Dies wird vor allem für Sicherheitskritische Bereiche gefordert.

d) Einbettung von Modulen anderer Hersteller
Um so eine qualitativ sehr hochwertige Software und hochwertige Produkte herstellen zu können, die Lösungen Out-of-the Box nutzen können.

e) Wartbarkeit während des gesamten Produktlebenszykluses

f) Software Updates und Upgrades während des gesamten Fahrzeuglebens

Die von AUTOSAR umgesetzten Lösungen für diese Eigenschaften sind:

1. Standardisierung des Austauschformats
Fortschritte in der Spezifikation, sowie die Möglichkeit eine lückenlose Werkzeugkette zu realisieren, da dasselbe Austauschformat benutzt wird.

2. Basic Software
Dadurch wird die Softwarequalität gesteigert und der Focus auf Funktionen gelegt die einen echten Wettbewerbsvorteil bringen und somit natürlich auch rentabler sind.

3. MicroController Abstraktion
Dadurch sind darauf aufsetzende Applikationen Mikrocontroller unabhängig und somit auf vielen unterschiedlichen Mikrocontroller einsetzbar.

4. RunTimeEnvironment
Die RTE ist eine Middleware Schicht und stellt Dienste zur Kommunikation für die Applikations Software (AUTOSAR Software Components und/oder AUTOSAR Sensor/Actuator components) zur Verfügung. Diese kommunizieren untereinander oder mit den Diensten via RTE. Siehe später VFB.
Ein echter Vorteil ist jedoch die Möglichkeit Funktionen zu druppieren und auf andere ECUs zu verlagern zu können. Desweiteren lassen sich mit diesem Konzept auch Funktionen kapseln, um so unabhängig von der Kommunikationstechnologie zu sein.

5. Standard von Schnittstellen, um Probleme beim Einbinden von Produkten verschiedener Hersteller zu vermeiden
   Dies vereinfacht vor allem die Implementierung hardwareunabhängiger Software und ermöglicht den Einsatz von standardisierten AUTOSAR Werkzeugen um automatisch generierten Code zu erzeugen.
   Desweiteren ermöglicht es auch den leichten Austausch von Softwarekomponenten, selbst wenn sie von verschiedenen Zulieferern kommen und was am wichtigsten ist: Die Wiederverwendbarkeit wird erhöht.

AUTOSAR ist jedoch kein autarker neuer Standard sondern baut auf vielen bekannten Standards, vor allem OSEK, auf und versucht Schwächen der anderen Standards zu lösen.

2. AUTOSAR Konzepte

AUTOSAR bietet eine Reihe interessanter Konzepte, von denen hier nur die vier wichtigsten vorgestellt werden sollen, um die obengenannten Ziele zu erreichen.
Wer mehr zu den anderen Konzepten wissen will, sei hier auf [AUTOSAR.org – Technical Overview] verwiesen.

1. AUTOSAR Software Component (AUTOSAR SW-C) + SW-C Description

Ein wichtiges Designkonzept in AUTOSAR ist die Teilung in Applikation und Infrastruktur. Eine Applikation besteht aus miteinander kommunizierenden AUTOSAR Softwarekomponenten.
Jede dieser Softwarekomponenten beinhaltet einen Teil der Gesamtfunktionalität der Anwendung. Eine Instanz einer solchen Komponente wird auch *Atomic Software Component* genannt, wenn sie nur exakt einer ECU zugeordnet werden kann. AUTOSAR unterstützt das generische Komponenten-Konzept. Logisch untereinander verbundene SW-Cs können zusammen in einer *Composition* aggregiert werden.
Eine AUTOSAR SW-C kann entweder als Objektcode oder Quellcode zusammen mit einer Softwarebeschreibung geliefert werden.

