Models of Communication - Kommunikation in eingebetteten Systemen

Héctor Díez-Cubeiro
243 778

Betreut von Dipl.-Inform. Christian Fuß

Zusammenfassung

Frühe virtuelle Prototypen sind wesentlich im Entwicklungsprozess eines Automobils. Insbesondere fordert der wachsende Einsatz neuer Funktionalität eine präzise und möglichst optimierte Modellierung der Kommunikation zwischen den funktionalen Einheiten. Zu diesem Zweck wurde von der Firma Cadence eine virtuelle Integrationsplattform entwickelt, welche die Erstellung solcher virtuellen Modelle vereinfacht. Eine ausführliche Beschreibung der Bestandteile dieses Systems sowie eine Einführung in die Bus-Kommunikation und Bussysteme wird gegeben.

Inhaltsverzeichnis

1 Einleitung

Der wachsende Einsatz neuer Funktionalität zur Verbesserung des Komforts, der Sicherheit und der Leistung im Automobil fordert eine neue Planung seiner Infrastruktur, eine nochmalige Überprüfung der bisher verwendeten Entwurfsmethoden und gewisse Veränderungen zur Optimierung des Entwicklungsprozesses.

Neue Funktionalität bedeutet in diesem Fall die Integration elektronischer Komponenten, welche durch sogennante elektronische Steuergeräte (*engl. electronic control unit*, ECU) realisiert werden. Zwischen 50 und 70 solcher ECUs werden heutzutage üblicherweise in einem Wagen der Luxusklasse verbaut [3]. Aufgrund dessen, dass manche dieser Einheiten lebenswichtige und extrem sicherheitsrelevante Aufgaben wie z.B. die Bremssteuerung übernehmen sollen, muss die Koordination zwischen allen Elementen äusserst fließend, reibungslos und möglichst fehlerfrei funktionieren. Dabei spielt die Kommunikation zwischen den Steuerungskomponenten eine wesentliche Rolle. Die Verbindung erfolgt mithilfe von Bussen. Eine Vielfalt an Kommunikationsprotokollen und Bussen werden in heutigen Kraftfahrzeugen eingesetzt. CAN, TTP, TTCAN, LIN, MOST oder in Zukunft auch FlexRay sind einige Beispiele, die häufig auftreten.

Im Entwicklungsprozess besteht der Bedarf nach einer frühen Simulation bzw. nach der Erstellung eines virtuellen Prototyps, damit mögliche Fehler oder Ungenauigkeiten im System rechtzeitig entdeckt werden und so die hohen Fertigungspreise reduziert werden können. Dies gilt natürlich auch für Modellierung und Entwurf eines Automobils und insbesondere auch im Fall der Kommunikation zwischen seinen elektronischen Komponenten. Zu diesem Zweck wurde 2001 eine virtuelle Integrationsplattform (engl. *Virtual Integration Platform*) innerhalb VCC (***V**irtual **C**omponent **C**o-Design*) von der Firma Cadence entwickelt. VCC ist eine Umgebung, die Entwürfe auf Systemebene erlaubt und vereinfacht [9]. Ein Teil dieses Systems ist der Untersuchung von Kommunikationsmöglichkeiten gewidmet. Dieses Teilsystem, UCM (***U**niversal **C**ommunication **M**odel*) genannt, erlaubt präzise Leistungsabschätzungen [4]. Diese Ausarbeitung beschäftigt sich hauptsächlich mit der Frage des Übergangs von einem funktionalen Modell zum Architekturmodell, d.h., wie ein Verhaltensmodell, das unter idealen Bedingungen erzeugt wurde, einer bestimmten Architektur mit ihren entsprechenden Einschränkungen zugeordnet werden kann.

Eine ausführliche Beschreibung der bisher erwähnten Komponenten wird in Kapitel 2 gegeben. Ein vertiefter Einblick in die Funktionalität und in die Logik der virtuellen Plattform und UCM wird in Kapitel 3 geliefert. Kapitel 4 erörtert die vorgestellte Methodologie, kritisiert einige Aspekte und hebt bestimmte exisitie-

rende Verbesserungen hervor. Diese Ausarbeitung wird mit einer kurzen Zusammenfassung abgeschlossen.

2 Grundlagen

2.1 Elektronische Steuerungssysteme

Die ganze Elektronik eines Kraftfahrzeugs stellt ein umfassendes verteiltes eingebettetes System (engl. *cluster* [5]), bestehend aus mehreren Knoten, einem relativ großen Softwareanteil und aus Verbindungen zwischen den Knoten dar. In diesem Zusammenhang sind die Knoten des verteilten Systems Steuerungseinheiten. Eine solche ECU besteht grundsätzlich aus drei Subsystemen: einem Host-Controller oder Host-Computer, einem Bus-Controller oder Kommunikations-Controller und einem physikalischen Bustreiber [4].

Der Host-Computer setzt sich aus einer CPU, Speicher, einer lokalen Echtzeituhr, einem architekturabhängigen Echtzeitbetriebsystem (RTOS, *real time operating system*) wie z.B. OSEK-OS und Anwendungssoftware zusammen. Der Zweck eines Host-Computers ist die Ausführung bestimmter Jobs der Anwendung innerhalb einer gegebenen Zeitgrenze durchzuführen. Mit dem Kommunikationskanal bilden die Bus-Controller der Knoten das Kommunikationssystem im Netzwerk. Controller können sowohl zeitgesteuert als auch ereignisgesteuert sein und somit charakterisieren sie das ganze Bussystem.

Ereignisgesteuerte Systeme sind für diejenigen Aktivitäten reserviert, die durch zeitlich nicht vorhersehbare Ereignisse ausgelöst werden. Sie übertragen sogenannte *Ereignis-Nachrichten*. Eine solche Nachricht wird unmittelbar nach einem bestimmten Ereignis gesendet [5]. Ereignissteuerung ist besonders geeignet für weiche Echtzeitsysteme, d.h., die Art von Systemen die typischerweise alle ankommenden Eingaben innerhalb einer akzeptablen Antwortzeit abarbeiten. Die Reaktionszeit ist also flexibel und eine etwas längere Verzögerung der Verarbeitung der Daten sowie der Lieferung von Ergebnissen verursacht keine kritische Situation. Aufgrund der Intuitivität der Ereignissteuerung ist sie weit verbreitet und in den wichtigsten Kommunikationssystemen eingesetzt worden (wie z.B. CAN, s. 2.3.1). Der wichtigste Vorteil dieses Paradigmas ist der geringe Zeitverlust in der Reaktion auf ein Ereignis. Trotzdem kann es leicht bei schnell aufeinanderfolgenden Ereignissen zu einer Überlastung des Echtzeitsystems kommen. Dies würde eine fehlerhafte Bearbeitung mancher Ereignisse oder eine Überschreitung der Zeitschranke bedeuten.

Zeitgesteuerte Kommunikations-Systeme sind Systeme, die zu bestimmten vorher festgelegten Zeitpunkten Nachrichten periodisch übertragen. Diese Nachrichten werden als *statisch* bezeichnet. In einem solchen System werden alte Nachrichten sukzessiv aktualisiert bzw. von neuem überschrieben. Eine genaue Zeitplanung ist vorhanden und somit sind Überlastungen ausgeschlossen. Die Kommunikation bleibt noch bei einer hohen Buslast deterministisch. Dieses Prinzip wird meistens auf harten Echtzeitsystemen angewendet. Bei solchen Systemen geht es darum, das korrekte Ergebnis immer innerhalb eines vorgegebenen Zeitintervalls zu liefern. Eine Überschreitung der Zeitschranke gilt als Scheitern des Systems. Der hohe Planungsaufwand und die geringe Busarbitrierung sind die Hauptnachteile der Zeitsteuerung.

Eine Kombination aus Ereignis- und Zeitsteuerung tritt in manchen Kommunikationssystemen, wie z.B. FlexRay, auf. Dabei wird versucht von den Vorteilen beider Paradigmen zu profitieren. Eine detallierte Beschreibung wird in 2.3.2 gegeben.

Typische Beispiele für ECUs sind Transmissions- und Motor-Steuerungsmodule, Anti-lock Brake System Modules (ABS), Mensch-Maschine Schnittstellen (MMI), Door Control Unit oder Climate Control Unit [1]. In Abb. 1 werden einige dieser Module dargestellt.

