ITEA project 00009

EAST-EEA
Embedded Electronic Architecture

Report Typ	Bericht
Report Name	Schlussbericht
Berichtszeitraum	01.05.2001 – 31.10.2004
Report Status	Öffentlich
Versionsnummer	Version 1.01
Erstellungsdatum	27.06.2005

Zuwendungsempfänger	Förderkennzeichen
DaimlerChrysler AG	*01 IS A01 A*
Laufzeit des Vorhabens	*01.05.2001 – 31.10.2004*

Das diesem Bericht zugrunde liegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung, Wissenschaft, Forschung und Technologie unter dem Förderkennzeichen 01 IS A01 A gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei den Autoren.

ITEA
INFORMATION TECHNOLOGY
FOR EUROPEAN ADVANCEMENT

Dieser Bericht wurde erstellt durch die DaimlerChrysler AG

Beiträge wurden verfasst von

Joachim Eisenmann DCAG

Versionsübersicht

Version	Datum	Grund
0.1	09.12.2004	Vorlage von J. Eisenmann, DCAG
0.9	25.01.2005	Kurzdarstellung
0.91	28.01.2005	Struktur Teil II
0.92	1.02.2005	Inhalte Teil II
0.93	14.02.2005	
0.94	20.04.2005	
0.98	16.06.2005	
0.99	22.06.2005	
1.0	24.06.2005	Endfassung
1.01	27.06.2005	Laufzeit korrigiert

Inhaltsverzeichnis

1 Kurzdarstellung

1.1 Aufgabenstellung

Ziel dieses Projektes ist die Schaffung einer standardisierbaren offenen Softwarearchitektur im Kfz, die auf existierenden Standards aufsetzt und die eine hardwareunabhängige Verteilung von Software-Modulen in vernetzten Kfz-Systemen zulässt.

Einerseits kann der Kfz-Hersteller dadurch Software für seine „embedded real-time systems“ als Lieferantenteil mit „Teilenummer“ auf die im Fahrzeug verbaute Hardware adaptieren, unabhängig vom jeweiligen Zulieferanten. Andererseits können neue Kfz-Funktionen allein in Software durch ineinandergreifende Verknüpfungen ebenfalls zulieferantenübergreifend realisiert werden.

Diese offene Softwarearchitektur soll durch eine geschichtete Softwarestruktur erreicht werden, die auf existierenden Lösungen (z.B. OSEK/VDX) aufsetzt. Fokus ist eine sogenannte Middleware-Schicht, die sowohl die Partitionierung der Funktionssoftware auf eine verteilte und vernetzte Hardware unterstützt als auch die Interoperabilität der unterschiedlichen SW-Module garantiert.

Die Vorteile dieser offenen Systemarchitektur sind:

- für die Kfz-Hersteller:

  Gemeinsame Wieder- und Mehrfachnutzung von nicht wettbewerbsrelevanten Softwaremodulen. Wettbewerbsrelevante Funktionen können jeweils separat / spezifisch entwickelt werden.

- für die Kfz-Zulieferer:

  Reduktion der Variantenvielfalt, die entsteht, wenn jeder Kfz-Hersteller seine eigenen Standards setzt.

  Damit kann die Effizienz in der Funktionsentwicklung gesteigert und die internationale Wettbewerbsfähigkeit gesteigert werden. Neue Geschäftsmodelle für Zulieferanten und die Senkung der Einstiegsschwelle für neue Firmen sind zu erwarten.

- für Toolhersteller:

  Einheitliche Schnittstellen zu den Entwicklungsprozessen. Transparente Schnittstellen fördern auch hier eine Senkung der Einstiegsschwelle für neue Firmen.

Insgesamt erfordert oder stimuliert die Kfz-Industrie somit Lösungen, die einen wesentlichen Beitrag zur Entwicklung der deutschen IT-Industrie (im europäischen Markt) bringt.

Zurzeit finden viele neue Funktionen Eingang in das Kfz, z.B. Navigation, Verkehrstelematik, Fahrwerksteuerung, adaptive Tempo-regelung usw. Diese Funktionen sind noch weitgehend autark. Eine wichtige technische Aufgabe der beteiligten Kfz-Hersteller ist die Integration unterschiedlicher Elektroniksysteme und deren Komponenten verschiedener Zulieferer zu einem kostengünstigen Netzwerk im Fahrzeug. Um die Entwicklungskosten zukünftiger Systeme möglichst niedrig zu halten, ist das Verfahren zur Integration für Anwendungs- und Kommunikations-Schnittstellen und Entwicklungsumgebungen möglichst herstellerunabhängig zu formulieren.

Dies soll über die Definition einer offenen Systemarchitektur erreicht werden. Wesentliches Element dieser Architektur ist ein Schichtenmodell unter Einschluss einer Middleware-Schicht, die Schnittstellen und Dienste bereitstellt, welche die Portabilität von eingebetteten Software-Modulen unter hohen Qualitätsstandards gewährleistet.

Die Projektarbeit wird in den folgenden Teilprojekten (Work Packages) abgearbeitet:

WP1: General Aspects

In diesem Teilprojekt werden die Grundlagen für die weitere Projektarbeit bereitgestellt. Folgende Arbeitspakete (Work Tasks, WT) werden behandelt:

WT1.1 Zukünftiges technisches Szenario

Hier wird eine Benutzersicht bezüglich der zukünftig zu erwartenden Integration elektronischer Systeme im Kraftfahrzeug unter Marketingaspekten, d.h. ohne Erläuterung technischer Hintergründe, dargestellt. Ergebnis ist die Vision des zukünftigen Umfeldes der im Projekt zu entwickelnden Fahrzeugarchitektur.

WT1.2 Definition der Allgemeinen Anforderungen (Requirement-Analyse)

Hier werden die Anforderungen der Projektpartner in Form von Anwendungsfällen (Use Cases) gesammelt. Hieraus werden zum einen die technischen Kriterien zur Validierung der beispielhaften Implementierungen abgeleitet und zum anderen die Rohdaten zur Ermittlung der Referenzarchitektur (siehe WT1.5) erzeugt.

WT1.3 Existierende Lösungen und ein Fahrplan zu ihrer Harmonisierung

Ziel der Arbeiten ist generell die Vermeidung teurer und unnötiger Neuentwicklungen durch Rückgriff auf existierende Lösungen. Hierbei sollen sowohl Anwendungen in den verschiedenen Domänen (Karosserieelektronik, Triebstrang, Telematik usw.) als auch bei der Bustechnologie berücksichtigt werden.

WT1.4 Verfahren zum Umgang mit geistigen Eigentumsrechten und zukünftige Geschäftsmodelle

Die im Rahmen des EAST/EEA-Projekts zu erarbeitenden Mechanismen sollen das Zusammenwirken von Softwaremodulen ermöglichen, die von unterschiedlichen Entwicklungspartnern (Fahrzeughersteller und -zulieferer) erstellt werden. Diese Vorgehensweise erfordert neue Verfahren und Methoden zum Umgang mit geistigen Eigentumsrechten. Dabei sind zwei verschiedene Aspekte zu berücksichtigen:

- Technische Aspekte:

  Es muss ein Modell entwickelt werden, das es ermöglicht, die unterschiedlichen Rollen bei der Softwareentwicklung für KfZ-Steuergeräte (z.B. Systemintegrator, Anwendungsentwickler oder Steuergeräte-Integrator) sowie deren Schnittstellen untereinander technisch zu definieren. Dies kann auf Basis moderner kryptographischer Verfahren erfolgen. Der Informationsfluss über diese Schnittstellen muss dabei bestimmten noch zu festzulegenden Regeln gehorchen.

  Dies stellt sicher, dass jeder Entwicklungspartner sein spezifisches Know-how schützen kann und den anderen Partnern nur

genau die Informationen zur Verfügung stellen muss, die diese von ihm benötigen. Die in diesem Arbeitspaket erarbeiteten Methoden sollen in die im WP3 zu spezifizierenden Entwicklungstools integriert werden.

- Rechtliche Aspekte:

  Hier müssen die Auswirkungen dieser verteilten Softwareentwicklung auf zukünftige Geschäftsmodelle der einzelnen Entwicklungspartner untersucht werden. Das Ergebnis umfasst sowohl Randbedingungen für die Vertragsgestaltung zwischen den Partnern mit besonderem Augenmerk auf Haftungsfragen als auch mögliche Vorschläge an den Gesetzgeber für die zukünftige Gestaltung von gesetzlichen Rahmenbedingungen, sofern diese notwendig erscheinen.

WT1.5 Ableitung der zukünftigen Referenzarchitektur

In diesem Arbeitspaket soll die allgemeine Referenzarchitektur ausgehend von den Ergebnissen aus WT1.2 und WT1.3 abgeleitet werden. Dies erfolgt im Domain Engineering Prozess:

Eine *Domänenarchitektur* wird als Beschreibung der Analyse der spezifischen Domänen des Fahrzeugs bestimmt.

Eine *Subsystem-Architektur* wird aus der Domänenarchitektur abgeleitet durch Spezifikation kooperierender funktionaler Subsysteme.

Die *Referenzarchitektur* enthält weitere detaillierte Definitionen in bestimmten Bereichen der Subsystem-Architektur. Wesentliches Element dieser Architektur ist ein Schichtenmodell unter Einschluss einer Middleware-Schicht, die Schnittstellen und Dienste bereitstellt, welche die Portabilität von Embedded Software-Modulen unter hohen Qualitätsstandards gewährleistet.

WP2: Interoperability and Portability

In diesem Teilprojekt soll die Spezifikation eines Middleware- und Kommunikationskonzepts erarbeitet werden. Die Middleware bietet dabei den Anwendungsfunktionen Dienste in Form eines APIs (Application Programmers Interface) an, das eine abbildungstransparente Interaktion zwischen verschiedenen Anwendungsfunktionen im Fahrzeug erlaubt (Interoperabilität). Die Kommunikationsschicht stellt der Middleware Basiskommunikationsdienste zur Verfügung, die über Device-Treiber an die Fahrzeugnetzwerke (z.B. CAN, MOST, IEEE1394) adaptiert werden.

Das Teilprojekt unterteilt sich in die folgenden Arbeitspakete:

WT2.1 General Implementation Framework

In diesem Arbeitspaket werden die Anforderungen an die Middleware und die Kommunikationsschicht erarbeitet. Dazu gehört sowohl die Erarbeitung subsystemspezifischer Anforderungen als auch die Berücksichtigung allgemeiner Anforderungen. Das Ergebnis des Arbeitspakets ist eine Beschreibung der Struktur sowie der von den einzelnen Schichten zur Verfügung gestellten Dienste.

WT2.2 Middleware Specification

Aus den Ergebnissen von WT2.1 wird die Spezifikation der Middlewareschicht einschließlich den der Anwendung bereitgestellten Diensten erarbeitet.

WT2.3 Middleware Implementation

Prototypische Implementierung der in WT2.2 erarbeiteten Spezifikation der Middleware zur Validierung der Spezifikation sowie als Basis für die in WP4 zu erstellenden Demonstratoren.

WT2.4 Communication Layer Specification

Aus den Ergebnissen von WT2.1 wird die Spezifikation der Kommunikationschicht einschließlich den der Middleware bereitgestellten Diensten erarbeitet.

WT2.5 Communication Layer Implementation

Prototypische Implementierung der in WT2.4 erarbeiteten Spezifikation des Communication Layers zur Validierung der Spezifikation sowie als Basis für die in WP4 zu erstellenden Demonstratoren.

WP3: Development and Validation Tools

Dieses Teilprojekt behandelt die Entwicklung und Validierung von Architekturen und Funktionsmodulen. Es berücksichtigt den vollständigen Entstehungsprozess von der Spezifikation über die Simulation, Implementierung, Einzeltests und Integrationstests bis zur Validierung. Ziel ist, auf bereits existierenden Ansätzen aufzusetzen. Daher werden den jeweiligen Definitionsphasen grundsätzlich Untersuchungen bestehender Lösungsansätze vorgeschaltet.

WT3.1 Architecture, Modelling Language and Exchange Format

Hier geht es um die Definitionen einer Beschreibungssprache für "Embedded Electronic"- Architekturen (ADL = Architecture Description Language) für eine offene Systemarchitektur, einer Interface-Sprache zur Nutzung bestehender Funktionsmodule, die Anforderungen an ein XML- Austauschformat, sowie Analyse- und Modifikations-Tools zur Struktur-Optimierung.

WT3.2 Functional Models/Modules Test & Validation

WT3.3 Architecture Allocation and Validation, Performance and Safety

WT3.4 Integration Test & Validation

Die Tasks 3.2, 3.3 und 3.4 behandeln die Test- und Validierungsphasen für die Funktionsarchitektur, die Elektronikarchitektur und die Integration. Es werden die erforderlichen Tools, Prozesse, Prozeduren und Methodiken definiert, um eine hohe Produktqualität hinsichtlich Anforderungserfüllung, Sicherheit, Leistung und Robustheit sicherzustellen.

WT3.5 Automatic Code Generation

Hier werden die Anforderungen an einen automatischen Code-Generator für Applikationen mit verteilten Systemstrukturen untersucht.

WP4: Domain Specific Implementation and Validation

Das Ziel dieses Teilprojekts ist es, die in WP2 erarbeiteten Konzepte unter Verwendung der in WP3 erarbeiteten Methoden an Hand domänenspezifischer Demonstratoren zu validieren. Dabei handelt es sich um Labordemonstratoren, die allerdings, soweit dies im Rahmen dieses Projekts möglich ist, auf Basis seriennaher Steuergeräte aufgebaut werden sollen. Die Erfahrungen aus der Validierung durch die

Demonstratoren sollen zur Optimierung der in den anderen Teilprojekten erarbeiteten Lösungen dienen.

Beim Aufbau der Demonstratoren soll gezeigt werden, dass es durch den Einsatz der in EAST-EEA erarbeiteten Konzepte und Methoden möglich ist, die spezifischen Herausforderungen bei der Entwicklung von Software für die einzelnen Fahrzeugdomänen zu meistern.

