Matthias Traub

Durchgängige Timing-Bewertung von Vernetzungsarchitekturen und Gateway-Systemen im Kraftfahrzeug

KIT Scientific Publishing

Matthias Traub

Durchgängige Timing-Bewertung von Vernetzungsarchitekturen und Gateway-Systemen im Kraftfahrzeug

Durchgängige Timing-Bewertung von Vernetzungsarchitekturen und Gateway-Systemen im Kraftfahrzeug

von
Matthias Traub

KIT Scientific Publishing

Dissertation, Karlsruher Institut für Technologie
Fakultät für Elektrotechnik und Informationstechnik, 2010

Impressum

Karlsruher Institut für Technologie (KIT)
KIT Scientific Publishing
Straße am Forum 2
D-76131 Karlsruhe
www.ksp.kit.edu

KIT – Universität des Landes Baden-Württemberg und nationales
Forschungszentrum in der Helmholtz-Gemeinschaft

KIT Scientific Publishing 2010
Print on Demand

ISBN 978-3-86644-582-6

Durchgängige Timing-Bewertung von Vernetzungsarchitekturen und Gateway-Systemen im Kraftfahrzeug

Zur Erlangung des akademischen Grades eines

D O K T O R - I N G E N I E U R S

an der Fakultät für

Elektrotechnik und Informationstechnik

Karlsruher Institut für Technologie (KIT)

genehmigte

D I S S E R T A T I O N

von

Dipl.-Ing. Matthias Alexander Traub

geb. in: Stuttgart

Tag der mündlichen Prüfung: 29.06.2010

Hauptreferent: Prof. Jürgen Becker

Korreferent: Prof. Wolfgang Rosenstiel

Juli 2010

Danksagung

Diese Arbeit entstand während meiner Tätigkeit als Mitarbeiter im Bereich Forschung und Vorentwicklung bei der Daimler AG. Die wissenschaftliche Betreuung der Arbeit erfolgte an der Fakultät für Elektrotechnik und Informationstechnik am Institut für Technik der Informationsverarbeitung des Karlsruher Instituts für Technologie.

An dieser Stelle möchte ich Allen, die zum Gelingen dieser Dissertation beigetragen haben, meinen persönlichen Dank aussprechen. Mein besonderer Dank gilt meinem Doktorvater Prof. Jürgen Becker für die hervorragende Betreuung und Unterstützung. Die gemeinsamen Besprechungen waren für mich von sehr großem Wert. Seine Anmerkungen und Hinweise haben mich bei der Entwicklung der Ansätze immer weitergebracht. Herrn Prof. Rosenstiel danke ich für die Übernahme des Zweitgutachtens.

Mein weiterer Dank gilt meinen Kolleginnen und Kollegen, die mir jederzeit mit Rat und Tat zur Seite standen. Mein Dank gebührt auch den zahlreichen Praktikanten und Diplomanden, die mich bei der praktischen Umsetzung meiner Ansätze tatkräftig unterstützen. Für die vielen gemeinsamen Diskussionen und Arbeiten möchte ich mich bei Dr. Kai Richter von Symtavision bedanken.

Für den moralischen Beistand bin ich meinen Freunden sehr dankbar, insbesondere Eva und Wolfgang die mir während der Endphase eine große Hilfe waren. Weiterhin möchte ich an dieser Stelle meinen Eltern ganz herzlich danken. Ihr Vertrauen bildete die Grundlage für diese Arbeit.

Mein ganz besonderer Dank gilt meiner geliebten Frau. Ihre stetige Motivation und ihr Verständnis hat mir insbesondere in schwierigen Phasen den Mut gegeben die Arbeit erfolgreich zu Ende zu bringen. Abschließend möchte ich noch meinem Sohn Lorenz danken, der in der letzten Zeit so oft zu kurz kam und mir in den wenigen gemeinsamen Stunden viel Kraft gegeben hat. Meinem im Mai geborenen Sohn Kilian danke ich für den Motivationsschub zur Vollendung dieser Arbeit.

Tübingen, im Juli 2010 *Matthias Traub*

Kurzfassung

Die steigende Anzahl von Elektrik-/Elektronik-Systemen im Automobil und damit verbunden das zunehmende Kommunikationsaufkommen stellen immer höhere Anforderungen an den Entwicklungsprozess. Bei der Entwicklung von zukunftsfähigen E/E-Architekturen spielt der Entwurf einer robusten Vernetzung eine zentrale Rolle. Die Auslegung und Absicherung einer solchen Vernetzungsarchitektur erfolgte bisher mittels einfacher Metriken, z.B. Berechnung der Buslast von zyklischen Botschaften. Durch die wachsende Komplexität sowie aufgrund gesetzlicher und sicherheitsrelevanter Anforderungen muss das Timing-Verhalten der Systeme zukünftig gezielt betrachtet und bewertet werden. In der vorliegenden Arbeit wird eine Methodik beschrieben, welche eine detaillierte und durchgängige Auslegung und Absicherung von Vernetzungsarchitekturen und Gateway-Systemen hinsichtlich deren Timing-Verhaltens ermöglicht.

In den letzten dreißig Jahren haben sich verschiedene Forschergruppen mit der Frage beschäftigt, wie das Timing-Verhalten von eingebetteten Systemen analytisch bestimmt werden kann. Aus diesen wissenschaftlichen Aktivitäten sind eine Vielzahl von Ansätzen und Verfahren entstanden. Ein Ziel der vorliegenden Arbeit ist es diese Verfahren auf ihre Eignung für Fragestellungen im Automobilbereich zu untersuchen. Ferner wird ein Verfahren zur Extraktion von Timing-Informationen vorgestellt, welches ein detailliertes Bild zum Stand der aktuellen E/E-Systeme im Fahrzeug liefert. Die gewonnen Daten können weiterhin für eine Modellverfeinerung in der Entwurfsphase von E/E-Architekturen verwendet werden. Um eine exakte Abbildung des Timing-Verhaltens in den entsprechenden Bewertungsverfahren zu ermöglichen, erfolgt die Definition von Modellierungsregeln. Für die Integration der Bewertungsverfahren in den existierenden E/E-Entwicklungsprozess wird eine durchgängige Bewertungsmethodik aufgezeigt. Weiterhin wird ein Konzept für die Ableitung von Routing-Testpattern auf der Basis von Timing-Analysen vorgestellt. Dieses ermöglicht eine gezielte Berücksichtigung des Timing-Verhaltens von Steuergeräten mit Gateway-Anteilen bei den Funktionstests am Komponenten-Prüfstand.

Anhand von Fallbeispielen wird die durchgängige Bewertungsmethodik validiert, um die Abdeckung der Eigenschaften und Anforderungen aus dem Vernetzungsbereich nachzuweisen. Die Ergebnisse zeigen die Tauglichkeit der Methodik für den Einsatz im Serienprozess. Der Entwurf und die Entwicklung von E/E-Architekturen werden durch die Methodik signifikant verbessert.

Summary

The growing number of electric-/electronic-systems in the automobile and therewith the increasing volume of communication are placing even higher demands on the development process. In the development of future electric-/electronic-architectures the design of a robust network-architecture plays a central role. The design and verification of such network-architectures has taken place up to now by means of simple metrics, for example the calculation of the bus loads of cyclic messages. Through the growing complexity and also based on legal requirements, the timing behaviour of these systems must be considered with the future in mind and evaluated accordingly. In this thesis a methodology is described, which enables a detailed design and verification of network-architectures and gateways with regard to their timing behaviour and resource demands.

In the past 30 years various researchers have focused on the question, how the timing behaviour of embedded systems can be analysed in a formal way. Several approaches and methods are created based on these scientific activities. The goal of this dissertation is to investigate the methods regarding the suitability of issues in the automobile field. Furthermore a method for the extraction of timing information will be introduced, which offers a detailed view over the actual state of the art of automotive e/e-systems. The extracted timing data can be used for a refinement of models during the early design phase. Moreover concepts of modeling rules were carried out, which guarantee an exact observation of the timing behaviour of the electric-/electronic-systems. For the integration of timing evaluation methods in the existing e/e-development process a seamless timing-evaluation will be introduced. Finally an concept for a generation of routing test patterns was developed, which allows a significant increase of the test coverage when examining gateway-control units on the hardware-in-the-loop test bench.

Based on case studies (e.g. such as CAN-bus, central gateway-control unit and distributed functions) the concepts were verified for the integrated evaluation methodology in order to prove the coverage of the typical properties and requirements from the area of network-architectures. The results of the investigation have demonstrated the suitability

of the approach for the integration into the existing development process. The use of the timing-evaluation methodology for systems in the automotive area was significantly improved and offers a large added value during the design and development of future automotive electric-/electronic-architectures.

Inhaltsverzeichnis

1. Einleitung

1.1. Vernetzungsarchitekturen im Kraftfahrzeug

Heutige Fahrzeuge, insbesondere der Ober- und Luxusklasse, sind mit einer großen Anzahl an Funktionen ausgestattet, die dem Fahrer und den Passagieren ein hohes Maß an Komfort und Sicherheit bieten. Beispiele hierfür sind: Die fahrdynamischen Sitze, welche bei starker Querbeschleunigung den Insassen mehr Halt geben, Navigationssysteme mit 3D-Darstellung und dynamischer Routenberechnung sowie sogenannte Assistenzfunktionen wie Abstandsregeltempomat, Fahrspur- und Lichtassistenten, die dem Fahrer Routineaufgaben abnehmen (siehe z.B. [140] und [19]). Die Basis für diese Funktionen sind in zunehmendem Umfang Elektrik-/Elektronik-Systeme. Keine andere Technologie hat das Auto in den vergangenen 30 Jahren so stark verändert wie die Elektronik. Elektrik und Elektronik (E/E) machen heute rund 30 Prozent der Wertschöpfung eines Mittelklassefahrzeugs aus und sind die wesentlichen Treiber für etwa 90 Prozent aller Innovationen im Automobil [30]. In Abbildung 1.1 ist die Zunahme dieser Systeme anhand der Anzahl an vernetzten Komponenten und Bussen am Beispiel zweier Modellreihen der Ober- und Luxusklasse über eine Zeitspanne von 15 Jahren skizziert.

Die steigende Funktionalität und Komplexität von Elektrik-/Elektronik-Architekturen (E/E-Architekturen) und das damit verbundene zunehmende Kommunikationsaufkommen stellen immer höhere Anforderungen an den Entwicklungsprozess. Eine zentrale Rolle bei der Entwicklung spielt die Auslegung robuster und zukunftsfähiger Vernetzungsarchitekturen [131]. In bisherigen Topologien dominierten CAN- und LIN-Busse. Aktuelle und zukünftige Vernetzungsarchitekturen werden durch die notwendige Einführung von neuen Bussystemen wie FlexRay und Ethernet wesentlich heterogener. Eine Folge sind zusätzliche Herausforderungen, die sich bei der Auslegung und Absicherung der Gateway-Steuergeräte ergeben, da diese nun das Routing von Informationen zwischen unterschiedlichen Protokollen durchführen müssen. Eine derzeit typische Vernetzungsarchitektur eines Fahrzeugs der Luxusklasse ist in Abbildung 1.2 am Beispiel

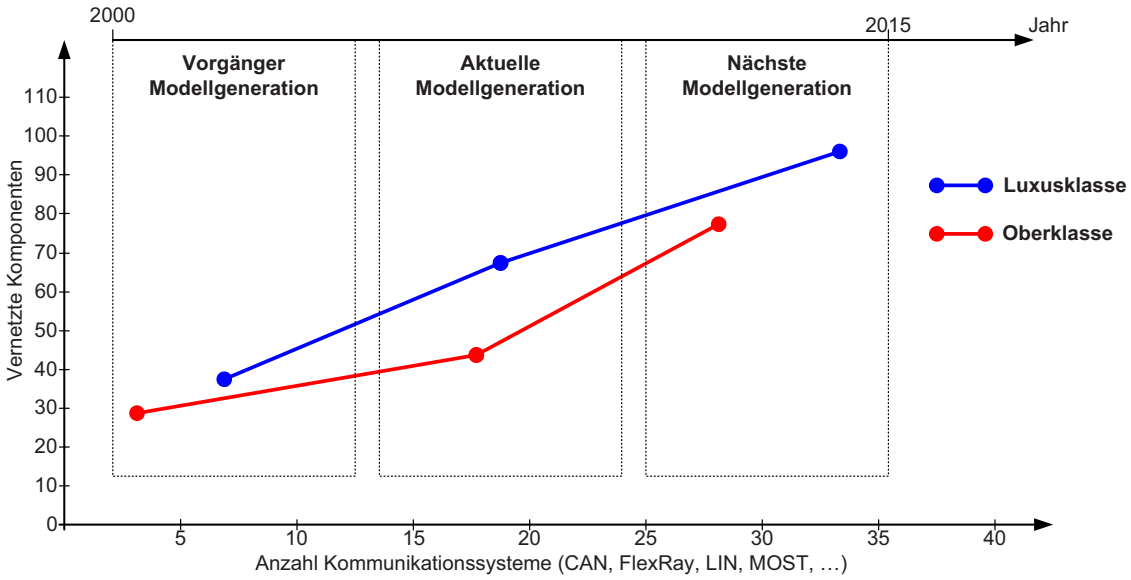


Abb. 1.1.: Anstieg der Anzahl an vernetzten Komponenten und Bussen am Beispiel zweier Modellreihen mit durchschnittlicher Ausstattung

eines aktuellen Fahrzeugs von BMW dargestellt. Die Abbildung zeigt die hohe Anzahl an Steuergeräten und Kommunikationssystemen sowie welche wichtige Rolle das zentrale Gateway-Steuergerät (ZGW) einnimmt.

Zusätzliche Komplexität bringt die funktionale Hochintegration mit sich, welche das Ziel hat die Anzahl der Steuergeräte im Fahrzeug zu reduzieren bzw. konstant zu halten. Insbesondere die Auslegung und die Absicherung des Schedules werden durch die Vielzahl an Funktionen wesentlich aufwändiger. Diese Herausforderungen gilt es im Entwicklungsprozess von E/E-Architekturen zu adressieren. Die funktionalen Anforderungen werden noch durch weitere Anforderungen hinsichtlich Qualität, Testbarkeit, Diagnostizierbarkeit ergänzt [33]. Ziel des Entwurfs von E/E-Architekturen ist ein Höchstmaß an Zuverlässigkeit und dies möglichst kostenoptimal. Die Skalierbarkeit darf dabei nicht vernachlässigt werden. Erstens dient eine E/E-Architektur meist als Plattform für mehrere Baureihen, zweitens erhöht sich das Kommunikationsaufkommen durch die Integration von neuen Funktionen, zum Beispiel bei der Modellpflege, innerhalb eines Produktlebenszyklus einer Baureihe. Hierfür sind bei der initialen Entwicklung einer E/E-Architektur entsprechende Reserven einzuplanen. Die zukünftigen Anforderungen können über die Abfrage der Produktstrategie abgeleitet werden und zeigen welche Funktionalinnovationen in den nächsten Jahren zur Umsetzung anstehen. In diesem Zu-

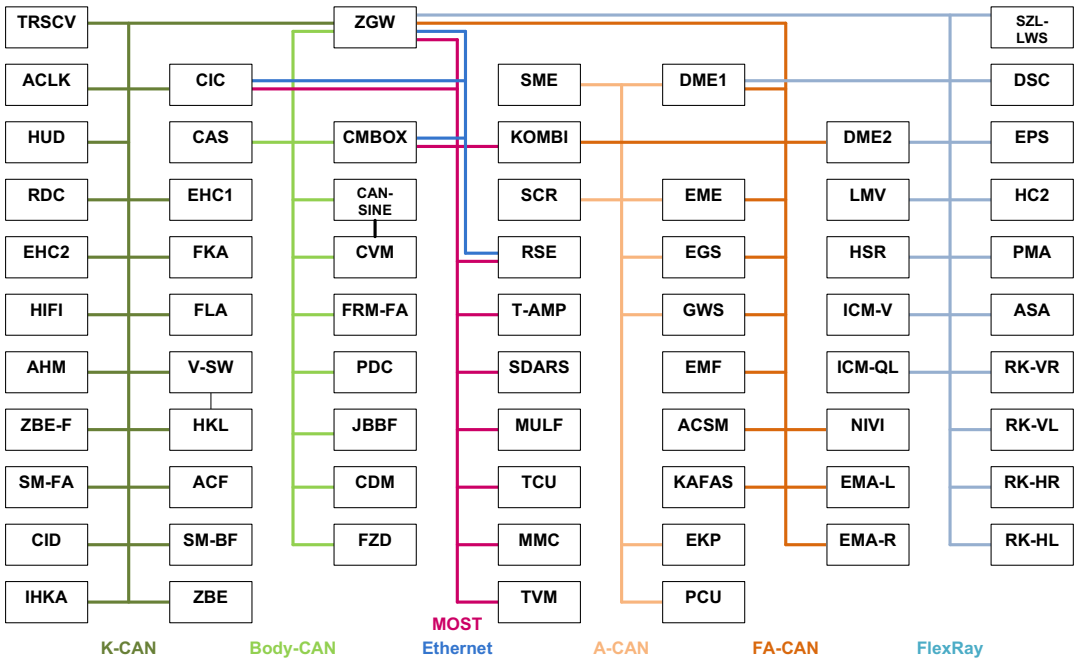


Abb. 1.2.: Vernetzungsarchitektur eines Fahrzeugs der Luxusklasse am Beispiel von BMW [40]

sammenhang spielen der Entwurf und die Auslegung der Vernetzungsarchitektur eine zentrale Rolle. Dabei gilt es die relevanten Anforderungen der Funktionen in der Entwurfsphase zu berücksichtigen und während der Integrationsphase abzusichern.

1.2. Zunehmende Anforderungen an die Bewertung von Vernetzungsarchitekturen

Die derzeit eingesetzten Verfahren zur Bewertung von Vernetzungsarchitekturen reichen nicht mehr aus, um eine vollständige Untersuchung der Systeme durchzuführen. Hierfür gibt es mehrere Gründe. Erstens werden die CAN-Busse immer öfter an deren Lastgrenze betrieben, so dass die bisher eingesetzten Metriken zur Buslastberechnung als einzige Absicherung nicht mehr genügen, zweitens treten durch die zunehmende Heterogenität der Vernetzungsarchitekturen Timing-Effekte auf, die es zu berücksichtigen gilt. Diese Effekte spielten in reinen CAN-Topologien keine Rolle. Drittens steigt die Anzahl der verteilten Funktionen kontinuierlich an. Ein Beispiel für den Anstieg der verteilten Funktionen ist in Abbildung 1.3 anhand verschiedener Baureihen der Audi AG dargestellt. Insbesondere im Bereich der Fahrerassistenzfunktionen wird deutlich sichtbar, dass die Verteilung stark zunimmt.

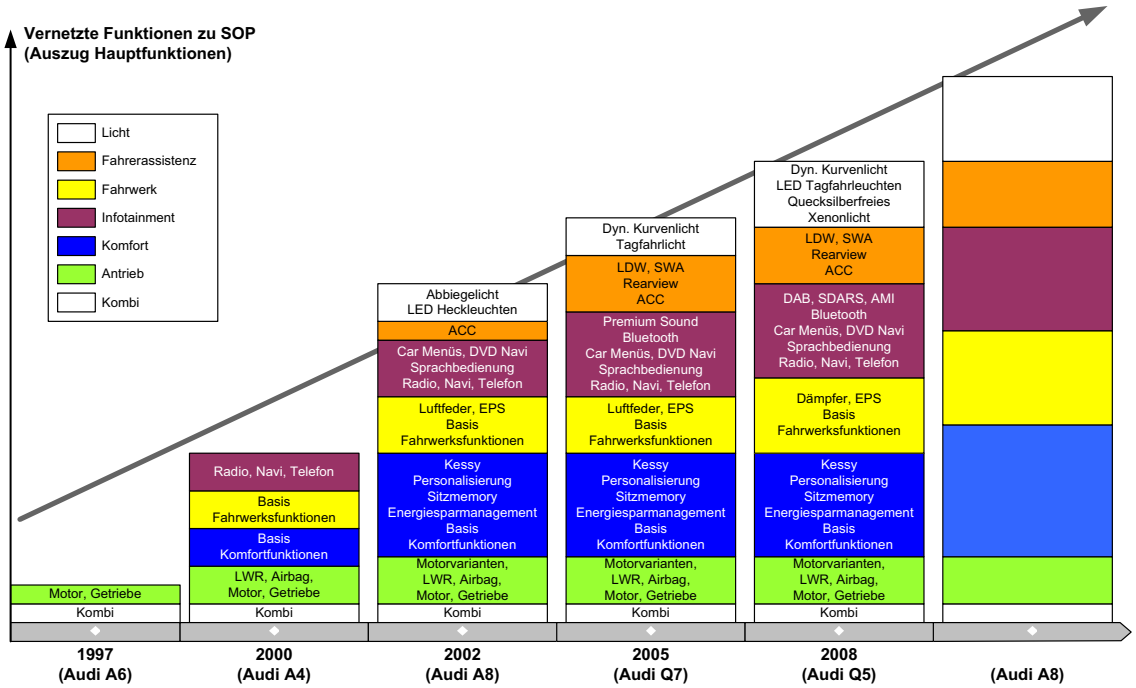


Abb. 1.3.: Steigende Anzahl an vernetzten Funktionen am Beispiel der Produktlinien der Audi AG [33]

Viele dieser verteilten Funktionen sind sicherheitskritisch und müssen daher einer erweiterten Absicherung unterzogen werden. Hinsichtlich des Timing-Verhaltens stehen dabei die Latenzzeiten und Jitter der über das Netzwerk übertragenen Botschaften und Signale im Fokus. Ein exemplarischer Ende-zu-Ende-Pfad von einem Sensor bis zu einem Aktuator ist in Abbildung 1.4 skizziert. Ein solcher Pfad kann mehrere Steuergeräte und Busse enthalten. Um für so einen Pfad eine Timing-Bewertung durchführen zu können, sind die einzelnen Pfadsegmente im Detail zu untersuchen. Es müssen die Latenzzeiten, die bei der Übertragung auf den Bussen entstehen, berücksichtigt werden ebenso die Latenzzeiten, die in beteiligten Steuergeräten selber auftreten.

Ein weiterer wichtiger Punkt für den Einsatz von Timing-Bewertungsverfahren im Entwicklungsprozess sind die gesetzlichen Anforderungen und Richtlinien, welche bei der Entwicklung einer E/E-Architektur zu berücksichtigen sind. Als Beispiele kann hier die Richtlinie für die *On-Board-Diagnose (OBD)* genannt werden sowie die Norm *ISO26262*. Diese Norm ist für sicherheitskritische elektrisch/elektronische Systeme in Kraftfahrzeugen zukünftig von Relevanz.

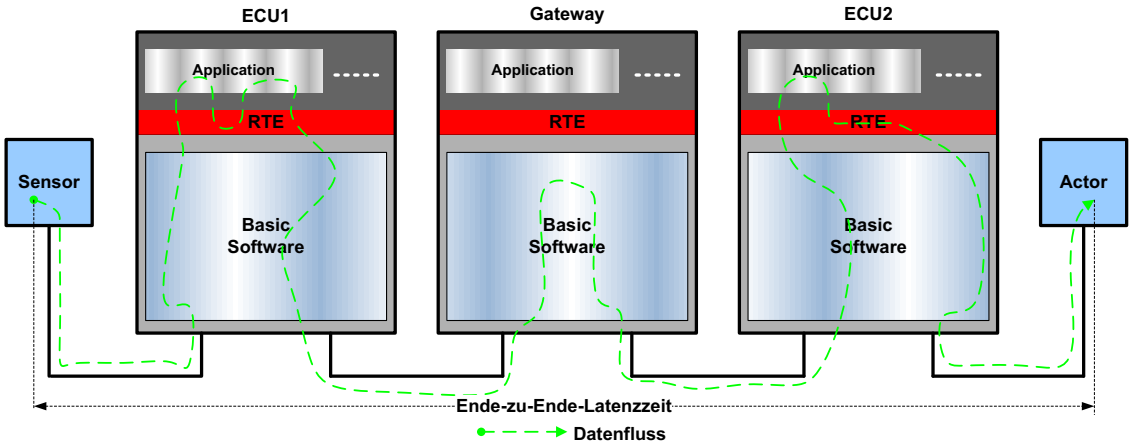


Abb. 1.4.: Beispiel für eine Ende-zu-Ende-Latenzzeit für ein Signal vom Sensor bis zum Aktor

1.3. Zielsetzungen der Arbeit

Die vorgestellten Trends und Anforderungen erfordern den Ausbau der Bewertungsmethodik für Vernetzungsarchitekturen, um auch zukünftige Fahrzeuge mit robusten und erweiterbaren E/E-Architekturen entwickeln zu können. Insbesondere bei sicherheitskritischen verteilten Funktionen müssen schon in einer frühen Entwicklungsphase Aussagen bezüglich deren Anforderungen an eine Vernetzungsarchitektur und deren Realisierbarkeit getroffen werden. Ein solches Vorgehen für eine frühzeitige und erweiterte Konzeptabsicherung führt zu einer Qualitätssteigerung und einer Reduzierung von Kosten für die Behebung von Fehlern, die ansonsten erst in einer viel späteren Entwicklungsphase zum Vorschein kämen [106].

Seit mehreren Jahren wird der Einsatz von Timing-Bewertungsverfahren im Entwicklungsprozess von E/E-Architekturen im Kraftfahrzeug vorangetrieben. Aktuell werden in verschiedenen Forschungsprojekten und Gremien die Methoden erarbeitet und die notwendigen Austauschformate spezifiziert, z.B. im *AUTOSAR-Konsortium* und im von der Europäischen Union geförderten *TIMMO-Projekt* ([15], [120]). Auch Fachzeitschriften beschäftigen sich zunehmend mit dem Thema (zum Beispiel in [57] und [58]). Einige Ansätze zur Integration der Verfahren in den Entwicklungsprozess wurden auch schon von OEMs und Zulieferern auf Konferenzen vorgestellt (zum Beispiel in [107], [95], [79] und [29]).