2.2 Bus-Kommunikation

Allgemein dienen Bussysteme zum Datenaustausch in Form von Datenpaketen oder Frames zwischen den Mitgliedern eines Netzwerks. Diese Mitglieder sind beispielsweise Rechner oder elektronische Steuerungseinheiten. Die Datenpakete haben je nach Bussystem unterschiedliche Komponenten. Typischerweise sieht der Aufbau einer solchen Nachricht folgendermassen aus:

- Trennzeichen zur Markierung von Beginn und Ende des Frames
- Ziel- und Quelladressen
- Steuerinformationen
- Daten des Pakets
- Prüfsummen zur Fehlererkennung

[1] Die englische Version der Wikipedia® liefert diese Beispiele für den Suchbegriff ”electronic control unit” im Zusammenhang mit der Ausstattung eines modernen Automobils. Der Leser möge diesen Begriff unter http://en.wikipedia.org/wiki/Electronic_control_unit nachschauen.

Abhängig vom Charakter des Bussystems (zeit- oder ereignisgesteuert), können die Frames zusätzliche Felder enthalten, wie z.B. das Arbitrierungsfeld bei CAN zur Darstellung des Identifiers. Mehr als ein Feld zur Fehlererkennung sind in manchen Bussystemen üblich.

Im nächsten Abschnitt werden einige konkrete in der Automobil-Industrie eingesetzte Bussysteme detalliert vorgestellt.

2.3 Busse im Automobil

Heutige Automobil-Netzwerke sind keine Sammlung von diskreten, Punkt-zu-Punkt Leitungen. Im Gegensatz dazu erlauben Bussysteme, dass jedes einzelne Gerät nur einmal an den Bus angeschlossen werden muss mit der entsprechenden Senkung des Gewichts, des Raumbedarfs und des Verkabelungspreises. Ferner besteht für alle an dem Bus angeschlossenen Einheiten die Möglichkeit miteinander zu kommunizieren. Ein weiterer Vorteil ist die Existenz von Diagnose-Geräten, welche für die Überwachung des Busses zuständig sind. Diese und weitere Eigenschaften von Bussystemen werden im darauffolgenden Abschnitt anhand konkreter Beispiele ausführlich besprochen.

Der Übergang von traditionellen mechanischen und hydraulischen Systemen zu X-By-Wire-Systemen muss einige Garantien mit sich bringen. An erster Stelle der technischen Entwicklung steht die Sicherheit und demzufolge müssen die neuen Systemen mindestens so sicher, zuverlässig und fehlertolerant wie die alten sein. Dies erfolgt im grössten Teil mittels der Bussysteme. Neben der Übertragungsgeschwindigkeit ist die Flexibilität und die Unterstützung verschiedener Topologien eine der wichtigsten Eigenschaften die heutzutage von Bussystemen verlangt wird. Ein breites Anpassungsspektrum übersetzt sich anschliessend in grössere Einsatzmöglichkeiten im Automobil. In dieser Richtung befindet sich auch die Idee der Vereinheitlichung von Bussen, und zwar ein einziger Bustyp sollte in möglichst vielen Bereichen einsetzbar sein. Damit wird eine Vermeidung der ”Spezialisierung” der Bussysteme in heutigen Kraftfahrzeugen gesucht. Beispielsweise wird der MOST-Bus für Multimedia- und Telematik-Anwendungen eingesetzt oder der LIN-Bus zur Ankopplung von Aktuatoren und Sensoren verwendet [8]. Abb.1 zeigt ein Netzwerk, in dem verschiedene Bussysteme verwendet wurden.

Außerdem ist auch die Unterstützung verschiedener Marken, Baureihen und Plattformen nötig. Somit wird erreicht, dass eine Elektronik-Architektur überall eingesetzt werden kann. Dafür wird eine blockweise Einteilung von Komponenten und Funktionen benötigt, die sich dabei gegenseitig möglichst wenig beeinflussen sol-

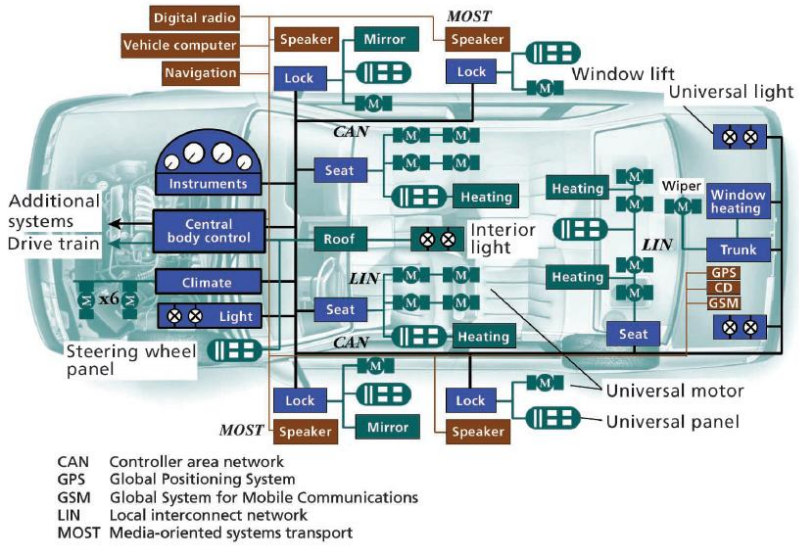

Abbildung 1: Ein Teilnetzwerk aus einem modernen Kraftfahrzeug [6]

len. Diese Einteilung des Systems in kleineren Stücken ermöglicht später eine einfachere Erweiterbarkeit. Durch die hohe Verbreitung einiger Bus-Komponenten (wie z.B. CAN-Knoten) sind diese sehr preisgünstig erhältlich.

Einige der geläufigsten Kommunikationsprotokolle werden als nächstes ausführlich beschrieben.

2.3.1 CAN

Das ***C**ontroller **A**rea **N**etwork* (CAN) wurde 1981 von Bosch und Intel entwickelt und wird seit Anfang der 90er Jahre in vielen Kraftfahrzeugen eingesetzt. CAN ist ein ereignisgesteuertes Kommunikationssystem. Ursprünglich wurde es für nicht-sicherheitskritische elektronische Anwendungen im Automobil entworfen [2].

Ein CAN-Netzwerk besteht aus Knoten mit einer CAN-Schnittstelle, die mittels eines CAN-Busses verbunden werden. CAN ist ein Multi-Master-Bus, d.h., jeder Knoten kann Master werden und seine Nachrichten abschicken. Da die Nachrichten keine Empfängeradressen enthalten, werden diese von jedem Teilnehmer gehört und anhand des Identifiers kann der Empfänger entscheiden, ob diese Nachricht für ihn relevant ist. CAN ist also ein Broadcast-Bus. Die Datenübertragungsgeschwindigkeit beträgt zwischen 10kBit/s und 1MBit/s, obwohl sich in der Realität bei einer Buslänge von 100 m diese 1MBit/s sich auf effektive 500kB/s re-

duzieren. Zur Realisierung des CAN-Busses werden üblicherweise Twisted-Pair Leitungen oder Eindrahtleitungen verwendet.

Die Kommunikation funktioniert folgendermassen: Zum Zeitpunkt, zu dem ein Sensor ein Ereignis wahrnimmt, wird von diesem Knoten versucht eine Nachricht durch den Bus zu schicken. Dabei kann es sein, dass Kollisionen auftreten, wenn mehrere Knoten zum selben Zeitpunkt auch ihre Nachrichten übertragen wollen. Die vom CAN-Bus verwendete Strategie zur Kollisionsvermeidung heißt CSMA/AMP (*Carrier Sense Multiple Access with Arbitration by Message Priority*) und basiert auf der Zuweisung jeder Nachricht zu einer bestimmten Priorität. Wenn der Bus frei wird (dies wird von einem Client überprüft, "Carrier Sense"), dann senden alle Knoten mit laufenden Nachrichten ihre Nachrichten durch den Bus. Falls mehrere Nachrichten gleichzeitig ankommen, verwendet der "Arbitration on Message Priority"-Block bestimmte Prioritätswerte um alle Nachrichten außer der mit der höchsten Priorität wegzuwerfen. Ein Knoten kann also nur dann seine Nachricht vollständig auf dem Bus schreiben wenn eine Anforderung des Host-Computers zugrunde liegt, der Kanal leer ist und die Priorität der Nachricht die höchste ist.