Die Verbreitung per Objektcode hat den Nachteil, dass dieser stark hardwareabhängig ist. Eine Implementierung als Quellcode ist dagegen unabhängig von der zugrunde liegenden Hardware. Auch muss nicht die gesamte benötigte Funktionalität auf derselben Hardwarekomponente zur

Verfügung stehen, da diese aufgrund der Kommunikationsmöglichkeiten von anderen Hardwarekomponenten über das Netzwerk bereitgestellt werden kann. Somit ist der Quellcode unabhängig von der verwendeten Netzwerktechnologie.
Die Beschreibung umfasst sowohl alle Operationen und Datenstrukturen die die SW-C liefert und benötigt, als auch die erforderlichen Hardwarevoraussetzungen (Speicher, CPU-Zeit, etc.) und Anforderungen an die Infrastruktur. Das Beschreibungsformat ist ebenfalls standardisiert.
Sensor und Aktuator SW-Cs sind Spezialformen, welche die Abhängigkeiten zu speziellen Sensoren oder Aktuatoren kapseln.

2. Virtual Functional Bus (VFB)

Der VFB ermöglicht die Kommunikation von *Software-Componenten (SW-C)* auf einem abstrakten technologisch unabhängigen Niveau. Dies bedeutet, dass den SW-Cs auch die Services der Basis Software, die ECU Abstraktion und komplexe Gerätetreiber (Complex Device Drivers) einfach per Kommunikationskanal zur Verfügung stehen.
Komponenten in AUTOSAR besitzen jeweils fest definierte Ports, die der Interaktion untereinander dienen.

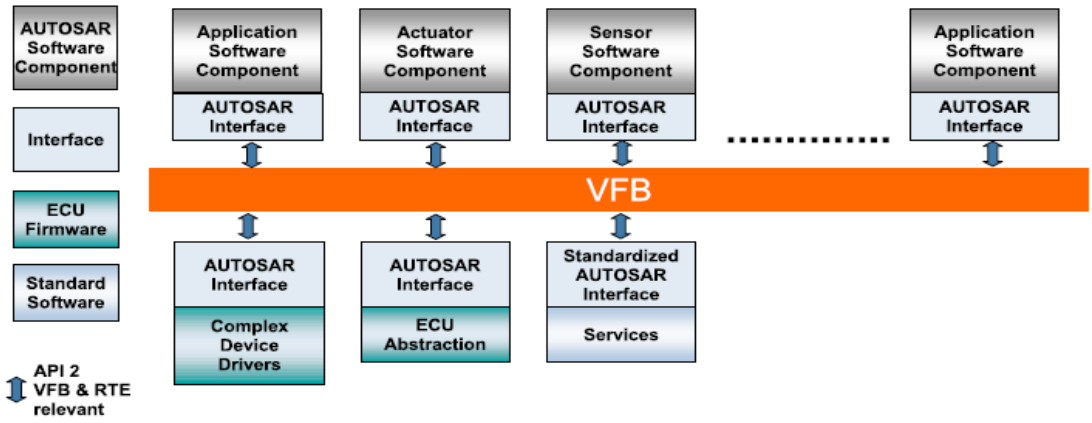

Abbildung 1: Virtual Functional Bus

Bei AUTOSAR soll bei der Entwicklung einer Komponente darauf geachtet werden, dass der Programmierer niemals API-Funktionen oder Funktionen sowie Variablen der einzelnen Module direkt aufruft, sondern jeweils die entsprechenden Ports mit ihren definierten Schnittstellen verwendet.
Ein Port ist eindeutig an eine Instanz einer Komponente gebunden, andererseits ist es aber möglich, dass es mehrere Instanzen einer Komponente gibt. Die Mehrfachinstanzierung dient unter anderem der Ausfallsicherheit.

Für die Kommunikation stehen im Wesentlichen zwei Muster zur Verfügung:

a) Client-Server
Bei dem ein oder mehrere Clients die Informationen anfordern und der jeweils anbietende Server antwortet. Sowohl synchron als auch asynchron unterstützt.

b) Sender-Receiver
Mit dem Nachrichten von einem an viele gesendet werden. Es wird ein asynchroner Modus angenommen, d.h. der Sender blockiert seine Befehlsausführung auch dann nicht, wenn wie bei einer synchronen Kommunikation eine Antwort zurück erwartet würde.