WT4.1 Body Electronics

Bei der heutigen Elektronikentwicklung werden die Steuergeräte vielfach als Einheit von Hardware und darauf laufender Software, also als „Black Box“, betrachtet. Die in EAST-EEA erarbeiteten Methoden erlauben eine Trennung von Soft- und Hardware, d.h. eine separate Betrachtung aller für die Softwareentwicklung notwendigen Aspekte unabhängig von der zugrunde liegenden Hardware. Durch den Demonstrator sollen die folgenden Vorteile der EAST-EEA-Vorgehensweise gezeigt werden:

- Software-Funktionen und Steuergeräte-Hardware können durch unterschiedliche Zulieferer entwickelt werden.
- Software-Funktionen eines Zulieferers können auf unterschiedliche Steuergeräte anderer Zulieferer verteilt werden.

Die in WP2 spezifizierte Middleware stellt den Funktionen unabhängig von der Verteilung einen Satz von vereinheitlichten Diensten zur Verfügung.

Die in WP3 erarbeitete Entwicklungsmethode erlaubt den Aufbau von Funktionsbibliotheken sowie den Austausch von Softwaremodulen zwischen Fahrzeugherstellern und Zulieferern unter Beachtung des Rechts auf geistiges Eigentum.

WT4.2 Powertrain

Die Implementierung der Middleware- und Communication-Layer aus WP 2 und die Methoden und Tools aus WP3 bilden die Grundlage für die Realisierung einer EAST-SW-Architektur. Die Teilnehmer an der WT4.2 haben sich auf folgende zu untersuchende Antriebsstrangkomponenten geeinigt:

- Verbrennungsmotor
- Starter/Generator
- Automatikgetriebe und/oder automatisiertes Getriebe.

Hauptaufgabe ist die Untersuchung und Validierung der Echtzeitfähigkeit und der Konfigurationsmöglichkeiten der in den vorhergehenden WP’s definierten Architekturen, Tools und Kommunikationsmechanismen an ausgewählten Applikationsbeispielen. Die Ergebnisse werden an firmenspezifischen Demonstratoren dargestellt. Zusätzlich besteht die Option Teilergebnisse an einem firmenübergreifenden gemeinsamen Demonstrator zu erproben.

WT4.3 Telematics

Mit Multimedia, Internet, Telekommunikation und Telematikdiensten sind Technologien ins Fahrzeug zu integrieren, die zum größten Teil von IT- und Consumerindustrie getrieben werden und von sehr verschiedenen Lieferanten kommen können. Typischerweise sind deren Produktzyklen deutlich kürzer als diejenigen von Fahrzeugen. Fahr-

zeuge stellen nur einen kleinen Markt, der jedoch technologisch hohe Anforderungen bei beherrschbaren Kosten stellt.

Die Leistungsfähigkeit der im EAST-EEA Projekt erarbeiteten Architekturkonzepte und Methoden sollen für die Telematik-Domäne durch die Realisierung eines oder mehrerer Szenarios zu Download und Installation von Software ins Fahrzeug untersucht werden. Hierzu stehen verschiedene Szenarien zur Auswahl:

- Szenario Nutzerprofile:

  Der Benutzer eines Fahrzeuges wird anhand einer Chipkarte oder eines biometrischen Merkmals erkannt, sein Profil von seinem Home-Server geladen und so stets die vertraute Umgebung eingestellt.

- Szenario Neuer Telematikdienst:

  Die Software zu einem bisher nicht im Fahrzeug verfügbarer Telematikdienst wird über eine Funkschnittstelle ins Fahrzeug geladen und automatisch installiert. Den Fahrzeuginsassen steht anschließend eine neue Funktionalität zur Verfügung.

- Szenario Funktionserweiterung:

  Über Mikrowellensensoren kann ein Hindernis um das Fahrzeug detektiert werden. Dies kann sowohl für die Einparkhilfe als auch für Abstandswarnung und Precrash-Meldung verwendet werden. Diese Funktionen werden mit denselben Sensoren ermöglicht. Eine zunächst nur als Einparkhilfe verkaufte Lösung kann durch Softwareupgrade um zusätzliche Funktionen erweitert werden. Eine Anbindung in das Fahrzeug-Netzwerk ist über CAN geplant.

Zu Beginn der Arbeiten in WT4.3 sind diese Szenarien bzgl. ihrer Leistungsfähigkeit zur Validation der Konzepte aus WP2 und WP3 zu analysieren. Eine den verfügbaren Ressourcen und adäquate Auswahl ist zu treffen.

WT4.4 Human Machine Interface

Ein Demonstrator für Human Machine Interface ist in Deutschland nicht geplant. Dieser wird im Gesamtprojekt unter italienischer Führung (Fiat, Magneti Marelli) unter Beteiligung einzelner deutscher Partner realisiert. Damit ist hinsichtlich der Validierung ein Rückfluss ins deutsche Konsortium gewährleistet.

WT4.5 Chassis

Der Validator im Bereich der Fahrwerkselektronik hat zur Aufgabe die Ergebnisse der Arbeitspakete WP2 und WP3 in Bezug auf die Erfüllung der Anforderungen aus diesem Technologiebereich zu validieren. Der Wesentliche nachzuweisende Punkt wird hierbei in einer offenen Systemarchitektur gesehen, in der die Verschiebbarkeit von Funktionsnetzwerken zwischen unterschiedlichen Plattformen (z.B. Steuergeräten) möglich ist. Eine rechtzeitige Bereitstellung von Architekturbeschreibungsmitteln (Sprache und Werkzeugumgebung) sowie einer domänenübergreifenden, einheitlichen Schnittstellenbeschreibung und einer zumindest prototypischen Implementierung der spezifizierten Middleware als Ergebnisse der Arbeitspakete 1, 2 und 3 ist dabei Voraussetzung.

Um den Nachweis der Verschiebbarkeit von Softwarekomponenten zu erbringen, wird der Demonstrator von 3 verschiedenen deutschen

Projekt-Partnern realisiert, wobei unterschiedliche Entwicklungsprozesse zum Einsatz kommen: BMW wird eine Überlagerungslenkung (Steer-by-wire), DaimlerChrysler eine fehlertolerante, redundant ausgeführte Pedaleinheit und Siemens VDO eine elektromechanische Bremse (EMB, brake-by-wire) realisieren.

Der Demonstrator wird in der Realisierung als Laboraufbau konzipiert sein und dadurch ausschließlich diesen Anforderungen genügen.

WP5: Project Management

Die technischen Teilprojekte WP1 bis WP4 werden ergänzt durch das WP5. Dieses Teilprojekt beinhaltet die interne und externe Koordination, das Dokumentenmanagement, das Glossar, die Arbeitsqualität, die Ergebnisverwertung sowie die Sammlung offener Punkte.

1.2 Voraussetzungen des Vorhabens

Um die ambitionierten Ziele zu erreichen, ist eine europäische Zusammenarbeit von Kfz-Herstellern, Zulieferern und Forschungseinrichtungen notwendig. Damit wird eine breite Basis für eine eventuell nachfolgende Standardisierung geschaffen.

Aus diesem Grund ging die Initiative zum Projekt auch von einem europäischen Teilnehmerkreis aus. Das deutsche Konsortium ist Teil des ITEA-Projekts „EAST-EEA“, an dem insgesamt 23 Partner aus Deutschland, Frankreich, Italien und Schweden beteiligt sind.

1.3 Planung und Ablauf des Vorhabens

Die Arbeiten im Vorhaben wurden im Wesentlichen gemäß dem in der Vorhabensbeschreibung vom 10.12.2001 enthaltenen Arbeitsplan durchgeführt. Vor Allem in der Anfangsphase des Projekts gab es zahlreiche Treffen auf Work-Package- und Work-Task-Ebene. Die Ergebnisse wurden in 29 Deliverables dokumentiert.

Zur Steuerung des Projekts wurden die folgenden Gremien installiert:

- Steering Committee (STC):

  Diesem Steuerkreis gehören alle Projektpartner an. Er tagt zweimal jährlich und stellt die höchste Instanz innerhalb des Projekts dar.

- Steering Support Committee (STSC):

  Es unterstützt das STC bei der Steuerung des Projekts und tagt vierteljährlich. Ihm gehören von deutscher Seite die Firmen DaimlerChrysler, BMW, Bosch und SiemensVDO an.

- Externe Projektkoordination:

  Für die Aufgaben des Projektmanagements wurde die Firma Irion Management Consulting (IMC) beauftragt.

Im Rahmen von ITEA wurden insgesamt 3 Projektreviews am 25.09.2002, 22.05.2003 und am 22.06.2004 durchgeführt. Am 21./22.06.2004 fand auf der Flugwerft Schleißheim die Abschlussveranstaltung des Projekts statt.

Daneben war das Projekt EAST-EEA auf den ITEA-Konferenzen 2002 in Amsterdam, 2003 in Leuven und 2004 in Sevilla vertreten.

1.4 Wissenschaftlich-technischer Stand

Im Bereich der Kraftfahrzeug-Elektronik gab es zu Beginn der 90er-Jahre die ersten Ansätze zur Standardisierung von Bussystemen, Protokollen und Standard-Software. Im Jahr 1991 kam das erste Fahrzeug mit einem CAN-Bus (Controller Area Network) [1] auf den Markt. Heute wird dieses Bussystem mit Datenraten bis zu 1 MBit/s von fast allen Kraftfahrzeug-Herstellern zur Vernetzung von Antriebsstrang- und Innenraum-Steuergeräten eingesetzt.

Im Bereich der Telematik hat sich der MOST-Bus (Media Oriented Systems Transport) [2] etabliert, ein optisches Bussystem mit Datenraten bis zu 22 MBit/s. Damit können sowohl Audio- als auch Video-Signale digital zwischen Telematik-Geräten (z.B. Headunit, CD-Wechsler, Soundsystem) übertragen werden.

Für sicherheitskritische Systeme im Fahrwerksbereich ergab sich die Notwendigkeit für ein neues deterministisches und fehlertolerantes Bussystem, das einerseits durch eine Zeitsteuerung die Realisierung verteilter Regelungen über den Bus ermöglicht, auf der anderen Seite durch Mechanismen wie Kanalredundanz und Bus-Guardian die Kommunikation auch bei Leitungsfehlern oder fehlerhaften Steuergeräten aufrecht erhält. Hier hat sich FlexRay [3] mit Datenraten bis zu 10 MBit/s zum De-Facto-Standard entwickelt.

Parallel zu der Entwicklung der Bussysteme gab es ab 1990 erste Überlegungen zur Neustrukturierung der Software auf Steuergeräten. Das erste Ziel war dabei, auch auf den im Kfz eingesetzten Microcontrollern mit beschränkten Ressourcen (8 bis maximal 16 Bit, maximal 2kByte RAM) eine Basisfunktionalität zur Verfügung zu stellen, die aus den Teilen Betriebssystem, Kommunikation und Netzmanagement besteht. Im Jahr 1993 wurde u. A. von BMW, Bosch, DaimlerChrysler, Opel und Siemens das OSEK-Konsortium gegründet, dem ein Jahr später auch die französischen Partner PSA und Renault beitraten. In diesem Rahmen wurde gemeinsam mit anderen Partnern daraus ein heute in der gesamten europäischen Automobilindustrie etablierter Standard geschaffen [4]. Zur Unterstützung zeitgesteuerter Architekturen wurde in 2001 die erste Version eines zeitgesteuerten Betriebssystems (OSEKTime) sowie eines fehlertoleranten Kommunikationsmoduls (OSEK FTCom) vorgestellt.

Interoperability and Portability

Als Folge der sich stark verändernden Randbedingungen (steigende Komplexität, zunehmender Kostendruck, kürzere Entwicklungszeiten, mehr Baureihen, weniger Entwickler) bei der Elektronik-Entwicklung für neue Kraftfahrzeuge ergibt sich die Notwendigkeit, Systeme mehr als in der Vergangenheit baureihenübergreifend wieder zu verwenden. Eine wichtige Voraussetzung für die Wiederverwendung ist eine Strukturierung der Anwendungen in Teilfunktionen, die über standardisierte Kommunikationsprotokolle miteinander kommunizieren.

Diese Portabilität und Interoperabilität soll durch eine so genannte Middleware-Schicht erreicht werden. Dazu existieren unterschiedliche Ansätze, die im Rahmen von EAST-EEA untersucht und harmonisiert werden sollen.

- **TITUS**

  Auf OSEK aufbauend wurde von DaimlerChrysler die TITUS Software-Architektur entwickelt, die auf einem Client-Server-Modell basiert und sich für den Entwurf von Steuerungsfunktionen im Kraftfahrzeug eignet [5, 6, 7]. Diese Funktionen können gemäß der TITUS-Methodik in einem Werkzeug entworfen werden und in einem separaten Prozessschritt auf eine, für die jeweilige Fahrzeugbaureihe optimierte Steuergerätetopologie abgebildet werden [8].

- **EEA (Embedded Electronics Architecture)**

  Von den französischen Projektpartner PSA, Renault, Valeo und SiemensVDO (F) existieren ebenfalls der Entwurf einer Middleware für vernetzte Fahrzeugsysteme sowie der Vorschlag für eine Architektur-Beschreibungssprache „Architecture Implementation Language (AIL)“ [9].

- **Multimedia-Middleware**

  Aus der Unterhaltungs- und Computerindustrie sind verschiedene Ansätze zur Vereinheitlichung und Abstraktion von Kommunikationssystemen gegenüber den darauf zugreifenden Applikationen bekannt. Beispiele für derartige Architekturen finden sich mit JINI auf der Basis der Programmiersprache JAVA, mit HAVi auf Basis des Kommunikationsstandards IEEE1394 (FireWire) und mit UPnP auf Basis der Metasprache XML.

  Im ITEA-Projekt VHE-Middleware wird die Konzeption, Entwicklung und Erprobung einer Software-Infrastruktur, bestehend aus kooperierenden Diensten, für den Heimbereich verfolgt [10]. Dies erfolgt unter besonderer Berücksichtigung der Mobilität des Benutzers und mobiler Endgeräte (z.B. Handy, Palm, PocketPC, PDA, Notebook, ...).

Weitere für das Projekt relevante Ansätze sollen im Verlauf der Projektarbeit identifiziert und bewertet werden.

Development and Validation Tools

Im Bereich der Entwicklungstools für eingebettete elektronische Systeme existieren heute Werkzeuge, die bestimmte Aspekte des Entwicklungsprozesses wie z.B. Requirements Engineering oder Test abdecken. Zur Realisierung einer durchgängigen Entwicklungs-Toolkette fehlt insbesondere eine Architekturbeschreibungssprache, die über alle Phasen des Entwicklungsprozesses eine konsistente Datenhaltung sicherstellt und entsprechende Austauschformate zwischen den Tools zu Verfügung stellt.