Die folgenden Punkte wurden in diesen Projekten und Arbeiten bisher jedoch nur unvollständig adressiert:

- Der Einsatz und die Einordnung der verschiedenen Timing-Bewertungsverfahren in den automotive E/E-Entwicklungsprozess wurden bisher nicht vollständig untersucht.

- Die Möglichkeiten der Timing-Bewertung von komplexen Vernetzungsarchitekturen, die vom OEM zu entwickeln sind, wurden bis heute nicht im Detail untersucht und aufgezeigt.

- Die detaillierte Timing-Bewertung von Gateway-Steuergeräten mit analytischen Verfahren wurde bislang nicht angegangen.

Diese offenen Punkte werden in der vorliegenden Arbeit adressiert und Lösungen hierfür erarbeitet. Der Hauptfokus dieser Dissertation liegt in der Entwicklung einer Methodik für die durchgängige Bewertung von Timing-Fragestellungen innerhalb des E/E-Entwicklungsprozesses von Vernetzungsarchitekturen und Gateway-Systemen im Kraftfahrzeug. In Abbildung 1.5 sind die wichtigsten Schritte anhand des V-Modells dargestellt. In der Entwurfsphase können erste Timing-Abschätzungen durchgeführt werden, wobei die daraus gewonnen Erkenntnisse dann als Grundlage für die Spezifikation und zur Auslegung der Systeme dienen können. Bei der anschließenden Absicherung kann das spezifizierte Timing-Verhalten verifiziert werden.

Das Ziel der Arbeit ist es, den durchgängigen Einsatz von Timing-Bewertungsverfahren im existierenden Entwicklungsprozess zu ermöglichen und die notwendige Methodik aufzuzeigen. Hierfür werden folgende Punkte in dieser Dissertation adressiert:

1. Darstellung des Stands der Technik im Bereich der Bewertung von automotive Vernetzungsarchitekturen. Einordnung von Timing-Bewertungsverfahren in den existierenden E/E-Entwicklungsprozess für die Beantwortung von Fragestellungen im Bereich der Vernetzungsarchitekturen und Gateway-Systeme.

2. Identifikation der notwendigen Daten, die für eine aussagekräftige Timing-Bewertung notwendig sind. In diesem Zusammenhang wird ein Verfahren zur Extraktion von Timing-Informationen aus Loggingdaten entwickelt.

3. Untersuchung der Wiederverwendung von Timing-Informationen aus existierenden Fahrzeugen für die Modellverfeinerung in der Entwurfsphase von E/E-Architekturen.

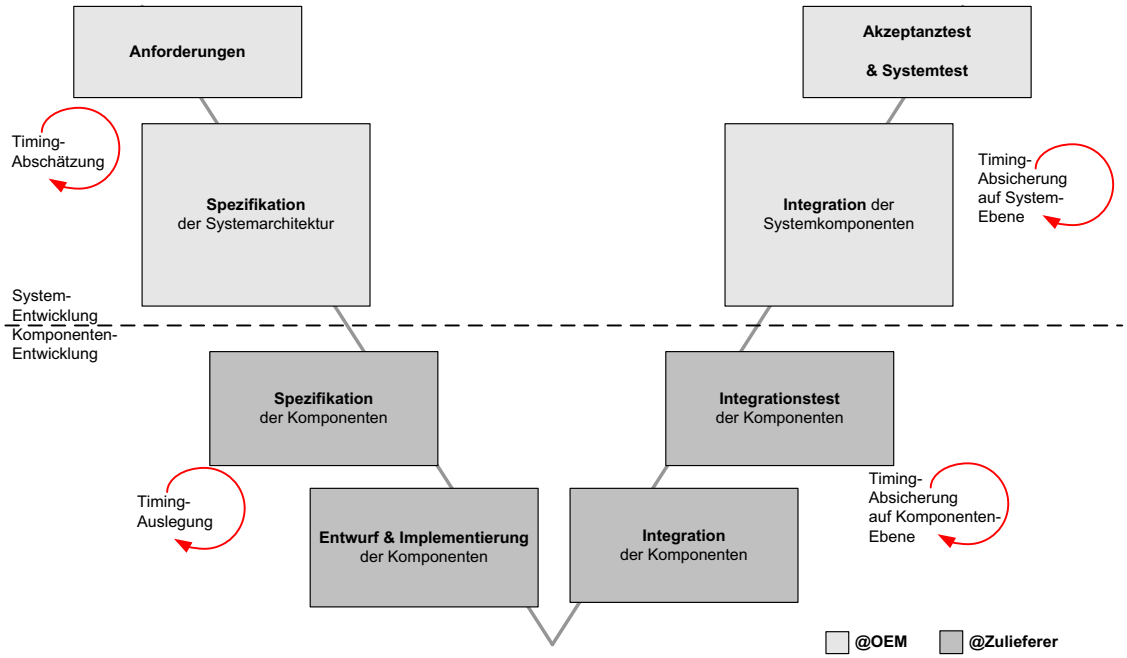


Abb. 1.5.: Timing-Bewertungen in den einzelnen Phasen des Entwicklungsprozesses von Elektrik-/Elektronik-Architekturen

4. Erweiterung und Verfeinerung der Timing-Bewertungsverfahren für automotive Fragestellungen, z.B. Erarbeitung von detaillierten Modellierungsregeln und Bewertungsmöglichkeiten für Vernetzungsarchitekturen und Gateway-Steuergeräten.

5. Aufzeigen einer durchgängigen Bewertungsmethodik für die Timing-Fragestellungen im E/E-Entwicklungsprozess.

6. Entwicklung eines Konzeptes zur Ableitung von Routing-Testpattern auf der Basis von Timing-Analysen, mit dem Ziel eine gezielte Berücksichtigung des Timing-Verhaltens von Steuergeräten mit Gateway-Anteilen am Komponentenprüfstand zu ermöglichen.

Die entwickelten Konzepte und Methoden dieser Arbeit werden anschließend auf deren Tauglichkeit überprüft und anhand von repräsentativen Beispielen aus der Praxis validiert. Die Evaluierung wird mit der entstandenen prototypischen Werkzeugkette durchgeführt, die es ermöglicht eine durchgängige Bewertung des Timing-Verhaltens von Vernetzungsarchitekturen durchzuführen. Bei der Daimler AG wurden in den letzten Jahren die formale Beschreibung sowie der werkzeuggestützte Entwurf von E/E-Architekturen

eingeführt und stetig weiterentwickelt (siehe [103]). Die in dieser Arbeit entwickelte Methodik soll zukünftig Anwendung in diesem Prozess finden und in die Serie überführt werden.

1.4. Gliederung der Arbeit

Die Arbeit gliedert sich wie folgt: In *diesem Kapitel* wurden die Motivation und die Zielsetzung für die Einführung von Timing-Analyse-Verfahren in den Entwurfsprozess von E/E-Architekturen im Kraftfahrzeug vorgestellt. In *Kapitel 2* wird die Thematik der Timing-Bewertung eingeführt sowie die in dieser Arbeit verwendete Terminologie vorgestellt. Darauf aufbauend erfolgt die Herleitung der Timing-Anforderungen, die sich in diesem Bereich stellen. *Kapitel 3* gibt einen Überblick über den aktuellen Stand von E/E-Architekturen. Weiterhin werden Kenngrößen eingeführt, welche für eine Timing-Bewertung von Bedeutung sind. In *Kapitel 4* werden Verfahren vorgestellt, die eine Timing-Bewertung ermöglichen. Im Anschluss an die Vorstellung der einzelnen Ansätze erfolgt eine Einordnung der Verfahren sowie eine Abgrenzung dieser Dissertation von anderen aktuellen Arbeiten in diesem Themengebiet. Davon ausgehend wird in dieser Arbeit eine Methodik entwickelt, welche eine durchgängige Integration der Timing-Bewertungsverfahren in die existierende Entwicklungskette ermöglicht. Es werden dabei Timing-Fragestellung berücksichtigt, die sich sowohl im Bereich Vernetzung als auch im Umfeld der Gateway-Entwicklung ergeben. In *Kapitel 5* wird ein Verfahren vorgestellt, welches die Möglichkeit bietet, Timing-Informationen aus existierenden Vernetzungsarchitekturen zu extrahieren. Weiterhin werden in *Kapitel 6* Modellierungsregeln definiert, die eine präzise Abbildung des Timing-Verhaltens der Systeme ermöglichen. Basierend auf den Erkenntnissen der vorangegangenen Arbeiten, wird in *Kapitel 7* die Methodik für eine durchgängige Timing-Bewertung anhand einer prototypischen Werkzeugkette aufgezeigt und deren Integration in den existierenden Entwicklungsprozess von E/E-Architekturen beschrieben. Weiterhin wird in *Kapitel 7* ein Konzept für die Generierung von Routing-Testpattern vorgestellt. Die Testpattern ermöglichen eine Absicherung von Steuergeräten mit Gateway-Funktionalität am Komponenten-Prüfstand während der Integrationsphase. In *Kapitel 8* erfolgt die Validierung der entwickelten Methodik anhand von Beispielen aus der Praxis. *Kapitel 9* fasst die erzielten Ergebnisse der Arbeit zusammen und gibt einen Ausblick auf mögliche Erweiterungen.

2. Grundlagen der Arbeit

Im vorangegangenen Kapitel wurde das Thema Timing-Bewertung von E/E-Architekturen im Kraftfahrzeug motiviert. Um bei den weiteren Ausführungen auf einer einheitlichen Terminologie aufzusetzen, werden im Folgenden die in dieser Arbeit verwendeten Begriffe eingeführt.

2.1. Eingebettete verteilte Echzeitsysteme

Eingebettete verteilte *Echtzeitsysteme (engl. Real-Time Systems)* unterscheiden sich hinsichtlich der Zeitanforderungen grundlegend von den allgemeinen Computersystemen (z.B. Büro-Computer), sogenannten *Nicht-Echtzeitsystemen*. Bei den Nicht-Echtzeitsystemen kommt es ausschließlich auf die Korrektheit der Datenverarbeitung und der Ergebnisse an [143]. Im Gegensatz dazu spielt bei den Echtzeitsystemen neben der Korrektheit der Ergebnisse auch die Einhaltung der Zeitanforderungen eine zentrale Rolle. Zeitbedingungen, deren Nichteinhalten zu einer *Katastrophe* führen können, heißen harte Zeitbedingungen [65]. Alle anderen Zeitbedingungen heißen weiche Zeitbedingungen. Weitere wichtige Anforderungen an Echtzeitsysteme sind: *Rechtzeitigkeit*, *Gleichzeitigkeit*, *Verfügbarkeit* (siehe [143]) sowie *Vorhersagbarkeit* und *Zuverlässigkeit* [102]:

- Mit Rechtzeitigkeit ist die Anforderung gemeint, die garantiert, dass eine Ausführung auf einer CPU bzw. eine Übertragung über einen Bus innerhalb der definierten Zeitschranke abgeschlossen wird.
- Ein Echtzeitsystem muss in der Lage sein, mehrere Ereignisse gleichzeitig verarbeiten können, damit die Rechtzeitigkeit für mehrere Aktionen gleichzeitig gewährleistet ist. Zum Beispiel durch (quasi-) parallele Ausführung auf einem Prozessor oder echte parallele Ausführung auf einem Mehrprozessorsystem [114].
- Innerhalb eines spezifizierten Zeitbereichs muss ein Echtzeitsystem immer uneingeschränkt zur Verfügung stehen, d.h. ohne Unterbrechung betriebsbereit sein.

- Die Anforderung der Vorhersagbarkeit ist die Forderung, dass alle von einem System zu verarbeitenden Ereignisse und Funktionen vor der Ausführung bekannt und deterministisch sein müssen [102].
- Die Zuverlässigkeit ist die Fähigkeit eines Systems, während einer vorgegebenen Zeitdauer bei zulässigen Betriebsbedingungen die spezifizierte Funktion zu erbringen [102].

2.2. Begriffsdefinitionen

Die folgenden Begriffsdefinitionen basieren zu großen Teilen auf dem Standardwerk von G. Buttazzo *Hard Real-Time Computing Systems* [21]. Generell gilt: Zeitpunkte werden immer mit *Kleinbuchstaben* annotiert, Zeiträume mit *Großbuchstaben*.

Definition 2.1 (Ausführungszeit/Übertragungszeit) *Die Ausführungszeit bzw. Übertragungszeit C_i ist die Zeit, welche benötigt wird, um einen Task w_i ohne Unterbrechung auszuführen oder eine Botschaft m_i zu übertragen.*

Definition 2.2 (Aktivierungszeitpunkt) *Der Aktivierungszeitpunkt* (engl. Release) *a_i ist der Zeitpunkt, an dem ein Task w_i bereit ist zur Ausführung bzw. eine Botschaft m_i zur Übertragung ansteht.*

Definition 2.3 (Startzeitpunkt) *Der Startzeitpunkt b_i ist der Zeitpunkt, an dem die Ausführung eines Tasks w_i startet bzw. die Übertragung einer Botschaft m_i beginnt.*

Definition 2.4 (Ende der Ausführung) *Das Ende der Ausführung* (engl. Termination) *eines Tasks w_i bzw. der Abschluss der Übertragung einer Botschaft m_i wird mit c_i angegeben.*

Definition 2.5 (Deadline) *Die Deadline d_i ist der Zeitpunkt, an dem eine Abarbeitung eines Tasks w_i oder die Übertragung einer Botschaft m_i abgeschlossen sein muss.*

Definition 2.6 (Antwortzeit) *Die Antwortzeit* (engl. Response Time) *R_i gibt die Zeitdauer, an die ein Task w_i bzw. eine Botschaft m_i tatsächlich für die Ausführung benötigt bzw. wieviel Zeit bei deren Übertragung ab dem Zeitpunkt der Aktivierung a_i vergangen ist.*

Abbildung 2.1 zeigt ein Beispiel für ein Prioritätsscheduling. Ein Task w_i, wird zum Zeitpunkt a_i aktiviert. Die Ausführung startet erst bei b_i, da noch höherpriore Tasks $w_j \in hp(w_i)$ aktiv sind. Nach der Ausführungszeit C_i wird der Task w_i zum Zeitpunkt c_i innerhalb der Deadline d_i beendet. Die Antwortzeit der Task ergibt sich aus der Ausführungszeit C_i und der Zeit, während der Task w_i seit dem Aktivierungszeitpunkt durch höherpriore Tasks verzögert wird. Prinzipiell kann der Task w_i auch während ihrer Ausführung durch höherpriore Tasks $w_j \in hp(w_i)$ unterbrochen und dadurch weiter verzögert werden. Dies ist in Abbildung 2.1 nicht mit aufgezeigt, um eine einfache Darstellung zu gewährleisten.

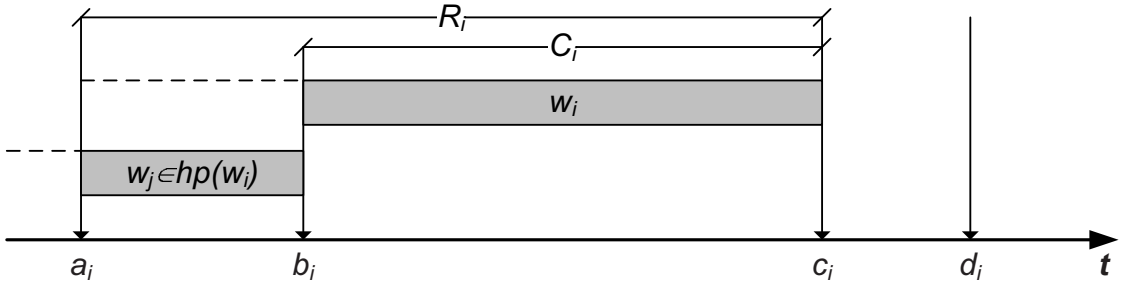


Abb. 2.1.: Timing-Eigenschaften eines Tasks w_i, die durch höherpriore Tasks $w_j \in hp(w_i)$ verzögert wird

Definition 2.7 (Idle-Zeit) *Die Idle-Zeit T_{idle} definiert die Zeitdauer, während der kein Tasks zur Ausführung oder Botschaften zur Übertragung anstehen. Die CPU bzw. das Übertragungsmedium ist in dieser Zeit nicht belegt.*

Definition 2.8 (Verspätung) *Die Verspätung* (engl. Lateness) *$T_{late,i}$ ist die Zeitdauer, die ein Task w_i oder eine Botschaft m_i in Bezug auf die Deadline d_i zu spät kommt. Wird die Ausführung bzw. Übertragung innerhalb der Deadline beendet, so ist die Verspätung $T_{late,i}$ negativ.*

$$T_{late,i} = c_i - d_i \qquad [2.1]$$

Definition 2.9 (Überschreitungszeit) *Die Überschreitungszeit* (Exeeding Time) $T_{exe,i}$ *definiert die Zeit, welche ein Task* w_i *oder eine Botschaft* m_i *nach Überschreitung der Deadline noch aktiv ist.*

$$T_{exe,i} = max(0, T_{late,i}) \quad [2.2]$$

Definition 2.10 (Schlupf) *Der Schlupf* (engl. slack) *definiert die Zeit, die ein Task* w_i *oder eine Botschaft* m_i *nach seiner Aktivierung maximal verzögert werden darf, damit dieser noch vor der Deadline vollständig ausgeführt werden kann.*

$$T_{slack,i} = d_i - a_i - C_i \quad [2.3]$$

In Abbildung 2.2 sind die weiteren Timing-Eigenschaften am Beispiel eines Tasks w_i dargestellt.

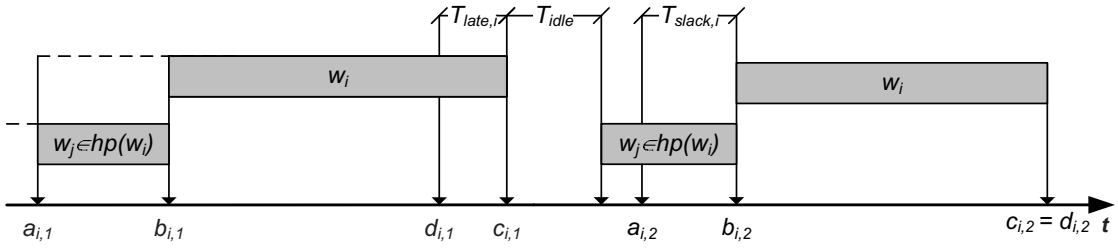


Abb. 2.2.: Weitere Timing-Eigenschaften eines Tasks w_i

Definition 2.11 (Ereignis, Ereignisstrom) *Ein Ereignis e ist ein Vorkommnis (z.B. ein Statuswechsel), das sich zu einem bestimmten Zeitpunkt t ereignet [65]. Ein Ereignisstrom* $E = \{e_1, e_2, e_3, ...\}$ *beschreibt die Abfolge von Ereignissen. Ein Ereignisstrom ist definiert über vier charakteristische Funktionen [96]:*

$$\eta^+ : \mathbb{R}^+ \mapsto \mathbb{N}^+, \quad [2.4]$$

$$\eta^- : \mathbb{R}^+ \mapsto \mathbb{N}^+, \quad [2.5]$$

$$\delta^+ : \mathbb{N}^+ \backslash \{0,1\} \mapsto \mathbb{R}^+ \text{ and} \quad [2.6]$$

$$\delta^- : \mathbb{N}^+ \backslash \{0,1\} \mapsto \mathbb{R}^+ \text{ and} \quad [2.7]$$

Die Funktion $\eta^+(\Delta t)$ beschreibt die maximale Anzahl an Ereignissen innerhalb eines Zeitintervalls Δt. Über die Funktion $\eta^-(\Delta t)$ wird die minimale Anzahl an Ereignissen in einem Zeitintervall Δt beschrieben. Die Funktionen $\delta^+(n)$ und $\delta^-(n)$ geben den maximalen bzw. minimalen Abstand zwischen n aufeinanderfolgenden Ereignissen an.

Definition 2.12 (Minimaler Auftritts-/Sendeabstand) *Der minimale Auftritts-/Sendeabstand* (engl. Minimum Distance) *$T_{min,i}$ gibt an, welcher Mindestabstand zwischen zwei aufeinanderfolgenden Tasks w_i oder Botschaften m_i eingehalten werden muss.*

Definition 2.13 (Jitter) *Der Jitter beschreibt die Abweichung eines Tasks w_i oder einer Botschaft m_i von dem definierten Aktivierungs-/Sendezeitpunkt (Periode). Es wird zwischen Eingangsjitter J_{in} und Ausgangsjitter J_{out} unterschieden.*

In Abbildung 2.3 sind exemplarisch ein Ereignisstrom mit Jitter J und Mindestsendeabstand T_{min} sowie die resultierende Funktion $\eta(\Delta t)$ aufgezeigt.

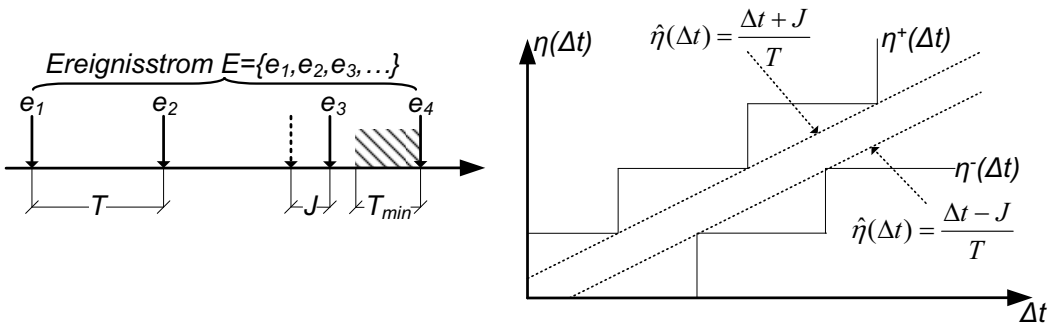


Abb. 2.3.: Exemplarischer Ereignisstrom und die resultierende Funktion $\eta(\Delta t)$

Definition 2.14 (Absoluter Jitter) *Der absolute Jitter J_a gibt die maximale Abweichung an, welche über die gesamte Zeit zwischen den Instanzen eines Tasks oder einer Botschaft auftritt. Dabei gilt für den absoluten Eingangsjitter:*

$$J_{a\_in,i} = \max_k (b_{i,k} - a_{i,k}) - \min_k (b_{i,k} - a_{i,k}) \qquad [2.8]$$

Für den relativen Ausgangsjitter gilt:

$$J_{a\_out,i} = \max_k (c_{i,k} - a_{i,k}) - \min_k (c_{i,k} - a_{i,k}) \qquad [2.9]$$

Definition 2.15 (Relativer Jitter) *Der relative Jitter J_r gibt die Abweichung von zwei aufeinanderfolgenden Task- oder Botschaftsinstanzen an. Für den relativen Eingangsjitter gilt:*

$$J_{r\_in,i} = \max_k \left| (b_{i,k} - a_{i,k}) - (b_{i,k-1} - a_{i,k-1}) \right| \quad [2.10]$$

Für den relativen Ausgangsjitter gilt:

$$J_{r\_out,i} = \max_k \left| (c_{i,k} - a_{i,k}) - (c_{i,k-1} - a_{i,k-1}) \right| \quad [2.11]$$

Definition 2.16 (Latenzzeit) *Latenzzeit L, in unterschiedlichen Zusammenhängen auch Reaktionszeit, Verweilzeit oder Verzögerungszeit genannt, ist der Zeitraum zwischen einer Aktion (bzw. einem Ereignis) und dem Eintreten einer verzögerten Reaktion. Bei einer Latenzzeit ist die Aktion verborgen und wird erst durch die Reaktion deutlich [142].*

In Abbildung 2.4 ist ein gemappter Task w dargestellt sowie ein Eingangsereignisstrom E_{in} und der resultierende Ausgangsereignisstrom E_{out} inklusive der Jitter.

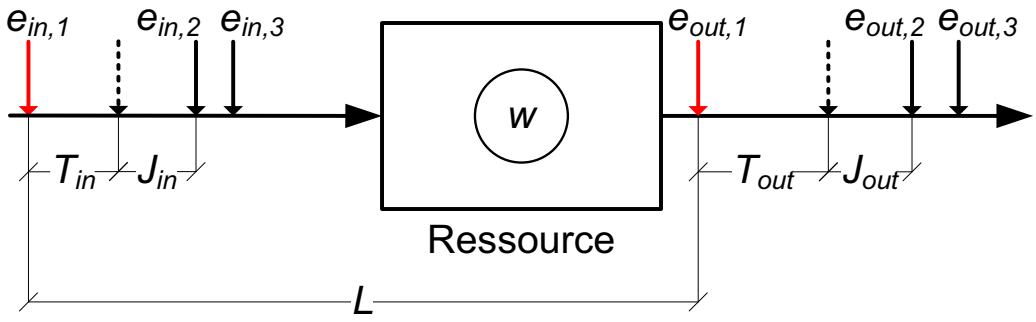


Abb. 2.4.: Beispiel für ein gemappter Task sowie ein Eingangsereignisstrom E_{in} und der resultierende Ausgangsereignisstrom E_{out}

Definition 2.17 (Hyperperiode oder Makroperiode) *Die Hyperperiode H oder auch Makroperiode genannt, gibt die Periode an, bei der sich der Plan für das Scheduling wiederholt. Sie ist das kleinste gemeinsame Vielfache aller Perioden der im System vertretenen Jobs [117].*

Definition 2.18 (Offset) *Der Offset T_{off} beschreibt die Zeit, die nach dem Systemstart gewartet wird, bis ein Task w_i ausgeführt bzw. eine Botschaft m_i verschickt wird.*

Definition 2.19 (Auslastung) *Die Auslastung U* engl. Utilization *beschreibt die Belegung einer Ressource. Beispielsweise kann für n unabhängige periodische Tasks w_i, für die gilt, dass deren Periode T_i gleich der Deadline d_i ist, wie folgt berechnet werden:*

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \qquad [2.12]$$

In der vorliegenden Arbeit wird oft der Begriff Architektur verwendet. Dessen Handhabung ist in der Literatur sehr unterschiedlich. Häufig findet sich eine starke Analogie zwischen den Begriffen Architektur und Topologie [106]. Um für die Arbeit eine einheitliche Terminologie festzulegen, wird für einige wichtige Begriffe eine eindeutige Definition gegeben, diese werden in dieser Arbeit wie folgt verwendet.