3. RunTimeEnvironment(RTE)

Die RTE wird gerne dem VFB gleichgesetzt, ist jedoch die Implementierung des VFB (nicht der VFB selber, der ja ein abstraktes Konzept ist) für eine spezielle ECU. Die RTE darf Funktionalitäten der VFB weitestgehend an die unter ihr liegenden Schichten delegieren, sofern dies möglich ist und muss nicht alles selbst implementieren.

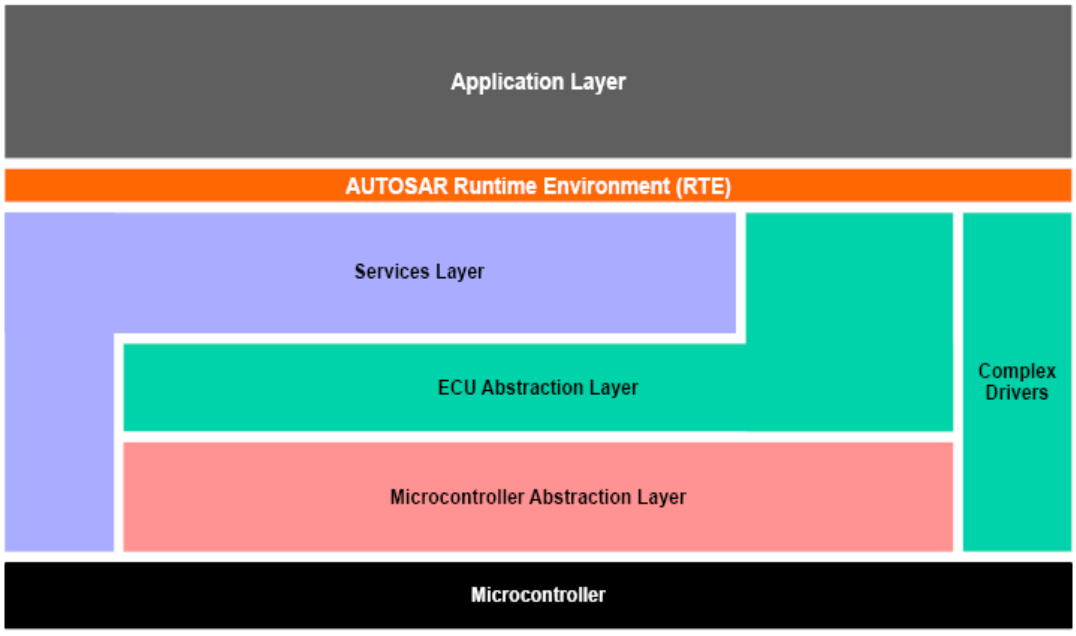

Abbildung 2:Schichtenmodell

4. Basic Software

Die Funktionalität einer ECU wird von der *Basic Software* zur Verfügung gestellt. Die Hardwarezugriffe erfolgen über diese Schicht. AUTOSAR

definiert eine Softwarearchitektur für Steuergeräte, die die Software von der Hardware des Gerätes entkoppelt.
AUTOSAR definiert für die Software die aus einzelnen Funktionsmodulen, den *Software-Components*, zusammensetzt ist und die unabhängig voneinander und durch verschiedene Hersteller entwickelt sein können, einen weitgehend automatisierten Konfigurationsprozess der die Softwarekomponenten zu einem konkreten Projekt zusammenbinden.