In der Arbitrierungsphase wird anhand des Identifiers entschieden welche Nachricht die höchste Priorität hat. Die CAN-Spezifikation benutzt für Higher- und Lower-Bits die beiden Bezeichner "rezessiv" und "dominant". Dominante Bits überschreiben immer rezessive. Wollen beispielsweise 2 Knoten gleichzeitig eine Nachricht auf dem Bus schreiben, so werden die beiden Knoten solange auf den Bus schreiben bis einer von beiden ein rezessives und der andere ein dominantes Bit schreibt. In diesem Fall kann nur der Knoten weiter schreiben, der die Arbitrierung gewonnen hat. Der andere muss aufhören und auf die nächste Arbitrierungsphase warten. Dadurch vergewissert sich das System, dass die wichtigste Nachricht weiterkommen kann und nie unterbrochen wird.

CAN unterstützt zwei Frameformate: der CAN-Base-Frame (CAN 2.0A) und der CAN-Extended-Frame (CAN 2.0B). Diese Formate unterscheiden sich an der Länge des Identifiers, der 11 Bit lang im Fall des Base-Frames und 29 Bit lang im Fall des Extended-Frames ist. Insgesamt ist ein Base-Frame max. 108 Bit lang und ein Extended-Frame max. 128 Bit. Eine grobe Skizze des Base-Frames sieht wie folgt aus (s. Abb. 2):

- Das Startfeld markiert den Beginn eines Frames durch ein dominantes Bit
- Das Arbitrierungsfeld setzt sich grundsätzlich aus dem 11-bittigen Identifier und einem Remote-Transmission Bit (RTR, *Remote Transmission Request*) zusammen. Das RTR-Bit ist dafür zuständig zwischen Data-Frame und Data-Remote-Frame zu unterscheiden

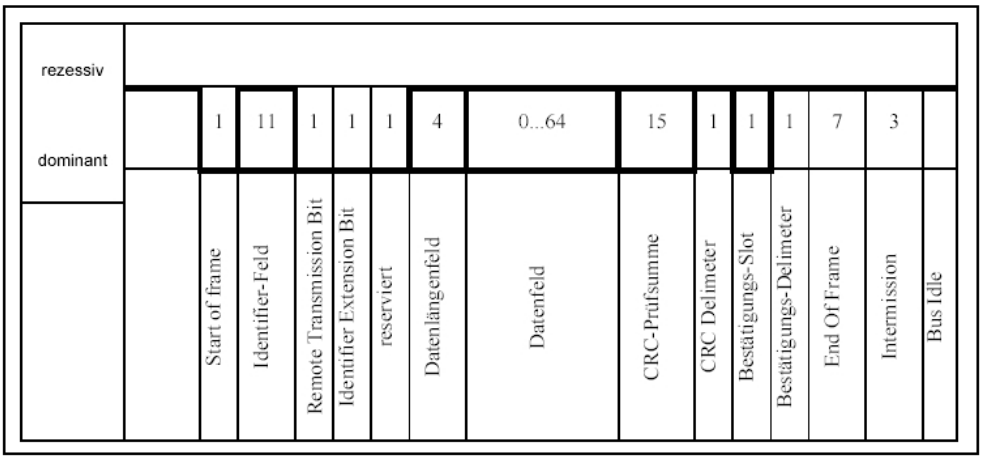

Abbildung 2: Aufbau eines CAN-Datenframes

- Das Kontrollfeld enthält ein Identifier-Extension Bit (IDE) und ein reserviertes Bit, um zwischen Base-Frame und Extended-Frame zu unterscheiden, und ein 4-bittiges Datenlängenfeld (DLC, *Data Length Code*), welches die Länge des Datenfeldes beschreibt
- Das Datenfeld beinhaltet die zu übertragenen Daten und kann maximal 8 Bytes lang sein
- Das CRC-Feld enthält die Prüfsumme. Dieses Feld sorgt dafür, eine fehlerfreie Übertragung des Frames zu erreichen, indem die Prüfsummen vor und nach der Übertragung verglichen werden
- Das ACK-Feld besteht aus dem Bestätigungs-Slot und Bestätigungs-Delimiter. Dieses wird für die Empfangsbestätigung empfängerseitig verwendet
- Der Frame wird durch 7 rezessive Bits abgeschlossen
- Das Intermission-Frame-Space (IFS) setzt den minimalen Abstand zwischen zwei nacheinander folgende Nachrichten

Die CAN-Spezifikation legt fest wie Fehlererkennung und Fehlerbehandlung durchgeführt werden müssen. Die folgenden fünf Methoden dienen zu diesem Zweck:

- *Monitoring*. Der Absender kontrolliert ob das im Bus geschriebene Bit das gleiche wie das gesendete ist. Wenn Unterschiede festgestellt wurden, ist ein Fehler aufgetreten.

- *Cyclic Redundancy Check*. Eine Prüfsumme wird abhängig von der Nachricht vor der Sendephase berechnet und zur Sendezeit mit der Nachricht übertragen. Der Empfänger testet mithilfe der Berechnungen vom Sender ob die Summe noch stimmt. Falls nicht, muss der Frame erneut gesendet werden.

- *Frame-Check*. Die Länge und Struktur des empfangenen Frames werden mit der Spezifikation verglichen. Falls ein Fehler auftritt, wird eine "Format Error"-Meldung vom Knoten zurückgegeben.

- *ACK-Fehler*. Jeder Knoten berichtet wenn der Empfang einer Nachricht erfolgreich war oder nicht, indem ein dominantes Bit (0) bzw. ein rezessives Bit (1) im ACK-Feld der Nachricht geschrieben wird.

- *Bitstuffing*. CAN benutzt die NRZ (Non-Return-to-Zero)-Kodierung um eine effiziente Fehlererkennung durchzuführen. Dabei kann das Problem auftreten, dass monotone Folgen von 0en oder 1en übertragen werden, wobei die Synchronisierung der Teilnehmer verhindert wird. Bitstuffing löst dieses Problem, indem nach fünf aufeinanderfolgenden gleichwertigen Bits ein sogenanntes "Stopfbit" mit dem komplementären Wert hinzugefügt wird.

2.3.2 TTP/FlexRay

TTP (***T**ime **T**riggered **P**rotocol*) ist ein zeitgesteuertes Kommunikationsprotokoll, welches insbesondere für den Einsatz in sicherheitskritischen Anwendungen bzw. harten Echtzeitsystemen mit hohen Abhängigkeitsanforderungen geeignet ist [5]. Es wird seit 15 Jahren an der Universität Wien entwickelt. FlexRay ist ein verwandtes Protokoll und wird seit 1999 von BMW und DaimlerChrysler in Zusammenarbeit mit Motorola und Philips entwickelt. Schon 2002 traten weitere Firmen wie Bosch und General Motors im Konsortium ein und seit kurzem, aufgrund der ähnlichen Eigenschaften, Anforderungen und Merkmale von TTP und FlexRay, hat sich ein vereinigtes Kommunikationsprotokoll ergeben.

Unter den Merkmalen dieses Bussystems stehen eine konfigurierbare synchrone und asynchrone Übertragung mit einer Bandbreite von etwa 10 – 25Mbit/s und höher. Verschiedene Bustopologien werden unterstützt, müssen aber einige Voraussetzungen erfüllen, wie z.B. eine mögliche redundante Kommunikation durch mehrere Kanäle, die Verbindung aller Knoten mit der Fahrzeugbatterie, die Möglichkeit alle Knoten über den Bus wecken zu lassen und die Bereitstellung von Power-Management für alle Knoten.

Das Kommunikationsverfahren basiert auf der Idee der Teilung der Zeit in zwei parallele wiederkehrende Intervalle, einen synchronen statischen Kanal für die zeitgesteuerten Nachrichten und einen asynchronen dynamischen Kanal für die ereignisgesteuerten Nachrichten.