Definition 2.20 (Architektur) *Eine Architektur ist die grundlegende Organisation eines Systems, verkörpert durch deren Komponenten, ihre Beziehungen zueinander und zur Umgebung sowie den Prinzipien, die das Design und die Evolution leiten [24].*

Bei einer E/E-Architektur im Kraftfahrzeug werden die Komponenten durch die einzelnen Busse und Steuergeräte repräsentiert.

Definition 2.21 (Topologie) *Eine Topologie (griech.: Topos = Ort) ist die Beschreibung jeglicher Art von Anordnung von Elementen und deren Verbindungen [106].*

Die Topologie umfasst folglich einen Teil der Architektur. Die Architektur beschreibt nicht nur die Anordnung, sondern zusätzlich auch die verwendeten Elemente und Technologien eines Systems sowie deren Schnittstellen.

Definition 2.22 (Vernetzungsarchitektur) *Die Vernetzungsarchitektur enthält die Netzwerktopologie, d.h. die einzelnen Busse sowie die Verbindungen/Schnittstellen der an die Busse angekoppelten Steuergeräte und enthält die verwendeten Kommunikationssysteme (Technologie).*

Weiterhin werden die Begriffe Hardwarearchitektur und Softwarearchitektur für die detaillierte Beschreibung von Steuergeräten verwendet. Die Hardwarearchitektur umfasst die technische Realisierung eines Steuergerätes, z.B. CPU, Speicher, Schnittstellen. Die Softwarearchitektur beschreibt die Applikations- sowie die Basissoftware eines Steuergerätes.

2.3. Anforderungen an den Timing-Bewertungsprozess

Zu jedem Zeitpunkt im E/E-Entwicklungsprozess muss eine ausreichende Menge an Timing-Informationen vorliegen, damit eine aussagekräftige Bewertung durchgeführt werden kann. Zu den Timing-Informationen zählen:

- Ausführungszeiten der Tasks und Übertragungszeiten der Botschaften
- Konfiguration des Betriebssystems und der Kommunikationssysteme
- Beschreibung des Timing-Verhaltens der Eingangsereignisse, z.B. der anwenderabhängigen Interaktion, um die entsprechenden Verhaltensmodelle abzuleiten
- Beschreibung des Sendeverhaltens der Botschaften.
- Weitere Timing-Informationen wie z.B. Jitter, Drift und Offsets

Das Ziel ist es, sowohl auf Komponentenebene (Steuergeräte und Busse) als auch auf Systemebene, eine fundierte Aussage über das Timing-Verhalten treffen zu können. Die hierfür notwendigen Timing-Informationen können bisher nur teilweise direkt aus den Spezifikationen entnommen werden. Einige Informationen sind implizit vorhanden und lassen sich über Regeln ableiten, z.B. die Übertragungszeit C_i einer Botschaft m_i kann anhand deren Länge p_i und der Übertragungsgeschwindigkeit V des Kommunikationssystems berechnet werden. Für die Beschreibung der noch fehlenden Informationen besteht meist ein direkter Zusammenhang mit der Interaktion des Fahrers und der Passagiere mit den E/E-basierten Funktionen. Dieses spontane Timing-Verhalten kann über die Beobachtung (Messung) der E/E-Systeme bestimmt werden.

Abbildung 2.5 zeigt diese Schwierigkeit anhand eines Steuergeräts. Aus der Systembeschreibung und über Auswertungen der Code-Laufzeiten können viele Timing-Informationen ermittelt werden. Insbesondere jedoch die Ereignisse, welche von außen ausgelöst werden und direkten Einfluss auf die Ausführungszeiten haben, lassen sich nur schwer spezifizieren.

In gleicher Weise wie bei Steuergeräten verhält es sich bei der Timing-Bewertung von Kommunikationssystemen (siehe Abbildung 2.6). Ohne die Kenntnisse über das dynamische Timing-Verhalten der Busteilnehmer (Steuergeräte) ist eine sinnvolle Bewertung der Buskommunikation nicht möglich. Diese Dynamik, welche in der Versendung von

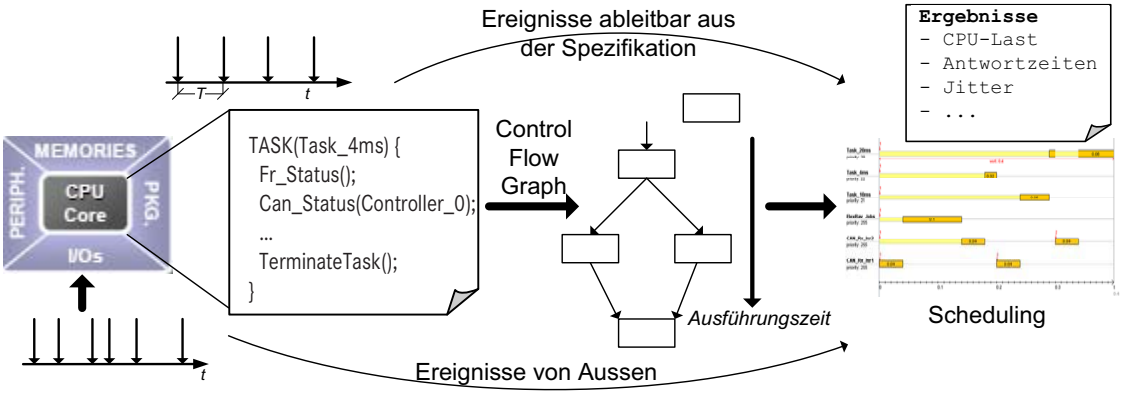


Abb. 2.5.: Einflüsse und Informationen für eine Timing-Bewertung von Steuergeräten

spontanen Botschaften resultiert, hängt in den meisten Fällen direkt mit Interaktionen der Insassen des Fahrzeugs zusammen.

Um das Problem der fehlenden Timing-Informationen zu lösen, wird in der vorliegenden Arbeit eine Methodik vorgestellt, die eine durchgängige und aussagekräftige Timing-Bewertung innerhalb des Entwicklungsprozesses von E/E-Architekturen ermöglicht und Wege aufzeigt, die dafür notwendigen Timing-Informationen bereitzustellen. Ferner können die ermittelten Informationen schrittweise als zusätzliche Attribute (Kriterien oder Anforderungen) in die Spezifikationen der E/E-Systeme einfließen. Auf deren Basis ist dann zukünftig eine genauere Auslegung der Systeme möglich. Ferner sind die ermittelten Timing-Informationen für eine Modellverfeinerung in der Entwurfsphase von E/E-Architekturen verwendbar.

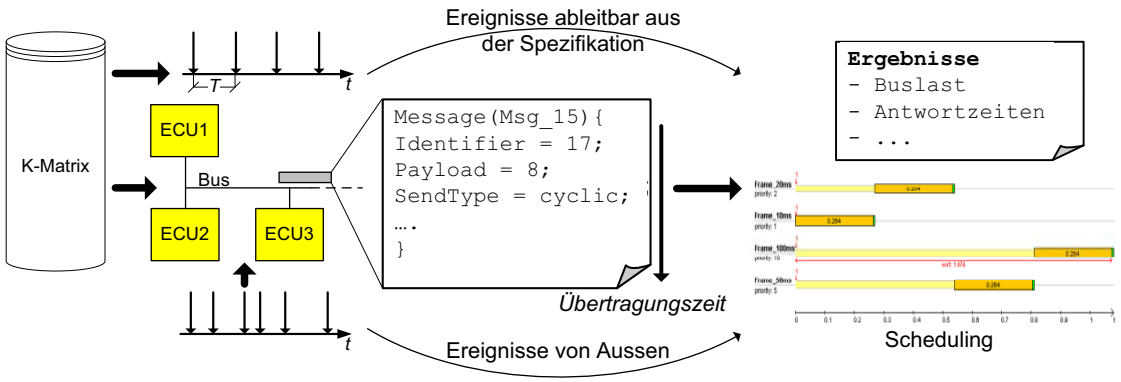


Abb. 2.6.: Einflüsse und Informationen für eine Bewertung von Kommunikationssystemen

3. E/E-Architekturen und Entwicklungsprozesse

In diesem Kapitel werden die Grundlagen der E/E-Architekturen im Kraftfahrzeug eingeführt, welche im Kontext der Timing-Bewertungen relevant sind. Weiterhin erfolgt die detaillierte Beschreibung der einzelnen Komponenten. Im Anschluss daran wird auf den aktuellen E/E-Entwicklungsprozess eingegangen und es werden die detaillierten Anforderungen für die Timing-Bewertungen abgeleitet. Abschließend erfolgt die Ableitung der hierfür wesentlichen Kenngrößen.

3.1. E/E-Architekturen im Kraftfahrzeug

Die Begriffsdefinitionen für Architektur, Topologie, etc. wurden in Abschnitt 2.2 gegeben. Eine E/E-Architektur im Kraftfahrzeug umfasst das Funktionsnetzwerk, das Netzwerk der Hardwarekomponenten, den Leitungssatz und die physikalische Topologie. Die Unterteilung in die verschiedenen Ebenen ist in Abbildung 3.1 dargestellt. *Top-Down* betrachtet werden die Anforderungen *(Requirements)* formuliert. Diese Anforderungen sind durch ein Netz kooperierender Funktionen realisiert *(Functional Network)*. Die Software-Anteile der Funktionen sind auf den einzelnen Steuergeräten integriert. Die Verbindung der Steuergeräte ist über verschiedene Kommunikationssysteme realisiert. Weiterhin sind Sensoren und Aktoren an die Steuergeräte gekoppelt. Dieser Verbund bildet das Hardwarekomponentennetzwerk *(Hardware Component Network)*. Der sogenannte *Schematic Layer* beinhaltet die Anschlüsse und Leitungen zwischen den einzelnen Komponenten sowie die Energieversorgung. Der Leitungssatz *(Wiring Harness)* ist in der nächsten Ebene zu finden. Die physikalische Topologie *(Physical Topology)* umfasst die Einbauorte und Kabeldurchführungen.

Die oberen drei Ebenen sind bei der Timing-Bewertung relevant. Dazu zählen die Hardware- und Software-Architektur der Steuergeräte sowie die Kommunikationssysteme. Die Topologieaspekte, wie Signallaufzeiten auf physikalischer Ebene und die hierfür notwendige Berücksichtigung der Leitungslängen, spielen bei der Betrachtung des

Timing-Verhaltens auf logischer Ebene keine Rolle und sind nicht Gegenstand dieser Arbeit. Die für die Timing-Bewertung relevanten Eigenschaften der Komponenten werden im Folgenden näher ausgeführt.

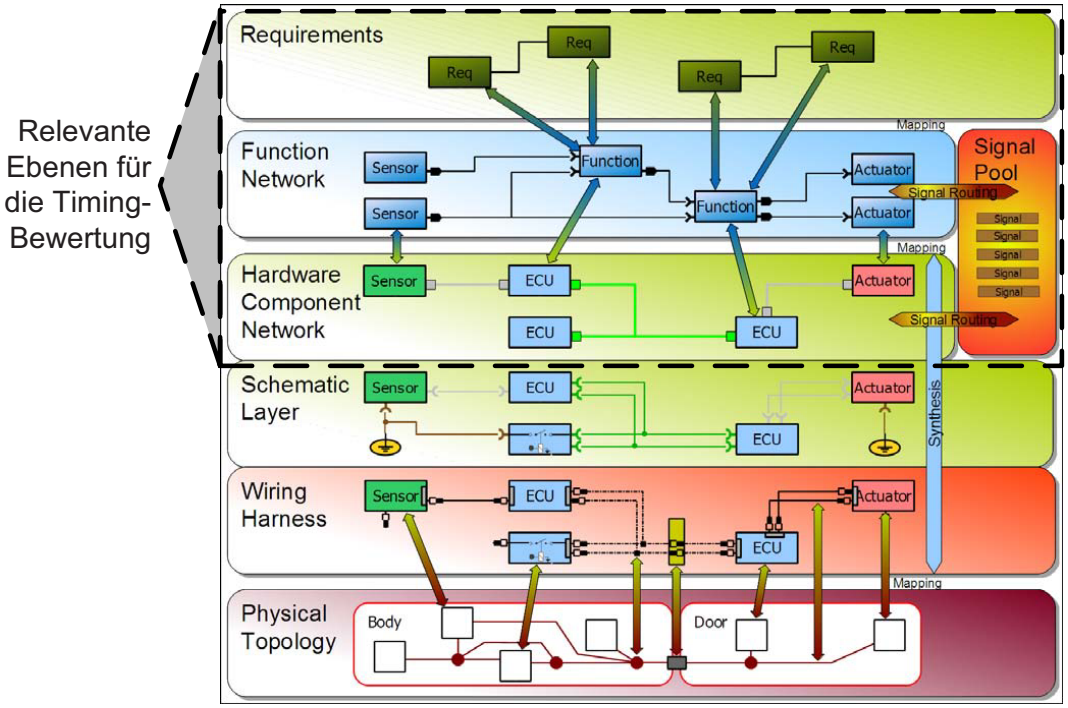


Abb. 3.1.: Die einzelnen Ebenen einer E/E-Architektur und die für die Timing-Bewertung relevanten Abschnitte [131]

3.1.1. Hardwarearchitektur von Steuergeräten

Die Hardware-Architektur von Steuergeräten umfasst die Steckerpins, das Gehäuse, die Platine sowie die elektronischen Baugruppen wie Spannungsversorgung, Leistungstreiber für die Ein- und Ausgänge und die Transceiver-Bausteine sowie die Recheneinheit, typischerweise ein Mikrocontroller. Ein Beispiel für einen heute eingesetzten 32-Bit Mikrocontroller ist in Abbildung 3.2 dargestellt. Der Baustein beinhaltet neben dem Rechenkern (hier: *V850E Core*) den flüchtigen und festen Speicher (*RAM* und *Flash*). Weiterhin werden diverse Schnittstellen zur Ankopplung von Sensorik und Aktorik sowie zur Anbindung verschiedener Kommunikationssysteme bereitgestellt.

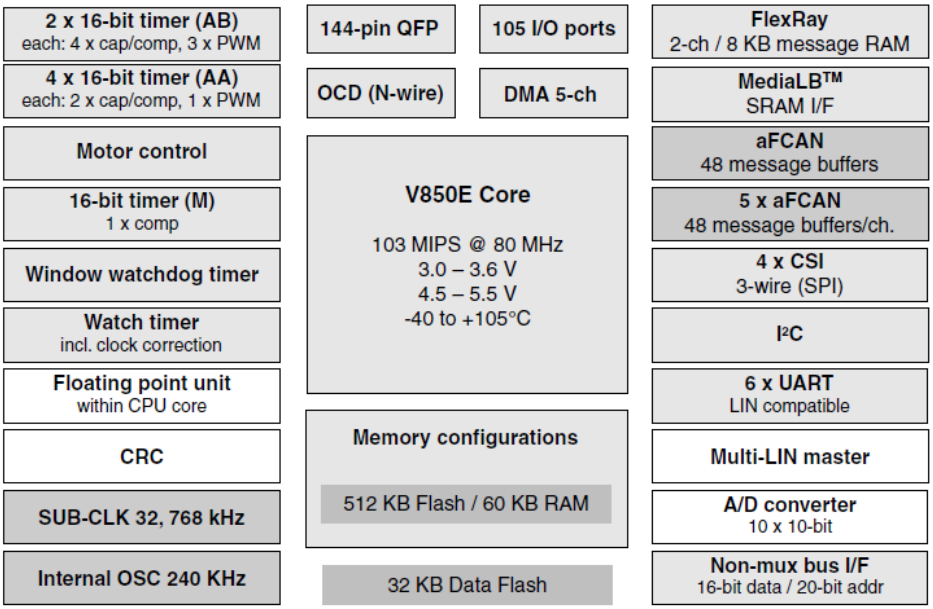


Abb. 3.2.: Blockdiagramm eines aktuellen Microkontrollers *V850/CAG4-M* von NEC [84]

Für das Timing-Verhalten eines Steuergerätes bzw. der Software ist die Auswahl eines für die jeweiligen Aufgaben geeigneten Mikrocontrollers von zentraler Bedeutung. Für ein Gateway-Steuergerät beispielsweise ist eine homogene Ankoppelung der Peripherie an die CPU sehr wichtig, d.h. die Peripherie, z.B. die Buscontroller sollten nicht eine vielfach kleinere Frequenz als die CPU haben, damit möglichst wenig *Wait States* von der CPU bei der Datenübertragung zu dem Controller eingefügt werden müssen.

Ein weiteres wichtiges Thema im Zusammenhang mit der Timing-Bewertung werden zukünftig die Multicore-Architekturen von Mikrocontrollern sein, welche in naher Zukunft vermehrt in Steuergeräten eingesetzt werden. Das Verhalten und die Bewertung solcher Multicore-Systeme ist nicht Bestandteil dieser Arbeit. Die weiteren Ausführungen beziehen sich immer auf Single-Core Architekturen. Viele der entwickelten Konzepte sind jedoch auf Multicore-Architekturen übertragbar.

3.1.2. Softwarearchitektur von Steuergeräten

Die Softwarearchitektur eines Steuergerätes umfasst neben der Laufzeitumgebung auch die Applikationssoftware. Die Laufzeitumgebung beinhaltet das Betriebssystem, die Treiber für die Ansteuerung der Peripherie sowie Basisdienste. In der Applikationssoftware sind die kundenerlebbaren Funktionen gekapselt.

Da über den Einsatz einer OEM-eigenen Laufzeitumgebung kein signifikanter Wettbewerbsvorteil erzielt werden kann, wird seit vielen Jahren an herstellerübergreifenden Standards gearbeitet. Die wichtigsten Gremien sind *OSEK/VDX (Offene Systeme für die Elektronik im Kraftfahrzeug/Vehicle Distributed Executive)*, die *Herstellerinitiative Software (HIS)* und *AUTOSAR (Automotitve Open Systems Architecture)* [87],[17]. Mit der Arbeit an OSEK/VDX wurde im Jahre 1995 begonnen und es ist aktuell in fast allen Steuergeräten aktueller Fahrzeuge zu finden. Mit dem Zusammenschluss vieler OEMs, Zulieferer und Toolhersteller im AUTOSAR-Konsortium erfolgte ein weiterer Schritt hin zu einer umfassenden Standardisierung. In AUTOSAR sind große Teile der Konzepte von OSEK eingeflossen.

OSEK/VDX

Das OSEK/VDX-System besteht aus mehreren Modulen. In Abbildung 3.3 sind die einzelnen Module dargestellt.

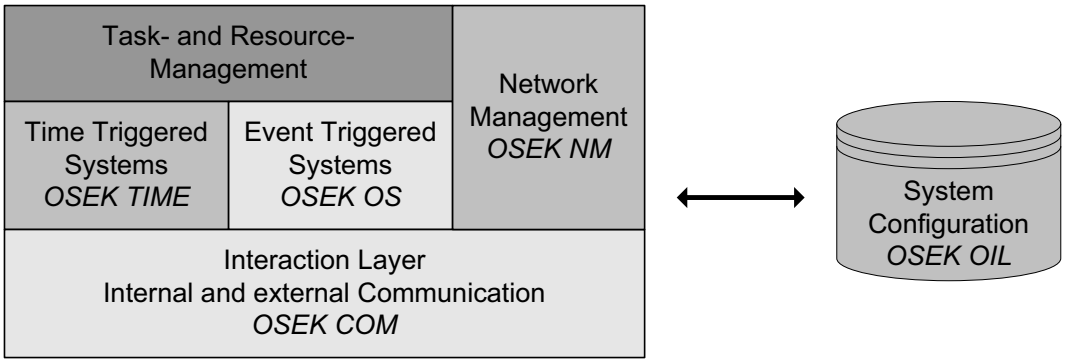


Abb. 3.3.: Grundkomponenten des OSEK/VDX-Systems [144]

1. *OSEK-OS (Operating System)* ist ein ereignisgesteuertes Echtzeit-Multitasking Betriebssystem, welches die Möglichkeit zur Task-Synchronisation und Ressourcenverwaltung bietet [53].

2. *OSEK-TIME* ist die zeitgesteuerte Variante.

3. *OSEK-COM (Communication)* beschreibt die Interaktionsschicht zum internen Datenaustausch zwischen den Tasks eines Steuergerätes und zum externen Datenaustausch mit anderen Steuergeräten über die entsprechenden Schnittstellen.

4. Über das *OSEK-NM (Networkmanagement)* wird die Überwachung und Verwaltung der Kommunikation mit anderen Steuergeräten realisiert, die über ein Kommunikationssystem stattfindet.

5. *OSEK-OIL (OSEK Implementation Language)* ist eine Beschreibungssprache zur Konfiguration der aufgeführten Module.

Die stärksten Einflüsse auf das Timing-Verhalten eines Steuergerätes hat das Betriebssystem und dessen Konfiguration. Aus diesem Grund werden im Folgenden die wichtigsten Eigenschaften des OSEK-OS diskutiert. Das OSEK-OS ist eingeteilt in vier Konformitätsklassen. Diese ermöglichen es je nach Anforderungen einen bestimmten Funktionsumfang des OSEK-OS für ein Steuergerät zu verwenden. Als Ausführungseinheiten für den Programmcode dienen die *Tasks*. Bei OSEK-OS wird zwischen *Basic Tasks* und *Extended Tasks* unterschieden. Basic-Tasks hängen nicht von äußeren Ereignissen ab, während Extended-Tasks auf Ereignisse von außen warten. Abbildung 3.4 zeigt das Zustandsmodell der OSEK-OS-Tasks.

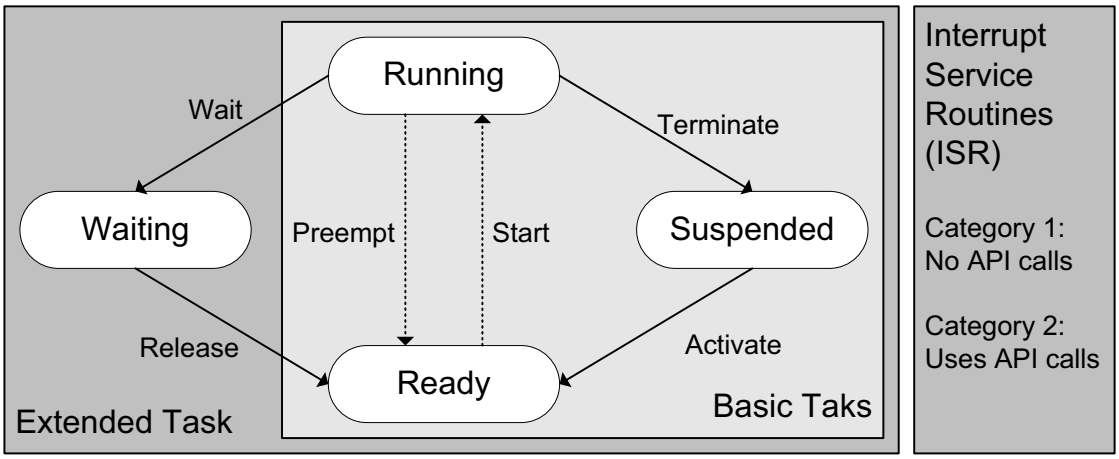


Abb. 3.4.: Betriebssystemezustände für *Basic Tasks* und *Extended Tasks* des OSEK-OS[88]

Nach der Aktivierung einer Task *(Suspended → Ready)* ist diese bereit zur Ausführung. Eine Task kann ausgeführt *(Ready → Running)* werden, falls: 1) Die CPU nicht belegt ist, 2) ein preemptiver Task mit niederer Priorität gerade ausgeführt wird und 3) keine Interrupts ausgelöst wurden.

Beim OSEK-OS wird über den Scheduler die CPU-Zeit den Tasks über statische Prioritäten zugewiesen [61]. Im Gegensatz dazu arbeitet das OSEK-TIME über das TDMA-Verfahren. Alle Tasks beim OSEK-OS können durch Interrupts unterbrochen werden. Preemptive Tasks sind auch von höherprioren Tasks unterbrechbar. Bei gemeinsam genutzten Ressourcen wird der korrekte Zugriff über das *Priority-Ceiling-Protokoll* gesteuert. Ein Beispiel für ein Deadlock-freies Task-Scheduling des OSEK-OS ist in Abbildung 3.5 dargestellt.