3. Schichtenmodell von AUTOSAR

Die Entkopplung von Hardware (ECU) und Software sowie den verschiedenen Softwarekomponenten erfolgt durch ein Grundsoftwarepaket der *Basic Software.* Die *Basic Software* lässt sich in die vier vertikalen Säulen unterteilen:

- Systemdienste,
- Speicherverwaltungsdienste,
- Kommunikationsdienste und
- Hardware-Ein-/Ausgabedienste

Jede dieser Säulen wird horizontal in die folgenden drei Schichten eingeteilt:

- *Service Layer*,
- *ECU Abstraction Layer* und
- *Microcontroller Abstraction Layer*

Eine fünfte Säule, die komplexen Treiber (Complex Drivers), liefert eine Migrationsstrategie für Softwaremodule die keiner Schicht eindeutig zuzuordnen sind. Diese können so erst einmal eingebunden werden und später den Schichten gemäß angepaßt werden.

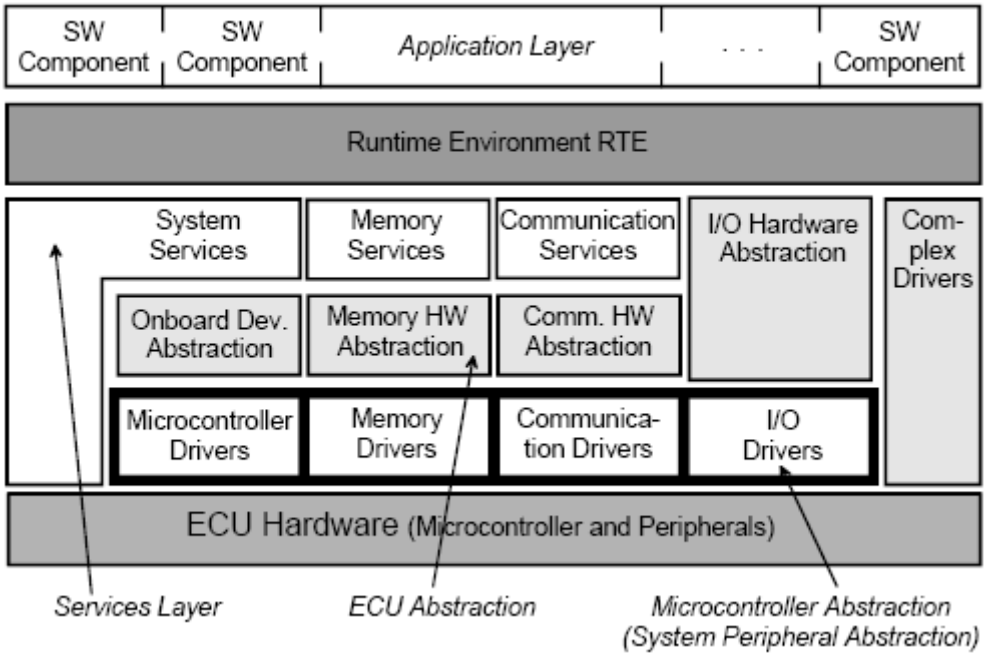

Abbildung 2: Verfeinertes Schichtenmodell

Service Layer

Stellt Systemdienste wie zum Beispiel Diagnose Protokolle, NVRAM, Flash und Speicher Management bereit.

ECU Abstraction Layer

Stellt ein Interface zum Zugriff auf die elektrischen Signale einer speziellen ECU bereit, um die darüber liegende Software von allen darunter liegenden Hardwareabhängigkeiten zu befreien.

Microcontroller Abstraction Layer

Kein direkter Zugriff auf den Microcontroller. Der Zugriff erfolgt nur über ein standardisiertes Interface. Dadurch sind Module darüberliegender Schichten unabhängig vom jeweiligen Controller. Module des *ECU Abstraction Layers* sind für externe Peripherie-ICs zuständig. Diese Schicht implementiert auch Notfikationsmechanismen für die verschiedenen Treibermodule.

Folgende Baugruppen sind zur Zeit spezifiziert:

- *Mikrocontroller-Grundfunktionen*
  Zum Startup Code eines MCU gehört unter anderem die Initialisierung des Taktgenerators und der CPU-Betriebsmodi, Speicherbus, Interrupts, Ansteuerung des Watchdog-Timers (WDT) und der Zeitgeberbaugruppe (GPT General Purpose Timer).