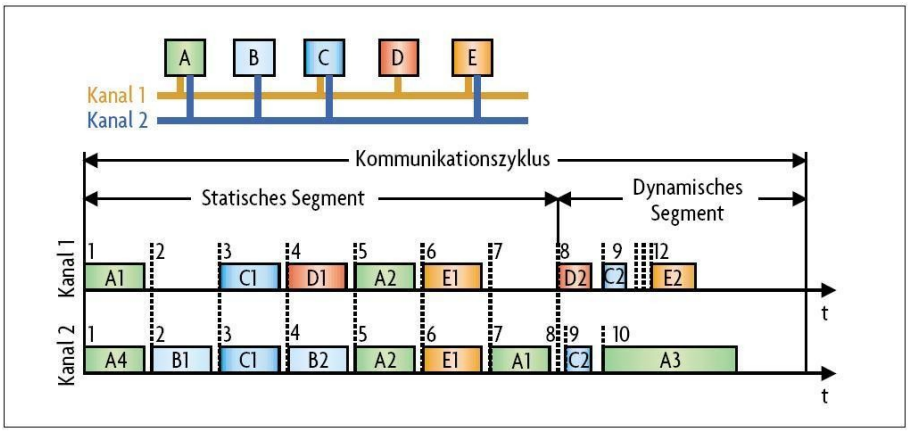

Abbildung 3: TTP/FlexRay-Kommunikationszyklus. Fünf Geräte kommunizieren über zwei Kanäle miteinander. Man beachte die Verwendung des für die Knoten A, B, C und E redundanten zweiten Kanals [8].

Die Slots gleicher Länge des statischen Segments sind für Nachrichten hoher Priorität reserviert. Auf den Bus wird nach dem TDMA (engl. *Time Division Multiple Access*) Verfahren zugegriffen. Dieses Verfahren ist ein Typ vom TDM-Verfahren (engl. *Time Division Multiplexing*). Im TDM-Verfahren geht es darum die Datenströme mehrerer Sender mit niedriger Bitrate zu einem Datenstrom hoher Bitrate durch Aufteilung der Datenströme auf einzelne Zeitslots zusammenzufassen. Bei TDMA erfolgt der Zugriff durch mehrere unabhängige Stationen und nicht nur durch einen einzigen Multiplexer (wie bei TDM). Knoten, die an beiden Kanälen angeschlossen sind, senden ihre Nachrichten synchron auf beiden. Wird eine Nachricht nicht gesendet, so bleibt die entsprechende Zeit ungenutzt. Alle Nachrichten haben in diesem Fall die selbe Priorität. Ihre Empfangs- und Sendezeiten sind vorhersagbar und berechenbar, deswegen sind Kollisionen ausgeschlossen und alle Nachrichten werden zu deterministischen Zeitpunkten übertragen. Eine globale Zeit steht für alle Knoten zur Verfügung. Das Protokoll schützt die Kanäle vor ”Babbling Idiots” über Buswächter. Eine Busüberlastung kann hier nicht auftreten und deswegen existiert auch keine Arbitrierungsphase. Nachrichten erhalten keine Empfänger- oder Absender-Adressen und die Knoten können somit wiederum alle Nachrichten hören.

Weniger kritische Ereignis-Nachrichten werden im dynamischen Segment übertragen. Im Gegensatz zum statischen Teil, sind hier die Slots variabel. Ein Un-

terschied in den Zeit-Slots von beiden Kanälen ist auch möglich. Der Buszugriff erfolgt hier nach dem FTDMA-Verfahren (engl. *Flexible Time Division Multiple Access*), bei dem die Einzelkanäle innerhalb der verfügbaren Bandbreite auf einen Teilbereich dieser Bandbreite zugreifen. Kollisionen werden auch im dynamischen Bereich durch bestimmte Zugriffsmechanismen im Sendevorgang vermieden. Dieses Prinzip wird später für die virtuelle Modellierung von Bussystemen angewendet (s. 3.2.3).

2.3.3 Weitere

Der ***L**ocal **I**nterconnect **N**etwork* Bus (LIN) ermöglicht eine kostengünstige Ankopplung von Aktuatoren und Sensoren da, wo CAN aus Kosten- und Platzgründen nicht sinnvoll ist. LIN hat eine Datenübertragungsgeschwindigkeit von 20 kBit/s und ist das kleinste und billigste der hier vorgestellten Bussysteme. Es ist ein serieller Broadcast-Feldbus mit einem Master und mehreren Slaves (max. 16). LIN unterstützt keine Kollisionsbehandlung, deswegen werden alle Nachrichten vom Master initiiert und maximal von einem Slave beantwortet. LIN-Netzwerke besitzen üblicherweise maximal 16 Knoten und werden durch die jeweiligen Master an einem CAN-Bus angeschlossen. Die Übertragung von Nachrichten erfolgt synchron und das verwendete Buszugriffverfahren ist ”Polling”. Dies beruht auf einem UART-Protokoll (Universal Asynchronous Receiver/Transmitter), bei dem der Master nacheinander jeden Slave abfragt und dieser daraufhin eine Nachricht versenden kann. Aufgrund der geringen Wichtigkeit und nicht sicherheitskritischen Charakters der durch diesen Bus verbundenen Komponenten, wird hier die Verwendung von redundanten Kanälen nicht unterstützt.

Ein weiterer Bus ist der ***M**edia **O**riented **S**ystems **T**ransport* Bus (MOST). Er wird seit 1998 von der MOST Cooperation entwickelt und vor allem in Multimedia- und Telematik-Anwendungen eingesetzt, d.h. überall dort, wo Audio, Video, Navigation und Telekommunikation auftreten. Er besitzt eine relativ hohe Datenübertragungsrate von 24,8 Mbit/s. MOST unterstützt sowohl eine asynchrone als auch eine synchrone Übertragung von Nachrichten. Als Buszugriffsverfahren wird bei MOST TDM (Time Division Multiplexing, s.2.3.2) und CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) benutzt. Das letzte Verfahren funktioniert ähnlich wie das CSMA/AMP (s.2.3.1). Im Gegensatz zu den bereits vorgestellten Bussystemen enthalten die Nachrichten hier eine Adresse, die als Absender- oder Empfänger-Adresse verwendet wird. Verschiedene Topologien wie Ring-, Stern- und Ketten-Topologien mit bis zu 64 Knoten werden unterstützt. Wie bei TTP/FlexRay kommt hier auch als Physical Layer ein optisches Medium (Plastic Optic Fibre) zum Einsatz. Weitere charakteristische Eigenschaf-

ten dieses Bussystems sind Flexibilität, niedrige Kosten und Kompatibilität mit der PC-Industrie.

3 Modellierung

Der Entwurfsprozess für die Elektronik eines Kraftfahrzeugs hat bisher hauptsächlich aus drei wohlgetrennten Phasen bestanden, nämlich eine erste Spezifikationsphase, eine Subsystem-Implementierungsphase und eine Integrationsphase. Dieses starre Designprinzip bietet nur wenige Möglichkeiten, das Produkt ausreichend zu testen und ist oft sehr teuer und manchmal auch fehleranfällig. Darüber hinaus verursacht die schnelle Evolution von Automobilen eine Steigerung der Komplexität in seinen Kraftfahrzeug-, Funk- und Multimedia-Anwendungen. Produkte verlangen eine enge multidisziplinäre Zusammenarbeit und kürzere Entwicklungszeiten.

Entwickler haben diese Tendenz wahrgenommen und beginnen neue Entwurfsmethoden in frühen Phasen des Entwicklungsprozesses zu verwenden. Der Entwurf eines Systems wandert heutzutage von einem experimentellen manuellen Prozess zu strengeren durch Werkzeuge unterstützten Ansätzen. Die Integrationsphase sollte innerhalb einer virtuellen Umgebung simuliert werden können, in der alle Szenarios dargestellt werden. Die Motivation für eine solche Transformation besteht darin eine schnellere, präzisere und kostengünstigere Produktentwicklung zu erreichen. Entwicklungszeiten von vier bis sechs Monaten sind heute für manche elektronischen Systeme durchaus möglich [9].

Die verwendeten Entwurfswerkzeugen basieren üblicherweise auf einer virtuellen Integrationsplattform und Funktionsarchitekturen. Modelle von Software- und Hardware-Komponenten werden erzeugt um das ganze Modell des verteilten Systems darzustellen. Die Kombination aus diesen beiden Elementen erlaubt eine einfachere Entwicklung von elektronischen Systemen, die aus Bausteinen zusammengesetzt sind und aus den unterschiedlichsten Anwendungsdomänen stammen.