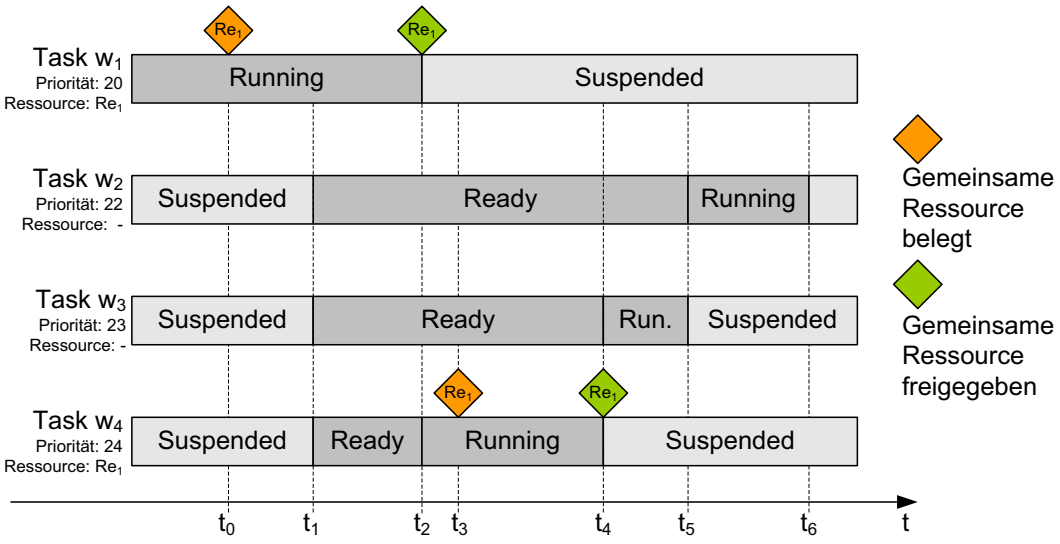


Abb. 3.5.: Beispiel für das Scheduling von Tasks unter Berücksichtigung des Priority-Ceiling-Protokolls

Der Task w_1 wird aktuell ausgeführt. Zum Zeitpunkt t_0 wird die Ressource Re_1 verwendet. Task w_4 steht ab dem Zeitpunkt t_1 zur Ausführung bereit. Trotz der höheren Priorität wird über das Priority-Ceiling-Protokoll dessen Ausführung erst bei t_2 gestartet, nachdem Task w_1 die Ressource wieder freigegeben hat (die Priorität von Task w_1 wird von 20 auf 24 angehoben).

AUTOSAR

Die AUTOSAR-Initiative definiert eine Softwarearchitektur für Steuergeräte. Diese entkoppelt die Software von der Hardware eines Gerätes. Weiterhin beinhaltet die Softwarearchitektur Funktionsmodule, die sogenannten Softwarekomponenten. Diese können unabhängig voneinander und durch verschiedene Hersteller entwickelt und dann in einem weitgehend automatisierten Konfigurationsprozess zu einem konkreten Projekt zusammengebunden werden [144]. In Abbildung 3.6 ist die Softwarearchitektur von AUTOSAR mit deren wichtigsten Modulen abgebildet. Die sogenannte Basissoftware enthält die Hardware-Schnittstellen (Treiber), die Services, das Betriebssystem und die Interaktionsschicht. Das Betriebssystem *AUTOSAR-OS* ist aufwärtskompatibel zu OSEK-OS und wurde zusätzlich um Konzepte aus OSEK-TIME erweitert [16]. Über diese Schicht wird eine klare Trennung auf der Basis standardisierter Schnittstellen realisiert. Dadurch ist der Austausch oder die Ergänzung von Applikationen leicht möglich, ohne dass der komplette Software-Stack geändert werden muss.

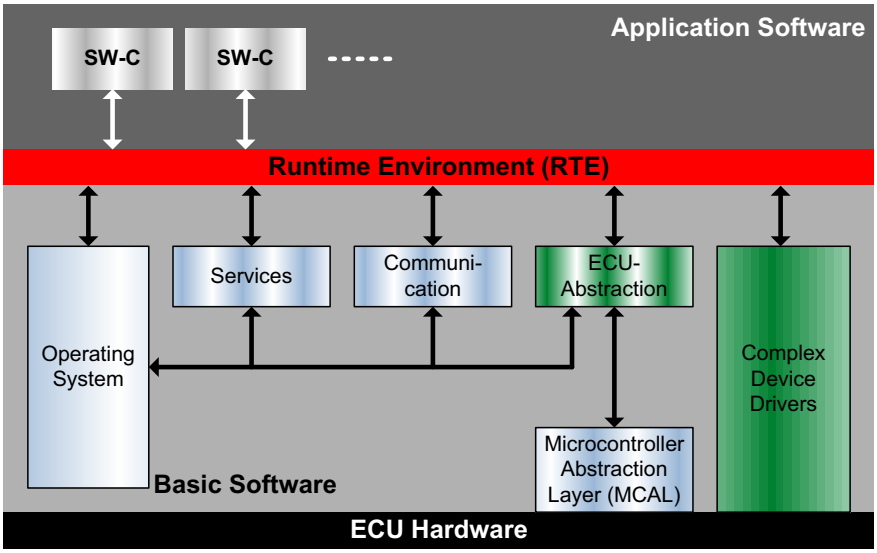


Abb. 3.6.: AUTOSAR Software-Architektur [8]

Innerhalb der Arbeit ist das Timing-Verhalten von Gateway-Steuergeräten ein wichtiger Bestandteil. Diese Komponenten sind die zentralen Elemente einer Vernetzungsarchitektur, deren Timing-Verhalten maßgeblich die Latenzzeiten der zu routenden Signale

und Botschaften beeinflusst. Aus diesem Grund werden im Folgenden die Module des AUTOSAR-Software-Stack näher beschrieben, welche für die Gateway-Funktionalität von Bedeutung sind. Abbildung 3.7 zeigt die relevanten Anteile für das Routing. Hierzu zählen die Treiber *(Drivers)*, die sogenannten *Interfaces*, der *PDU-Router*, das *COM-Modul* und das Netzwerkmanagement *(Networkmanagement)*. Der Treiber und das Interface kapseln die busspezifischen Eigenschaften. Im PDU-Router erfolgt das Weiterleiten von ganzen Datenpaketen, der sogenannten *PDUs (Protocol Data Unit)*. Das COM-Modul ist für das Routing einzelner Signale zuständig. Über das Netzwerkmanagement steuert das Gateway-Steuergerät das Schlaf- und Weck-Verhalten des gesamten Netzes.

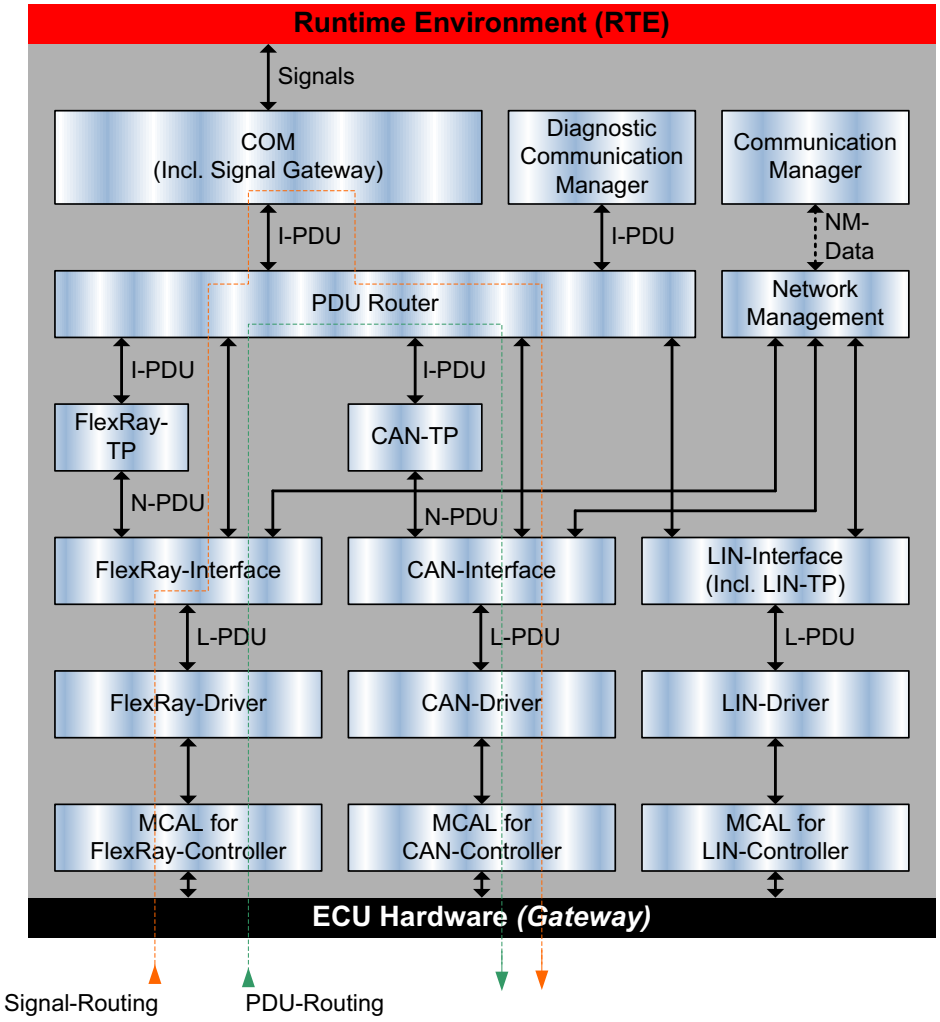


Abb. 3.7.: Detaillierte Sicht auf die einzelnen Module der AUTOSAR-Basis-Software, welche für das Routing relevant sind

Ab der Projektphase für das AUTOSAR-Release 4 wird das Timing-Thema in einer Arbeitsgruppe bearbeitet und in einer eigenen Spezifikation *Specification of Timing-Extensions* beschrieben [15]. Der Hauptfokus richtet sich auf die Bereitstellung einer konsolidierten und konsistenten Timing-Beschreibung. Diese bietet die Möglichkeit in den verschiedenen Schritten des Entwicklungsprozesses Timing-Informationen und deren Anforderungen zu beschreiben sowie eine Bewertung auf Basis dieser Daten durchzuführen. Ein Beispiel für die Beschreibung eines Systems ist in Abbildung 3.8 dargestellt.

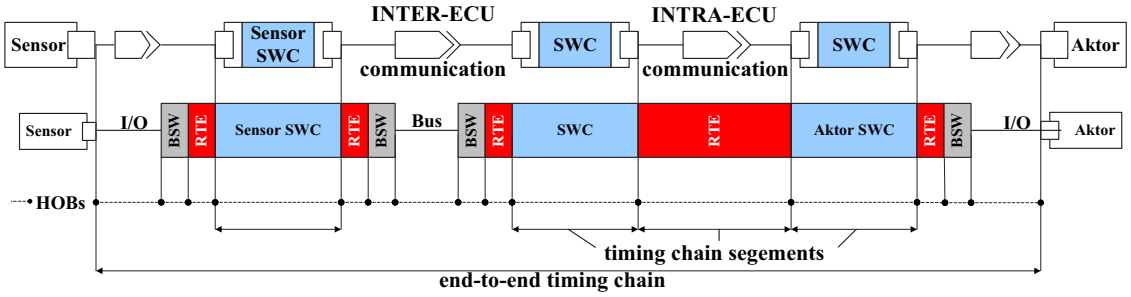


Abb. 3.8.: Beispiel für eine Ende-zu-Ende-Timingkette, von Abfrage des Sensors bis zur Auslösung des Aktors [15]

Über die Notationsmöglichkeiten der AUTOSAR-Timing-Spezifikation kann eine solche sogenannte Timing-Kette *(End-to-end timing-chain)* von der Abfrage eines Sensors bis zum Aktor beschrieben werden. Die Timingkette teilt sich in mehrere Untersegmente auf, die sogenannten *Timing chain segments*. Die Segmentierung ermöglicht eine Timing-Beschreibung auf verschiedenen Abstraktionsebenen. Die hierfür notwendige Deadline kann über die *AUTOSAR Timing-Extensions* beschrieben werden, ebenso die Latenzzeiten und Ausführungszeiten der einzelnen Segmente.

3.1.3. Kommunikationssysteme

In einer E/E-Architektur im Kraftfahrzeug kommen unterschiedliche Kommunikationssysteme zum Einsatz. Der Fokus liegt dabei auf den Kommunikationssystemen CAN und FlexRay, da diese in aktuellen und zukünftigen Vernetzungsarchitekturen die zen-

trale Rolle einnehmen. In den folgenden Abschnitten werden die für die Arbeit relevanten Aspekte erläutert. Für eine umfassende Beschreibung wird auf die wichtigsten Dokumente verwiesen.

Controller Area Network

Das *Controller Area Network (CAN)* ist das am häufigsten eingesetzte Kommunikationssystem im Kraftfahrzeug. Der CAN-Bus wurde Anfang der 90er Jahre von der Robert Bosch GmbH entwickelt. Die Spezifikation definiert eine maximale Übertragungsgeschwindigkeit V von 1MBit/s. In aktuellen Vernetzungsarchitekturen liegt die Übertragungsgeschwindigkeit auf den CAN-Bussen zwischen 100kBit/s und 500kBit/s. Höhere Übertragungsgeschwindigkeiten als 500kBit/s führen zu teilweise starken Topologieeinschränkungen im Kraftfahrzeug. Aufgrund des steigenden Kommunikationsaufkommens werden mittlerweile mögliche Topologien mit Übertragungsraten bis zu ein 1MBit/s untersucht. Die Busarbitrierung beim CAN-BUS erfolgt mittels des CSMA/CR-Verfahrens *(Carrier Sense Multiple Access/Collision Resolution)*. Über den Identifier wird die Nachrichtenpriorisierung geregelt, d.h. der niedrigste Wert hat die höchste Priorität. Der CAN-Bus arbeitet ereignis-getrieben, d.h. die Spezifikation sieht keine festen Sendezeitpunkte für bestimmte Botschaften vor. Es ist jedem Busteilnehmer erlaubt, zu einer beliebigen Zeit auf den Bus zuzugreifen. Pro Nachricht können maximal acht Byte Nutzdaten übertragen werden. In Abbildung 3.9 ist der Aufbau eines CAN-Datenframes dargestellt. Die Daten werden NRZ-codiert *(Non-Return to Zero)*. Weiterhin wird für einen Teilbereich der Botschaft Bitstuffing angewandt. Dieses wird zur fortlaufenden Synchronisation der CAN-Knoten verwendet.

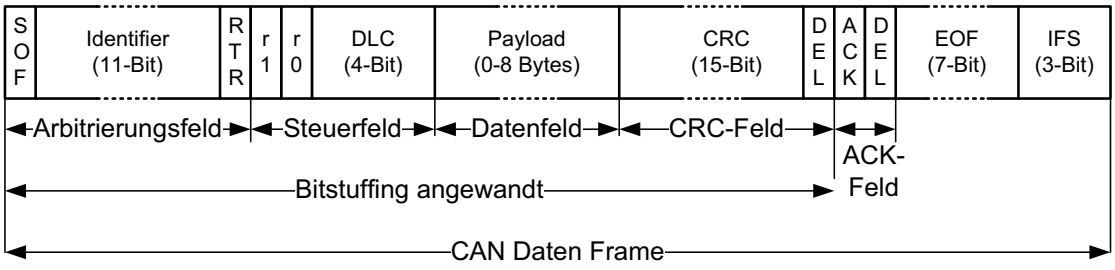


Abb. 3.9.: Aufbau eines Daten- oder Datenanforderungsframes im CAN-Standardformat [45]

Sowohl für die Berechnung der Buslast U als auch für die Ermittlung der Übertragungszeit einer Botschaft m_k müssen deren Stuffbits mit berücksichtigt werden. Die maximalen Stuffbits $n_{stuff,k}$ einer Botschaft m_k berechnen sich wie folgt:

$$n_{stuff,k} = \left\lfloor \frac{34 + 8 \cdot p_k - 1}{4} \right\rfloor \qquad [3.1]$$

Für die Berechnung der Übertragungszeit C_k einer CAN-Botschaft m_k sind alle Steuerbits *(Header und Trailer)* und deren Payload p_k (Anzahl der Nutzdatenbytes) zu berücksichtigen. Für die Berechnung der Busauslastung ist zusätzlich noch noch der *Interframe-Space* $T_{ifs} = 3Bit$ den Steuerbits hinzuzufügen.

$$C_k = \frac{8 \cdot p_k + 44 + n_{stuff}}{B} \qquad [3.2]$$

Da eine schon gestartete Botschaftsübertragung nicht unterbrechbar ist, kann eine weitere Botschaft m_k erst mit dem Buszugriff beginnen, wenn im schlechtesten Fall die längste niederpriore Botschaft vollständig übertragen ist. D.h. beim Sendevorgang findet eine Umkehr der Prioritäten statt. Der Sender, der den Bus belegt, hat bis zum Abschluss seiner Übertragung die höchste Priorität. Weitere wichtige Parameter des CAN-Busses, die eine Auswirkung auf das Zeitverhalten der Botschaften haben, sind:

- Die Offsets der CAN-Botschaften, oft auch als *StartDelayTime* (siehe *Specification of DBKOM-Attributes* [138]) oder *ComTxModeTimeOffsetFactor* (siehe *Specification of Communication* in AUTOSAR [10]) referenziert. Über die Offsets wird das Versenden der zyklischen Botschaften eines Steuergerätes entzerrt, so dass es zu keiner blockweisen Übertragung kommt.

- Die Sendetypen der Botschaften beschreiben, wie eine Botschaft von deren jeweiligem Sender verschickt wird. Die Sendetypen sind OEM-spezifisch. In Tabelle 3.1 sind die Sendetypen der CAN-Botschaften beschrieben, so wie diese bei der Daimler AG zum Einsatz kommen.

Tab. 3.1.: Sendetypen der CAN-Botschaften, so wie diese bei der Daimler AG zum Einsatz kommen [138]

Sendetyp	**Zeitliches Verhalten**
zyklisch *cyclicX*	immer aktiv mit fester Periode T_{cycle}
spontan *spontanous*	spontanes Auftreten, relevant für die Modellierung ist der Mindestsendeabstand T_{min}
bei aktiver Funktion (BAF) *cyclicIfActive*	zeitweise aktiv (wenn Funktion aktiviert) mit fester Periode
zyklisch und spontan (csx) *cyclicAndSpontanWithDelay*	immer aktiv mit fester Periode T_{cycle} zusätzlich kann die Botschaft auch noch spontan innerhalb der Periode unter Berücksichtigung des Mindestsendeabstandes T_{min} verschickt werden
schnell *cyclicIfActiveFast*	immer aktiv mit zwei festen Perioden: Lang- same Periode T_{slow} bei nicht aktiver Funktion und schnelle Periode T_{fast} bei aktiver Funktion
geändert *cyclicWithRepeatOnDemand*	werden abhängig von der Anzahl der definierten Wiederholungen zyklisch gesendet
Keine *none*	Kein Verhalten definiert

Für weiterführende Literatur zum Thema CAN-Bus wird auf die folgende Literatur verwiesen [34], [104] und [69].

FlexRay

Das Kommunikationssystem *FlexRay* wurde ab 2001 innerhalb eines Konsortiums verschiedener OEMs, Zulieferer und Halbleiterhersteller entwickelt. In den neuen Fahrzeuggenerationen ist das System mittlerweile auch im Einsatz. Auf physikalischer Ebe-

ne erlaubt FlexRay den ein- oder zweikanaligen Betrieb. Es kann eine Umsetzung in Linien- oder Sterntopologie sowie in einer Mischform erfolgen. Die maximale Übertragungsgeschwindigkeit liegt bei $V = 10MBit/s$.

FlexRay ist wie CAN ein nachrichtenorientiertes Kommunikationssystem. Der Zugriff auf den Bus erfolgt jedoch bei FlexRay zeitgesteuert über das sogenannte *Time Division Multiple Access (TMDA)* Verfahren. Jedes Steuergerät erhält zu festgelegten Zeitpunkten den exklusiven Buszugriff. Das TDMA-Verfahren macht es notwendig, dass eine Synchronität zwischen allen Busknoten vorliegt. Hierzu sind verschiedene Verfahren zur Uhrensynchronisation (Raten- und Offsetkorrektur) notwendig. Die Zeit ist bei FlexRay auf der Basis von *Makroticks* und *Microticks* definiert. Ein Makrotick ist global für das gesamte Netz festgelegt. Die Microticks sind knotenspezifisch. Abbildung 3.10 zeigt die Timing-Ebenen von FlexRay. Oberhalb der Makrotick-Ebene befindet sich die *Arbitration Grid* Ebene, hier erfolgt die Abbildung der Makroticks auf die Slots des statischen Segmentes sowie auf die Minislots des dynamischen Segmentes. Auf der obersten Ebene ist der Kommunikationszyklus (Zyklus) definiert. Dieser beinhaltet ein statisches und optional ein dynamisches Segment für die Datenübertragung. Zusätzlich sind die *Network Idle Time* und das optionale *Symbol Window* Bestandteile eines Zyklus.

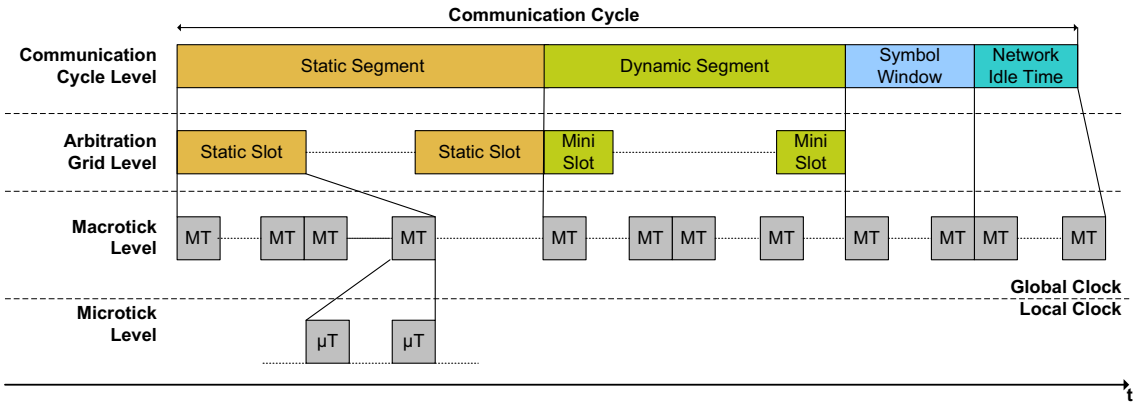


Abb. 3.10.: FlexRay-Timing-Ebenen [37]

Der Aufbau eines FlexRay-Frames (Botschaft) ist in Abbildung 3.11 dargestellt. Diese besteht aus einem Header und einem Trailer sowie einem Datenfeld mit maximal 254Byte.

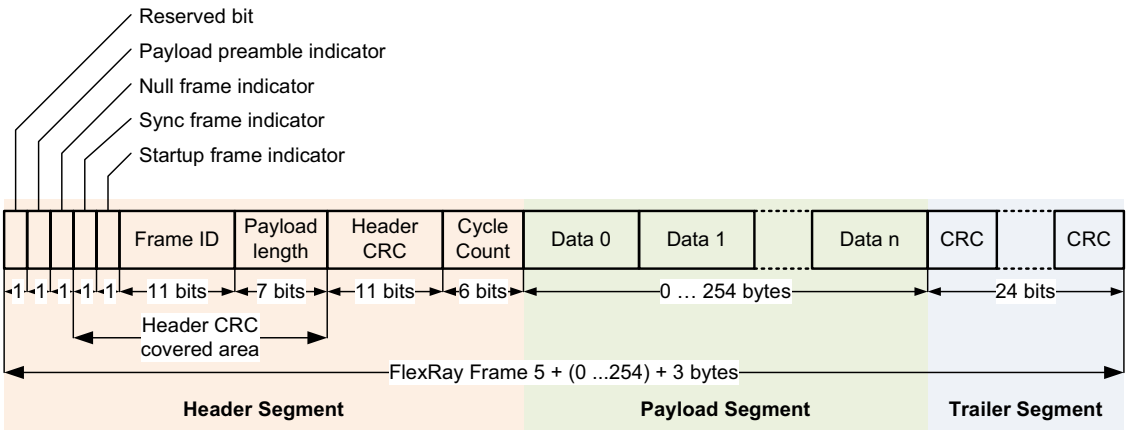


Abb. 3.11.: Aufbau eines FlexRay-Frames [37]

Die Übertragungszeit C_k einer Botschaft m_k hängt von der Übertragungsgeschwindigkeit V und der Länge der Botschaft ab. Im statischen Segment von FlexRay haben alle Botschaften dieselbe Länge. Die Botschafts- bzw. Framelänge *aFrameLenght* in Bit berechnet sich wie folgt (siehe *FlexRay-Specification 3.0* auf Seite 293 [37]:

$$\begin{aligned} aFrameLength[Bit] = & gdTSSTransmitter[Bit] + cdFSS[Bit] + 80Bit + \\ & aPayloadLength[two-byteword] \\ & \cdots 20Bit/two-byteword + cdFES[Bit] \end{aligned} \quad [3.3]$$

Dabei ist $gdTSSTransmitter$ die sogenannte *Transmission Start Sequence*, diese kann im Wertebereich zwischen $3...5Bit$ liegen. Der Parameter $cdFSS$ steht für die Dauer der Startsequenz, diese hat die Länge von $1Bit$. Die Endsequenz $cdFES$ hat eine Länge von $2Bit$. Da jedes übertragene Byte auf dem FlexRay $10Bit$ entspricht, kommen noch für den $8Byte$ großen Header $80Bit$ hinzu sowie die entsprechende Payload. Daraus ergibt sich für die Übertragungszeit:

$$C_k = \frac{aFrameLength}{V} \quad [3.4]$$

Im dynamischen Segment ergibt sich die Übertragungsdauer aus der Botschaftslänge. Die entsprechende Formel ist in der *FlexRay-Specification 3.0* auf Seite 321 zu finden [37]. Ein weiterer wichtiger Punkt bei FlexRay-Steuergeräten ist die korrekte Konfiguration der FlexRay-Jobs (siehe auch [11]). Diese sind in der Art zu definieren, dass

das Schreiben eines Frames in den *Message Ram* des FlexRay-Controllers unmittelbar vor dem nächsten Sendezeitpunkt des Frames stattfindet. Gleiches gilt für das Lesen aus dem Message Ram. Dieses Vorgehen ist notwendig, um eine optimale Übermittlung der Daten sicherzustellen und Wartezyklen zu vermeiden. In Abbildung 3.12 ist beispielhaft ein solches Zusammenspiel zwischen FlexRay-Jobs und -Schedule dargestellt. Beispielsweise bedient der FlexRay-Job *Tx2* die Slots $40 - 88$. Um die zeitlich optimale Übergabe der Daten zu gewährleisten, muss der Job vor dessen Deadline mit der Ausführung fertig sein.