- *Digital- und Analog-Ein-/Ausgabe*
  Grundkonfiguration sämtlicher Mikrocontroller-Anschlüsse (PORTs), Digital-Ein-/ Ausgänge (DIO), Input-Capture-Einheit (ICU), Pulsbreiten-modulierte Signalausgänge (PWM) und Analog-Digital-Umsetzer (ADC)

- *Speicherbausteine*
  Ansteuerung von internen und externen Flash-ROM und EEPROM-Bausteinen

- *Kommunikationsschnittstellen*
  Treiber für interne und externe SPI-, CAN-, LIN- und FlexRay-Controller.
  Neben den eigentlichen Treibern sind einige Module mit Selbsttestfunktionen

  Alles was in diesen Schichten nicht angeboten wird, steht im Normalfall nicht zur Verfügung, sondern muss über spezielle Schnittstellen angesprochen werden. Im Fall der Verwendung solcher Spezialfunktionen, müssen die Dienste gegebenenfalls verlegt werden.
  Bezüglich der Basissoftware setzt AUTOSAR auf die Vorarbeiten von OSEK, HIS, ASAM und ISO sowie der Industriekonsortien für CAN, FlexRay, LIN

und versucht die dort definierten Konzepte, z.T. auch Standards für Betriebssystem, Hardwaretreiber und Protokolle zu integrieren, wobei neben funktionalen Erweiterungen insbesondere eine Durchgängigkeit der Schnittstellen und Kommunikationsmechanismen zwischen den Softwaremodulen vorgesehen ist.

4. AUTOSAR-OS

Das Betriebssystem AUTOSAR-OS, ohne dass keine Funktionalität möglich wäre, basiert auf den Vorarbeiten von OSEK, genauer gesagt auf OSEK-OS.
Im Gegensatz zu OSEK-OS gehören zu den Features von AUTOSAR von vornherein Echtzeitverhalten und Zeitüberwachung, sowie Sicherheitsmechanismen zur Laufzeit. Sollten zum Beispiel bei *Telematic* oder *Infotainment* andere Betriebssysteme besser geeignet sein, so können diese natürlich gegen AUTOSAR-OS ausgetauscht werden, sofern sie kompatibel zu den Interfaces von AUTOSAR sind.
Je nach Verwendungszweck läßt sich AUTOSAR-OS skalieren. Den Umfang beschreiben die *Scalability Classes* 1-4 (Abb. 3)

Category	Scalability Class 1	Scalability Class 2	Scalability Class 3	Scalability Class 4
OSEK OS (all conformance classes)	✓	✓	✓	✓
Counter Interface	✓	✓	✓	✓
Schedule Tables	✓	✓	✓	✓
Stack Monitoring	✓	✓	✓	✓
Protection Hook		✓	✓	✓
Timing Protection		✓		✓
Global Time/Synchronization Support		✓		✓
Memory Protection			✓	✓
OS Applications			✓	✓
Service Protection			✓	✓
Call Trusted Function			✓	✓

Abbildung 3: Einteilung der Scalability Classes (nach Freescale)

Durch diese Klasseneinteilung kann je nach Anwendungsbereich ein angepaßtes und schlankes OS benutzt werden.

5. AUTOSAR Methodologie

Die AUTOSAR Methode ist weder ein vollständiger Prozess noch ein Business Modell; es sind auch keine Rollen oder Zuständigkeiten in dieser Methode erfaßt. Die Methode stellt lediglich eine Vorgehensmethodik zum Erstellen der Software bereit.
Hierbei handelt es sich um eine Kette von Schritten, welche in einer ausführbaren ECU Komponente endet.