Auf diesem Gebiet sind einige interessante Ansätze bekannt:

- **Produktlinien-Ansatz**

  Eine der Grundlagen für den Produktlinien-Ansatz ist „The Product Line Practice Initiative“ des Software Engineering Institutes (SEI) der Carnegie Mellon Universität (CMU) in den USA [11]. Das SEI hat in [12] eine Reihe von Informationen zum Produktfamilien-Engineering zusammengestellt. Es hat so genannte „Practice Areas“ definiert und beschrieben, die für Produktfamilien wichtig sind. Diese Beschreibungen sind ausführlich, allerdings geben sie nur sehr wenige Verfahren und Methoden an und zählen nur auf, welche Dinge wichtig sind.

- **ITEA-ESAPS**

  Das europäische ESAPS-Projekt ist das bisher größte Projekt zum Thema Produktfamilien, und die dort getätigte Arbeit repräsentiert im Wesentlichen den Stand der Technik in diesem Bereich. Dies wurde u.a. auf der First Product Line Conference [13] im August 2000 bestätigt, die einen guten Überblick über den State-of-the-Art in der Produktfamilienentwicklung bot.

- **ITEA-DESS**

  Ziel dieses Projektes ist die Definition einer innovativen, komponentenbasierten Software-Entwicklungsmethodik für eingebettete Echtzeitsysteme auf der Basis von UML-Konzepten, die Erstellung unterstützender Werkzeugumgebungen durch die Integration moderner (UML-)Werkzeuge und der Nachweis der Zweckmäßigkeit der Methodik durch die Implementierung mehrerer Testfälle zur Validierung [14]. Unter eingebetteten Systemen werden in diesem Projekt hauptsächlich Systeme verstanden, die aus Microcontrollern mittlerer bis höherer Leistungsfähigkeit (16-32bit) bestehen und deren Anwendung in den meisten Fällen Echtzeit-Charakter hat, also das Einhalten von festen, periodischen Terminen im Bereich von 1-100ms.

- **Projekt SETTA**

  Im Rahmen der Definition neuer Architekturen für X-by-wire-Systeme, d. h. sicherheitskritischer Fahrwerkssysteme ohne mechanische Rückfallebene, arbeiteten u. A. die Projektpartner DaimlerChrysler, SiemensVDO (D) und Renault in einem EU-Projekt namens SETTA [15, 16], welches sich mit dem Entwurfsprozess verteilter, zeitgesteuerter Systeme beschäftigt. Die Arbeiten beschränken sich in dem Projekt im Wesentlichen auf den simulationsbasierten Entwurf. Mit Hilfe von EAST-EEA sollen offene Punkte, wie z.B. das Austauschformat von Designinformationen und die Integration der Modelle in reale Hardware geschlossen werden.

Über diese Aktivitäten in konkreten Forschungsprojekten hinaus gibt es zahlreiche Initiativen zur Integration von unterschiedlichen Entwicklungstools zu einer durchgängigen Toolkette.

1.5 Zusammenarbeit mit anderen Stellen

Soweit es zur Informationsgewinnung bezüglich der oben genannten Projekte und Initiativen über deren Publikationen hinaus notwendig war, wurden Gespräche mit deren Vertretern geführt.

Eine darüber hinausgehende Zusammenarbeit mit anderen Stellen fand nicht statt.

2 Eingehende Darstellung

2.1 Erzieltes Ergebnis

Bei der eingehenden Darstellung der Ergebnisse sind nachfolgend nur die Arbeitspakete beschreiben, an denen die DaimlerChrysler AG beteiligt war.

WP1: General Aspects

WT1.2 Definition der Allgemeinen Anforderungen (Requirement-Analyse)

In diesem Arbeitspaket wurden die Anforderungen (Requirements) an das Gesamtsystem, d.h. an eine Middleware-orientierte Software-Architektur, erarbeitet. Der Requirements-Prozess wurde dabei in 2 Phasen unterteilt:

1. Identifikation von Requirements
2. Analyse und Klassifikation der Requirements

Bei der Identifikation von Requirements ist es zunächst notwendig, die Beteiligten und deren Rollen im Prozess zu verstehen. Dazu wurde eine sogenannte Stakeholder-Analyse durchgeführt. Normalerweise haben diese „Interessensvertreter“ einen unterschiedlichen historischen Hintergrund und damit auch ein unterschiedliches Verständnis von Requirements, hauptsächlich bezüglich der angemessenen Granularität. Da im EAST-EEA-Projekt sowohl Automobilhersteller als auch Zulieferer, Tool-Hersteller und Forschungseinrichtungen vertreten sind, hilft diese Analyse beim Verständnis der Anforderungen, die von den unterschiedlichen Beteiligten eingebracht werden. Abbildung 1 zeigt die Projektpartner sowie deren Rolle im Projekt.

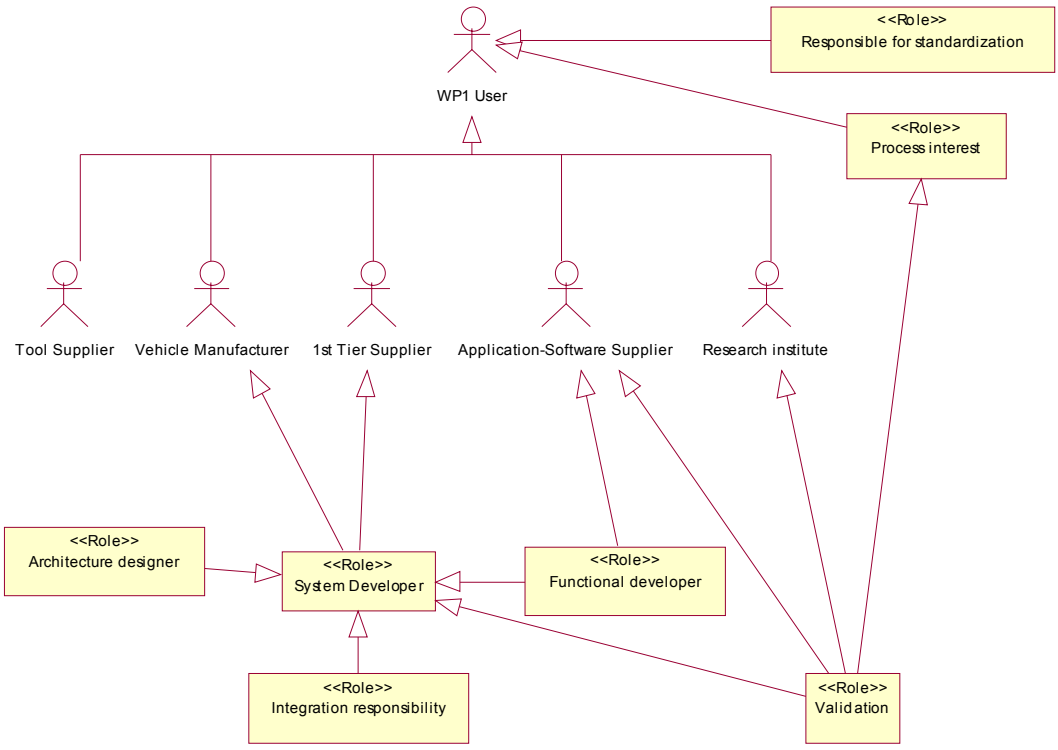

Abbildung 1: Stakeholder-Analyse

Die Erfassung der Requirements wurde in Form von Fragebögen durchgeführt, die anschließend auf Arbeitstreffen mit allen Beteiligten diskutiert wurden. Durch die einheitliche Form wurde die technische Basis für den nachfolgenden Analyseprozess gelegt. Ergebnis waren die folgenden Listen:

- Liste funktionaler Requirements
- Liste nichtfunktionaler Requirements
- Liste von Constraints (Rahmenbedingungen)

Das Ziel der Requirements-Analyse war es, ausgehend von obigen Listen einen Satz von Anforderungen zu erhalten, die

- korrekt,
- eindeutig,
- vollständig,
- konsistent,
- verständlich und
- nachvollziehbar

sind.

Das Ergebnis dieses Requirements-Analyse-Prozesses ist in einem sogenannten Feature-Modell beschrieben. Dabei wird ein einem Feature-Modell sowohl die Abhängigkeit als auch die Interaktion von Features untereinander beschrieben. Abbildung 2 zeigt die Notationen für die Abhängigkeiten „is a“ (Verallgemeinerung von …) und „contains“ (besteht aus …).

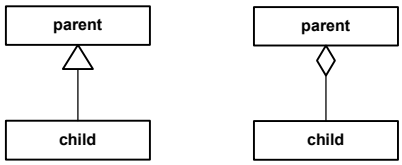

Abbildung 2: Feature-Abhängigkeiten „is a …“ und „contains …“

Für die Beschreibung der Interaktionen werden gestrichelte Pfeile verwendet.

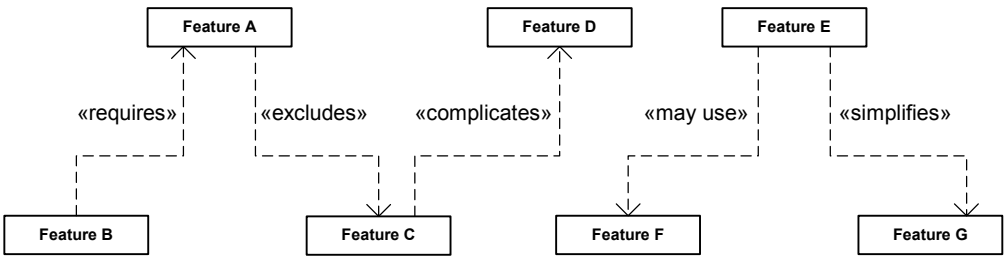

Abbildung 3 zeigt alle möglichen Interaktionen:

- requires (setzt voraus)
- excludes (schließt aus)
- complicates (kompliziert)
- may use (kann benutzen)
- simplifies (vereinfacht)

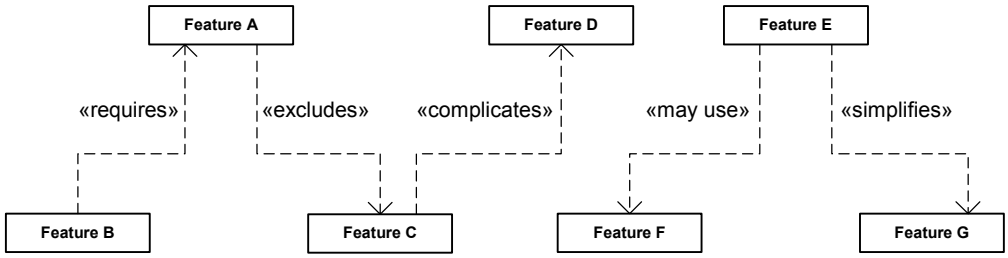

Abbildung 3: Feature-Interaktionen

Mit Hilfe dieser Beschreibungsmittel wurden die in Listenform erfassten Requirements beschrieben und zueinander in Beziehung gesetzt. Die sich daraus ergebenden Feature-Bäume bilden die Basis für die Spezifikationen in den anderen Workpackages.:

- In WP2 werden die Requirements für den Entwurf der Architektur benötigt.
- In WP3 werden die Requirements für die Entwicklung der Architecture Description Language (ADL) genutzt.
- In WP4 sind die Requirements für den Aufbau der Validatoren notwendig.

Um die Requirements in den anderen Workpackages nutzen zu können, wurden die Kriterien zur Einhaltung der Anforderungen in Form von „Use Cases“ erarbeitet und in einem weiteren Deliverable beschrieben. Diese beziehen sich direkt auf die weiter oben beschriebenen Feature-Modelle.

Deliverables	
D1.2	General Requirements
D1.3	Criteria for Evaluation to the Fulfilment of the Requirements

WT1.3 Existierende Lösungen und ein Fahrplan zu ihrer Harmonisierung

In diesem Arbeitspaket wurde der Stand der Technik beim Entwurf von Elektroniksystemen im Fahrzeug zu Beginn des Projekts erarbeitet. Dies bezieht sich auf:

- Entwicklungsmethodik
- Entwicklungswerkzeuge
- Technologien
- Architekturen
- Problemlösungsansätze

Das Ergebnis zeigt, dass in den beteiligten Firmen zwar teilweise gleiche Tools eingesetzt werden, die Ansichten über die optimale Entwicklungsmethodik aber sehr divergent sind.

Im zweiten Teil des Arbeitspakets wurde ein Konvergenzplan erarbeitet, wie die im Projekt entstehenden Lösungen für

- Middleware-Technologien
- Entwicklungsmethodik auf Basis einer Architecture Description Language (ADL)

auf Basis der heute existierenden Lösungen in der europäischen Automobilindustrie umgesetzt werden können.

Dazu wurden Vorschläge für kurz-, mittel- und langfristige Realisierungsmöglichkeiten ausgearbeitet.

Deliverables	
D1.4	State-of-the-Art Report
D1.6	Convergence Plan Report

WT1.4 Verfahren zum Umgang mit geistigen Eigentumsrechten und zukünftige Geschäftsmodelle

Die im Projekt erarbeiteten technischen Möglichkeiten zum Austausch von Software auf Steuergeräten erfordern neue Geschäftsmodelle zwischen Automobilherstellern und Zulieferern. Wird die Software als eigenständiges Wirtschaftsgut betrachtet, so muss neben dem klassischen Zulieferer ein Softwarehersteller als weiterer Lieferant in die Überlegungen einbezogen werden.

Vor diesem Hintergrund wurden unterschiedliche Geschäftsmodelle untersucht. Ein Schwerpunkt lag dabei auf der Vertragsgestaltung, um insbesondere die Aspekte Intellectual Property und Haftungsfragen im Zusammenhang mit Abnahme und Freigabe von Software abzudecken.

Die Sichtweisen der einzelnen Projektpartner wurden dabei durch Fragebögen erfasst und ausgewertet. Dabei ergaben sich zwei mögliche zukünftige Geschäftsmodelle:

1. Rahmenvertrag zwischen Automobilhersteller und Softwarelieferant, kombiniert mit Einzellizenzverträgen zwischen Softwarelieferanten und Zulieferern
2. General-Lizenzvertrag: der Automobilhersteller tritt als Lizenznehmer des Softwarelieferanten auf und übernimmt die Weiterlizensierung der Software an die Zulieferer.