Für weiterführende Literatur zum Thema FlexRay wird auf die folgenden Dokumente verwiesen [94], [37] und [38].

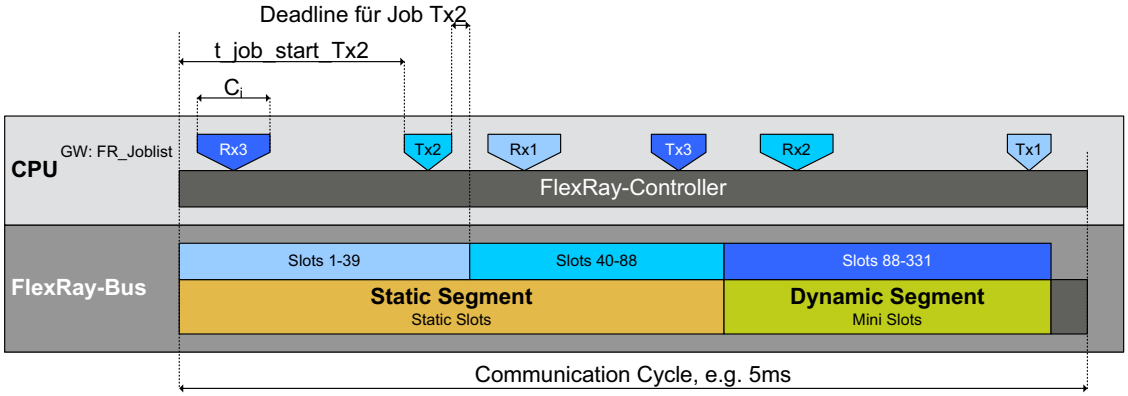


Abb. 3.12.: Zusammenspiel zwischen den FlexRay-Jobs im Steuergerät und dem FlexRay-Schedule

Weitere Kommunikationssysteme

Weitere Kommunikationssysteme, welche in aktuellen Vernetzungsarchitekturen von Kraftfahrzeugen zum Einsatz kommen, sind in Abbildung 3.13 dargestellt.

- Das *Local Interconnect Network (LIN)*, ein serielles Bussystem, welches nach dem Master/Slave-Prinzip arbeitet. Die maximale Übertragungsgeschwindigkeit des LIN-Bus liegt bei 20kBit/s [71],[72].

- Der *Media Oriented System Transport (MOST)*, dieser ist für Multimedia-Anwendungen konzipiert und hat eine maximale Übertragungsgeschwindigkeit von 150MBit/s. Der MOST-Bus ist in Ringtopologie ausgeführt [82], [83].

- Das *Low Voltage Differential Signaling (LVDS)* wird zur Hochgeschwindigkeitsdatenübertragung verwendet, beispielsweise für die Übertragung von Rohdaten von Kameras.

- Das *Ethernet-IP (Internet Protocol)*, das derzeit als schneller Flash-Zugang eingesetzt wird und bei BMW für die Anbindung des *Rear Seat Entertainments*. Aktuell wird an weiteren Einsatzmöglichkeiten geforscht (z.B. für Kameravernetzung und als Backbone-Bus) [110], [93].

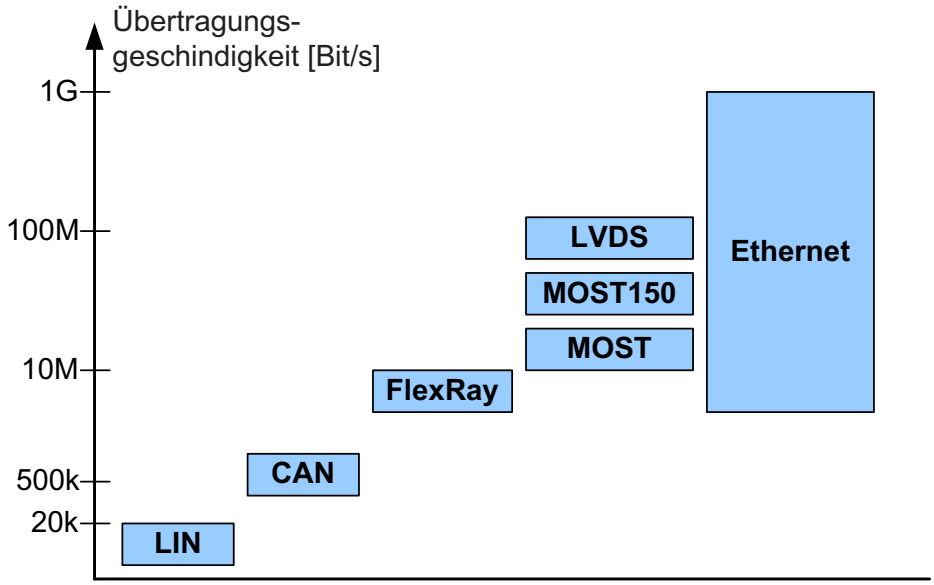


Abb. 3.13.: Gegenüberstellung der Kommunikationssysteme im Kraftfahrzeug anhand deren Übertragungsgeschwindigkeiten

Die folgende Tabelle 3.2 stellt die wichtigsten Eigenschaften der einzelnen Kommunikationssysteme gegenüber.

Tab. 3.2.: Gegenüberstellung der wichtigsten Eigenschaften der Automotive-Kommunikationssysteme

Eigenschaften	**LIN**	**CAN**	**FlexRay**	**MOST**
Max. Brutto-Übertragungsgeschwindigkeit [MBit/s]	0.02	1.0	10.0	150.0
Max. Payload [Byte]	8	8	254	384
Zugriffsverfahren	Token Passing	CSMA/CR	TDMA	Token Passing
Topologie	Bus	Bus/Stern	Bus/Stern	Ring
Synchronisation	Master Slave	Keine	Verteilte	Master Slave
Kanäle	1	1	1-2	1
Max. Knoten	60	keine	64	keine
Physical Layer	half duplex	half duplex	half duplex	full duplex

3.2. Entwicklungsprozess für E/E-Architekturen

Durch den hohen Anteil an E/E-Systemen im Kraftfahrzeug ist die Existenz eines durchgängigen und konsistenten Entwicklungsprozesses von zentraler Bedeutung für die Qualität der Produkte. Die Entwicklung eines Kraftfahrzeugs lässt sich grob in fünf Phasen einteilen (siehe auch Abbildung 3.14):

- In der *Strategiephase* wird die Entscheidung über die Entwicklung einer neuen Baureihe getroffen. Hierfür werden Marktanalysen durchgeführt und die Eckpunkte für die grobe strategische Ausrichtung festgelegt sowie die Leitplanken (Meilensteine) für die Entwicklung vorgegeben.

- Während der *Konzeptphase* erfolgt die Evaluierung unterschiedlicher Alternativen einer Realisierung des Fahrzeugs. Dabei bilden die vorgegebenen Randbe-

dingungen aus der Strategiephase die Grenzen des Entwurfsraumes. Für die E/E-Architektur wird in dieser Phase ein Gesamtkonzept erarbeitet, welches die Grundlage für die nächste Entwicklungsphase bildet.

- Die Entwicklung des eigentlichen Fahrzeugs wird in der gleichnamigen *Entwicklungsphase* durchgeführt. Die einzelnen Artefakte eines Fahrzeugs werden vom Prototypenstadium bis hin zur Serienreife entwickelt.
- Die *Serienphase* beschreibt den Zeitraum, in dem ein Fahrzeug produziert wird. Während dieser Zeit werden meist noch kleinere Verbesserungen am Fahrzeug vorgenommen. In der Mitte der Serienphase erfolgt fast immer die sogenannte *Modellpflege*, welche teilweise größere Änderung am Fahrzeug zur Folge hat.
- Die *Nach Serienphase* ist der Zeitraum, in dem das Fahrzeug nicht mehr produziert wird, aber noch entsprechende Ersatzteile für die Wartungen und Reparaturen verfügbar sein müssen.

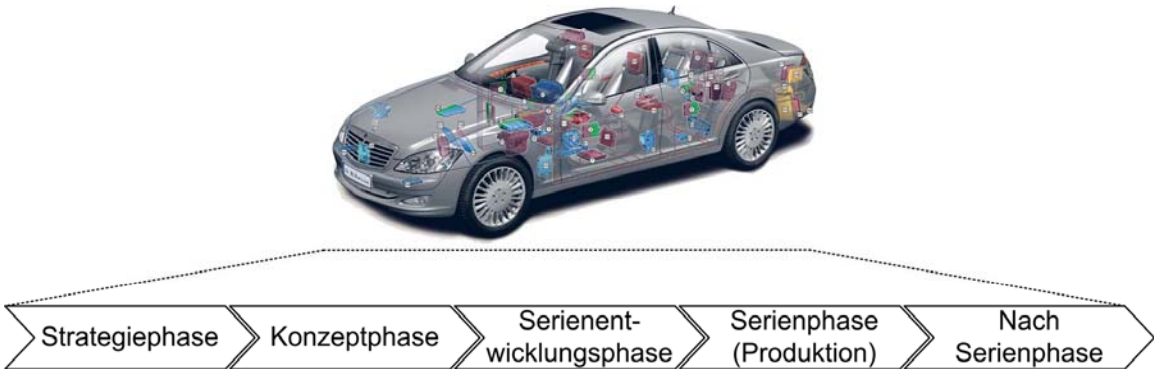


Abb. 3.14.: Phasen innerhalb des Lebenszyklus eines Automobils

Die eigentliche Entwicklung einer E/E-Architektur beginnt in der Konzeptphase. In diesem Zeitraum werden verschiedene Architekturvarianten anhand definierter Kriterien miteinander verglichen. Der Einsatz neuer Technologien wird bewertet und die Innovationen werden integriert. In Abbildung 3.15 ist der grobe Ablauf für die Konzeptphase dargestellt. Über das sogenannte *Frontloading* wird auf der Basis existierender Architekturen ein initiales Modell erstellt. Dieses wird um die Innovationen erweitert. Der

Trend zum Frontloading hat zum Ziel, die Zahl der Änderungen innerhalb einer E/E-Architektur zu reduzieren oder zumindest in eine frühere Phase im Entwicklungsprozess zu verschieben. Frühzeitig wird versucht, Erkenntnisse über die Fahrzeugentwicklung zu gewinnen und die entworfenen Konzepte, z.B. auf digitaler Basis, abzusichern [141].

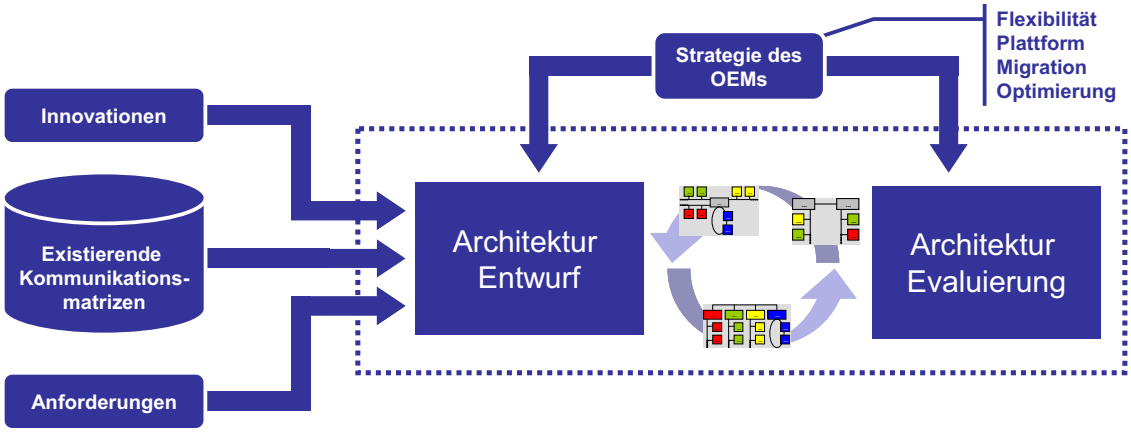


Abb. 3.15.: Vorgehensweise beim Entwurf von E/E-Architekturen während der Konzeptphase [129]

Da sich diese Arbeit auf die Bewertung von Vernetzungsarchitekturen fokussiert, wird im Folgenden auf die hierfür relevanten Prozessschritte eingegangen. Die Entwicklungsschritte der Komponenten sowie des Gesamtsystems einer Vernetzungsarchitektur orientiert sich am sogenannten *V-Modell*. In Abbildung 3.16 sind die einzelnen Schritte aufgezeigt.

Nach der Analyse der Anforderungen erfolgt die Spezifikation der logischen Systemarchitektur, d.h. in diesem Schritt wird das Funktionsnetzwerk festgelegt sowie die Schnittstellen der Funktionen und der Kommunikation zwischen den Funktionen der gesamten logischen Systemarchitektur beschrieben. Anschließend wird die logische Systemarchitektur analysiert und die Spezifikation der technischen Systemarchitektur abgeleitet. In dieser Phase des Entwicklungsprozesses sind erste Timing-Abschätzungen möglich. Nach erfolgreicher Spezifikation der technischen Systemarchitektur erfolgt die Ableitung der Spezifikation für die einzelnen Komponenten (Busse und Steuergeräte). In dieser Phase ist auch die Timing-Auslegung für die Systeme durchzuführen. Im Anschluss daran werden die Komponenten anhand der Spezifikation entworfen, implemen-

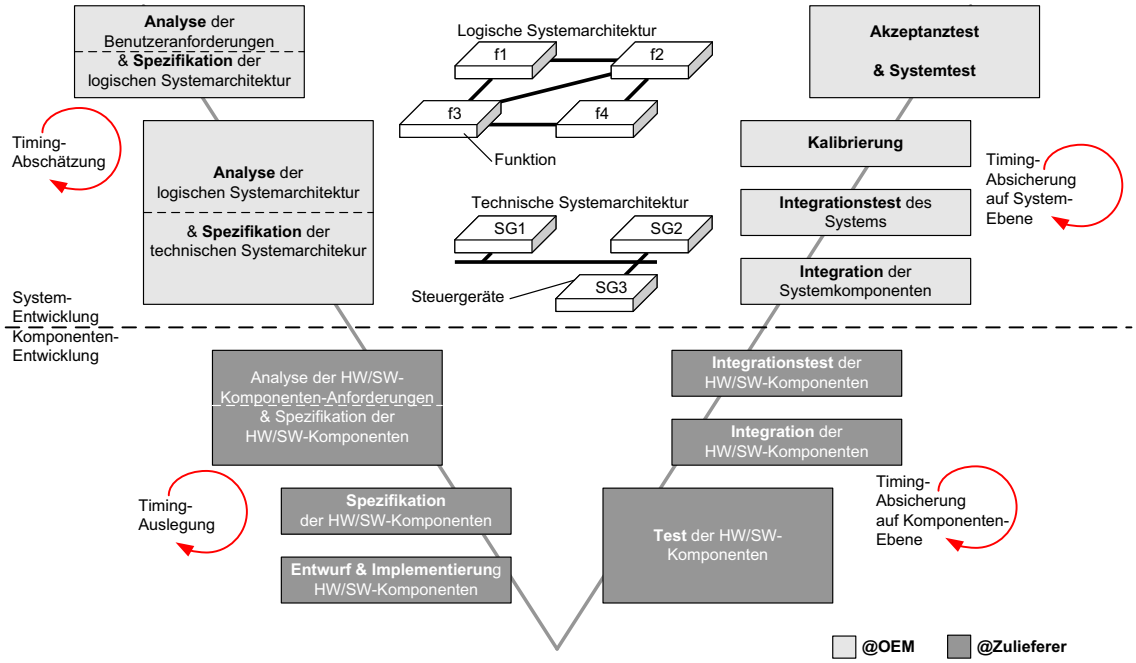


Abb. 3.16.: V-Modell für die Entwicklungsschritte von Vernetzungsarchitekturen im Kraftfahrzeug, ergänzt mit relevanten Timing-Bewertungsschritten (in Anlehnung an *Automotive Software Engineering* [111])

tiert und getestet. Im nächsten Schritt werden die einzelnen Komponenten auf Systemebene integriert und getestet. Im kompletten rechten Arm des V-Modells können in den einzelnen Schritten die Module, Komponenten und Systeme auf deren Timing-Verhalten abgesichert werden.

3.3. Bewertung des Stands der Technik

Durch die stetige Zunahme der E/E-Systeme im Kraftfahrzeug sind dementsprechend auch die Anforderungen an den Entwicklungsprozess gestiegen. Weiterhin sind Themen bezüglich Produkthaftung zukünftig mit zu berücksichtigen, z.B. die Norm *ISO26262*. Im aktuellen Entwicklungsprozess ist meist folgende Aufteilung anzutreffen: Der OEM entwirft das Gesamtsystem, spezifiziert die Einzelsysteme, integriert die einzelnen Komponenten und testet den Verbund [39]. Die Zulieferer übernehmen den Entwurf, die Implementierung und den Test der einzelnen Komponenten. Dabei kann der OEM auch einzelne Anteile an den Komponenten mit beisteuern, z.B. wird ein Teil der Applikationen (Software-Komponenten) vom OEM entwickelt. Durch die zunehmende Vernetzung

von Funktionen sowie aufgrund der immer höheren Auslastung der Systeme reichen die bisher eingesetzten Verfahren für eine vollständige Absicherung des Timing-Verhaltens nicht aus. Abbildung 3.17 zeigt ein typisches Beispiel für ein verteiltes System.

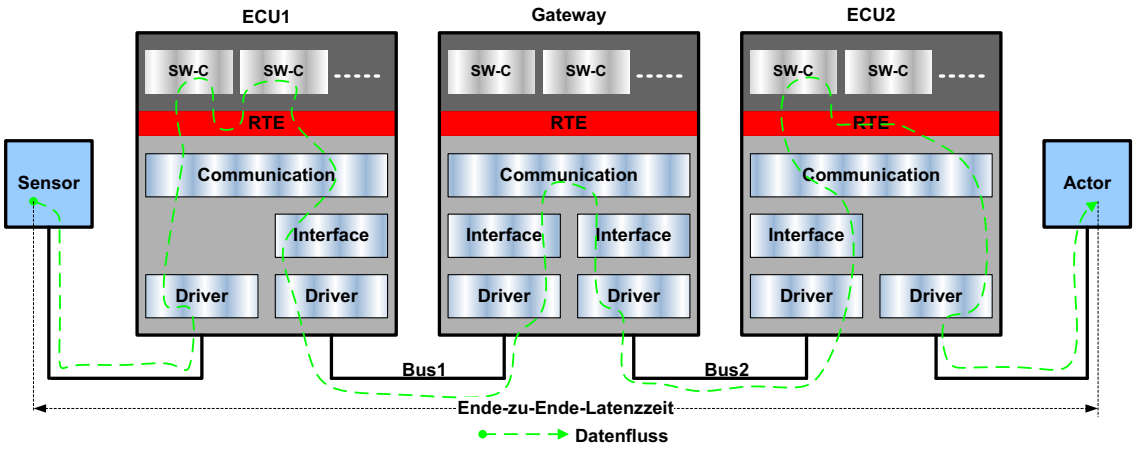


Abb. 3.17.: Beispiel für einen typischen Ende-zu-Ende-Pfad vom Sensor bis zum Aktor

Um einen solchen Ende-zu-Ende Pfad bewerten und absichern zu können, ist das Timing-Verhalten der beteiligten Komponenten im Detail zu berücksichtigen. Insbesondere für die Evaluierung der Kommunikationssysteme wurden bisher rein kapazitive Untersuchungsmethoden verwendet. Um hier zukünftig eine detaillierter Aussage treffen zu können, wird in dieser Arbeit eine durchgängige Methodik für die Timing-Bewertung aufgezeigt. In diesem Zusammenhang sind folgende Fragestellungen von Bedeutung:

1. Wie kann das Timing-Verhalten von Software- und Hardware Systemen und deren Subsysteme im Kraftfahrzeug gezielt vorhergesagt und abgesichert werden?

2. Welche Daten sind für eine aussagekräftige Timing-Bewertung notwendig und stehen zu welchem Entwicklungszeitpunkt zur Verfügung?

3. Inwieweit sind Timing-Aussagen in der frühen Entwurfsphase möglich?

4. Wie ist eine exakte Abbildung des Timing-Verhaltens in den entsprechenden Bewertungsverfahren umzusetzen?

5. Wie lassen sich die Timing-Bewertungsverfahren optimal in den existierenden E/E-Entwicklungsprozess integrieren?

Zu Punkt 1 werden in Kapitel 4 Ansätze und Verfahren beschrieben, welche eine Timing-Bewertung ermöglichen. Zu Punkt 2 erfolgt in Kapitel 5 die Vorstellung eines Verfahrens zur Extraktion von Timing-Informationen aus existierenden Fahrzeugen. Diese Informationen können dann z.B. die Datenqualität des Frontloadings in der Konzeptphase steigern (Punkt 3). Weiterhin werden zu Punkt 4 in Kapitel 6 Regeln abgeleitet, um eine aussagekräftige Timing-Bewertung von Vernetzungsarchitekturen und Gateway-Systemen zu ermöglichen. Die Eingliederung der Bewertungsverfahren (Punkt 5) sowie die Entwicklung einer durchgängigen Bewertungsmethodik wird in Kapitel 7 diskutiert.

Im AUTOSAR-Konsortium wurden in den letzten Jahren Beschreibungsmöglichkeiten und Ansätze für die Bewertung des Timing-Verhaltens erarbeitet. Die für eine Timing-Bewertung notwendigen Attribute können ab dem AUTOSAR Release 4.0 in einem eigenen Template hinterlegt werden [15]. Zusätzlich werden verschiedene Ansätze vorgeschlagen, um Timing-Fragestellungen in den unterschiedlichen Phasen des Entwicklungsprozesses sowie auf verschiedenen Ebenen bewerten zu können. Abbildung 3.18 zeigt die einzelnen Sichtweisen für eine Bewertung auf.

- Über das *SWC-Timing* kann das interne Timing-Verhalten einer Software-Komponente bewertet werden.
- Das *VFB-Timing* berücksichtigt das Timing-Verhalten zwischen den einzelnen miteinander interagierenden Software-Komponenten auf der Ebene des *Virtual Function Bus (VFB)*.
- Das *BSW-Module-Timing* beschreibt das interne Timing-Verhalten der Basis-Software-Module.
- Mit dem *ECU-Timing* kann das Timing-Verhalten des gesamten Software-Stacks eines Steuergerätes beschrieben werden.
- Über das *System-Timing* kann das Timing-Verhalten von mehreren Steuergeräten und Bussen im Verbund untersucht werden.

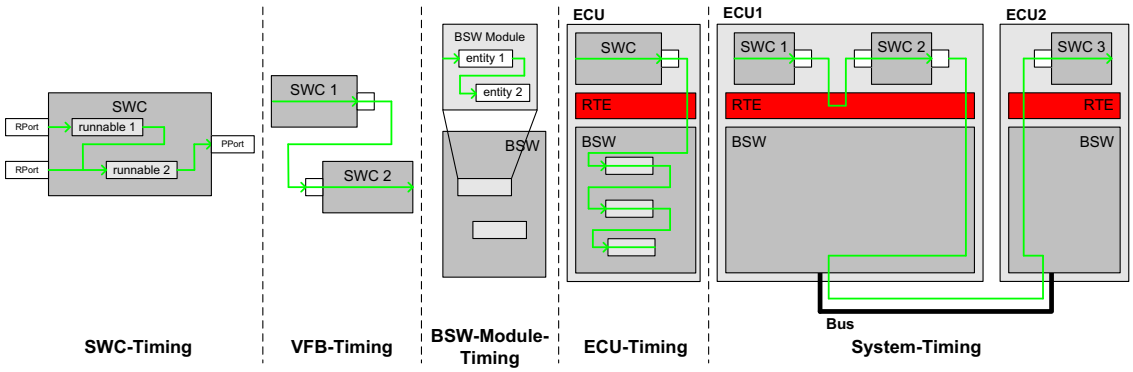


Abb. 3.18.: Timing-Modelle, welche in der Timing-Group für das Release 4.0 der AUTOSAR-Spezifikation erarbeitet wurden

Die vorliegende Arbeit behandelt im Detail die Timing-Modellierung und -Bewertung der Kommunikationssysteme sowie der Basis-Software. Die Timing-Bewertung der Software-Komponenten oberhalb der RTE ist nicht Gegenstand der Arbeit. Die Bereitstellung der notwendigen Timing-Informationen für die Bewertung wurde bisher nicht im Detail von anderen Arbeiten beleuchtet. Hier zeigt diese Dissertation Lösungsmöglichkeiten auf. Ein weiterer wichtiger Punkt ist die exakte Modellierung des Timing-Verhaltens. In AUTOSAR sind die Timing-Attribute spezifiziert, deren Einsatzmöglichkeiten und Wechselwirkungen bislang nicht detailliert für die Bewertung von Vernetzungsarchitekturen und Gateway-Systeme untersucht wurden.

3.4. Kenngrößen für die Timing-Bewertung

Für die Timing-Bewertung von Vernetzungsarchitekturen und Gateway-Systemen werden im Folgenden die wichtigsten Kenngrößen definiert. Aus Vernetzungssicht lassen sich diese Kenngrößen in mehrere Bereiche einteilen: Bewertung der Steuergeräte, Bewertung der Busse und Bewertung des Verbundes (Gesamtsystems). Für die einzelnen Kenngrößen wird im Folgenden eine Beschreibung gegeben, deren Aussagen sich anhand dieser ableiten lassen. Die formale Definition, sofern diese nicht explizit ausgeführt ist, kann in Abschnitt 2.2 nachgeschlagen werden.