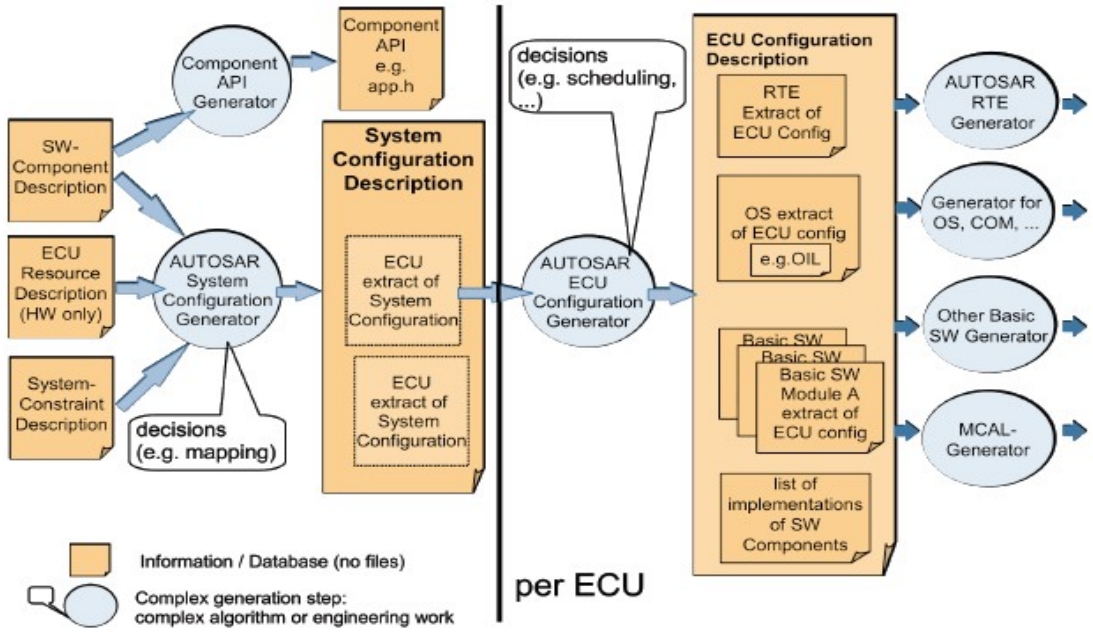

Abbildung 4:AUTOSAR Methodik 1

Alle Beschreibungsdokumente sind in XML verfaßt. Als erstes werden die entsprechenden Hardware- und Softwarebestandteile sowie die notwendiger Bedingungen ausgewählt und festgelegt.
Für diesen Schritt stehen entsprechende Vorlagen zur Verfügung.
Der nächste Schritt befasst sich mit der Zuordnung von Software- zu Hardwarekomponenten. Es erfolgt eine Auswertung der benötigten und der von der Hardware zur Verfügung gestellten Ressourcen. Basierend auf diesen Daten wird das Mapping durchgeführt. Die vorliegende Beschreibung (System Configuration Description) enthält die Beschreibung des Systems, wie unter anderem Topologie, BUS Mapping und welche Software auf welche ECU kommt.
Die nun folgenden Schritte sind für jede ECU einzeln durchzuführen. Anschließend wird die ECU Information einzeln aus der System-Konfigurationsbeschreibung extrahiert, die für eine detaillierte Beschreibung der ECU verwendet wird (Schritt: Configure ECU).