Virtual Component Co-design

Als Beispiel für eine solche Plattform steht das ”Virtual Component Co-design”-Werkzeug (kurz VCC) seit 1997 von der Firma Cadence Design Systems entwickelt. Diese Umgebung stellt eine Vielfalt an Werkzeugen zur Erstellung von funktionalen und Architektur-Modellen zur Verfügung. Konzeptionell orientiert sich diese Plattform am physikalischen Aufbau einzelner ECUs und ECU-Netzwerken. Sie verfügt über Modelle von typischen Bussystemen, von Echtzeit-

Betriebsystemen, von Mikroprozessoren und weiteren Bestandteilen eines solchen Systems. Darüber hinaus, wird die Wichtigkeit der Mensch-Maschine-Interaktion auch berücksichtigt. Tools, die die verschiedenen Teilnehmer am Entwicklungsprozess unterstützen sollen, wurden im VCC integriert.

Die Anforderungen, die VCC mittlerweile erfüllt, sind folgende [3]:

- Unterstützung von Entwurfsverfeinerung und Optimierung der Kommunikation und Berechnung von abstrakten Modellen zu Implementierungsmodellen
- Wiederverwendung von Komponenten ist auch ein wichtiges Thema in diesem Kontext. Daraus werden weitere Entwürfe abgeleitet
- Modellierung des Verhaltens eines Systems unter idealen Bedingungen und unabhängig von der Implementierung innerhalb wechselnden Bedingungen in jedem Abstraktionsniveau und verschiedenen Netzwerk-Topologien
- Eine Schnittstelle zur Erweiterung des Systems ist auch vorhanden

Universal Communication Model

Die Hauptaufgabe des UCM ist grundsätzlich ein Framework bereitzustellen, welches die Leistung der zwei wichtigsten Kommunikationsparadigmen, Zeit- und Ereignissteuerung, modelliert [4]. Diese Umgebung arbeitet innerhalb VCC und erlaubt präzise Leistungsabschätzungen. Dabei werden auch Verzögerungen im Bus-Netzwerk mit gerechnet. Die hohe Wiederverwendung und die flexible Parametrisierung ermöglichen die Modellierung von zahlreichen verschiedenen Kommunikationsprotokollen. Dieses Prinzip kann sowohl die schon existierende Bussysteme modellieren und simulieren als auch neue entwerfen und untersuchen. Die UCM Arbeitsweise kann als ein Verfeinerungsprozess betrachtet werden. Der VCC Benutzer versucht solange die Parameter einzustellen bis die gewünschte Konfiguration erreicht wird. Somit wird die Erstellung eines virtuellen Prototyps ganz am Anfang im Entwurfsprozess stark vereinfacht.

Abb. 4 zeigt ein grobes Schema von der Arbeitsweise dieser virtuellen Integrationsplattform. Die drei Hauptbestandteile dieses Diagrams sind die externe IP Zulieferer, der Entwicklungsprozess eines Automobilsystems und die VCC-Umgebung. Innerhalb des Entwicklungsprozesses befindet sich die Funktion von VCC zwischen der Spezifikations- und Implementierungsebene. Die Zulieferer besorgen der Plattform eine Reihe von SW-Komponenten, wie beispielsweise Matlab-, C- oder ASCET-Projekten, und virtuellen Architekturbausteinen wie

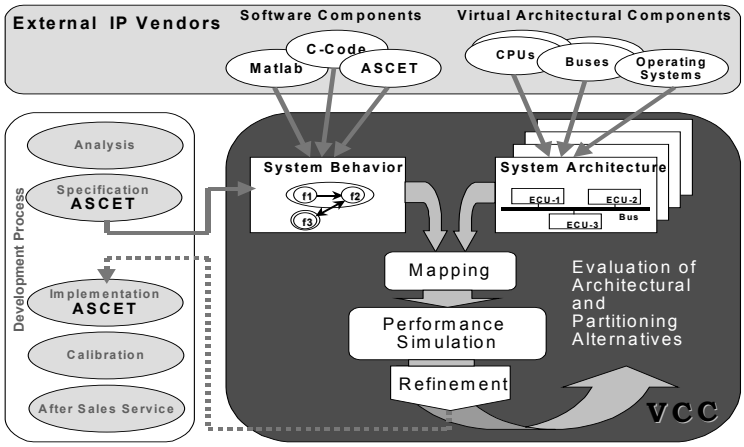

Abbildung 4: Schema der virtuellen Integrationsplattform innerhalb VCC [4].

CPUs oder Betriebsysteme. Daraus werden im Rahmen der VCC-Umgebung das Systemverhalten und Systemarchitektur(-en) separat und unabhängig voneinander gebildet. Die Abbildung des funktionalen Modells auf eine bestimmte Architektur erfolgt nach der Erzeugung der beiden Modelle. Das Leistungsmodell, das sich aus dieser Abbildung ergibt, wird verwendet um die nötigen Simulationen und Verfeinerungsiterationen innerhalb des UCM durchzuführen bis das gewünschte Ergebnis erreicht wird. Dieses Ergebnis wird im nächsten Schritt des Prozesses (Implementierung) benutzt. Eine detallierte Beschreibung des Ablaufs wird in 3.1 gegeben.

3.1 Von der abstrakten Ebene zum Architekturmodell

Ein nahtloser Übergang von einer idealen Welt zur Realität, in der die entwickelten Anwendungen ausgeführt werden, scheint eine wesentliche Voraussetzung für wertvolle und nutzbare, aus der Prototyping-Phase erhaltenen, Ergebnisse zu sein. Im Zentrum dieser Methodologie steht die Idee der Trennung zwischen Verhaltensmodell (ideale, abstrakte Welt) und Architekturmodell (reale Welt). Ein Verhaltensmodell entsteht unter Abwesenheit von Softwareausführungs- und Kommunikations-Latenzzeiten. Dieses Modell enthält die eigentliche Funktionalität des Systems. Ein Architekturmodell stellt eine konkrete Implementierungsvariante dar. Die bereits erwähnte Transition von der abstrakten Ebene zur konkreten Implementierung erfolgt, indem die Funktionalität auf die gewünschte Architektur abgebildet wird. Dabei entsteht eine bestimmte System-Partitionierung und die

ideale Leistung des Verhaltensmodells wird nun durch die Einschränkungen der Hardware limitiert [4].

Die wichtigsten Schritte in der Arbeitsweise von VCC sind folgende:

1. Definition eines Verhaltensdiagramms, indem funktionale Software-Komponenten importiert werden
2. Generierung einer idealen Kommunikation zwischen den Komponenten im Verhaltensmodell
3. Erzeugung eines Architektur-Diagramms
4. Abbildung der Software-Bausteine auf entsprechende ECUs
5. Generierung des CPU-Scheduling
6. Funktionale Simulation
7. Wiederverteilung der Funktionalität und Scheduling-Eichung einzelner CPUs
8. Initialisierung des UCM-Modells
9. Simulation unter realen Bedingungen
10. Parameter-Studie zur Definition einer spezifischen Busprotokoll-Implementierung
11. Simulation unter Beachtung der Bus-Latenzzeiten

Im folgenden werden die relevantesten Aspekte dieses Prozesses ausführlich beschrieben.

Die Funktionalität des Systems wird in VCC mittels des ASCET-SD Werkzeugs realisiert. Die ASCET-SD Entwurfsumgebung der Firma ETAS ist besonders geeignet für die Entwicklung von Automobilanwendungen. Sie erlaubt dem Benutzer die Spezifikation ausführbarer Software-Modelle für jede einzelne ECU, die Simulation dieser Modelle und die Code-Generierung sowohl für die Zielhardware als auch für die Prototyping-Systeme. Diese Umgebung setzt während der Simulation keine Verweilzeit der Ausführung voraus. Algorithmen werden auch in ASCET-SD definiert und weiterentwickelt.