Kenngrößen für die Bewertung von Steuergeräten

Grundlast

Mit der Grundlast wird die Auslastung der CPU auf der Basis der zyklischen Tasks bewertet. Über diesen Wert kann eine Einschätzung für die noch freie CPU-Zeit, welche für die dynamische Last zur Verfügung steht, getroffen werden.

Dauer von Spitzenlasten

Über die Dauer der Spitzenlast für eine bestimmte Zeitdauer Δt können Rückschlüsse auf das dynamische Verhalten des Systems getroffen werden.

Ausführungszeiten

Die Ausführungszeiten C_k der Tasks und Funktionen geben ein detailliertes Bild darüber, welche Funktionalität wie viel an Rechenzeit benötigt. Mit Hilfe dieser Informationen können besser Integrationsentscheidungen getroffen werden. Insbesondere bei bereits existierenden Steuergeräten, auf die weitere Funktionen integriert werden sollen, lassen sich hierüber und in Verbindung mit den Antwortzeiten exakte Aussagen ableiten, inwieweit die zusätzliche Funktionalität das Gesamtsystemverhalten beeinflusst.

Antwortzeiten

Über die Antwortzeiten R_k kann die Betriebssystemkonfiguration abgesichert werden. Die Werte können als Information für die Funktionsentwickler dienen, z.B. um die Stabilität von Regelkreisen abzusichern. Weiterhin sind darüber Mehrfachaktivierungen von Tasks sicher bestimmbar.

Interruptverhalten

Das Verhalten der Interrupts ist eine wichtige Größe, welche maßgeblichen Einfluss auf das Timing-Verhalten des Gesamtsystems hat. Weiterhin können mögliche Interruptverluste identifiziert und eliminiert werden. In diesem Kontext sind folgende Eigenschaften zu berücksichtigen:

1. Die Interruptsperrzeiten, welche eine direkte Auswirkung auf alle Interrupts des Systems haben.

2. Das dynamische Auftreten der Interrupts ist ein wichtiges Kriterium, d.h. die Frage wie viele Interrupts maximal gleichzeitig auftreten können.

3. Über Punkt 1. und 2. lassen sich die Einflüsse der Interrupts auf die zyklischen Tasks bewerten.

Jitter

Wie auch die Antwortzeit können die ermittelten Jitterwerte J_k als zusätzliche Information für die Funktionsentwickler dienen.

Optimierungspotentiale

Mögliche Optimierungspotentiale ergeben sich durch die Identifikation von Engpässen über die Bestimmung der Ausführungszeiten. Weiterhin kann auf der Basis der Kenntnis der Latenz- und Ausführungszeiten die Konfiguration des Betriebssystems angepasst werden.

Kenngrößen für die Bewertung von Kommunikationssystemen

Periodische Grundlast

Die periodische Grundlast umfasst alle Botschaften, die laut K-Matrix zyklisch auf dem Bus versendet werden. Botschaften, welche nur bei aktiver Funktion zum Senden anstehen, werden dabei nicht berücksichtigt. Der Wert beschreibt die Grundauslastung eines Busses und stellt somit die untere Schranke dar.

Periodische Spitzenlast

Bei der periodischen Spitzenlast werden alle Botschaften berücksichtigt, die ein zyklisches Verhalten haben, d.h. auch die Botschaften, welche nur bei einer aktiven Funktion versendet werden. Mit dem Wert der Spitzenlast kann eine Aussage getroffen werden, wie viel Platz im schlimmsten Fall noch für dynamische Kommunikation verfügbar ist.

Dauer von Spitzenlasten

Die Dauer der Spitzenlast B, auch *Burst* genannt, auf den Bussen beschreibt kritische Bereiche der Kommunikation, während der ohne Unterbrechung Botschaften gesendet werden. Ursache hierfür ist meist ein hoher Anteil an dynamischer Last (spontane Kommunikation).

Jitter

Der Jitter von Botschaften entsteht beim CAN-Bus durch die Arbitrierung und beim FlexRay im dynamischen Segment. Die Priorität einer Botschaft hat direkten Einfluss auf deren Jitter. Insbesondere in der Spezifikationsphase der K-Matrizen und des FlexRay-Schedules sollte dieser Einfluss berücksichtigt werden.

Antwortzeiten

Die Antwortzeiten R_k geben die Zeiten an, welche für den Zugriff und die Übertragung der Botschaften benötigt werden.

Relative Antwortzeiten

Die relativen Antwortzeiten $R_{rel,k}$ sind ein Maß für das zeitliche Verhalten einer Botschaft im Bezug auf deren Deadline d_k. Ähnlich wie über den Slack (siehe Abschnitt 2.2 kann damit eine Aussage getroffen werden, wie viel Restzeit noch vorhanden ist. Für die relative Antwortzeit gilt:

$$\begin{aligned} &R_{rel,k} = \frac{R_k}{d_k} \text{ mit} \\ &R_{rel,k} < 1 : \text{Die Botschaft wird innerhalb der Deadline versendet} \\ &R_{rel,k} \geq 1 : \text{Die Botschaft wird nicht innerhalb der Deadline versendet} \end{aligned} \quad [3.5]$$

Optimierungspotentiale

Mögliche Optimierungspotentiale bietet die Änderung der Schedulekonfiguration, z.B. durch geänderte Vergabe der Prioritäten bzw. Slots. Speziell beim CAN-Bus lassen sich durch die gezielte Vergabe der Offsets *(StartDelayTime)*, in der AUTOSAR-Spezifikation als `ComTxModeTimeOffsetFactor` referenziert, die Antwortzeiten minimieren (siehe hierzu auch in [10]: *To avoid bursts in start-up a time offset can be configured per I-PDU*.

Kenngrößen für die Bewertung von verteilten Systemen

Datendurchsatz

Der Datendurchsatz *(in Bit/s)* ist für die Ermittlung der Übertragungskapazität sehr wichtig, um Aussagen bezüglich der Übertragungszeiten von Transportprotokollen und hinsichtlich der Flashzeiten einer Architektur zu bekommen.

Ende-zu-Ende Latenzzeiten

Bei den Ende-zu-Ende Latenzzeiten kann zwischen reaktiven und regelungstechnischen Systemen unterschieden werden. Bei reaktiven Systemen ist die Reaktionszeit L_{ft} von Interesse, d.h. die längste Zeitdauer, welche für die Übertragung benötigt wird. Bei Regelungssystemen sind der Jitter und das maximale Alter der Daten L_{ma} wichtig. Abbildung 3.19 zeigt die Unterschiede der beiden Ende-zu-Ende-Latenzzeiten auf. Eine exakte mathematische Beschreibung ist in [35] zu finden.

Ende-zu-Ende Jitter

Die Kenntnis der Ende-zu-Ende Jitter J_{end} ist für die Auslegung von verteilten Regelungssystemen wichtig, um die Regler entsprechend stabil auslegen zu können.

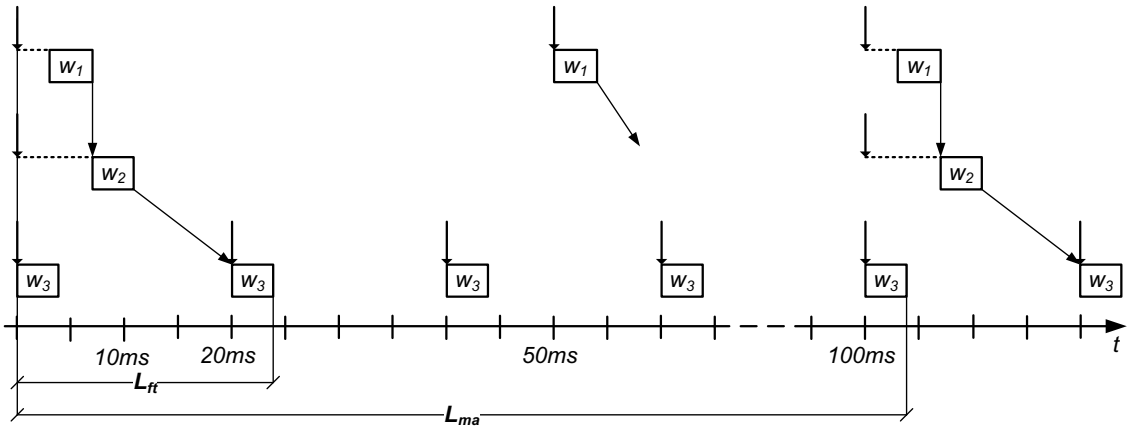


Abb. 3.19.: Beispiel für die beiden Ende-zu-Ende-Latenzzeiten: Reaktionszeit L_{ft} und maximales Alter L_{ma}

Ende-zu-Ende Jitter

Die Kenntnis der Ende-zu-Ende Jitter J_{end} ist für die Auslegung von verteilten Regelungssystemen wichtig, um die Regler entsprechend stabil auslegen zu können.

Routingzeiten

Die Routingzeit L_{route} gibt an, wie lange die maximale Verarbeitungszeit für das Routing einer Botschaft oder eines Signals im Gateway-Steuergerät dauert. Die Information ist für die Bestimmung der Ende-zu-Ende Latenzzeiten von Relevanz.

Puffergrößen

Insbesondere bei Gateway-Steuergeräten ist die Auslegung der Puffergrößen von zentraler Bedeutung, um auch bei hoher Routinglast keine Botschaften zu verlieren.

Optimierungspotentiale

Eine globale Optimierung von Ende-zu-Ende Pfaden kann durch die in den vorherigen Abschnitten aufgezeigten lokalen Optimierungsschritte erreicht werden. Dabei besteht die Möglichkeit einen oder mehrere Schritte durchzuführen und diese je nach Kritikalität und Änderungsaufwand auszuwählen.

4. Ansätze und Verfahren zur Timing-Bewertung

Im vorangegangenen Kapitel wurden Kenngrößen definiert, die eine umfassende Bewertung des Timing-Verhaltens von Vernetzungsarchitekturen ermöglichen. In diesem Kapitel werden Verfahren vorgestellt, die zur Bewertung des Timing-Verhaltens verwendbar sind. Diese lassen sich in zwei Kategorien einteilen (siehe auch Abbildung 4.1):

1. **Experimentelle Verfahren:** Bei diesen Verfahren erfolgt die Timing-Bewertung auf der Basis von Simulationsmodellen oder durch die Verwendung von realer Hardware. Ferner ist eine Kombination der beiden Verfahren möglich.

2. **Analytische Verfahren:** Diese Verfahren bestimmen das Timing-Verhalten auf analytischem Wege anhand von mathematischen Modellen. Die formalen Verfahren lassen sich in zwei Gruppen einteilen. Erstens die sogenannten holistischen Verfahren, diese erweitern die klassischen Analysemethoden der Schedulingtheorie auf verteilte eingebettete Systeme [89]. Das resultierende heterogene Modell, welches die verschiedensten Analysetechniken kombiniert, wird jedoch bei großen Systemen schnell unübersichtlich. Zweitens der modulare Analyseansatz, dieser wird oft auch als *Compositional Analysis* referenziert. Bei diesem Ansatz werden die Komponenten eines verteilten Systems parallel und lokal analysiert. Das resultierende Ausgangsereignismodell wird als Eingangsereignismodell für den nächsten Analyseschritt verwendet. Diese Art der iterativen Analyse stellt ein Fix-Punktproblem dar [100]. Stellt sich nach mehreren Iterationen Konvergenz ein, war die Analyse erfolgreich.

Die experimentellen Verfahren befinden sich schon seit vielen Jahren im breiten Serieneinsatz. Sowohl bei der Auslegung von Vernetzungsarchitekturen im Kraftfahrzeug und deren einzelnen Komponenten als auch bei der Absicherung in der Integrationsphase werden Simulations- sowie Test-Techniken eingesetzt. Mit diesen Verfahren lassen sich anhand von Testfällen verschiedenste Untersuchungen durchführen. Es können Verteilungen ermittelt und obere Schranken bestimmt werden. Je nach Testtiefe und -Umfang

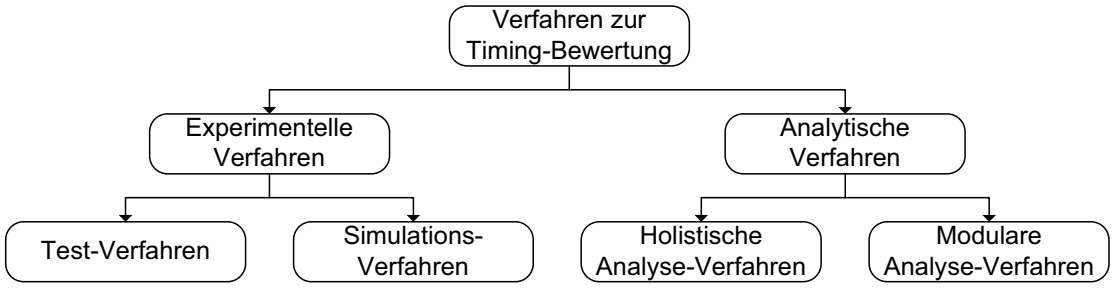


Abb. 4.1.: Übersicht möglicher Verfahren zur Timing-Bewertung

stellen diese jedoch nicht immer die oberen Schranken sicher dar. Für eine sichere Bestimmung ist der Einsatz von formalen Verfahren notwendig

An formalen Analyseverfahren für die Bewertung von Echtzeitsystemen wird schon seit vielen Jahren in der Wissenschaft geforscht. In der Halbleiterindustrie sind formale Verfahren schon seit längerer Zeit in breitem Serieneinsatz, während in Luftfahrt und Automobilindustrie deren Anwendung sich im E/E-Entwicklungsprozess erst allmählich etabliert. Mit formalen Analyseverfahren sind die Systemauslastungen sowie die unteren und oberen Schranken z.B. von Ausführungszeiten oder die maximalen Ressourcenanforderungen auf der Basis von Modellen zuverlässig bestimmbar.

Abbildung 4.2 zeigt einen exemplarischen Vergleich der verschiedenen Verfahren am Beispiel der Bestimmung der maximalen Latenzzeit. Sowohl bei der Messung als auch bei der Simulation wird das Zeitverhalten anhand von Testmustern untersucht. Dabei kann der Fall eintreten, dass das real mögliche Maximum nicht erzeugt wird (siehe Abbildung 4.2 gestrichelte Linie).

Mit formalen Analyseverfahren können die oberen Schranken dagegen sicher bestimmt werden. Mit diesen Verfahren kann es jedoch zu Überschätzungen kommen, die berücksichtigt werden müssen. Je nach Einsatzzweck bietet das eine oder das andere Verfahren gewisse Vorteile. Mit der Simulation und Tests lassen sich beliebige Details eines Systems untersuchen und die Abläufe können exakt nachvollzogen werden. Hierfür ist es jedoch notwendig, dass für alle zu überprüfenden Fälle ein entsprechendes Testpattern vorhanden sein muss. Die analytischen Verfahren abstrahieren dagegen gezielt Details, identifizieren die kritischen Randfälle automatisch und ermitteln zuverlässig die oberen Schranken [128]. Die Bestimmung von Häufigkeitsverteilungen kann

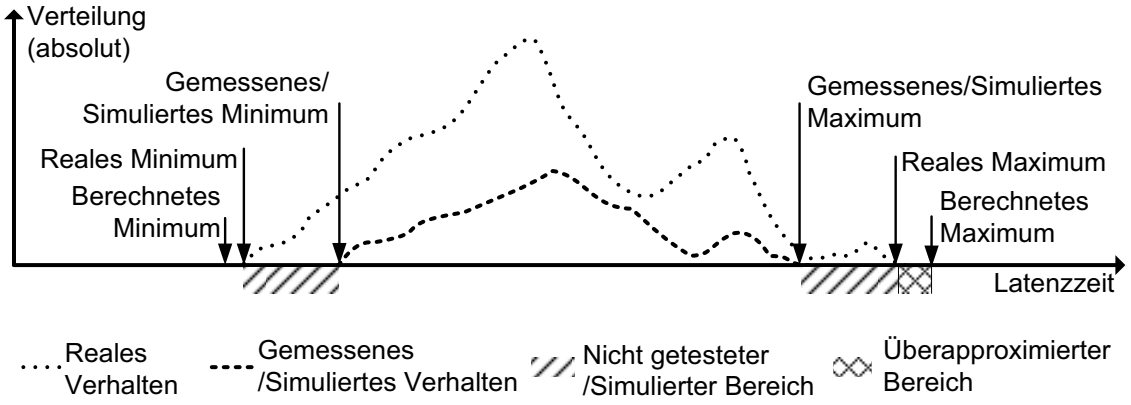


Abb. 4.2.: Vergleich der verschiedenen Timing-Bewertungsverfahren

sowohl mit Simulation und Test als auch mit analytischen Verfahren ermittelt werden (siehe z.B. [116]).

Alle Verfahren bieten die Möglichkeit, Timing-Bewertungen auf Komponentenebene (von Steuergeräten oder Bussen) und auf Systemebene (von gesamten Vernetzungsarchitekturen) durchzuführen. Im Folgenden werden die einzelnen Verfahren anhand der verschiedenen Ebenen vorgestellt.

4.1. Timing-Bewertung von Steuergeräten

Die Kriterien, welche für die Timing-Bewertung von Steuergeräten eine Rolle spielen, wurden in Abschnitt 3.4 vorgestellt. Abbildung 4.3 zeigt die einzelnen Schritte, die heute vom Entwurf bis zur Absicherung durchlaufen werden. Mit der Einführung von AUTOSAR und dem Trend einer zunehmenden modellbasierten Entwicklung von Funktionen ist die virtuelle Integration von Software-Komponenten ein immer wichtigeres Thema. Diese wird sowohl beim OEM als auch beim Zulieferer durchgeführt (linker Teil von Abbildung 4.3). Die vollständige Integration der Software wird in den meisten Fällen vom Zulieferer übernommen. Dieser prüft nach vollständiger Implementierung das Verhalten gegen die Spezifikation. Der Integrationstest des Systems wird beim OEM durchgeführt. Dabei ist das Steuergerät als *Black-Box* im Gesamtverbund integriert.

Für die Timing-Bewertung werden aktuell hauptsächlich Simulations- und Testverfahren eingesetzt. Insbesondere für sicherheitsrelevante Steuergeräte wird aber mittler-

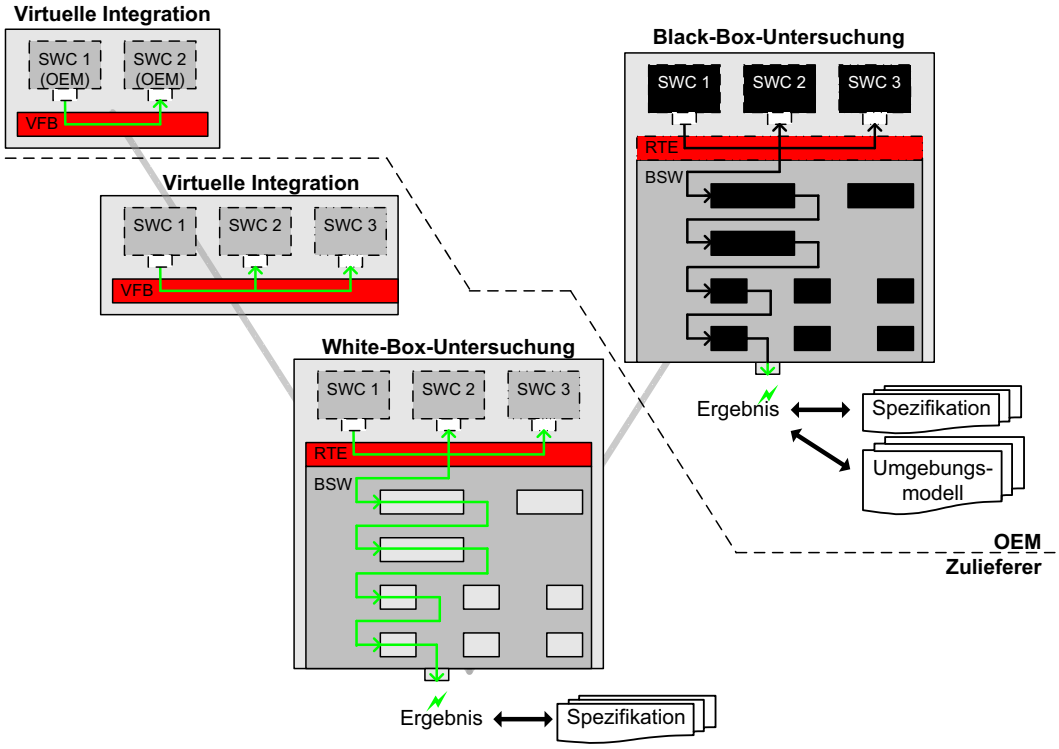


Abb. 4.3.: Die einzelnen Bewertungsmöglichkeiten innerhalb des Entwicklungsprozesses von Steuergeräten

weile auch auf statische Analyseverfahren zurückgegriffen (z.B. bei der Lenkungsentwicklung von BMW und Volkswagen [60] und [57]).

4.1.1. Simulation und Test von Steuergeräten

Simulation und Tests von Steuergeräten werden seit ca. 10 Jahren erfolgreich in der Praxis angewandt. Hierzu sind die verschiedensten Werkzeuge und Ansätze im Einsatz. Die Timing-Absicherung wird häufig über die Code-Instrumentierung auf der realen Hardware durchgeführt. Neuere Mikrocontroller bieten erweiterte Debugging-Möglichkeiten, um auch das Timing-Verhalten online zu überprüfen, ohne vorher zusätzliche Annotationen im Code durchführen zu müssen (z.B. über Nexus-Debugging-Schnittstelle [85]).

Für eine Timing-Simulation von Steuergeräten stellen verschiedene Hersteller passende Prozessormodelle zur Verfügung, dadurch ist die virtuelle Integration und anschließende HW/SW-Simulation des kompletten Steuergeräte-Codes möglich (z.B. in [112], [136], [55]). Über eine solche Simulation kann das Timing-Verhalten online überprüft

werden oder über die Aufzeichnung einer Loggingdatei nachgelagert ausgewertet werden. Ein Beispiel für eine solche virtuelle HW/SW-Simulation auf der Basis von SystemC ist in Abbildung 4.4 aufgezeigt.

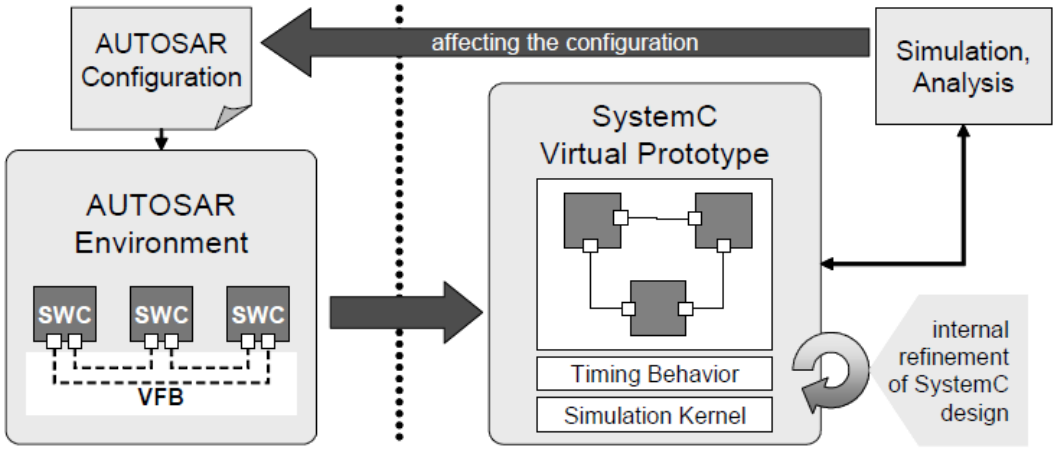


Abb. 4.4.: Abbildung von AUTOSAR-Komponenten auf einen virtuellen Prototyp in SystemC [67]

Durch die Einführung von AUTOSAR ist die virtuelle Integration und Simulation von Software-Komponenten (Funktionen) oberhalb der RTE möglich. Ein solcher Ansatz zur frühzeitigen Validierung von Software-Komponenten, wird in [78] vorgestellt. Unterstützt wird diese Art der virtuellen Integration von verschiedenen Werkzeugen, z.B. Matlab/Simulink von Mathworks [75] in Kombination mit SystemDesk von dSpace [28].

4.1.2. Statische Timing-Analysen von Steuergeräten

Die Anfänge der Scheduling-Theorie gehen auf die Arbeiten von C.L. Liu und J.W. Layland zurück, die Anno 1973 einen Scheduling-Algorithmus für harte Echtzeitsysteme vorstellten [73]. Aufbauend auf dieser Theorie haben verschiedene Forschungsgruppen sich mit dem Thema *Scheduling* beschäftigt und die Theorie stetig weiterentwickelt. In [121] und [7] wird ein Verfahren zur Analyse des unterbrechbaren *(preemptive)* Schedulings mit statischen vergebenen Prioritäten beschrieben. Diese Art von

Scheduling-Mechanismen kommen auch bei den automotive Betriebssystemen OSEK-OS und AUTOSAR-OS zum Einsatz (siehe Abschnitt 3.1.2 und 3.1.2).