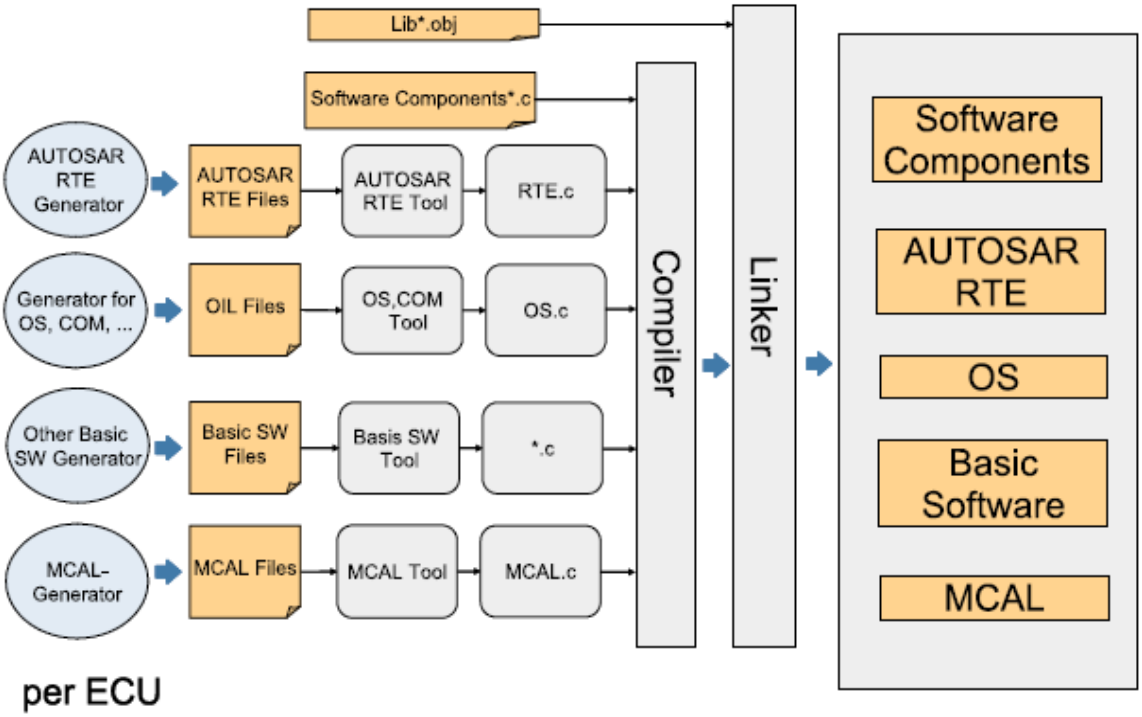

Abbildung 5:AUTOSAR Methodik 2

Im letzten Schritt wird aus den gesammelten bzw. generierten Informationen ein ECU Executable erzeugt. Hier wird dann z.B. der Code für die RTE und die Basic Software generiert. Ebenso wird der vorliegende Sourcecode der Softwarekomponente kompiliert. Sollte dieser schon als Objektcode vorliegen, so muss er natürlich nicht mehr kompiliert werden. Zu guter Letzt werden die einzelnen Bestandteile in einer ausführbaren Komponente zusammengefasst.

6. Fazit zu Autosar:

AUTOSAR ist der neueste Schrei in der Automobilindustrie. Obwohl es schwer ist Informationen zu sammeln, sobald es tief ins Detail geht, hier bleiben dann oft nur noch die Spezifikationen, gibt es doch einige Diskussionen im Web über AUTOSAR.

Ihnen scheint zur Zeit gemein zu sein, dass AUTOSARs Ziele längst noch nicht überall erreicht sind. Ein Hauptziel von AUTOSAR die Software von der darunterliegenden Hardware zu abstrahieren und unabhängig zu machen und durch die entsprechenden Werkzeuge eine automatische und performante Verteilung der Softwaremodule zu garantieren scheitert zur Zeit anscheinend noch an der Tatsache dass „viele der tool-gestützten Prozesse noch nicht durchgängig implementiert“ seien.

Vielfach wird davon geredet, dass „AUTOSAR für mehr Transparenz bei Projekten sorgt, die in Kooperation von OEM und Zulieferer entwickelt werden. Aufgaben und Teilbereiche können durch die modulare Struktur der Funktionen fast beliebig verteilt werden, ohne bei der Integration Gefahr zu laufen, sich gegenseitig negativ zu beeinflussen.“

Natürlich ist das was technischer Seite wünschenswert ist, noch auf der kaufmännischen Seite zu klären. Gerade die Austauschbarkeit der Komponenten und Modulen durch solche von Drittanbietern scheint einigen Kaufleuten noch Unbehagen bereiten. Hier wird wahrscheinlich nur die Zeit zeigen, wie dies in entsprechende *Business Rules* bzw. Geschäftsmodelle umgemünzt werden kann.