Abb. 5 zeigt die zunächst voneinander unabhängige Teile des Modells, in denen die Definition der ECU-Bestandteile ”gespeichert” wird. Diese heißen Projekte in ASCET-SD und sind hierarchisch organisiert. Die Module innerhalb der Projekte, sind in der zweiten Ebene der Hierarchie angesiedelt. Ein Modul ist die

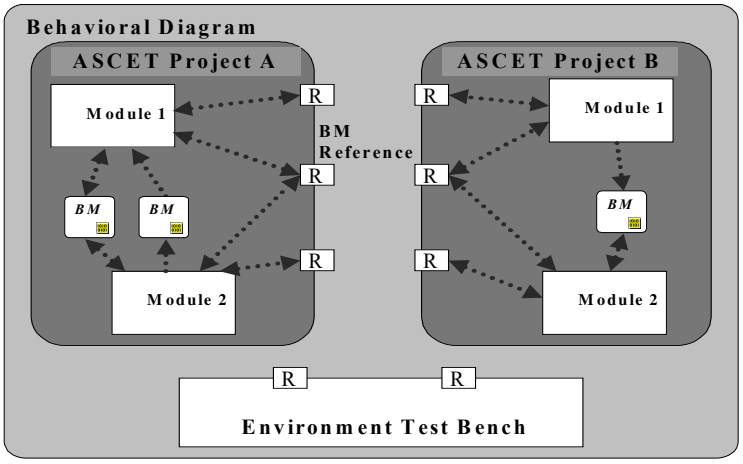

Abbildung 5: Importierte ASCET Projekte in VCC [4]

kleinste abbildbare Komponente, die über das System-Netzwerk verteilt werden kann. Es kann mehrere Prozesse haben, die die zugewiesene Funktion ausführen. Prozesse sind die kleinsten verwaltbaren Einheiten und repräsentieren die Blätter des Hierarchie-Baumes. Verhaltensmodelle können anschliessend automatisch aus den Prozessen oder manuell als C- oder Matlab-Simulink-Code importiert werden. VCC erlaubt eine schnelle und flexible Verteilung der Funktionalität über verschiedene Architekturen mit dem Ziel eine möglichst optimierte Konfiguration zu erreichen. Module kommunizieren mithilfe von globalen Projekt-Variablen und Nachrichten (geschützte Variablen) miteinander.

Die Software-Bausteine im Funktionalen Modell kommunizieren über gemeinsam genutzte Speichereinheiten (engl. *shared memories*). Diese Einheiten werden in VCC *behavioral memories* (BM) genannt. Wie Abb. 6 zeigt, besteht dabei die Möglichkeit aus unabhängigen ASCET-Projekten ein funktionales Netzwerk zu erzeugen, indem die Ports (BM-Referenzen), die an manchen Stellen der Projekten in Abb. 5 zu sehen sind, manuell miteinander verbunden werden. BMs können also mehrere Module miteinander verknüpfen und ferner das ganze Netzwerk mit einem bestimmten Umgebungsmodell verbinden. Dieses Modell ist dafür zuständig das funktionale Modell zu stimulieren. Auf BMs greifen die Module mittels Read- und Write-Operationen zu.

Das Ergebnis dieser Modellierung ist ein architekturunabhängiges Modell. Im nächsten Schritt wird dieses Modell einem Architekturmodell zugeordnet.

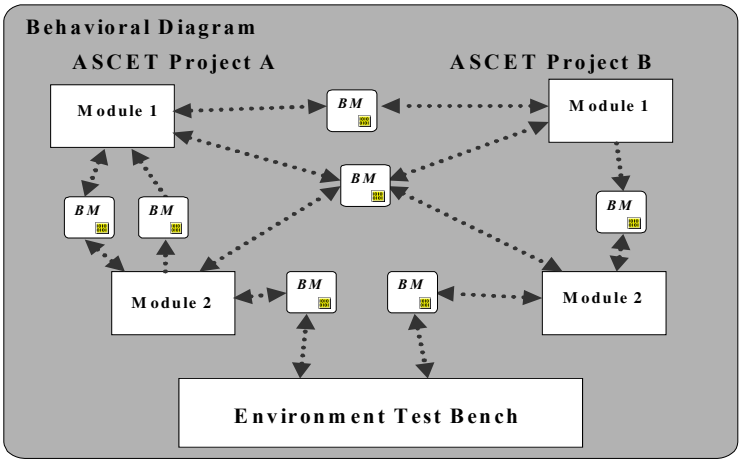

Abbildung 6: Architekturunabhängiges Modell [4]

3.2 Architekturmodell für Kommunikation

Wie bereits in Abschnitt 2 erwähnt, besteht einen Systemnetzwerk für Automobile aus einer Menge von ECUs, die jeweils mit einem Bus verbunden sind. Abb. 7 schematisiert den Aufbau einer physikalischen ECU, indem die vier für die Modellierung relevantesten Bestandteile hervorgehoben werden, nämlich Anwendung, Kommunikationsebene, Bus-Controller und Bus-Treiber.

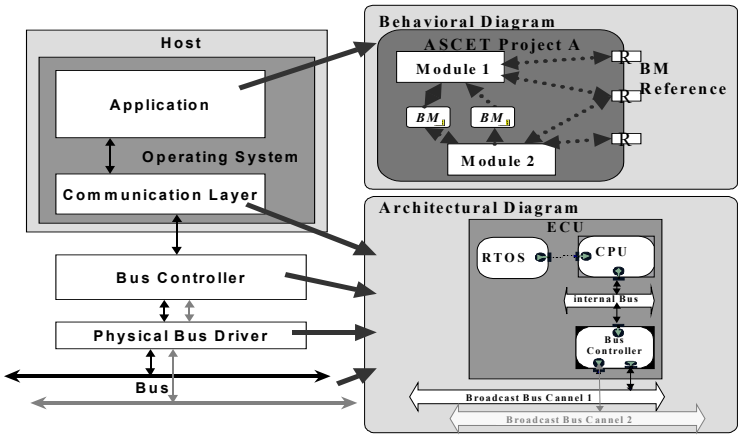

Abbildung 7: Modellierung eines ECUs innerhalb VCC [4]

Die Zeiger von den ECU-Komponenten in die VCC-Diagrammen verraten die Weise, in der diese Elemente modelliert werden. Aufgrund dessen, dass die einzige architekturunabhängige Komponente der ECUs die Anwendungssoftware ist,

wird diese als Verhaltensmodell in VCC importiert. Die restlichen Komponente der ECUs werden als Architekturkomponenten modelliert. Damit ist es unmöglich, dass ein Einfluss von der Architektur in das Verhaltensmodell zustande kommt. Die Schnittstelle zwischen Anwendung und Architektur bleibt unabhängig von der Bus-Implementierung.

3.2.1 UCM Services

Die Modellierung der Leistung und teilweise der Funktionalität eines Bausteins der Architektur in VCC wird mihilfe von virtuellen C++ Funktionen durchgeführt. Solche Funktionen werden als Services oder Dienste bezeichnet. Sie sind das Fundament der Modellierung von UCM und liefern eine anständige Simulationsplattform für ECUs. Der Service-Stack aus Abb. 8 modelliert das Echtzeitbetriebsystem und die CPU eines ECUs sowie den Bus-Controller und stellt eine gemeinsame Schnittstelle zwischen diesen Komponenten zur Verfügung.

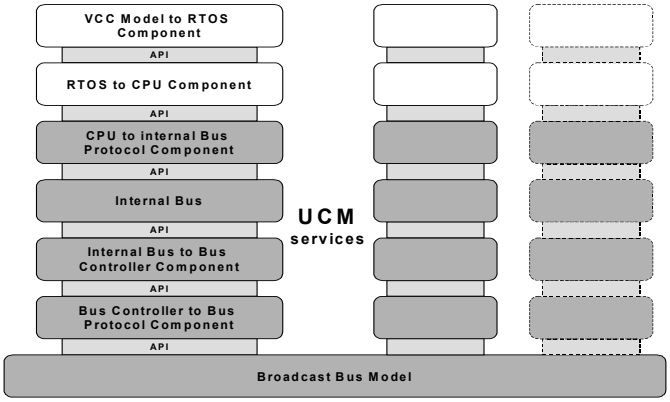

Abbildung 8: Modell eines ECUs mittels UCM Services [4]

Wie in Abb. 8 dargestellt, die schichtenweise Anordnung der Services ähnelt sehr stark dem Aufbau einer ECU (s. den linken Teil der Abb. 7). Zum Zeitpunkt der Durchführung einer Simulation sucht VCC den Pfad aus der Architekturtopologie-Netzliste und verbindet die nötigen Protokollkomponenten innerhalb des UCM. Beispielsweise würden die beiden ersten Schichten des Stacks in einer frühen funktionalen Simulation zum Einsatz kommen, denn in diesem Fall werden die Kommunikationslatenzen nicht betrachtet. Eine Simulation unter realen Bedingungen würde die untersten Schichten verwenden, die eine konkrete Protokoll-Implementierung unterstützen. Graue und weiße Bereiche unterscheiden die Schichten, die die Buskomponenten modellieren von den restlichen.