Deliverables	
D1.8	Analysis of the Current Situation regarding existing Business Models
D1.10	Proposal of an EAST-EEA Business Model

WP2: Interoperability and Portability

WT2.1 General Implementation Framework

Das Hauptziel von EAST-EEA war die Definition einer middlewarebasierten Softwarearchitektur für eingebettete Elektroniksysteme in allen Fahrzeugdomänen. Dabei sind die Anforderungen an diese Middleware in den einzelnen Fahrzeugdomänen sehr unterschiedlich. Während in der Domäne "Innenraum" die Forderung nach einer Verteilbarkeit von Funktionen über mehrere Steuergeräte dominiert, sind es in der "Antriebsstrang"-Domäne die harten Echtzeitbedingungen. Demgegenüber steht in der "Fahrwerk"-Domäne die Einhaltung besonderer Sicherheits-anforderungen im Vordergrund. Allen diesen klassischen Fahrzeug-domänen ist gemeinsam, dass selbst in naher Zukunft keine dynamischen Veränderungen der Konfiguration zur Laufzeit vorgesehen sind. In der Telematik ist es jedoch genau diese Forderung nach einer dynamischen Erweiterbarkeit des Systems durch Nachladen von neuen Applikationen, welche die Softwarearchitektur bestimmt.

Das Ergebnis der Erfassung von Anforderungen ist eine strukturierte Liste von Anforderungen einschließlich einer Bewertung hinsichtlich ihrer Relevanz für die einzelnen Fahrzeugdomänen. Diese Anforderungen sind innerhalb des Projektes in die folgenden Kategorien eingeteilt:

- Application Component Notification
- Communication Configuration
- Information Exchange (z.B. Medien-Unabhängigkeit)
- Safety (z.B. Fehlertoleranz)
- System Configuration Information
- Software Download
- Network Resource Management
- General Services (z.B. Monitoring)
- Security (z.B. Authentifikation, Encryption)
- System Configuration (z.B. Skalierbarkeit)
- Mode Management (z.B. Powermanagement)
- Portability
- Verification & Testing
- Diagnosis

Um einen Transfer der Projektergebnisse in Serienprojekte zu gewährleisten, ist es notwendig, einerseits auf laufende Standardisierungsaktivitäten (z.B. OSEK) aufzubauen. Andererseits gilt es, Konzepte von einzelnen Firmen oder anderen Projekten und Konsortien, hinsichtlich ihrer Relevanz für EAST-EEA zu bewerten (z.B. HIS, AEE, TITUS).

Von Seiten DaimlerChrysler wurde dabei die TITUS-Software-Architektur eingebracht. Im TITUS-Projekt entwickelte DaimlerChrysler eine Client/Server-basierte Softwarearchitektur, die einen Middleware-Layer zur abbildungstransparenten Kommunikation zur Verfügung stellt. Sie baut auf dem Betriebssystem OSEK und anderen Standard-Softwareschichten gemäß Abbildung 4 auf:

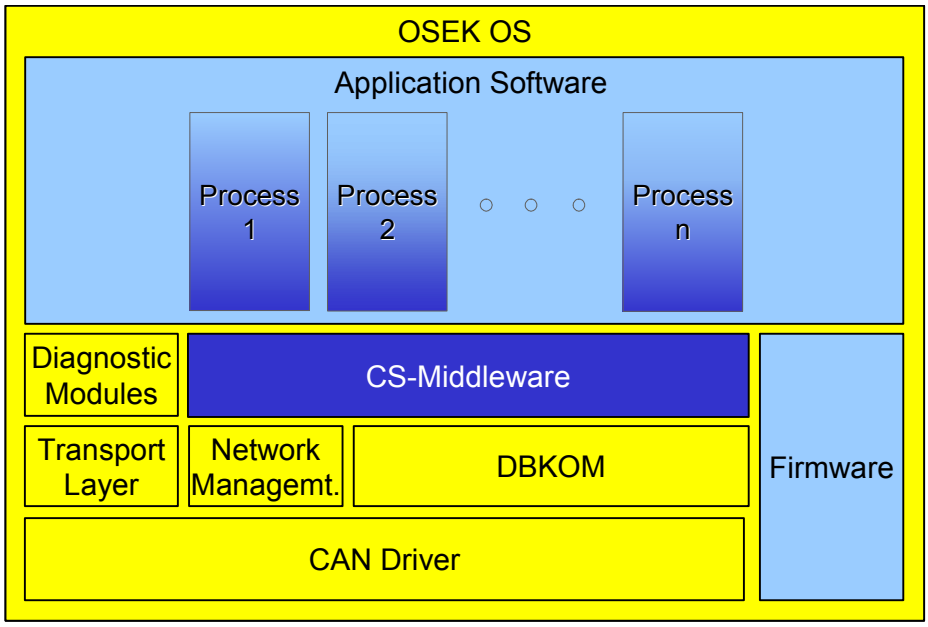

Abbildung 4: TITUS-Software-Architektur

CAN Driver

Standardisierter Device-Treiber für den CAN-Bus, für viele unterschiedliche Hardwareplattformen verfügbar.

DBKOM

Kommunikationsmodul, stellt der Anwendung eine signalorientierte Schnittstelle zur Verfügung (in Unterscheidung zu OSEK-COM V2.0, das nur eine Nachrichten-Schnittstelle hat). DBKOM unterstützt sowohl interne Kommunikation (innerhalb eines Steuergeräts) als auch externe Kommunikation (über den Bus mit anderen Steuergeräten).

Network Management

OSEK Network Management

Transport Layer

ISO Transport Layer

Diagnostic Module

Keyword Protocol 2000

Firmware

Hardware-Abstraktionsschicht zum standardisierten Zugriff auf Sensorik und Aktorik.

Application Process

Ein Applikationsprozess ist die kleinste Einheit von Anwendungsfunktionen. Typischerweise besteht eine Anwendung aus mehreren Prozessen, die auf unterschiedlichen Steuergeräten ablaufen.

CS Middleware

Client Server Middleware: Sie stellt den Anwendungen ein Kommunikationsprotokoll basierend auf dem Request/Reply-Mechanismus zur Verfügung. Der Datenaustausch zwischen den Anwendungsprozessen geschieht über vordefinierte Protokolle, die wiederum Methodenaufrufe enthalten. Die Middleware wird zur Konfigurationszeit automatisch generiert bleibt statisch über die gesamte Lebenszeit.

Das Deliverable D2.2 wurde federführend von DaimlerChrysler erstellt und umfasst eine Kurzbeschreibung der heute in der Automobilelektronik eingesetzten Standard-Softwaremodule.

Deliverables	
D2.1	Embedded Software Structure Requirements
D2.2	Description and Analysis of existing Solutions

WT2.2 Middleware Specification

Zu Beginn des Projekts wurde das Ziel verfolgt, einen einheitlichen Middleware-Ansatz für alle Fahrzeugdomänen zu erarbeiten. Bei der Projektarbeit wurde im Zuge der Erfassung von Requirements (WT2.1) festgestellt, dass die Anforderungen an die Middleware-Services aus den Fahrzeugdomänen heraus sehr unterschiedlich sind.

Daher wurde im Konsens aller Projektpartner folgende Vorgehensweise festgelegt:

1. Domänenspezifische Definition von Middleware-Services
2. Erarbeitung von Gemeinsamkeiten zwischen den Domänen in WT2.4

Die in 1. spezifizierten Middleware-Dienste wurden in WT2.3 implementiert und im Workpackage 4 validiert.

DaimlerChrysler hat dabei in den folgenden Domänen mitgearbeitet:

Body Electronics

Der Bereich "Innenraum" oder "Body Electronics“ heutiger Mittel- bis Oberklasse-Fahrzeuge besteht aus einer großen Anzahl von Steuergeräten unterschiedlichster Hersteller, die z.B. über ein CAN-Bussystem miteinander kommunizieren. Typische Funktionen, die diese Steuergeräte ausführen, sind z.B. Fensterheber, Klimaanlage, Sitzverstellung und Lichtsteuerung. Ein wesentliches Merkmal dieser Funktionen ist, dass sie verteilt, d.h. von unterschiedlichen Steuergeräten, ausgeführt werden.

Bei der heutigen Elektronikentwicklung werden die Steuergeräte vielfach als Einheit von Hardware und darauf laufender Software, also als „Black Box“, betrachtet. Die in EAST-EEA erarbeiteten Methoden erlauben eine Trennung von Soft- und Hardware, d.h. eine separate Betrachtung aller für die Softwareentwicklung notwendigen Aspekte unabhängig von der zugrunde liegenden Hardware.

Durch die EAST-EEA-Vorgehensweise sollen die folgenden Vorteile aufgezeigt werden:

- Trennung von Hardware und Software: Unterschiedliche Zulieferanten für Funktions-Software und Steuergeräte-Hardware
- Portabilität: Software-Funktionen eines Zulieferers können auf Steuergeräte anderer Zulieferer verteilt werden.
- Interoperabilität: Die EAST-EEA-Middleware stellt einheitliche Dienste zur Verfügung.

Die daraus resultierende Software-Architektur in der Domäne "Innenraum" besteht aus den in Abbildung 5 dargestellten Bausteinen. Jeder dieser Bausteine ist verantwortlich für eine Reihe von Diensten. Anwendungsfunktionen (Application) haben keinen direkten Zugriff auf diese Module, sondern müssen die von der Middleware angebotenen Dienste nutzen.

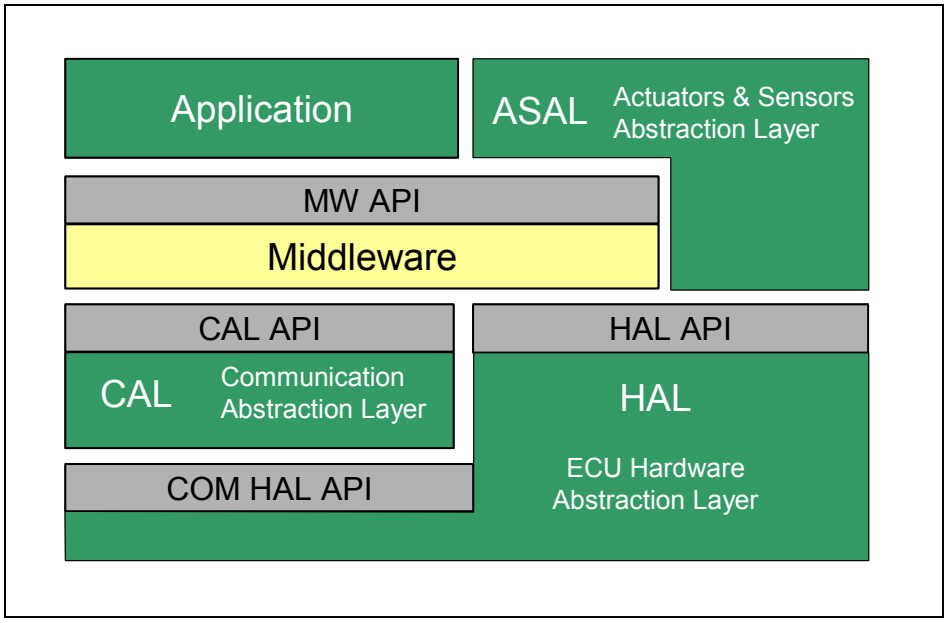

Abbildung 5: Software-Architektur Innenraum

Bausteine der Software Architektur für den Innenraum:

- HAL (Hardware Abstraction Layer):
  Abstraktion der Funktionalität von der Steuergeräte-Hardware und -Peripherie (I/O)
- ASAL (Actuator and Sensor Abstraction Layer):
  Abstraktion der Funktionalität von Sensorik / Aktuatorik
- CAL (Communication Abstraction Layer):
  Abstraktion der Funktionalität vom Kommunikationsmodul (Busse, Protokolle)
- Middleware:
  Dienste für die Portabilität und den Datenaustausch verteilter Anwendungsfunktionen
- Application:
  Implementiert die Anwendungsfunktion

Portabilität von Anwendungsfunktionen

Um das Ziel der Verschiebbarkeit von Applikationen zu erreichen, muss der Datenaustausch zwischen lokalen und entfernten Applikationen in einer einheitlichen Art und Weise ablaufen. Abbildung 6 zeigt zwei vernetzte Steuergeräte mit einer Anzahl von Applikationen. Dabei kommuniziert Applikation K sowohl mit einer Applikation L auf demselben Steuergerät A als auch mit einer Applikation N auf einem entfernten Steuergerät B. Alle Applikationen nutzen dabei ausschließlich das API (Application Programmers Interface) der Middleware.

Das Routing der Daten, d.h. die Entscheidung, ob die Daten lokal oder über den Bus versendet werden müssen, erfolgt in der Regel durch die Middleware (zur Generierungszeit). Sollte der CAL wie im Fall von OSEK-COM V3.0 über diese Funktionalität verfügen, so kann diese durch die Middleware genutzt werden. In Abbildung 6 ist dies durch eine gestrichelte Linie angedeutet.

Durch die statische Struktur im Innenraum ist es damit möglich, nach der Platzierung aller Applikationen auf die Steuergeräte die Middleware entsprechend den notwendigen Kommunikations-verbindungen automatisch zu generieren. Dabei kann der Ressourcenverbrauch auf den Steuergeräten z.B. dadurch optimiert werden, dass lokale Kommunikation über globale Variablen realisiert wird.