Um die Antwortzeiten R_k der Tasks und Funktionen sowie die Auslastung U der CPU berechnen zu können, müssen die Ausführungszeiten C_k der relevanten Tasks und Funktionen bekannt sein. Diese können entweder über die in Abschnitt 4.1.1 beschriebene Instrumentierung des Programmcodes bestimmt werden oder aber es wird hierfür auch auf analytische Verfahren zurückgegriffen. Diese sogenannten statischen Verfahren analysieren den Maschinencode der zu untersuchenden Softwarefunktionen auf der Basis mathematischer Modelle. Hierfür ist die CPU sowie alle relevanten Peripheriekomponenten in einem Modell abgebildet und jede Operation ist mit konkreten Ausführungszeiten hinterlegt. Über eine Modellierung des Maschinencodes kann somit die Kostenfunktion des Kontrollflusspfades erstellt werden. Für jede mögliche Kombination von Registerbelegungen, Eingangswerten und Schleifengrenzen kann eine Ausführungszeit für eine Softwareroutine bestimmt werden. Die Maximierung der Kostenfunktion führt dann zu einer global gültigen maximalen Ausführungszeit der Routine. Im Werkzeug *aiT* der Firma AbsInt ist das statische Analyseverfahren umgesetzt. Abbildung 4.5 zeigt die einzelnen Schritte. Im ersten Schritt wird aus dem *Executable* der Kontrollfluss rekonstruiert. Anschließend wird eine Schleifentransformation durchgeführt (Umwandlung der Schleifen in rekursive Funktionen). Innerhalb der statischen Analyse erfolgt die Schleifanalyse, dabei werden die Schleifeniterationen und Rekursionen behandelt. Weiterhin erfolgt die Werte-Analyse, welche die Ziele der Speicherzugriffe berechnet sowie die Cache-Analyse, für die Berechnung der Cache-Treffer, um Speicherzugriffe vorherzusagen. Ferner wird das Pipelineverhalten vorausgesagt. Im nächsten Schritt erfolgt die Pfad-Analyse auf der Basis einer ganzzahligen linearen Funktion (Kostenfunktion). Weiterführende Arbeiten auf diesem Themengebiet wurden unter anderem von *Ringler* und *Montag* durchgeführt. Ringler zeigt einen Weg auf, wie auf der Basis von Funktionsmodellen eine modellbasierte WCET-Analyse möglich ist [102]. In den Arbeiten von Montag wird ein Weg für die Ermittlung von Annotationen für die WCET-Analyse aufgezeigt [81], [80].

Mit den ermittelten Ausführungszeiten lassen sich dann die Prozessorlast U anhand der Formel 2.12 bestimmen. Für die Berechnung der Antwortzeiten R_k kann mittels einer Scheduling-Analyse erfolgen. Diese ermittelt auf der Basis der Ausführungszeiten für jeden Task die Antwortzeit. Für die Berechnung gilt folgende Gleichung:

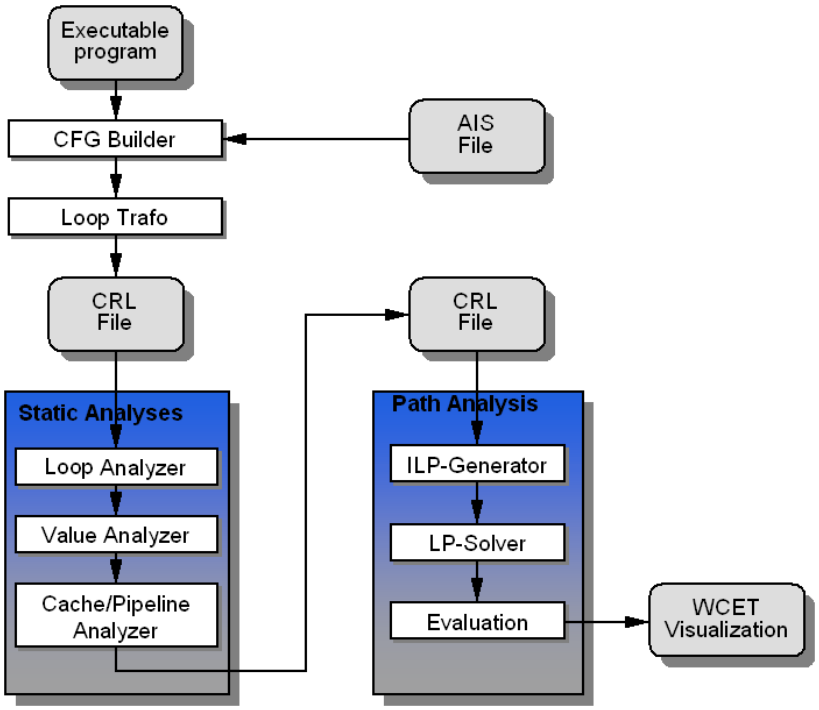


Abb. 4.5.: Beispiel für den Ablauf einer statischen Code-Analyse mit aiT von AbsInt [1]

$$R_k^{n+1} = C_k + T_{block} + \sum_{\forall j \in hp(k)} \left\lceil \frac{R_k^n}{T_j} C_j \right\rceil \text{ mit} \qquad [4.1]$$

i) T_{block} : Blockierzeit einer Task durch *Priority Ceilling* oder Semaphore

ii) $hp(i)$: Menge an Tasks mit höherer Priorität als Task w_i

Die Gleichung kann iterativ gelöst werden, Start mit $R_k^0 = 0$ und terminiert mit $R_k^{n+1} = R_k^n$. Es wurde nachgewiesen, dass $R_k^{n+1} \geq R_k^n$ gilt und die Gleichung konvergiert, wenn die Prozessorlast $\leq 100\%$ ist [7]. Bei einer Scheduling-Analyse werden sowohl die Scheduling Mechanismen (z.B. prioritätsbasiert wie bei OSEK) als auch die Aktivierungsmodelle (Ereignisse) der Tasks berücksichtigt. Diese notwendigen Modelle werden in Abschnitt 4.3.2 näher erläutert. Eine Scheduling-Analyse kann u.a. mit dem Werkzeug SymTA/S [116] durchgeführt werden, welches die notwendigen automotive Bibliotheken für OSEK und AUTOSAR zur Verfügung stellt.

Eine sehr gute Übersicht über die verschiedenen Scheduling-Mechanismen ist im Lehrbuch von G. Buttazzo zu finden [21].

4.2. Timing-Bewertung von Kommunikationssystemen

Eine Vorstellung der relevanten Bewertungskriterien für die Timing-Bewertung der Kommunikationssysteme erfolgte in Abschnitt 3.4. Der folgende Abschnitt stellt die gängigen Verfahren für Simulation und Test kurz vor. Weiterhin erfolgt die Diskussion der prinzipiellen Grundlagen, welche den formalen Methoden für die Bewertung von Kommunikationssystemen zu Grunde liegen.

4.2.1. Simulation und Test von Kommunikationssystemen

Für die Simulation und das Testen von Kommunikationssystemen gibt es eine Vielzahl an Werkzeugen auf dem Markt, die eine umfangreiche Auslegung und Absicherung ermöglichen. Über die sogenannten Restbussimulatoren wird das reale Verhalten der Kommunikationssysteme nachgebildet. Dabei besteht die Möglichkeit, dass ein Teil der Knoten eines Busses simuliert und der andere Teil als reale Systeme miteinander gekoppelt wird. Solche Werkzeuge werden unter anderem von den Firmen Vector-Informatik, IXXAT und Elektrobit angeboten [139], [56] und [32].

Weiterhin kann mittlerweile in den Modellierungswerkzeugen für die Funktionsentwicklung das Kommunikationsverhalten der Busse in einfacher Form nachgebildet werden (siehe z.B. in Simulink [75] oder SystemDesk [28]). Auch über entsprechende Modelle in SystemC ist die Simulation der Buskommunikation möglich [67], [68].

4.2.2. Statische Timing-Analysen von Kommunikationssystemen

Aufbauend auf den in Abschnitt 4.1.2 vorgestellten Analyseverfahren für Steuergeräte, wurden in den letzten Jahrzehnten auch im Bereich der Kommunikationssysteme die mathematischen Methoden für die Bewertung weiterentwickelt. Im Folgenden werden einige Verfahren für die automotive Kommunikationssysteme CAN und FlexRay vorgestellt.

Statische Timing-Analysen für CAN

In [123], [124] und [125] werden Verfahren diskutiert, die eine statische Bewertung des CAN-Busses ermöglichen. Diese Ansätze werden in [25] noch verfeinert und an einigen Stellen korrigiert. Für die statische Analyse eines CAN-Busses sind zusätzlich zu der

Übertragungszeit C_k einer Botschaft m_k, noch einige weitere Eigenschaften von zentraler Bedeutung (siehe Abschnitt 3.1.3), um eine vollständige und exakte Berechnung der maximalen Antwortzeiten R_k durchführen zu können. Diese setzen sich aus der Arbitrierungsphase, d.h. dem Warten auf den Buszugriff und der eigentlichen Übertragungszeit C_k zusammen (die Berechnung von C_k ist in Abschnitt 3.1.3 beschrieben).

Da eine einmal begonnene Übertragung nicht mehr unterbrochen bzw. abgebrochen werden kann, auch wenn eine höherpriore Botschaft zum Senden ansteht, ist dies bei der Berechnung der maximalen Antwortzeit einer Botschaft mit zu berücksichtigen. Für diese sogenannte *Sperrzeit* T_{block} gilt:

$$T_{block} = \max_{\forall j \in lp(m_k)} (C_j + T_{ifs}) \text{ mit } T_{ifs} = \frac{IFS}{V} \qquad [4.2]$$

Dabei ist $lp(m_k)$ die Menge an niederprioren Botschaften im Vergleich zu m_i. Mit T_{ifs} wird der Zwischenraum (*Interframe-Space (IFS)* mit $IFS = 3Bit$) zwischen zwei aufeinander folgenden Botschaften bezeichnet. V steht für die Übertragungsgeschwindigkeit des CAN-Busses.

Auf der Basis der Übertragungszeiten C_k und unter Berücksichtigung der Sperrzeit T_{block} lässt sich die maximale Antwortzeit einer Botschaft wie folgt berechnen [45]:

$$R_k = T_{block} + C_k + \sum_{\forall j \in hp(m_k)} (C_j + T_{ifs}) \cdot \left\lceil \frac{R_k}{T_j} \right\rceil \qquad [4.3]$$

Dabei gilt:

- $hp(m_k)$ ist die Menge an höherprioren Botschaften im Vergleich zu m_i.
- $C_j + T_{ifs}$ ist die belegte Zeit, welche von Botschaften mit höherer Priorität verursacht wird.
- mit $\left\lceil \frac{R_k}{T_j} \right\rceil$ wird die Anzahl der Wiederholungen angegeben, die in einem Zeitraum Δt auftreten können.

In Abbildung 4.6 ist der kritische Fall für die maximale Antwortzeit dargestellt. Die Botschaft *Frame_50ms* mit Priorität 5 kann im schlimmsten Fall durch eine Botschaft mit niederer Priorität (hier *Frame_100ms mit Priorität* 10) und durch die höherprioren

Botschaften *Frame_20ms* mit Priorität 2 und *Frame_10ms* mit Priorität 1 verzögert werden.

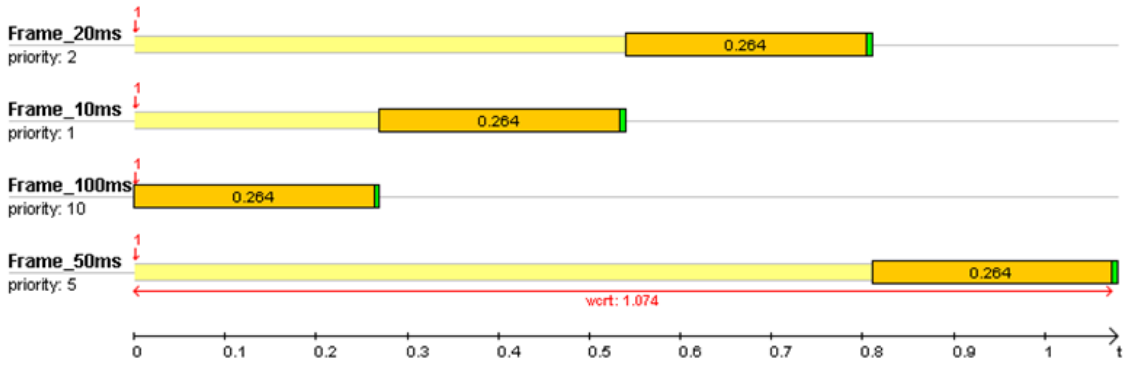


Abb. 4.6.: Beispiel für die Analyse eines CAN-Busses

Bei dem CAN-Bus, so wie dieser im automotive Bereich eingesetzt wird, handelt es sich um ein lokal synchrones und global asynchrones System. D.h. jedes einzelne Steuergerät hat einen festen Schedule, die gemeinsame Kommunikation über den Bus erfolgt jedoch ohne die Verwendung einer gemeinsamen Uhr. Dieses Verhalten ist für eine exakte Analyse des CAN-Busses von zentraler Bedeutung. Hierzu sind die Offsettabellen der Steuergeräte zu berücksichtigen. Diese legen die initiale Verteilung der zyklisch zu sendenden Botschaften eines Steuergerätes fest (siehe [138] und [10]). Über diese lokale Synchronisation zwischen den zyklischen Botschaften eines Steuergerätes können die maximalen Bursts bzw. demzufolge auch die maximalen Antwortzeiten reduziert und optimiert werden. Einen ersten Ansatz für die Berücksichtigung der Offsets bei der Scheduling-Analyse ist in [122] zu finden. Weitere Optimierungsansätze für CAN-Busse stehen in [86] und [43].

Statische Timing-Analysen für FlexRay

Das Kommunikationssystem FlexRay wurde in Abschnitt 3.1.3 eingeführt. Bei der statischen Timing-Analyse müssen das statische und das dynamische Segment von FlexRay unterschiedlich behandelt werden.

Im statischen Segment erfolgt der Buszugriff über das TDMA-Verfahren, d.h. jede Botschaft (Frame) kann immer nur zu einem bestimmten Zeitpunkt innerhalb des Zyklus gesendet werden. Eine Botschaft muss vor deren Sendeslot bereit zur Übertragung sein und folglich im Botschaftsspeicher (*Message RAM*) des FlexRay-Controllers abgelegt werden. Über den Message RAM wird dann die Botschaft zum Zeitpunkt des Sendeslots auf den Bus gelegt. Der vorherige Eintrag der Botschaft in den Botschaftsspeicher ist notwendig, um die Übertragung von deterministischen Daten sicherzustellen.

Aus diesem Verhalten heraus ergibt sich die Antwortzeit R_k einer Botschaft m_k, die im statischen Slot übertragen wird, wie folgt:

$$R_k = n \cdot T_{cycle} + C_k \qquad [4.4]$$

T_{cycle} steht dabei für die Dauer eines FlexRay-Zyklus und C_k für die eigentliche Übertragungszeit der Botschaft m_k (siehe auch Abschnitt 3.1.3). Mit n wird der *Repetition Factor* des *Slot-Multiplexings* berücksichtigt. Beispielsweise wird bei $n = 1$ die Botschaft in jedem Zyklus übertragen.

Die Berechnung der maximalen Antwortzeit für die Botschaften des dynamischen Segments ist aufwendiger. Der Buszugriff im dynamischen Segment findet prioritätsgesteuert über die Minislots statt. Je nach dem, ob nun eine Botschaft innerhalb des für sie reservierten Zeitschlitzes (Botschafts-ID = Minislot-Nummer) zum Senden ansteht, können die Sendezeitpunkte einer Botschaft variieren. Dadurch kann das Senden einer Botschaft mit hohen Minislots innerhalb des dynamischen Segments nach hinten verschoben werden bzw. nicht mehr innerhalb des aktuellen Zyklus versendet werden. Relevant hierfür ist der Parameter *pLatestTx* der den spätesten Zeitpunkt angibt, so dass die Botschaft innerhalb des dynamischen Segments noch vollständig übertragen wird. In [45] wird ein exaktes sowie ein approximatives Verfahren für die Analyse des dynamischen Segments vorgestellt. Ferner wird in [46] ein Ansatz zur Bewertung des dynamischen Segments diskutiert.

4.3. Timing-Bewertung auf Systemebene

Bei der Timing-Bewertung auf Systemebene werden nicht mehr einzelne Komponenten wie Steuergeräte oder Busse getrennt voneinander betrachtet, sondern es erfolgt die Untersuchung des Timing-Verhaltens im Verbund. Diese Art der Timing-Bewertungen von

Vernetzungsarchitekturen wird ausschließlich beim OEM durchgeführt. Im Anschluss daran erfolgt eine kurze Darstellung der simulativen Verfahren und der Testverfahren. Weiterhin wird für die statischen Analysen der holistische Ansatz sowie das modulare Bewertungsverfahren vorgestellt.

4.3.1. Simulation und Test auf Systemebene

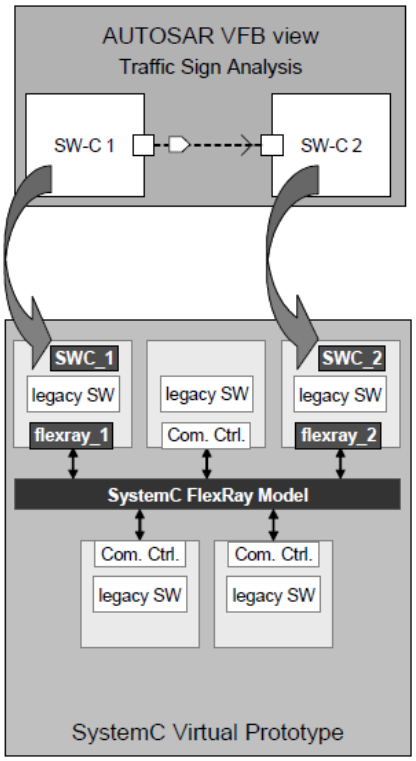
Die Simulations- und Test-Verfahren für die Bewertung auf Systemebene sind schon seit vielen Jahren in der Automobilindustrie erfolgreich im Einsatz. Die Anwendungsszenarien für die Verfahren sind breit gefächert:

- Restbussimulation: Simulation der Kommunikation und abstrahiertem Steuergeräteverhalten
- Komponentenprüfstand: Ein oder mehrere Steuergeräte liegen real vor und der Rest wird simuliert
- Brettaufbau: Alle Steuergeräte werden im Verbund getestet
- E-Fahrzeug: Test der Steuergeräte und der Buskommunikation im realen Fahrzeug

In den letzten Jahren wurden vermehrt Ansätze vorgestellt, welche die Simulation auf Systemebene ohne jede Art von Steuergeräte-Prototypen ermöglichen. Beispielsweise kann mit ChronSim der Firma Inchron der komplette Software-Code auf einem virtuellen Prozessormodell in Kombination mit der Buskommunikation simuliert werden [54].

In [67] beschreibt Krause et. al. einen Ansatz zur Simulation von verteilten Systemen unter der Verwendung von SystemC. Abbildung 4.7 zeigt exemplarisch ein solches SystemC-Modell mit zwei Steuergeräten, die über einen Bus miteinander verbunden sind.

Ein weiterer Ansatz für die Virtualisierung der Steuergeräte-Hardware wird in [26] von Delphi beschrieben. In diesem Beispiel wurde der Prozessorsimulator *Meteor* [136] von VaST mit der Restbussimulation CANoe [139] gekoppelt, um Tests über Systemgrenzen hinweg auf simulativer Basis durchführen zu können. Der prinzipielle Aufbau ist in Abbildung 4.8 aufgezeigt.




```
SC_MODULE (SWC_1) {
   sc_port<sc_flexray_if> s_port; ....
   SC_CTOR(SWC_1) {
      SC_THREAD(send_data); ....}
   void send_data() { ....
      s_port.write(sdata); ....}
};

SC_MODULE (SWC_2) {
   sc_port<sc_flexray_if> r_port; ....
   SC_CTOR(SWC_2) {
      SC_THREAD(receive_data); ....}
   void receive_data() { ....
      rdata = r_port->read(); ....}
};

int sc_main(int arc, char *argv[]) {
   sc_flexray_bus_model flexray_bus;
   sc_flexray_controller flexray_1;
   flexray_1.port_to_bus(flexray_bus);
   sc_flexray_controller flexray_2;
   flexray_2.port_to_bus(flexray_bus);
   SWC_1 SWC_1_instance;
   SWC_1_instance.s_port(flexray_1);
   SWC_2 SWC_2_instance;
   SWC_2_instance.r_port(flexray_2);
   .....
}
```

Abb. 4.7.: Beispiel für ein Gesamtsystemmodell in SystemC mit zwei Steuergeräten, die über einen Bus miteinander verbunden sind [67]

4.3.2. Statische Timing-Analysen auf Systemebene

Die statischen Timing-Analysen auf Systemebene lassen sich in zwei Klassen einteilen. Die holistischen Verfahren und die modularen Verfahren. Eine Beschreibung und Einordnung der beiden Klassen wurde bereits am Anfang dieses Kapitels gegeben. Im Folgenden wird der holistische Ansatz sowie drei Beispiele für die modulare Bewertung vorgestellt.

Holistische Analyse

Um eine holistische Analyse von verteilten Systemen zur ermöglichen, wurden schon viele Ansätze vorgeschlagen, die eine Erweiterung der klassischen Analysetechniken (siehe 4.1.2 und 4.2.2) auf diese aufzeigen. Die Herausforderung bei dieser Art der Analyse besteht darin, alle Komponenten innerhalb eines mathematischen Modells beschreiben zu können. Eine der Schwierigkeiten ist, dass sich die Modelle meist nicht genera-

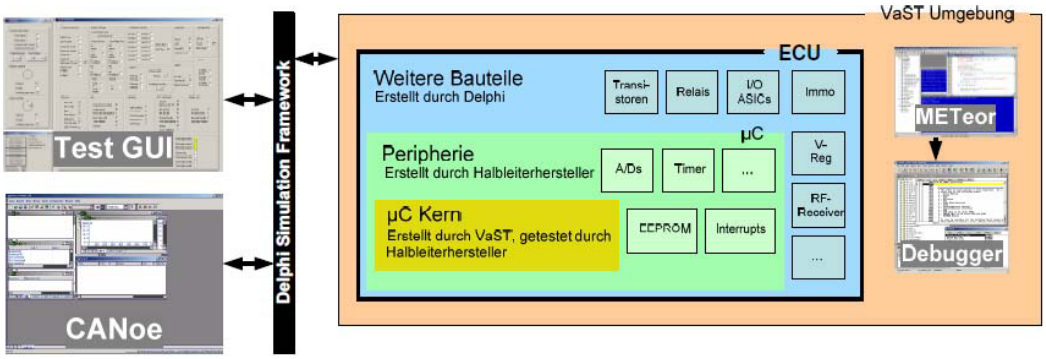


Abb. 4.8.: Prinzipieller Aufbau des kombinierten Simulators von Delphi [26]

lisieren lassen und somit für jedes neue Ereignismodell und Kommunikationsprotokoll eine neue Umsetzung, d.h. ein neues oder erweitertes formales Modell, zu erarbeiten ist. Im Folgenden sind einige Beispiele für holistische Ansätze aufgeführt:

- An an der Universität von Cantabria (Spanien) wurde ein Framework unter dem Namen *Modeling and Analysis Suite for Real-Time Applications (MAST)* entwickelt, welches einige der holistischen Analyse-Verfahren beinhaltet. Der prinzipielle Aufbau von MAST ist in Abbildung 4.9 aufgezeigt. [27]. Über einen graphischen Editor und über eine spezifische Beschreibungssprache können Systeme inklusive deren Echtzeiteigenschaften beschrieben werden.

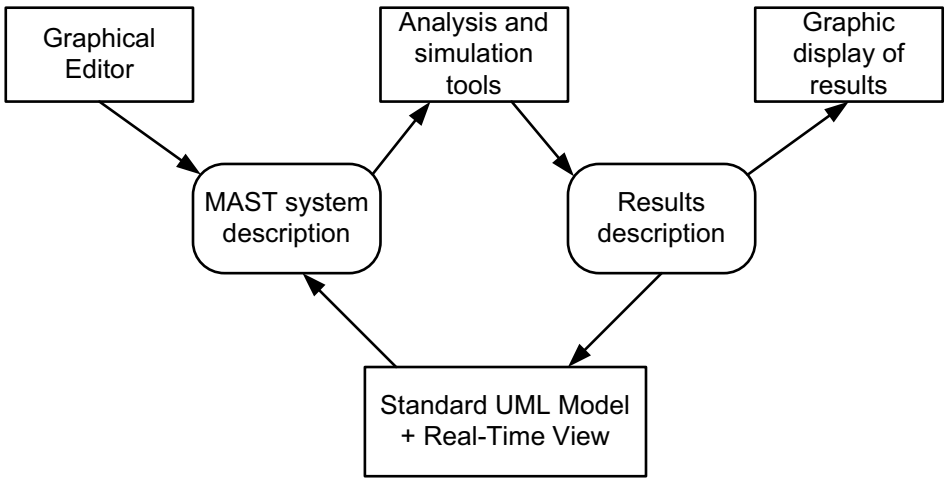


Abb. 4.9.: Aufbau des Frameworks der Modellierung und Analyse Suite MAST [27]

- In [126] beschreiben Tindell et. al. die Kombination des Schedulings mit festen Prioritäten auf einem Prozessor und eines TDMA-basierten Kommunikationssystems. Es werden dabei sowohl die Latenzzeiten des Protokollstacks berücksichtigt als auch die resultierenden Antwortzeiten der zu übertragenden Botschaften. Anhand eines Beispiels (drei Prozessoren an einem Bus) wird die Anwendung der vorgestellten Verfahren diskutiert.

- Ferner existiert ein weiterer Beitrag zur holistischen Timing-Analyse von Pop et. al. [90]. Es erfolgt die Diskussion der Möglichkeiten zur Untersuchung von Systemen mit Kontroll- und Datenabhängigkeiten anhand eines umfangreichen Beispiels. Dabei wird ein Prozessorsystem mit asynchroner (*Ereignis-getriggert*) und synchroner (*Zeit-getriggert*) Eigenschaften untersucht sowie ein daran angekoppeltes Kommunikationssystem.