Ein Vorteil dieses Ansatzes ist, dass eine Veränderung der Architekturtopologie keine neue Modellierung der UCM Services fordert.

Der UCM Ansatz unterstützt busspezifische Kommunikationsverzögerungen, die z.B. durch Umwandlung (”Verpackung”) von Nachrichten in Frames verursacht wird. Die Latenzzeit zwischen Host und Bus-Controller wird nicht berücksichtigt, denn sie ist vernachlässigbar klein (in der Ordnung von nsec) im Vergleich mit der globalen Leistung des Systems.

3.2.2 Broadcasting

Der Begriff Broad- oder Multicast ist wesentlich in heutigen Bussystemen. Wie bereits unter 2.3 gesehen, enthalten weder die Nachrichten eines CAN- noch eines TTP/FlexRay-Knotens Sende- oder Empfängeradressen. Demzufolge werden alle übertragenen Nachrichten von allen Knoten gehört. Aus dem Grund ist die Entwicklung eines Broadcast-Bus-Modells in VCC erforderlich.

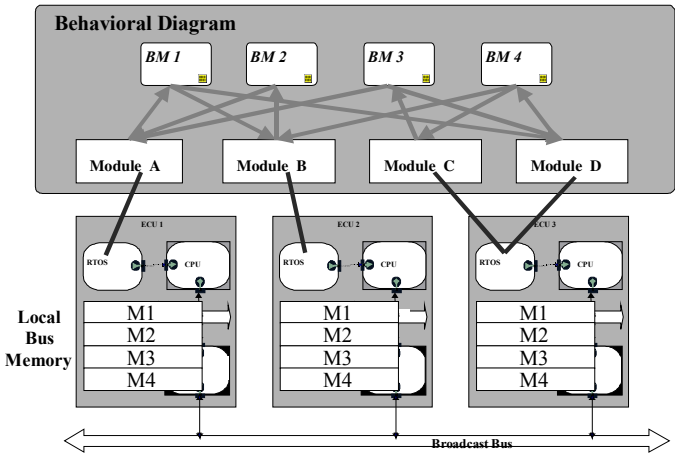

Abbildung 9: Broadcast-Modellierung in VCC [4]

Abb. 9 skizziert ein einfaches Netzwerksmodell zur Veranschaulichung der Arbeitsweise eines Broadcastsmodells in VCC. Die Modellierung erfolgt mittels lokaler Architektur–Busspeichereinheiten (engl. *local bus memory*, LBM), die essentiell für die Darstellung der Latenzzeiten der Kommunikation während Leistungssimulationen sind. Die auf unterschiedlichen ECUs abgebildeten funktionalen Komponenten (Module A – Module D) können verschiedene Versionen der aktuellen Nachricht auf dem Bus lesen. Diese Nachricht wird im Verhaltensdiagramm als *bus type behavioral memory* (BBM) dargestellt, d.h., ein BM, das über

den Bus gesendet wird (im Beispiel BM 1 – BM 4). Aufgrund der fehlenden Modellierung der Schnittstelle zwischen Host-Anwendung und Bus-Controller, wird vorausgesetzt, dass eine Read-/Write-Komponente immer den aktuellsten Wert der Nachricht im Bus lesen wird. Wie in Abb. 9 gezeigt wird, besitzt jedes BBM ein LBM in jedem Knoten des Netzwerks. Ausgehende Pfeile von den Modulen zu den BMs entsprechen Schreibe-, eingehende Lesetransaktionen. Dementsprechend bekommt jeder Knoten eine Bus-Transaktion, obwohl die in ihr enthaltene Information irrelevant für das Verhalten mancher ECUs sein könnte.

Auf eine Bus-Nachricht wird von den Modulen direkt mittels der LBMs eines Knotens zugegriffen. Dieser Zugriff hängt von den Lese-/Schreibe-Operationen der BBMs ab und somit auch vom Scheduling des Echtzeitbetriebsystems. Gleichzeitig werden die LBMs von den UCM Services asynchron gesteuert und aktualisiert. Buslatenzzeit wird dabei auch beachtet. LBM Vektoren einzelner ECUs werden danach in einer sogenannten Kommunikationsmatrix zusammengefasst, die zur Verfeinerung der Kommunikation und der Buseinstellungen in späteren Iterationen dient. Fehlerbehandlung oder Diagnosemechanismen wurden in UCM nicht implementiert.

3.2.3 Kommunikationszyklus

Der Kommunikationszyklus in UCM ist mit dem von FlexRay (s. 2.3.2) vergleichbar. Er basiert auf der Einteilung des Bus-Modells in statischen und dynamischen Bereichen. Statische Zonen werden ausschliesslich für zeitgesteuerte Frames reserviert, dynamische für ereignisgesteuerte. Die Flexibilität dieses Prinzips gestattet dem Entwerfer viele verschiedene Kommunikationsprotokolle zu untersuchen.

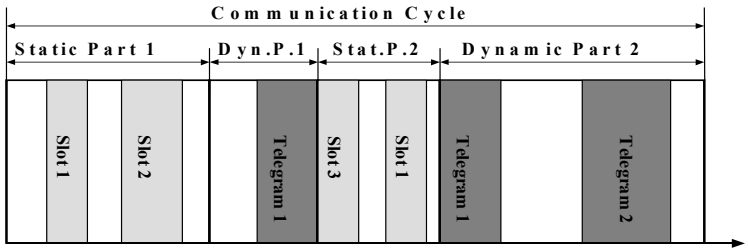

Abbildung 10: Kommunikationszyklus in UCM [4]

Genauso wie bei TTP werden bei UCM die zeitgesteuerten Frames als Slots nach dem TDMA Schema übertragen. Im dynamischen Bereich wird der Buszugriff mittels eines Arbitrierungsalgorithmus entschieden. Ereignisgesteuerte Frames

werden in UCM *telegrams* genannt. Eine globale Zeit steuert den Kommunikationszyklus.

4 Bewertung

Der in dieser Ausarbeitung beschriebene Ansatz der virtuellen Integrationsplattform wurde hauptsächlich dafür entworfen und entwickelt, um die Kosten, die Dauer und die Komplexität des Entwurfs- und Entwicklungsprozesses eines Automobils zu reduzieren. Die zur Verfügung gestellte Umgebung erlaubt die kostengünstige Erstellung und Simulation von zahlreichen virtuellen Versionen des möglichen endgültigen Produktes. Eine Optimierung des Produktes kann dabei erreicht werden, indem die Parameter der virtuellen Plattform solange verändert werden, bis eine optimale Lösung eines bestimmten Designs gefunden wird. Aus dem Grund, dass die Arbeitsweise der vorgestellten Methodologie kein starres Wasserfallmodell sondern ein zyklisches Modell darstellt, nimmt das Risiko eines derartigen Entwurfs ab. Die Berücksichtigung der Sicherheit in der Simulation in Kombination mit den restlichen typischen Anforderungen eines Automobil-Systems, bringt gute Voraussetzungen für die Qualität (zumindest was die Verlässlichkeit angeht) des Endproduktes mit. Diese Aspekte haben logischerweise eine deutliche Senkung der Fertigungssdauer und Produktionskosten zur Folge. Dabei wird aber das Testen mit realen Hardware-Prototypen auf der Strecke oder im Labor nicht völlig abgeschafft, sondern dieses Werkzeug soll im Grunde einen für das gewünschte Verhalten geeigneten Hardware-Prototypen auswählen und somit eine gezielte Testphase durchführen können. Entwurfsfehler können vor der Integrationsphase erkannt und behoben werden. Zusätzlich vereinfacht diese Methodologie die Interaktion zwischen den unterschiedlichen System-Entwicklern, indem spezifische Tools für die verschiedenen Anwendungsdomänen geliefert werden.