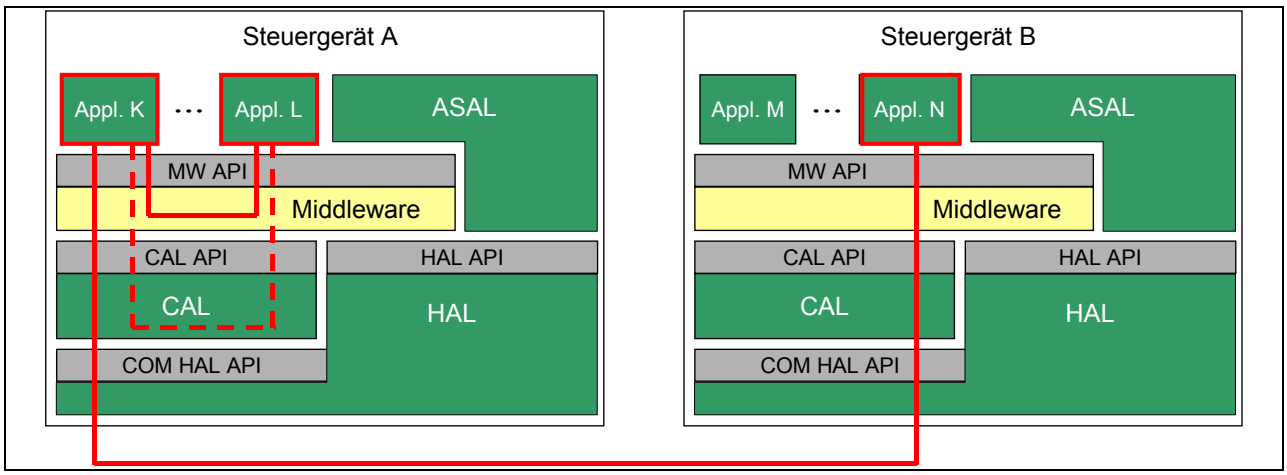

Abbildung 6: Kommunikation zwischen lokalen und entfernten Applikationen

Abstraktion der Hardware

Ein weiterer wichtiger Aspekt ist der einheitliche Zugriff auf Sensorik und Aktuatorik. Das EAST-EEA- Konzept sieht dabei zusätzlich zu dem vom HIS-Konsortium vorgeschlagenen HAL (Hardware Abstraction Layer) einen ASAL (Actuator and Sensor Abstraction Layer) vor, der die spezifischen Eigenschaften von bestimmten Sensoren und Aktuatoren kapselt. Die grundsätzliche Struktur ist in Abbildung 7 dargestellt.

In dem gezeigten Beispiel kommuniziert eine Applikation K auf Steuergerät B mit einem Sensor S1, der an Steuergerät B angeschlossen ist. Dabei kann es sich z.B. um einen Lichtsensor handeln, wobei der ASAL die Vorverarbeitung der Sensorinformation übernimmt und der Applikation eine aufbereitete Information (z.B. Tag/Nacht) zur Verfügung stellt. Da der Informationsaustausch über

die Middleware abläuft, kann diese Applikation ohne Modifikation auch auf das Steuergerät A verschoben werden, wobei der Sensor an Steuergerät B verbleibt.

Durch die beschriebenen Mechanismen können zur Generierungszeit somit Applikationen flexibel auf Steuergeräte verteilt bzw. verschoben werden. Dies kann Bestrebungen unterstützen, Funktionalität auf wenigen Rechenknoten zusammenzufassen und damit ggf. Steuergeräte zu reduzieren.

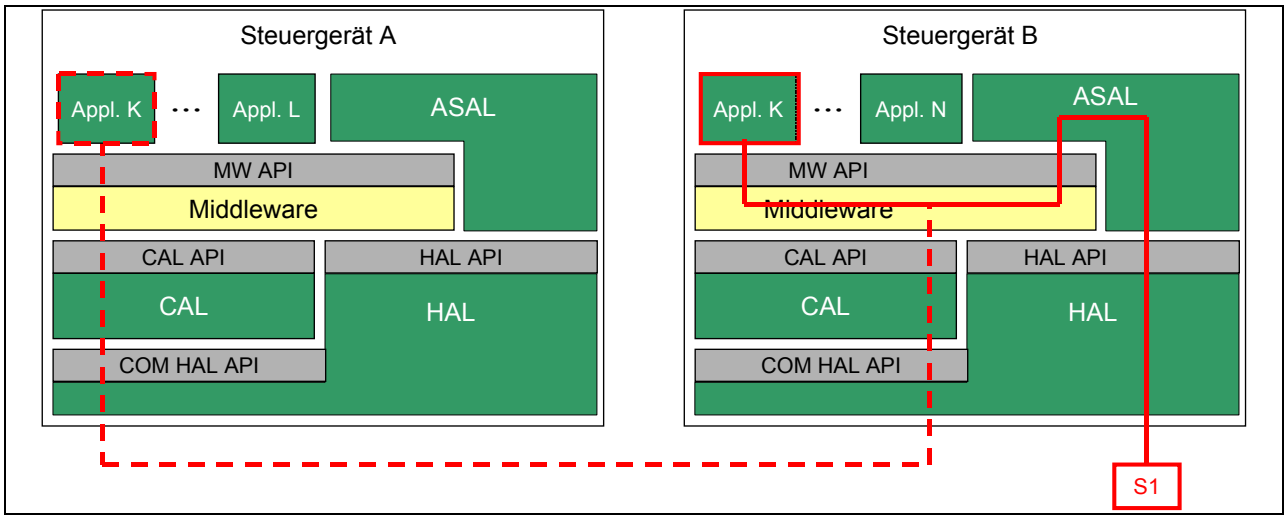

Abbildung 7: Sensorzugriff von lokalen und entfernten Applikationen

Chassis Electronics

Im Bereich der Fahrwerks-Elektronik wurde im Verlauf der Requirementsanalyse festgestellt, dass ein großer Teil der zur Integration unterschiedlicher Anwendungen auf einem Steuergerät notwendigen Dienste durch die heute eingesetzten Standard-softwaremodule (OSEKtime, OSEK-COM, FTCom, …) zur Verfügung gestellt werden (siehe Abbildung 8).

Beim Aufbau sicherheitskritischer Systeme im Fahrwerk ist es notwendig, wichtige Ereignisse wie z.B. den Wechsel von Systemzuständen nachvollziehbar zu dokumentieren.

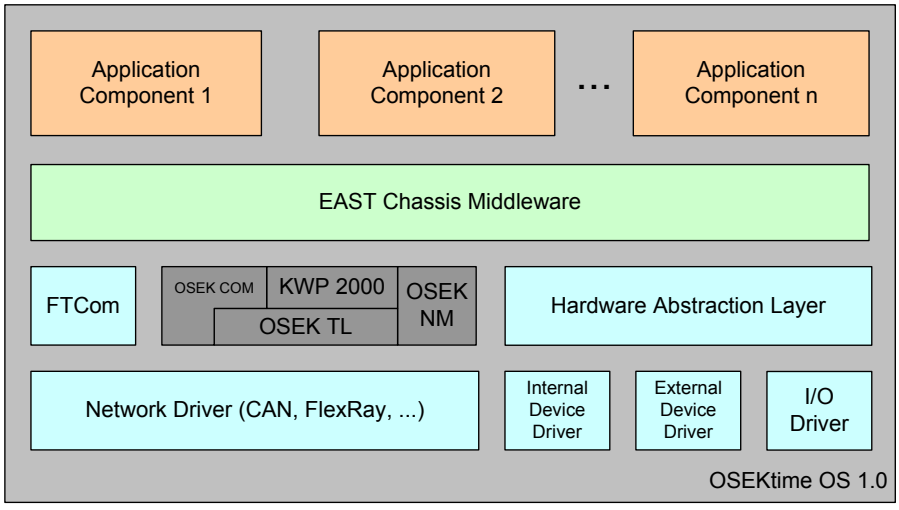

Abbildung 8: Software-Architektur Chassis

Defizite bestehen dabei vor allem in der zeitgetreuen Erfassung von Daten und der Sicherstellung von konsistenten Zuständen im Netzwerk.

Daher wurde unter Anderem ein Middleware-Dienst „Logging-Service“ spezifiziert:

Die Middleware stellt diesen Dienst allen Applikationen auf allen Steuergeräten zur Verfügung (siehe Abbildung 9). Auf einem ausgewählten Steuergerät (ECU 2) wird ein Logfile angelegt, in das wichtige Daten der Applikationen mit einem entsprechenden Zeitstempel eingetragen werden. Der Dienst ist für die einzelnen Applikationen völlig transparent, da die Übertragung der Daten zum richtigen Steuergerät durch die Middleware übernommen wird.

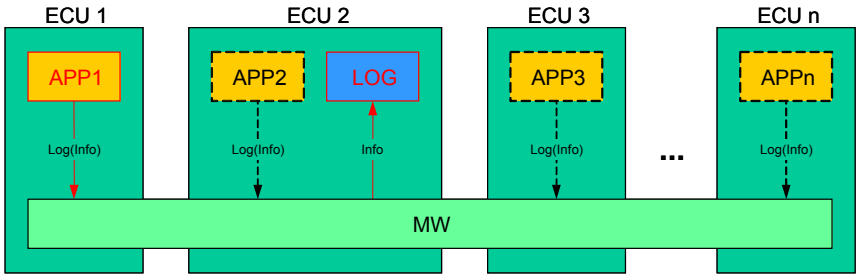

Abbildung 9: Middleware-Service: Logging

Telematik

Wie bereits im Kapitel Stand der Technik ausgeführt, gibt es in der Unterhaltungs- und Computerindustrie verschiedene Ansätze zur Vereinheitlichung und Abstraktion von Kommunikationssystemen gegenüber den darauf zugreifenden Applikationen.

Im Projekt wurde als viel versprechender Ansatz HAVi [17] auf Basis des Kommunikationsstandards IEEE1394 (FireWire) ausgewählt. HAVi definiert eine Middleware mit folgender Funktionalität:

- Automatische Erkennung von Teilnehmern im Netzwerk
- Koordination der Funktionen unterschiedlicher Geräte
- Installation von Anwendungen und Bedieneroberflächen
- Interoperabilität zwischen Geräten unterschiedlicher Hersteller

Im Bereich der Fahrzeug-Telematik hat sich jedoch der MOST-Bus (Media Oriented Systems Transport) [2] etabliert, ein optisches Bussystem mit Datenraten bis zu 22 MBit/s. Damit können sowohl Audio- als auch Video-Signale digital zwischen Telematik-Geräten (z.B. Headunit, CD-Wechsler, Soundsystem) übertragen werden.

Um die HAVi-Funktionalität im Fahrzeug nutzen zu können, wurde die Software-Architektur so modifiziert, dass sowohl IEEE1394-Geräte als auch MOST-Geräte in einem Systemverbund betrieben werden können (siehe Abbildung 10). Dabei wurden auf der einen Seite existierende HAVi-Komponenten adaptiert, andererseits mussten neue Komponenten zur Interaktion mit dem MOST-Bus eingeführt werden.

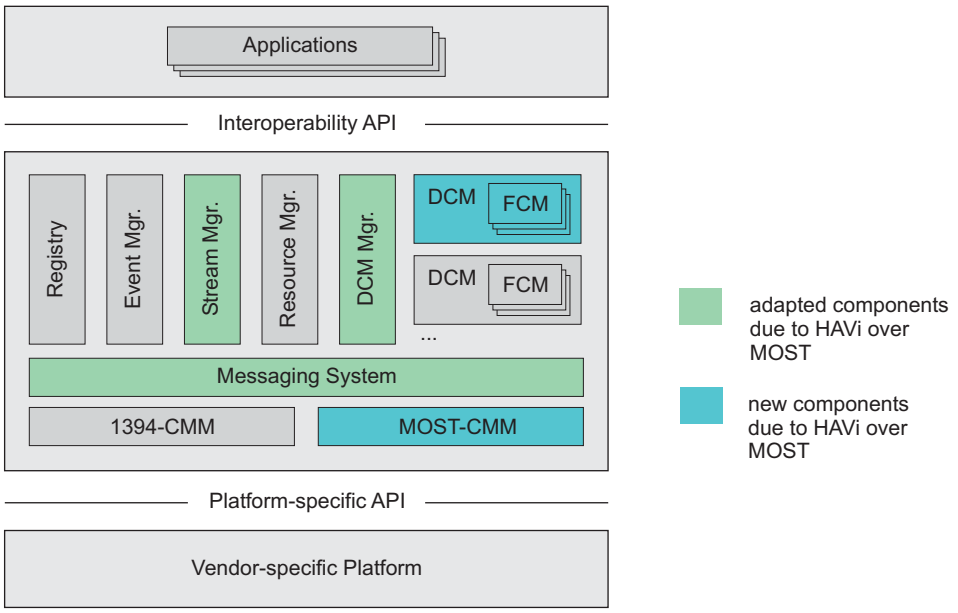

Abbildung 10: Modifizierte HAVi-Software-Architektur für MOST

Deliverables	
D2.3	Middleware Specification, 1st Release
D2.7	Embedded Basic Software Structure Specification, 2nd Release

WT2.3 Middleware Implementation

Die in Worktask 2.2 spezifizierten Middleware-Dienste wurden im Rahmen dieser Worktask zum Zwecke der Validierung implementiert. Dies geschah in enger Abstimmung mit den Worktasks 4.x. Dabei gab es in den einzelnen Domänen unterschiedliche Vorgehensweisen:

- Implementierung in Worktask 2.2, Code wurde Worktask 4.x zur Verfügung gestellt.
- Beispielimplementierung in Worktask 2.2, Implementierung auf Zielsystem wurde in Worktask 4.x durchgeführt.
- Implementierungsanleitung in Worktask 2.2, vollständige Implementierung in Worktask 4.x.

Details der Implementierung werden als vertrauliche Information betrachtet und stehen nur den an der Worktask beteiligten Partnern zur Verfügung.

Deliverables	
D2.4	Embedded Basic Software Implementation Users Guide
D2.5	Embedded Basic Software Implementation

WT2.4 Communication Layer Specification (Common Parts)

Im Zuge des Projekts wurde festgestellt, dass im Bereich der Basis-Kommunikationsschichten kein Spezifikationsbedarf besteht, da die gesamte für die Middleware notwendige Funktionalität von den heute verfügbaren Standard-Softwarekomponenten bereits erbracht wird.

Daher wurde dieses Arbeitspaket dazu genutzt, Gemeinsamkeiten zwischen den einzelnen Fahrzeugdomänen herauszuarbeiten.

Eine zunehmende Anzahl von Diensten überschreiten die historisch gewachsenen – sowohl technischen als auch organisatorischen – Domänengrenzen. Diese Dienste haben jedoch unterschiedliche Auswirkungen auf das Gesamtsystem. Durch eine Betrachtung der Auswirkungen der Dienste auf das Gesamtsystem ergibt sich die folgende Kategorisierung:

1. Dienste, die ein fahrzeugweites Konzept voraussetzen (z.B. Fahrzeug-Energiemanagement)
2. Dienste, die eine Wiederverwendung über Domänengrenzen hinweg ermöglichen. Diese Kategorie unterteilt sich in die Wiederverwendung
   a) von Konzepten: z.B. Fehlermanagement.
   b) von Algorithmen und Software-Code: z.B. Flashloader für den Software-Download
   c) von steuergeräte-internen Schnittstellen: z.B. Middleware Schnittstellen zur Verschiebbarkeit
3. Dienste, die einen domänenübergreifenden Zugriff auf Schnittstellen oder Daten ermöglichen.
   a) Domänenübergreifenden Zugriff über identische Schnittstellen: z.B. Diagnosekonzept
   b) Domänenübergreifende Verwendung von Daten: z.B. Fahrzeugglobale Sensorsignale

Diese Dienste bieten ein hohes Potential für Vereinheitlichung und Vereinfachung, können aber auch für einzelne Beteiligte in der Wertschöpfungskette Mehraufwände bedeuten. Daher sind in jedem Einzelfall die Vor- und Nachteile abzuwägen. Um das tun zu können, müssen die Auswirkungen genau verstanden sein. Im EAST-EEA-Projekt werden spezifische Auswirkungen, die domänenübergreifende Dienste erzeugen, anhand konkreter Beispiele analysiert. Eines dieser Beispiele ist der Flashloader für den Software-Download in ein Steuergerät.

Software-Download ist heute in der Steuergerätewelt des Fahrzeugs eng verknüpft mit der Diagnose. Um eine flexible Lösung für einen separaten Software-Download zu ermöglichen, muss die enge Verknüpfung zur Diagnose (Fehlererkennung, Fehlerbehandlung, und Reaktion auf Fehler), die unter anderem auch im ISO Standard 14230 (KWP2000) verankert ist, aufgebrochen werden.

Zunächst wird ein Downloadvorgang in Form eines Szenarios beschrieben. Anschließend werden die sich daraus ergebenden domänenübergreifenden Aspekte betrachtet.

Beschreibung des Szenarios:

Auf einem Steuergerät (Empfänger) ist ein Flashloader und die EAST-EEA-Middleware installiert. Auf einem zweiten Steuergerät (Sender), welches über ein Bussystem mit dem ersten Steuergerät verbunden ist, ist die zu ladende Software vorrätig.

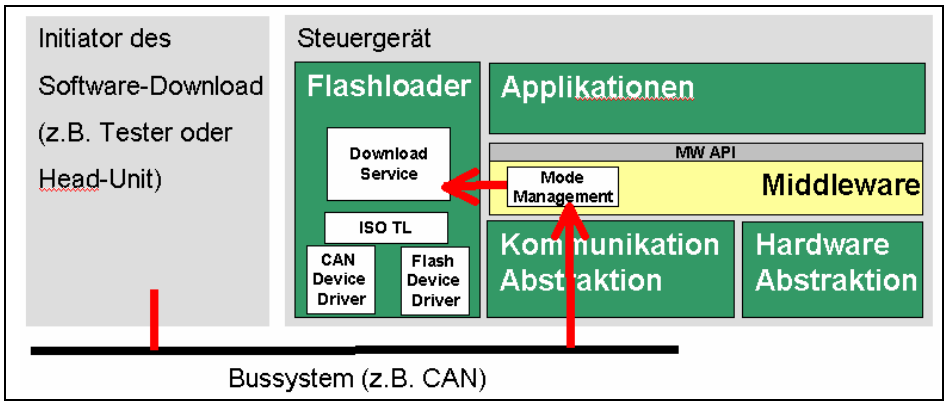

Abbildung 11: Initiierung des Software-Download Vorgangs

Der Sender schickt ein Signal an den Empfänger, damit dieser sich selber in den Diagnosemodus versetzt. Die Middleware des Empfängers initiiert den Wechsel des aktiven Modus auf dem Steuergerät - d.h. Applikationen werden heruntergefahren, Flashloader wird in RAM geladen, usw. (Abbildung 11).

Mit aktiviertem Flashloader wird der Download Vorgang durchgeführt (Abbildung 12). Nach einer Sicherheitsüberprüfung auf Authentizität des Senders und der zu ladenden Software wird die Software über das Bussystem angefordert. Nach Beendigung des Ladevorgangs wird die Software im Flashspeicher initialisiert. Je nach Art und Umfang der Änderungen muss ein Boot-Vorgang oder ein spezieller Moduswechsel erfolgen.

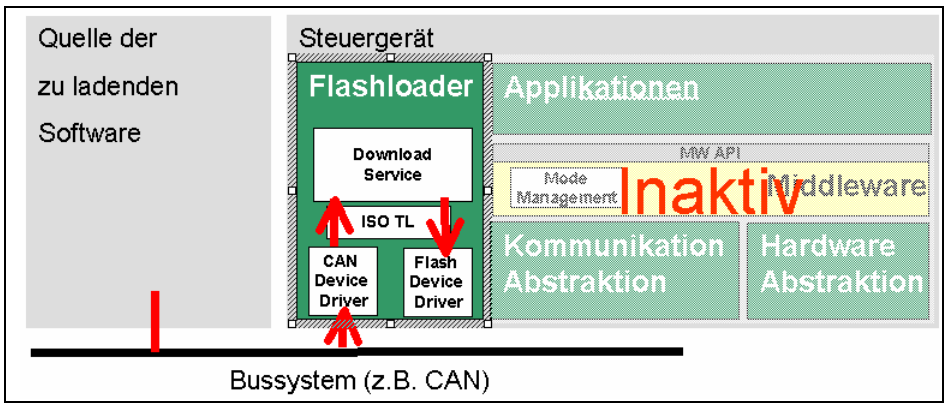

Abbildung 12: Aktiver Flashloader

Die neue Software läuft auf dem Empfängersteuergerät. Der Flashloader ist deaktiviert. Es ist der Normalmodus aktiviert.

Analyse in Hinblick auf domänenübergreifende Aspekte:

Für das Szenario sind drei Softwareanteile notwendig.

- Software auf dem Sender zur Initiierung des Szenarios. Diese Software sollte auf allen Steuergeräten dieselbe Schnittstelle vorfinden, um den Download Vorgang zu initiieren.

- Software auf dem Empfänger zum Modewechsel und zur Aktivierung des Download-Mode.
- Software auf dem Empfänger zur Ausführung des Downloads. Darunter fällt auch die Sicherheitsüberprüfung.

Die Analyse bereits eines einzigen Szenarios zeigt die Vielfältigkeit der domänenübergreifenden Dienste auf. Erst eine genaue Analyse und ein Verständnis der einzelnen darin verborgenen Aspekte ermöglichen eine Aussage über Kosten und Nutzen dieser Dienste.

Deliverables	
D2.6	Embedded Basic Software Structure – Common Parts Specification

WP3: Development and Validation Tools

Neben der Betrachtung der Middleware und deren Dienste ist eine Methodik zur Komplexitätsbeherrschung beim Entwurf verteilter, eingebetteter Kfz-Steuerungssoftware notwendig. Ein Möglichkeit dazu ist die Gliederung in mehrere Abstraktionsebenen.

Eine auf Abstraktionsebenen basierende Modellierungsmethodik verbindet horizontale Entwurfsinformationen (Simulationen auf einer Abstraktionsebene, Verifikation & Validierung) mit vertikalen Entwurfsinformationen (Verfeinerungen zwischen den Abstraktionsebenen). Zentraler Bestandteil einer solchen Methodik ist die EAST-EEA-ADL (Architecture Description Language). Sie besteht aus folgenden Abstraktionsebenen, hier Architekturen genannt:

- Fahrzeugprojekt
- Hardware Architektur
- Funktionale Analyse Architektur
- Technische Architektur
- Funktionale Design Architektur
- Ausführungsarchitektur
- Logische Architektur

Der Informationsfluss zwischen den Abstraktionsebenen ist in Abbildung 13 dargestellt. Die Modellierungselemente der Architekturen werden nachfolgend skizziert.

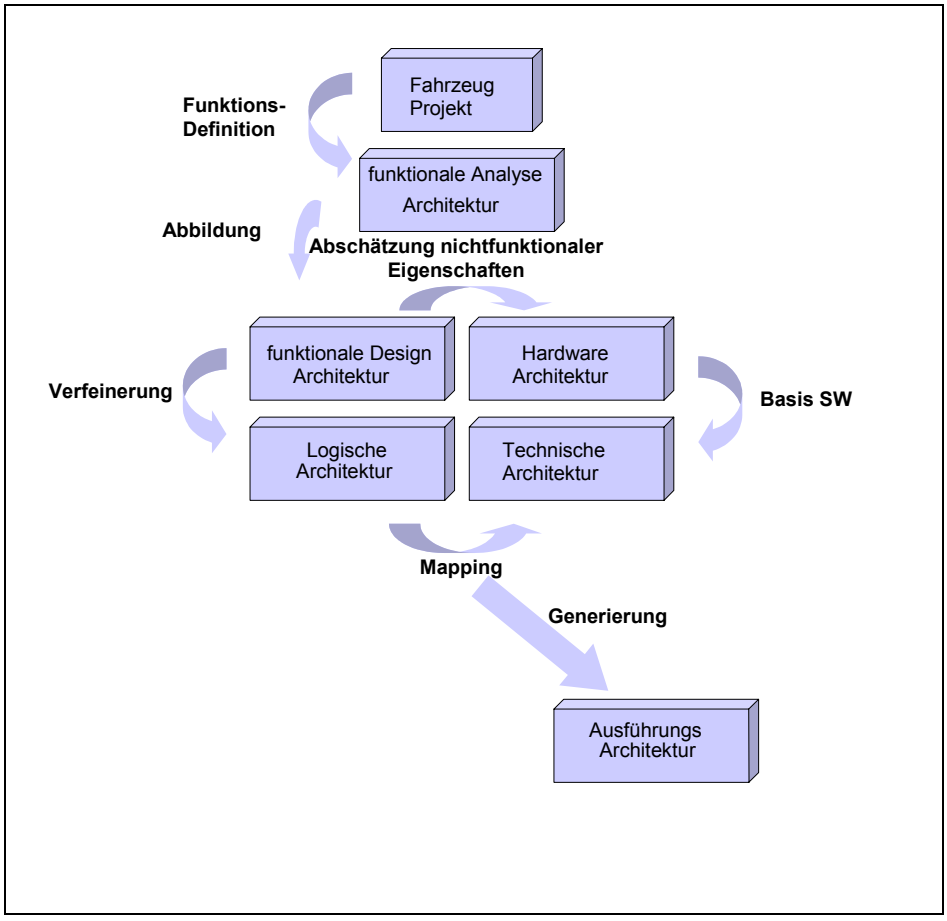

Abbildung 13: Abstraktionsebenen der EAST-EEA-ADL

Das **Fahrzeugprojekt** beschreibt alle elektronischen Funktionen mit den zugehörigen Varianten aus der Kundensicht. Ein geeignet aufgebautes Repository verwaltet somit alle möglichen Varianten einer Elektronik-Architektur.

Die Struktur der elektrischen Funktionen wird formal über deren Ein- und Ausgänge in der **funktionalen Analyse Architektur** beschrieben. Diese Struktur kann mit Verhaltensbeschreibungen hinterlegt werden. Funktionen besitzen Interfaces und können Subfunktionen enthalten. Spezielle Subfunktionen repräsentieren Sensoren und Aktuatoren als abstrakte Signale.

Die **funktionale Design Architektur** ist die höchste Abstraktionsebene, deren Struktur sich im Seriensteuergerätecode wiederfindet. Die Modellierungselemente auf dieser Ebene sind Funktions- und Signaltypen, Hierarchien sowie Signalflüsse. Funktionen lassen sich in zusammengesetzte Softwarefunktionen (Composite Software Function) oder elementare Softwarefunktionen (Elementary Software Function) klassifizieren. Während mit zusammengesetzten Funktionen Hierarchien beschrieben werden, stellen elementare Softwarefunktionen Blätter im Funktions-Hierarchiebaum dar. Das Verhalten von Funktionen kann je nach Aufgabenstellung auf zwei Arten beschrieben werden:

- Spezifikationsverhalten – dient primär der Simulation.
- Implementierungsverhalten – berücksichtigt notwendige Eigenschaften für eine spätere Verteilung auf Seriensteuergeräte sowie eine effektive Codegenerierung.

Die Verhaltensbeschreibung selbst wird mittels endlicher Automaten, Datenflußdiagrammen, Wahrheitstabellen oder textuellen Beschreibungen vorgenommen. Zeitanforderungen können an elementaren Softwarefunktionen spezifiziert werden. Die Unterscheidung in Funktionstypen und Instanzen führt neben besseren Kapselungs- und Wiederverwendungseigenschaften zu einer optimalen Nutzung von ROM in späteren Phasen des Entwurfs. Die Einführung von Implementierungsvarianten ermöglicht die automatische Anpassung von Funktionen und Signalen an unterschiedliche Sensor- und Aktuatorauflösungen.

Die **logische Architektur** hingegen beschreibt das System aus reiner Instanzensicht. Dazu werden alle Hierarchien aufgelöst und alle elementaren Softwarefunktionen instanziiert[1]). Alle Varianten sind zu diesem Zeitpunkt bereits bestimmt. Die Signale werden auf sogenannte IPC[2])-Buffer abgebildet. Nach einem Mapping der Funktionsinstanzen auf Steuergeräte stellt die Summe aller IPC-Buffer eine steuergeräteglobale Mailbox zur Interprozesskommunikation zur Verfügung. Diese Mailbox läßt sich effektiv in Seriensteuergerätesoftware umsetzen.

Steuergeräte-Eigenschaften wie Art- und Anzahl von Mikrocontrollern, Sensoren- und Aktuatoren sowie Netzwerkanschlüssen werden in der **Hardware Architektur**

[1]) Instanzen elementarer Softwarefunktionen werden als Funktionsinstanzen bezeichnet.

[2]) Inter-Process-Communication

beschrieben. Bestandteil dieser Beschreibung sind weiterhin Fahrzeugnetzwerke sowie etwaige Gateways .

Konfigurierbare Softwarekomponenten wie eine Hardware-Abstraction-Layer, Kommunikationssoftware und Betriebssysteme bilden die Grundlage für die **technische Architektur**. Auf diese Ebene ist auch die EAST-EEA-Middleware definiert.

Die **Ausführungsarchitektur** beschreibt das ablauffähige System. Man erhält sie durch ein Mapping von Funktionsinstanzen auf die Mikrocontroller der Steuergeräte, die Zuordnung der Funktionsinstanzen auf Betriebssystemtasks sowie der Abbildung der über ECU-Grenzen zu übertragenden Signale auf Nachrichten eines Kommunikationssystems. Letztere Abbildung definiert die Kommunikations-Matrix und dient zur Konfiguration der EAST-EEA-Middleware sowie der Communication Abstraction Layer Software.