Ein wichtiger Begriff, wie die Validierung der modellierten Architektur, fehlt in diesem Kontext. Theoretisch ist also ein Beweis der Korrektheit der funktionalen Architektur mit der vorgestellten Methodologie nicht möglich. [1] befasst sich mit diesem Thema im Zusammenhang mit AUTOSAR. Innerhalb VCC existieren auch einige Ansätze zur Analyse von unterschiedlichen Design-Implementierungen mit dem Multi-Criteria Decision Making (MCDM) als Basis der Evaluierung [7].

Auf der technischen Seite kann diese Methodologie noch weiter entwickelt und verbessert werden. Bezüglich des universellen Kommunikationsmodell innerhalb VCC scheint die Fehlerbehandlung ein Bereich zu sein, in dem noch weitere

wichtige Fortschritte gemacht werden können. Beispielsweise sind Fehler an Bus-Hardware-spezifischen Komponenten nicht behandelt. Dieses Feature wird zukünftig eingesetzt, indem entweder eine Verfeinerung der UCM Services durchgeführt wird oder spezifische Busprotokoll-Modelle explizit importiert werden.

Ferner existiert theoretisch die Möglichkeit in VCC, dass mehrere Knoten die selbe Nachricht über das Bus-Netzwerk übertragen. Dies ist die Folge des von der Methodologie angebotenen breiten Mapping-Spektrums und die fehlende Modellierung der Software für die Bus-Kommunikationsschicht. Praktisch wird dieses Problem mithilfe des Broadcast-Bus-Modells überwunden.

Ein weiterer etwas widersprüchlicher Aspekt ist die Implementierung eines Arbitrierungsalgorithmus in UCM. Eine richtig modellierte zeitgesteuerte Kommunikation, aufgrund ihres deterministischen Charakters, benötigt keinen solchen Algorithmus. Sogar eine optimierte Modellierung des dynamischen Segments, ähnlich wie die bei FlexRay, würde eine kollisionsfreie Kommunikation ergeben und somit wäre die Arbitrierung wiederum unnötig.

Die hier vorgestellte Methodologie stellt im Grunde den klassischen Ansatz zur Modellierung eines Automobilsystems dar. Wünschenswert wäre in diesem Prozess eine möglichst vollständige Automatisierung des Deployments, obwohl dies an bestimmten Stufen äusserst kompliziert sein würde, wie beispielsweise die Erstellung einer systemspezifische Partitionierung nach der Abbildung der Funktionalität auf eine Architektur. Dieser Schritt wird meistens manuell durchgeführt und repräsentiert den schwierigsten Schritt im Entwicklungsprozess.

Ein weiterer Ansatz, der in diesem Zusammenhang erwähnt sein muss, ist das vorher angesprochene AUTOSAR. AUTOSAR unterstützt eine allgemeinere und abstraktere Entwicklung von Automobilsystemen und basiert auf der Teilung der Systeme in einzelnen Komponenten. Daher wird dieses Paradigma als komponentenbasierte Softwareentwicklung gekennzeichnet. Die Erzeugung mancher Komponente hängt von anderen ab. Solche Abhängigkeiten zwingen dem Entwickler einer bestimmten Arbeitsweise zu gehorchen, nämlich die AUTOSAR-Methodik. Diese Methodik basiert auf einer Reihe von Arbeitsprodukten und Werzeugen, die sukzessiv, falls gewisse Voraussetzungen erfüllt sind, weitere Komponenten erzeugen. Die Hauptphase dieser Methodik ist die Generierung einer vollständigen Beschreibung der Konfiguration der ECUs. Diese Beschreibung enthält beispielsweise Information über das Betriebsystem, die Laufzeit oder die unterschiedlichen Implementierungsmöglichkeiten für die funktionale Bausteine. Diese Elemente können dann in ausführbaren Dateien automatisch umgewandelt werden und auf den entsprechenden ECUs laufen lassen. Die Entscheidung über welche ECUs an dieser Stelle in Frage kommen, wird während der Mapping-Phase getroffen. Das Mapping wird in dieser Methodik manuell in der Systemkonfiguration früh im

Entwicklungsprozess durchgeführt. Ähnlich wie bei der Arbeitsweise von VCC werden die verschiedenen Schritte dieses Prozesses mehrfach durchgeführt bis die gewünschte Konfiguration des Systems erreicht wird. Simulationsmöglichkeiten für die Kommunikation im Netzwerk, wie die von VCC, existieren in diesem Prozess nicht. In der Tat, abstrahiert AUTOSAR zu stark von der Netzwerktopologie für eine solche Simulation.

5 Zusammenfassung

Die Kommunikation innerhalb eingebetteter Systemen ist ein umfangreiches und interessantes Thema, von dem in dieser Ausarbeitung eine kleine Teilmenge betrachtet wurde. Diese Teilmenge enthält die Kommunikation innerhalb der Elektronik eines Kraftfahrzeugs. Dieses System besteht grundsätzlich aus einem Netzwerk von Steuerungseinheiten, die über einen Bus verbunden sind. Die Komplexität dieses Systemnetzwerks kann unter Umständen bis zu einem Punkt wachsen, an dem die Übersicht verloren geht. Dieses exponentielle Wachstum in der Integration neuer Funktionalität in Form von elektronischen Steuerungskomponenten im Automobil fordert eine Neustrukturierung des bisherigen Entwicklungsprozesses. Ein ”virtueller Zweig” muss im Entwurfsmodell hinzugefügt werden. Zu diesem Zweck wird seit 1997 das Virtual Component Co-Design Tool (VCC) entwickelt, welches die Erstellung eines frühen virtuellen Prototypen des Systems erlaubt. VCC basiert auf dem Prinzip der getrennten Modellierung von Funktionalität bzw. Verhalten des Systems und Architektur. Die anschliessende Abbildung des Verhaltens auf die Architektur liefert eine bestimmte Systempartitionierung und ein Performancemodell, welches die Untersuchung des Systems unter realen Bedingungen erlaubt. Darüber hinaus, ist zur Modellierung der Kommunikation das Universal Communication Model (UCM) in VCC zuständig. Diese Umgebung modelliert die Performance der zwei grundlegenden Kommunikationsparadigmen, nämlich Zeit- und Eventsteuerung.

Literatur

[1] AXEL KASKE, GUILLIAUME FRANÇOIS, *et al.*: *Virtual Prototyping for validation of functional architectures*. `en.etasgroup.com/about/tradeshows/documents/ERTS_VirtualPrototyping_Francois_2006-01-24.pdf`.

[2] BOSCH: *CAN Specification, Version 2.0*, 1991. http://www.semiconductors.bosch.de.

[3] BRUNEL, JEAN-YVES und ELIANE FOURGEAU: *Automotive Virtual Integration Platforms: Why's, What's, and How's*. IEEE Computer Society, Seite 370, 2002.

[4] DEMMELER, THILO und PAOLO GIUSTO: *A universal communication model for an automotive system integration platform*. In: *DATE '01: Proceedings of the conference on Design, automation and test in Europe*, Seiten 47–54, Piscataway, NJ, USA, 2001. IEEE Press.

[5] KOPETZ, HERMANN: *A comparison of CAN and TTP*. Technischer Bericht, Technishe Universitat Wien, Austria, 1998. citeseer.ist.psu.edu/kopetz98comparison.html.

[6] LEEN, GABRIEL und DONALD HEFFERNAN: *Expanding Automotive Electronic Systems*. www.cs.umd.edu/class/spring2002/cmsc818m/doc/0220/expanding.pdf.

[7] PRAERIT GARG, ASEEM GUPTA und JERZY W. ROZENBLIT: *Performance Analysis of Embedded Systems in the Virtual Component Co-Design Environment*. In: *ECBS '04: Proceedings of the 11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS'04)*, Seite 61, Washington, DC, USA, 2004. IEEE Computer Society.

[8] RALF BELSCHNER, *et al.*: *FlexRay - ein Kommunikationssystem für das Automobil der Zukunft*. Elektronik Automotive, 2002.

[9] SCHIRRMEISTER, FRANK und ALBERTO SANGIOVANNI-VINCENTELLI: *Virtual Component Co-Design – Applying Function Architecture Co-Design to Automotive Applications*, 2001. http://ieeexplore.ieee.org/iel5/7612/20759/00961757.pdf?arnumber=961757.