Auf allen Abstraktionsebenen bestehen Verbindungen zu Requirementsdokumenten. Letztere sind geeignet zu strukturieren. Die Strukturierungsregeln sind Bestandteil der EAST-EEA-ADL.

Die Beschreibungselemente der EAST-EEA-ADL basieren auf ausgewählten Beschreibungselementen der UML 2.0. Zur Zeit ist die EAST-EEA-ADL prototypisch realisiert und wird in den EAST-EEA-Domänen Triebstrang, Fahrwerk, Karosserie und Infotainment validiert. Nach erfolgreicher Validierung dient sie als Spezifikation für die Umsetzung in kommerzielle Werkzeuge.

Deliverables	
D3.1	EAST-ADL V0.3
D3.2	EAST-ADL V0.4
D3.3	EAST-ADL V0.5
D3.4	EAST-ADL V0.6a
D3.5	EAST-ADL V0.6b
D3.6	EAST-ADL V1.0

WP4: Domain Specific Implementation and Validation

In Workpackage 4 wurden die Validatoren aufgebaut. Dabei gilt allgemein für alle diese domänenspezifischen Validatoren, dass sie möglichst unter Verwendung seriennaher Komponenten und Steuergeräte aufgebaut wurden. Aus Gründen der Vertraulichkeit kann auf Details der Validatoren hier nicht eingegangen werden.

Durch die Validatoren sollte der Nachweis erbracht werden, dass

- Software kann von der Hardware getrennt betrachtet werden, d.h. Softawre und Hardware kann von unterschiedlichen Zulieferern entwickelt werden.
- Software-Funktionen von einem Zulieferer können auf Steuergeräte anderer Zulieferer verschoben werden.
- Die Komplexität einer verteilten firmenübergreifenden Entwicklung kann durch den Einsatz geeigneter Schnittstellen reduziert werden.
- Die in WP 2 spezifizierte Middleware stellt die geeigneten Dienste unabhängig von der Verteilung von Software-Funktionen auf Steuergeräte zur Verfügung.
- Die in WP 3 definierte Entwicklungsmethode unterstützt diese verteilte Entwicklung durch den flexiblen Austausch von Software-Modulen zwischen Herstellern und Zulieferern unter Gewährleistung des Rechts auf geistiges Eigentum.

Neben den domänenspezifischen Validatoren wurde durch die Kopplung des Body- mit dem Telematik-Validator exemplarisch das Validierungsszenario „Telematik-moderierter Software-Download“ realisiert.

WT4.1 Body Electronics

Der Innenraum-Validator wurde auf Basis der Innenraum-Elektronik einer Mercedes C-Klasse aufgebaut. Dabei sind die Steuergeräte mit ihrer Sensorik und Aktorik auf einem sogenannten Brettaufbau integriert (siehe Abbildung 14).

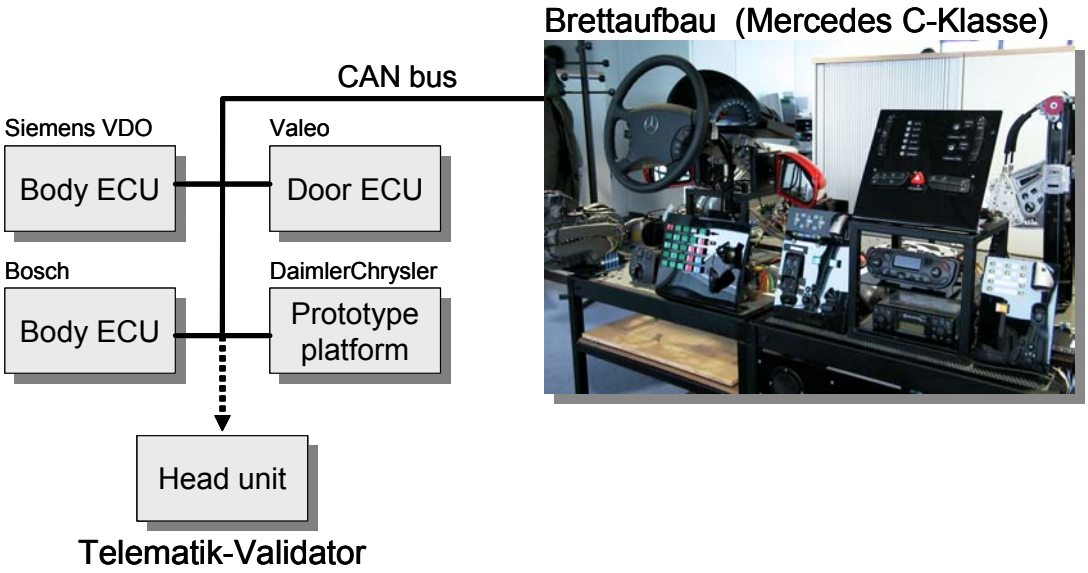

Abbildung 14: Brettaufbau Body-Validator

DaimlerChrysler stellte diesen Brettaufbau einschließlich eines zusätzlichen Steuergeräts „Prototype Platform“ als Worktask-Verantwortlicher zur Verfügung. Von den anderen Projektpartnern SiemensVDO, Bosch und Valeo wurden weitere Steuergeräte über den Fahrzeug-CAN-Bus an den Brettaufbau angeschlossen. Zusätzlich besteht die Möglichkeit der Kopplung mit dem Telematik-Validator.

Die folgenden Validierungs-Szenarien wurden definiert:

- Applikations-Softwarekomponenten von unterschiedlichen Zulieferern können auf einem Steuergerät integriert werden
- Applikations-Softwarekomponenten können alternativ auf unterschiedliche Steuergeräte platziert werden.
- Applikations-Softwarekomponenten können per Software-Download über den CAN-Bus auf die Steuergeräte geladen werden.
- Der Software-Download über die Telematik-Headunit in das Innenraum-System ist möglich.

Die Ausgangskonfiguration (Abbildung 15) repräsentiert im Wesentlichen den heutigen Stand in der Innenraum-Elektronik. Es existieren viele Steuergeräte, wobei jedes dieser Geräte nur wenig Funktionalität enthält.

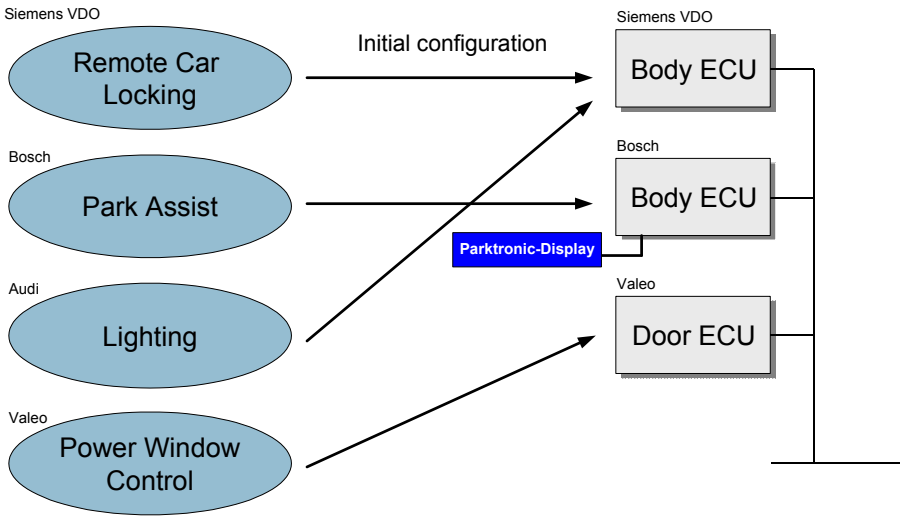

Abbildung 15: Ausgangskonfiguration

Zur Validierung der Verschiebbarkeit von Software-Komponenten wurde eine zweite Konfiguration definiert (Abbildung 16) definiert, bei der die Funktionen „Park Assist“ (Einparkhilfe), „Lighting“ (Lichtsteuerung) und „Power Window Control“ (elektrischer Fensterheber) gemeinsam auf demselben Steuergerät ablaufen. Das verbleibende Türsteuergerät (Door ECU) dient dabei nur noch als Schnittstelle zu der Sensorik und Aktorik in der Tür und kann dann kleiner und damit kostengünstiger ausgeführt werden.

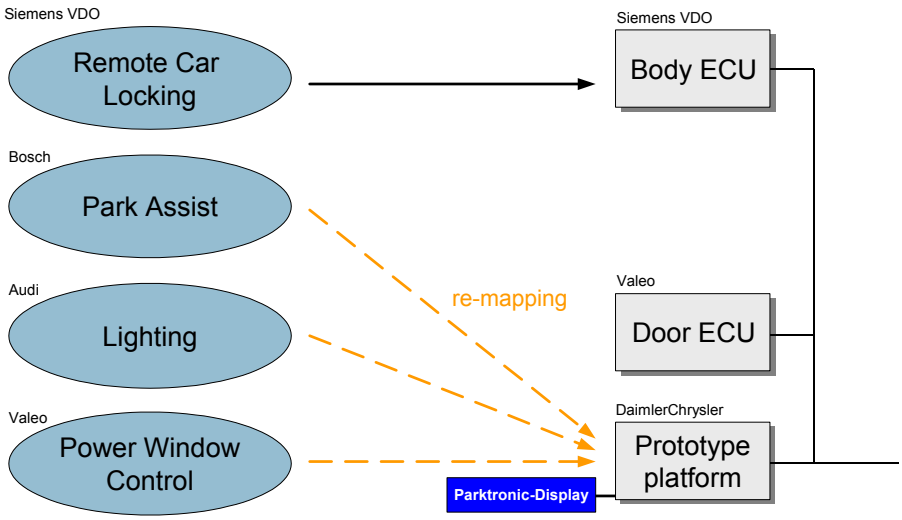

Abbildung 16: Zielkonfiguration

Neben der Möglichkeit, die Software über ein am CAN-Bus angeschlossenes Testgerät auf die Steuergeräte zu laden (z.B. in der Werkstatt), kann dies auch über das Telematik-System im Fahrzeug geschehen. Dieses Szenario ist in WT 4.3 beschrieben.

WT4.3 Telematics

Im Bereich Telematik wurden unter Federführung von DaimlerChrysler die folgenden Validierungs-Szenarien definiert:

- Möglichkeit zur Integration von Telematik-Geräten mit unterschiedlicher physikalischer Schnittstelle in ein gemeinsames System mit einheitlicher Bedienerschnittstelle
- Software-Update über das Telematik-System in die anderen Domänen des Fahrzeugs

Zur Validierung des ersten Szenarios wurde ein Validator mit MOST- und mit IEEE1396-Geräten aufgebaut (siehe Abbildung 17). Die Head-Unit fungiert dabei als Gateway zwischen diesen Bussystemen. Durch die Middleware-Dienste von HAVi ist es möglich, die angeschlossenen Geräte (z.B. CD-Wechsler) von jedem der angeschlossenen Bediensysteme aus zu steuern.

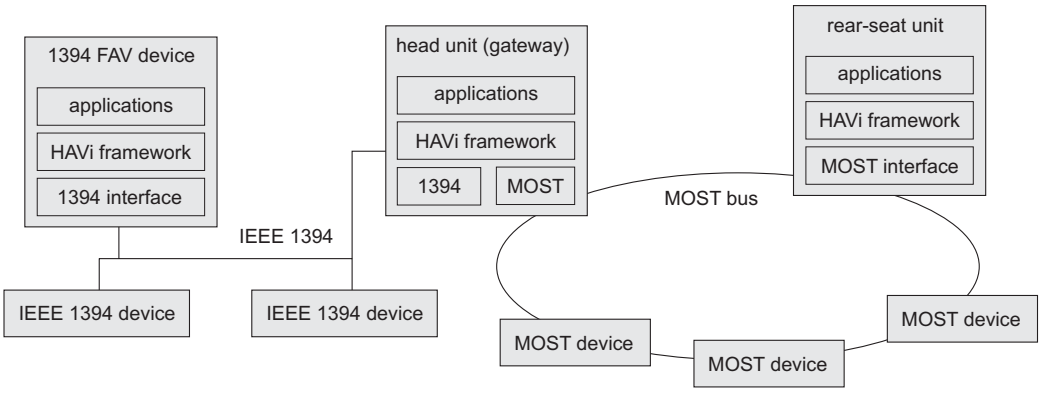

Abbildung 17: Telematik-Validator 1

Zur Validierung des zweiten Szenarios wurde auf der Head-Unit ein OSGi Service Gateway aufgebaut. Dieses empfängt die auf einem Innenraum-Steuergerät zu installierende Anwendung als OSGi-Bundle von der Infrastruktur (siehe Abbildung 18). Diese ist beim Validator durch einen über Ethernet mit der Head-Unit verbundenen PC nachgebildet. Durch die Verwendung des Internet-Protokolls kann diese Verbindung aber auch auf eine Luftstrecke (z.B. Wireless LAN, UMTS) abgebildet werden.

Das von der Head-Unit empfangene Flash-Bundle wird entpackt und über das Protokoll KWP2000 an das Innenraum-Steuergerät übertragen. Anschließend kann das Bundle auf der Head-Unit wieder gelöscht werden, um den Speicherplatz für andere Anwendungen frei zu machen.

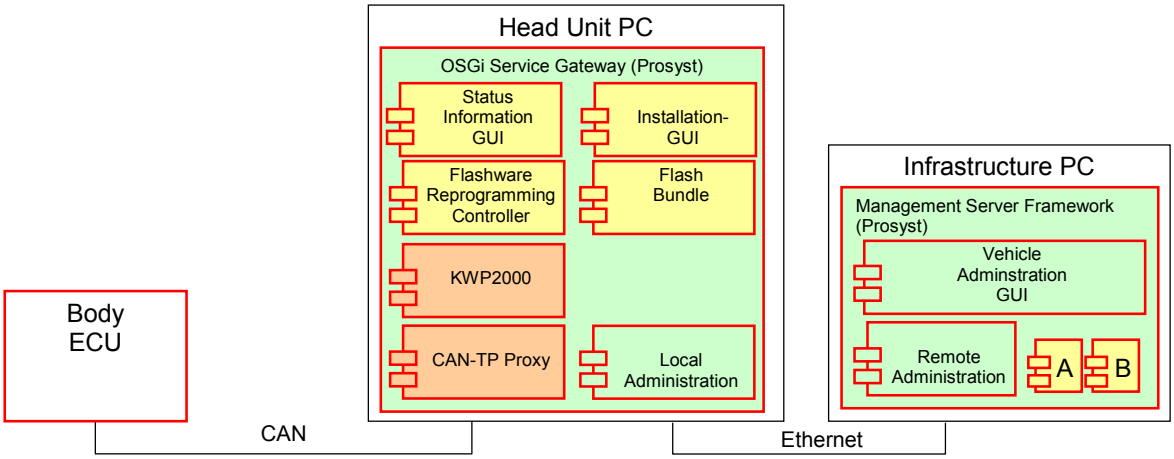

Abbildung 18: Telematik-Validator 2

WT4.5 Chassis

Der Chassis-Validator stellt ein sogenanntes „x-by-wire-System“ dar, bei dem sowohl bezüglich Lenkung (steer-by-wire) als auch Bremse (brake-by-wire) keine mechanische Verbindung zwischen Sensorik und Aktorik besteht. Die Verfügbarkeit des Systems wird hier durch den eingesetzten FlexRay-Bus gewährleistet (siehe Abbildung 19).

Der Beitrag von DaimlerChrysler besteht aus einem Bremspedal-Modul, das in einen inneren Regelkreis mit einem elektrischen Bremsaktuator von SiemensVDO und einem Lenkrad-Modul von BMW integriert ist.

Als verschiebbare Applikationskomponente wurde ein Fahrzeugmodell von Volvo realisiert, das hier exemplarisch auf dem Bremspedal-Modul abläuft. Weitere Applikation sind z.B. ESP (Stability Program) oder ACC (Adaptive Cruise Control), die sich über FlexRay in den Systemverbund integrieren lassen.

Zur Visualisierung der Funktionsweise des Gesamtsystems wurde zusätzlich ein Diagnosetool angeschlossen.

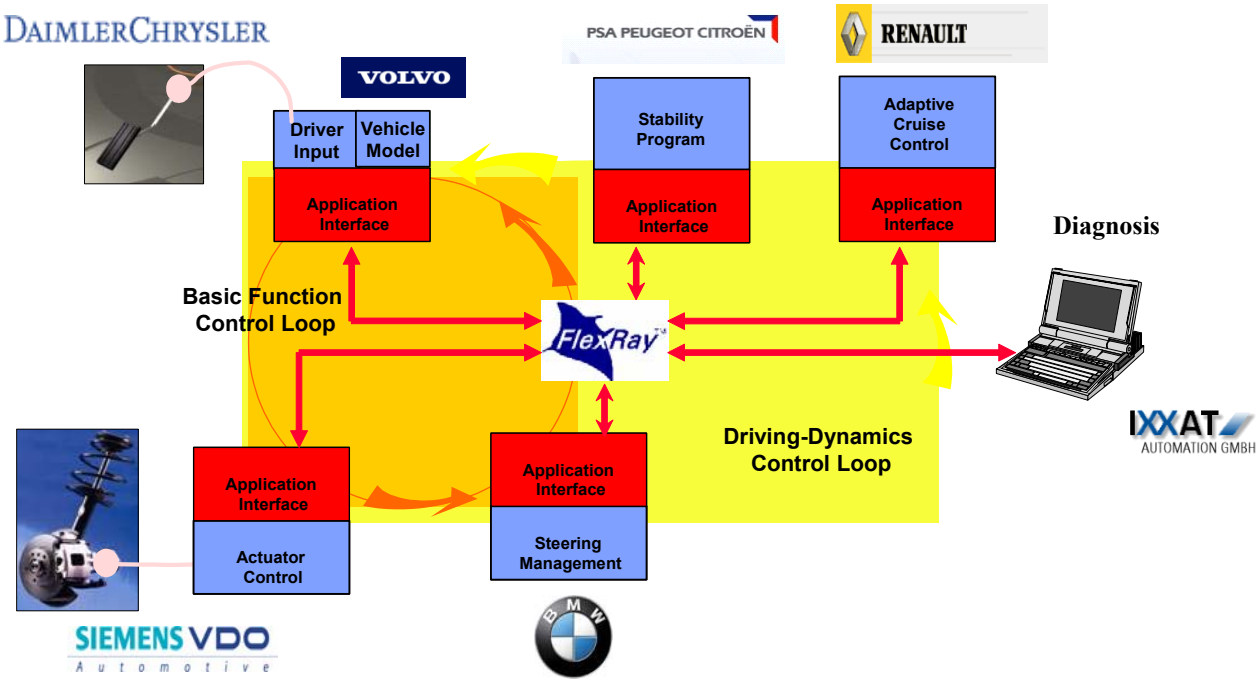

Abbildung 19: Chassis-Validator

Deliverables	
D4.1	Validator Requirements
D4.2	Validator Specification
D4.3	Validator Application Implementation
D4.4	Validator Middleware and Communication Layer Implementation
D4.5	Validator Test and Validation

2.2 Voraussichtlicher Nutzen

Im Automobil wird im Elektronikbereich ein hohes Potential an kundenorientierten, deshalb wettbewerbsdifferenzierenden Funktionserweiterungen gesehen. Von Komfortfunktionen über Fahrerassistenzsysteme, bis hin zu übergeordneten Verkehrsführungssystemen reicht die Palette der Erweiterungsmöglichkeiten. Eine Fortführung des klassischen steuergerätezentrierten Architekturansatzes wird langfristig als nicht mehr zielführend angesehen.

Dieser Erkenntnis folgend, haben sich 23 Partner (Kfz-Hersteller, Zulieferanten, internationale Forschungseinrichtungen) gefunden, um

gemeinsam Lösungen zu entwerfen, die den speziellen Anforderungen und Randbedingungen der Automobilindustrie gerecht werden. Das ITEA EAST-EEA-Projekt hat sich zum Ziel gesetzt, die nahtlose Integration softwarebasierter Funktionen in eine offene Elektronikarchitektur durch die Definition einer domänen- und hardwareunabhängigen Middleware zu ermöglichen und entsprechende Spezifikationen für zukünftige Standards vorzuschlagen.

Der EAST-EEA-Projektplan reflektiert den gewählten ganzheitlichen Ansatz zur Erreichung der gesteckten Ziele. Die Analyse der Trends und den sich daraus ergebenden Anforderungen an die zu entwickelnden Lösungen diente zur Formulierung der Gesamtsystemanforderungen. Basierend auf diesen Anforderungen wurde ein Architekturmodell entworfen, das als Referenz für die Spezifikation der Inhalte der integrierenden Softwareinfrastruktur – EAST-EEA-Middleware – diente.

Um sowohl künftige Anforderungen, als auch schon heutige Herausforderungen zu adressieren, erfolgte die Spezifikation von Middleware Diensten auf zwei Wegen. Zunächst wurden die Zwänge und Nöte, die sich aus der Implementierung heutiger Systeme ergeben, domänenspezifisch untersucht und kategorisiert. Diese Kategorien wurden mit einer anhand der Referenzarchitektur abgeleiteten Top-Down Betrachtung gegenübergestellt und abgeglichen.

Zur Validierung der Ergebnisse wurde für jede einzelne Domäne eine Laborumgebung, ein so genannter Validator, aufgebaut. Die Strukturen der Aufbauten orientieren sich weitgehend an den Architekturen der heute bekannten Dömanensubsysteme und werden überwiegend aus Seriensteuergeräten oder Rapid Prototyping Hardware aufgebaut. Ergänzt werden die Aufbauten punktuell mit Technologieelementen der nächsten Generation. Dies sind z.B. IEEE1394 Knoten in der Domäne Telematik oder das neue Bussystem FlexRay in der Domäne Chassis. Die Validierung selbst erfolgt praxisnah anhand der Definition von typischen Szenarien, z.B. der Re-Allokation von in Software implementierter Funktionalität auf Steuergeräten unterschiedlicher Hersteller.

Eine der wesentlichen Eigenschaften der Anwendung einer offenen Systemarchitektur ist die Vielfalt der zu einem Gesamtsystem beitragenden Partner. Gerade diese Tatsache wirft Fragen auf. Zum einen ist zu beantworten, wie Informationen zwischen den Partnern ausgetauscht werden, so dass jeder Partner das gleiche Verständnis der Situation hat. EAST-EEA hat sich aus diesem Grund in einer Querschnittsaufgabe an die Definition einer Architekturbeschreibungssprache ADL gewagt, die alle entscheidenden Phasen des Entwicklungsprozesses begleiten muss, und die, gemäß dem Anspruch, offen und standardisierbar sein soll. Die Antwort hierauf ist die Formulierung der EAST-ADL als mögliche Basis für ein Automotive Profile der UML 2.0 (Unified Modelling Language).

Zusammenfassend lässt sich nach ca. 3 Jahren Projektlaufzeit feststellen, dass EAST-EEA einen wesentlichen Beitrag zur Etablierung einer offenen Systemarchitektur geleistet hat. Es wäre jedoch falsch anzunehmen, dass eine so große Aufgabe in einem einzigen Projekt vollständig gelöst werden kann. Folgeprojekte werden notwendig sein, um die Umsetzung auf Gesamtfahr-

zeugebene voranzutreiben und dabei auch die Funktionsfähigkeit der neuen Businessmodelle nachzuweisen. Eine Reihe weiterer, auch europäischer Projekte, die zum erweiterten Themenkreis offener Systemarchitekturen zählen, zum Beispiel ARTIST[3]), AUTOSAR[4]), Veesa[5]) und EASIS[6]) sind oder werden mit EAST-EEA und dessen möglichen Folgeaktivitäten vernetzt. Wünschenswert wäre darüber hinaus auch eine Vernetzung des Gesamtthemenkomplexes offener Systemarchitekturen mit den Ansätzen der US-amerikanischen und der asiatischen Automobilindustrie.

2.3 Fortschritt auf diesem Gebiet bei anderen Stellen

Fortschritte bei anderen Stellen sind keine bekannt.

2.3 Erfolgte und geplante Veröffentlichungen

Vollmer, V.; Simons, M.; Leonhardi, A.; de Boer, G.: *Standardbasierte Architekturbausteine für interoperable Multimediaanwendungen im Fahrzeug.* In: Telematik im Kraftfahrzeug, VDI-Berichte Nr. 1728, Düsselforf, Germany, 2002.

Hardung B.; Wohlgemuth F.; Wernicke M.; Krüger A.; Wagner G.: *Development process for networked electronic systems.* In: Electronic Systems for Vehicles, Baden-Baden, Germany, VDI Berichte 1789, September 2003.

Thurner T.; Eisenmann J.; Haneberg M.; Voget S.; Geiger R.; Virnich U.: *The EAST-EEA project - a middleware based software architecture for networked electronic control units in vehicles.* In: Electronic Systems for Vehicles, Baden-Baden, Germany, VDI Berichte 1789, September 2003.

Leonhardi A.; Gschwandtner Th.; Simons M.; Vollmer V.: *Networking In-vehicle Entertainment Devices with HAVi.* In: 8th International Forum on Advanced Microsystems for Automotive Applications (AMAA 2004), Berlin, German, March 2004.

Vollmer V. ; de Boer G. ; Simons M.; Leonhardi A.: *Architektur-komponenten für zukünftige Multimediasysteme im Fahrzeug.* In: 24. Tagung Elektronik im Kfz, Essen, Germany, June 2004.

[3]) ARTIST, Advanced Real Time Systems, http://www.artist-embedded.org/

[4]) AUTOSAR, Automotive Open System Architecture, http://www.autosar.org/

[5]) VEESA, Vehicle e-safety architecture, http://www.cordis.lu/ist/projects/projects.htm

[6]) EASIS, Electronic Architecture and Systems Engineering for Integrated Safety Systems

Literaturverzeichnis

[1] CAN-Bus Spezifikation,
URL: http://www.can.bosch.com/docu/can2spec.pdf

[2] MOST Cooperation, URL: http://www.mostnet.de

[3] FlexRay Consortium, URL: http://www.flexray.com

[4] OSEK/VDX-Konsortium, *Spezifikation eines Betriebssystems, eines Netzwerkmanagemts und einer Kommunikationsschicht*, URL:http://www.osek-vdx.org, 2001.

[5] J. Eisenmann, M. Köhn, P. Lanchès, A. Müller. *Entwurf und Implementierung von Fahrzeugsteuerungsfunktionen auf Basis der Client/Server-Architektur*, VDI-Tagung „System-Engineering in der KFZ-Entwicklung“, 1997.

[6] P. Lanchès, J. Eisenmann, M. Köhn, J. Holland. *Client/Server Architecture - Managing New Technologies for Automotive Embedded Systems*, Convergence, 1998.

[7] A. Müller. *Client/Server-Architektur für Steuerungsfunktionen im KFZ*, it+ti – Informationstechnik und Technische Informatik, 1999.

[8] A. Müller. *Entwurfsmethodik und automatisierte Verteilung für Steuerungssoftware in einem verteilten Rechnersystem in der Automobilelektronik*. Dissertation Universität Tübingen, 1999.

[9] S. Boutin, *Architecture Implementation Language (AIL),* 1er Forum AEE, Guyancourt, März 2000,
URL: http://aee.inria.fr/forum/14032000/SB_Renault.pdf

[10] ITEA-Projekt VHE-Middleware,
URL: http://www.c-lab.de/vhe-middleware/

[11] Carnegie Mellon University, Software Engineering Institute, URL: http://www.sei.cmu.edu/plp/

[12] P. Clements, L. Northrop: *“A Framework for Software Product Line Practice”*, SEI Report, July 1999,
URL: http://www.sei.cmu.edu/plp/frameworkv2.7.pdf

[13] P. Donohoe, ed*.: “Software Product Lines; Experience and Research Directions”*; Proc. of the First Software Product Line Conference (SPLC 1), Denver, Aug. 2000, Kluwer Academic Press, 2000

[14] ITEA-Projekt Dess, URL: http://www.dess-itea.org/

[15] C. Scheidler, et. Al., *Systems Engineering of Time-Triggered Architectures – The SETTA Approach*, *DCCS 2000: 16th IFAC Workshop on Distributed Computer Control Systems,* Sydney, Australia, 29th November – 1st December, 2000.

[16] J. Ruh, U. Virnich, Hugo G. Chalé-Góngora, *The SETTA automotive demonstrator - A design approach on the perspective of the automotive industry, ECEC2001, 8th European Concurrent Engineering Conference,* Valencia, Spain 18-20 April, 2001.

[17] HAVi Initiative, URL: http://www.havi.org

Anlage 1: Erfolgskontrollbericht

Siehe separates Dokument

Anlage 2: Kurzfassung

Siehe separates Dokument