Kompositionelle Analyse

Die kompositionelle Analyse ist ein Analyseverfahren, welches an der Universität Braunschweig unter der Leitung von Prof. Ernst entwickelt wurde. Das Verfahren hat den Namen *Symbolic Timing-Analysis* kurz *SymTA/S*. SymTA/S ist ein modulares Analyse-Verfahren auf System-Ebene für die Bewertung von heterogenen *System on Chips (SoCs)* und verteilten eingebetteten Systemen [50] [96]. Im Gegensatz zu den holistischen Verfahren wird ein Ansatz vorgeschlagen, die klassischen Analyse-Verfahren auf Teilumfänge von verteilten Systemen abzubilden, liegt dem Verfahren von SymTA/S ein modularer Ansatz zu Grunde. Die Kernidee besteht darin, die einzelnen Komponenten eines verteilten Systems mit den klassischen Analysetechniken (siehe Abschnitt 4.1.2 und 4.2.2) zu untersuchen. Die lokal berechneten Ergebnisse werden dann über die definierten Schnittstellen den anderen Komponenten zur Verfügung gestellt. Abbildung 4.10 zeigt die prinzipielle Vorgehensweise des SymTA/S-Ansatzes. Im ersten Schritt werden die Konfigurationsdaten und die Informationen über das Umgebungsmodell (z.B. Ereignisse von Außen) den einzelnen Komponenten übergeben. Auf der Basis dieser Daten erfolgt eine lokale Analyse, um die Timing-Eigenschaften der Komponenten zu erhalten sowie deren Ausgangsereignismodelle zu generieren. Bei einer erfolgreichen lokalen Analyse, d.h. wenn die Konfiguration schedulbar ist, werden die resultierenden Ausgangsereignismodelle im darauffolgenden Schritt auf die Eingangsereignismodelle der

Komponenten abgebildet. Dieser Vorgang wird so lange wiederholt bis die Ereignisse durch das gesamte Systemmodell propagiert wurden und diese konvergieren.

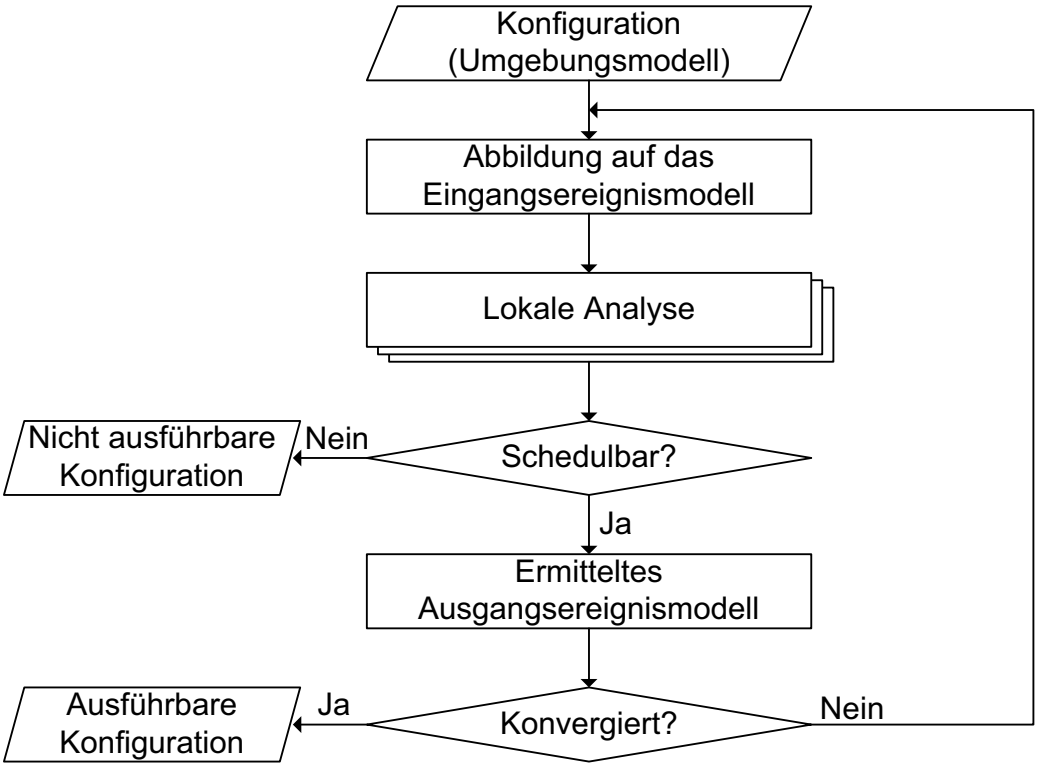


Abb. 4.10.: Prinzipielle Funktionsweise des SymTA/S-Ansatzes

Für die Koppelung der einzelnen Komponenten kommen zwei Schnittstellen zum Einsatz. Die *Event Model Interfaces (EMIFs)* und die *Event Adaption Functions (EAFs)* [99]. Die EMIFs ändern nicht die Timing-Eigenschaften eines resultierenden Ereignisstroms, sondern transformieren nur dessen mathematische Beschreibung. Abbildung 4.11 zeigt die möglichen Transformationen auf.

Die EAFs sind notwendig, wenn keine direkte Transformation im EMIF möglich ist. In einem solchen Falle werden die Eigenschaften des Ereignisstroms so angepasst, dass diese dem geforderten Eingangsmodell entsprechen. Abbildung 4.12 zeigt eine solche Transformation unter der Verwendung einer EAF auf.

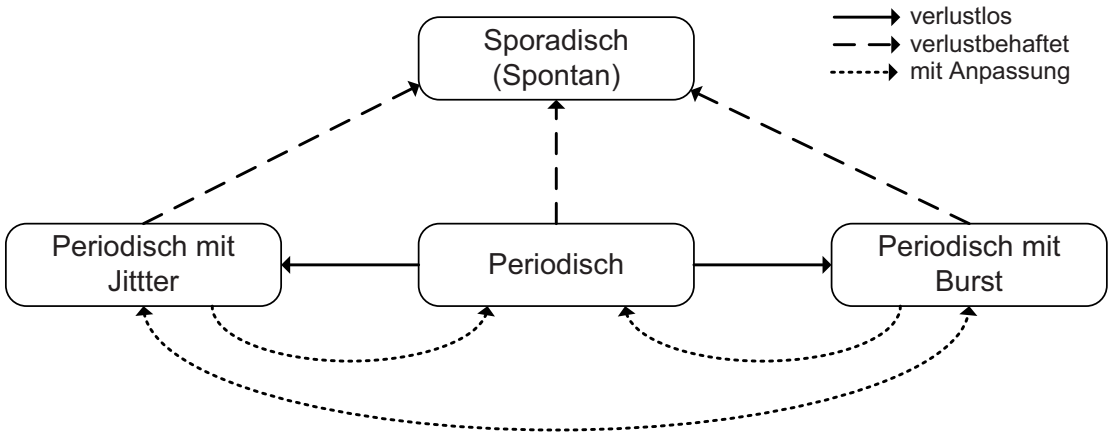


Abb. 4.11.: Ereignismodell Schnittstellen des SymTA/S-Ansatzes

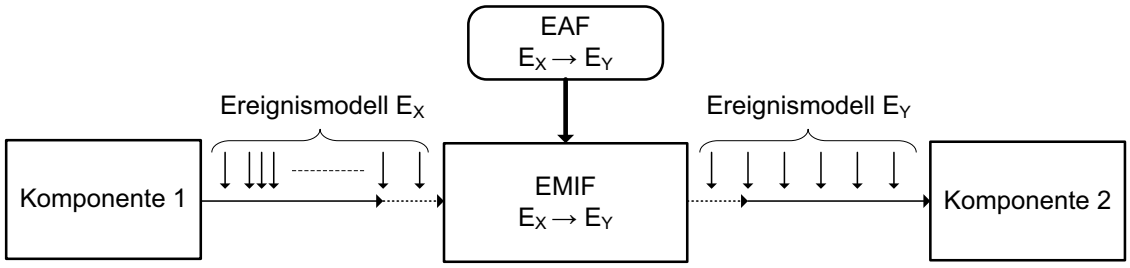


Abb. 4.12.: Beispiel für ein *EMIF* mit dazugehöriger Anpassungsfunktion *EAF*

Die zugrunde liegenden Ereignismodelle des SymTA/S-Ansatzes sind in [96] detailliert beschrieben. Die folgende Tabelle 4.1 zeigt einen Überblick über die Modelle und gibt eine kurze Erklärung über die notwendigen Parameter und Bedingungen.[1].

Weiterführende Arbeiten im Kontext des SymTA/S-Ansatzes beschäftigen sich mit der Hierarchisierung von Ereignisströmen sowie mit Datenabhängigkeiten (Kontext-Sensitivität) [105], [59]. Ferner wird die Timing-Optimierung und die Sensitivitätsanalyse für eingebettete Echtzeitsysteme in [91] und [92] diskutiert. Der SymTA/S-Ansatz wurde mittlerweile in einem Werkzeug gleichen Namens von der Firma Symtavision umgesetzt [116]. Dieses Werkzeug kombiniert dabei die bereits in Abschnitt 4.1.2 und Abschnitt 4.2.2 erwähnten Analysen für OSEK, AUTOSAR OS, CAN und FlexRay.

[1] In dieser Arbeit werden *sporadisch*, *aperiodisch* und *spontan* synonym verwendet

Tab. 4.1.: Standard Ereignis Modelle des SymTA/S-Ansatzes

Modell	**Kurzform**	**Parameter**	**Bedingungen**
Periodisch	P	$< T >$	keine
Periodisch mit Jitter	P+J	$< T, J >$	$J < T$
Periodisch mit Jitter und Burst	P+B	$< T, J, T_{min} >$	$J \geq T \vee T_{min} > T - J$
Sporadisch	A	$< T >$	keine
Sporadisch mit Jitter	A+J	$< T, J >$	$J < T$
Sporadisch mit Jitter und Burst	A+B	$< T, J, T_{min} >$	$J \geq T \vee T_{min} > T - J$

Realtime Calculus

Der *Real-Time Calculus (RTC)* ist ein Werkzeug für die Timing-Analyse von verteilten, eingebetteten Systemen [119]. Der RTC wurde an der Eidgenössischen-Technischen-Hochschule-Zürich entwickelt und basiert auf dem *Network Calculus* [70], einer Theorie über deterministische Warteschlangen für Kommunikationsnetzwerke und der Max-Plus/Min-Plus-Algebra [18]. Mit dem RTC können Netzwerke und deren Komponenten anhand von Ereignisströmen analysiert werden [89].

Als Beschreibungsmodell für das Verfahren dienen die sogenannten *Ankunfts- und Servicekurven (engl. Arrival- and Service-Curves)*. Die Ankunftskurve beschreibt die Anzahl an Ereignissen, welche z.B. eine Task aktivieren. Ein Ereignisstrom wird dabei von zwei Ankunftskurven repräsentiert: Der minimalen α^l und der maximalen Ankunftskurve α^u. Die Ankunftskurven können entweder automatisch generiert werden, sofern die notwendigen Informationen vorliegen, oder aber aus Messungen (via Traces) rekonstruiert werden. In Abbildung 4.13 ist ein Beispiel für einen Ereignisstrom und die entsprechenden Ankunftskurven dargestellt.

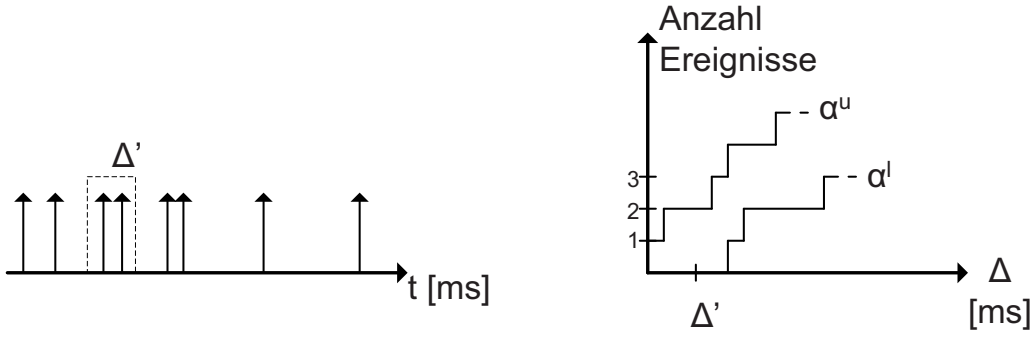


Abb. 4.13.: Beispiel für einen Ereignisstrom und die entsprechenden Ankunftskurven α^l und α^u

Die Ankuftskurven für jedes mögliche Zeitintervall I der Länge Δ sind wie folgt definiert:

$$\begin{aligned} &\alpha^l(t-s) \leq R[s,t) \leq \alpha^u(t-s)\ \forall s < t \text{ mit} \\ &\text{i) } R[s,t) : \text{ Anzahl der Ereignisse im Intervall } [s,t) \\ &\text{ii) } \alpha^l(t-s), \alpha^u(t-s) \in \mathbb{R}^+ \end{aligned} \qquad [4.5]$$

Mit der Servicekurve wird die Verfügbarkeit einer Ressource beschrieben, d.h. die Rechenzeit, die einer Task in einem gewissen Zeitabschnitt Δ zur Verfügung steht. Hierbei wird auch die minimale β^l und die maximale Verfügbarkeit (Service) β^u angegeben. In Abbildung 4.14 ist die Verfügbarkeit einer Ressource und die entsprechenden minimalen und maximalen Servicekurven abgebildet.

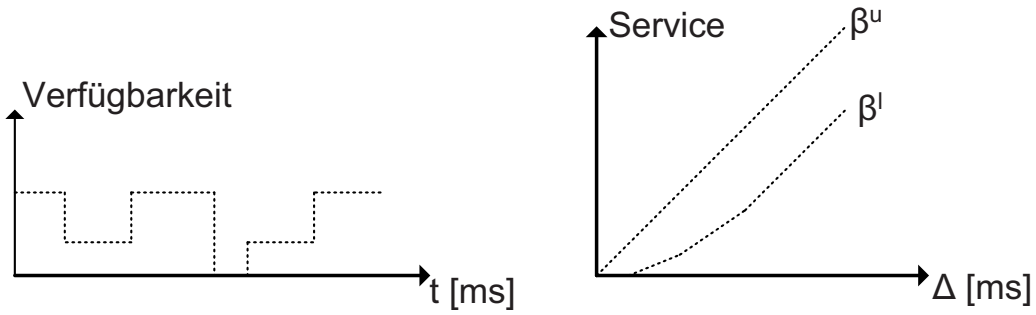


Abb. 4.14.: Beispiel für die Verfügbarkeit einer Ressource und die entsprechenden Servicekurven β^l und β^u

Der Service steht für die entsprechende Einheit, zum Beispiel für die Rechenzeit einer CPU (in Taktzyklen) oder die Anzahl an Bytes, die über einen Bus übertragen werden können. Die Servicekurve ist wie folgt definiert:

$$\begin{aligned} &\beta^l(t-s) \leq C[s,t) \leq \beta^u(t-s)\ \forall s < t \text{ mit} \\ &\text{i) } C[s,t) : \text{Anzahl an freien Ressourcen im Intervall } [s,t) \\ &\text{ii) } \beta^l(t-s), \beta^u(t-s) \in \mathbb{R}^+ \end{aligned} \quad [4.6]$$

Über die sogenannte *Greedy Processing Component* werden Ressourcen modelliert, die einen Ereignisstrom verarbeiten. In Abbildung 4.15 ist eine solche Komponente dargestellt. Der Eingangsereignisstrom ist über die Ankuftskurven α^l und α^u modelliert und triggert die Komponente. Diese werden am Eingang der Komponente in einem FIFO-Puffer zwischengespeichert. Die Verarbeitung erfolgt auf die *Greedy*-Art und wird durch die Verfügbarkeit der Ressource begrenzt. Diese ist durch die Servicekurven β^l und β^u beschrieben. Am Ausgang der Komponente wird der resultierende Ereignisstrom über die Ankuftskurven $\alpha^{l'}$ und $\alpha^{u'}$ ausgegeben. Die restliche Verfügbarkeit der Ressource ist über die Ausgangsservicekurven $\beta^{l'}$ und $\beta^{u'}$ angegeben.

Die Transformation der Ankunfts- und der Servicekurven, vom Eingang zum Ausgang an einer Komponente, sind über die folgenden Gleichungen abgebildet [22], Details dazu sind in [70] und [18] zu finden:

$$\alpha^{l'}(\Delta) = min\{\inf_{0\leq\mu\leq\Delta}\{\sup_{\lambda>0}\{\alpha^l(\mu+\lambda)-\beta^u(\lambda)\}+\beta^l(\Delta-\mu)\}, \beta^l(\Delta)\} \quad [4.7]$$

$$\alpha^{u'}(\Delta) = min\{\sup_{\lambda>0}\{\inf_{0\leq\mu<\Delta+\lambda}\{\alpha^u(\mu)+\beta^u(\lambda+\Delta-\mu)\}-\beta^l(\lambda)\}, \beta^u(\Delta)\} \quad [4.8]$$

$$\beta^{l'}(\Delta) = \sup_{0\leq\lambda\leq\Delta}\{\beta^l(\lambda)-\alpha^u(\lambda)\} \quad [4.9]$$

$$\beta^{u'}(\Delta) = \min\{\inf_{\lambda>\Delta}\{\beta^u(\lambda)-\alpha^l(\lambda)\}, 0\} \quad [4.10]$$

Durch den modularen Ansatz sind mit dem RTC große Systeme analysierbar, dabei wird ein System über die Beschreibung der einzelnen Ressourcen anhand der Greedy-Komponente zu einem Gesamtmodell, zu einem analysierbaren Gesamtmodell zusammengefügt. In Abbildung 4.15 ist noch ein Beispiel für die Verknüpfung von zwei Tasks zu einem Gesamtmodell (hier: *Fixed-Priority Scheduling*) abgebildet.

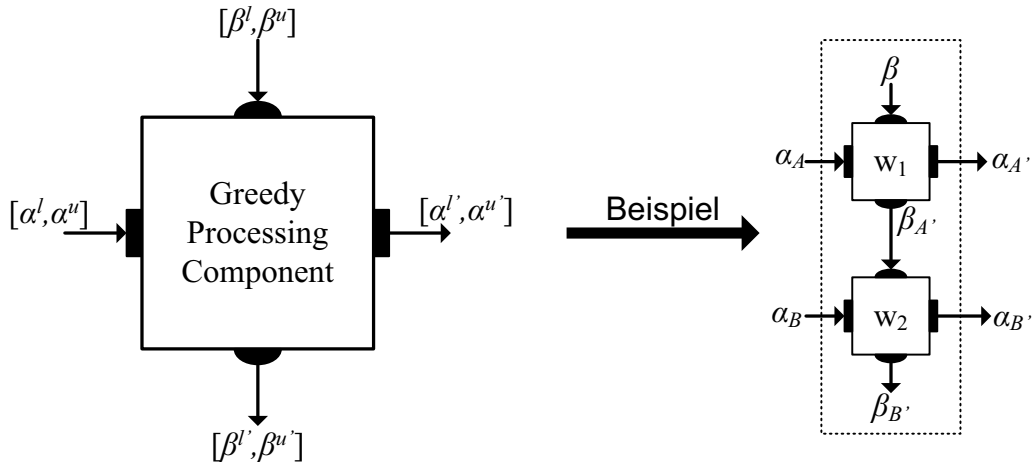


Abb. 4.15.: *Greedy Processing Component* des RTC und ein Beispiel für ein *Fixed Priority* Modell mit zwei Tasks w_1 und w_2 auf einer Ressource

Erste Modellierungen mit dem RTC von automotive Systemen wurden für die CAN-Kommunikation in [23] aufgezeigt. Ferner erfolgte die Diskussion einer ersten Analyse eines FlexRay-Netzwerks in [46].

Erweitertes Hierarchisches Ereignisstrommodell

Das *Erweiterte Hierarchische Ereignisstrommodell, engl. Advanced Hierarchical Event-Stream Model* wurde an der Universität Ulm am Lehrstuhl von Professor Slomka entwickelt. Das Verfahren verknüpft das Standard Ereignis Modell aus Abschnitt 4.3.2 und die Ereignisbeschreibung der Ankunftskurven aus Abschnitt 4.3.2. Mittels dieses Ansatzes können sowohl diskrete Ereignisse, als auch kontinuierlich auftretende Ereignisse innerhalb eines Modells beschrieben werden. Weiterhin ist die kontinuierliche Ereignis-

funktion des Real-Time Calculus über eine Approximation, mit frei wählbarer Exaktheit, modellierbar.

Als Beschreibungsgrundlage dient das von Gresser eingeführte Ereignisstrommodell [44]. Dieses ermöglicht eine verallgemeinerte Beschreibung von Ereignissen. Ein Ereignisstrom Θ enthält eine Anzahl an Ereignissen $e = (T, T_{off})$, T ist die Periode und T_{off} der Offset. Anhand einer Begrenzungsfunktion Φ kann die maximale Anzahl an Ereignissen, die innerhalb eines Intervalls I auftreten können, beschrieben werden (siehe [2] und [5]).

$$\Phi(I,\Theta) = \sum \left\lfloor \frac{\Delta I - T_{off,e}}{T_e} + 1 \right\rfloor \text{ mit } \Delta I > T_{off,e} \text{ und } e \in \Theta \qquad [4.11]$$

Weiterhin wird für die Beschreibung eine Anforderungsfunktion Ψ benötigt, welche einer Task w_i einen Ereignisstrom Φ zuordnet:

$$\begin{aligned} &\Psi(\Delta I, W) = \sum_{\forall w_i \in W} \Phi(\Delta I - d_i, \Theta_i) C_i \text{ mit} \\ &W: \text{ Menge an Task} \\ &d_i: \text{ Deadline der Task } w_i \end{aligned} \qquad [4.12]$$

Über die Anforderungsfunktion Ψ kann die *Schedulability* überprüft werden. Ein Task w_i ist schedulbar solange die Anforderungsfunktion innerhalb eines jeden Intervalls ΔI kleiner oder gleich der verfügbaren Kapazität $F(\Delta I)$ des Systems ist [3]. Diese Anforderungsfunktion kann aufgrund deren Linearität wie folgt approximiert werden, dabei sei $\Delta I_{e,k} = d_i + T_{off,e} + kT$:

$$\Psi(\Delta I, e_i, w_i, k) = \begin{cases} \Psi(\Delta I, e) + \dfrac{C_i}{T_e}(\Delta I - \Delta I_{e,k}) & \text{mit } \Delta I > \Delta I_{e,k} \\ \Psi(\Delta I, e) & \text{mit } \Delta I \leq \Delta I_{e,k} \end{cases} \qquad [4.13]$$

Eine solche Funktion ist in Abbildung 4.16 dargestellt. Für die ersten k Ereignisse erfolgt eine exakte Berechnung. Ab $\Delta I_{e,k}$ werden die Ereignisse approximiert.

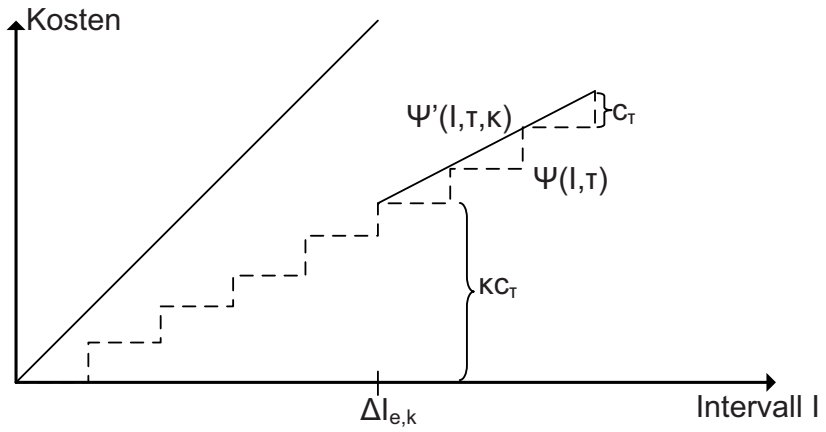


Abb. 4.16.: Beispiel für ein approximiertes Element eines Ereignisstroms [4]

Weiterhin kann das Modell auf hierarchische Ereignisströme erweitert werden. Dabei enthält ein hierarchischer Ereignisstrom $\widehat{\Theta} = \{\widehat{e}\}$ eine Menge an hierarchischen Ereignissen $\widehat{e}$. Jedes dieser Elemente beschreibt eine periodische Abfolge von Ereignissen und ist wie folgt definiert:

$$\widehat{e} = (T, T_{off}, \eta, \Gamma, \widehat{\Theta}) \qquad [4.14]$$

Mit η wird die maximale Anzahl an Aktivierungen innerhalb einer Periode T annotiert. Der Gradient Γ und der hierarchische Ereignisstrom $\widehat{\Theta}$ beschreiben das Muster, in dem die Ereignisse auftreten. Auf ähnliche Weise wird auch die hierarchische Anforderungsfunktion $\widehat{\Psi}$ abgebildet.

Der große Vorteil der Methodik liegt darin begründet, dass sowohl periodische als auch beliebige Ereignisse zusammen in einem Modell exakt beschreibbar sind. Durch die zusätzliche Möglichkeit der Approximation ist eine effiziente Analyse möglich, dabei kann je nach Anforderungen der Grad der Exaktheit für die Berechnung vorgegeben werden. Das Verfahren steht seit kurzem in dem kommerziellen Werkzeug *ChronVal* der Firma Inchron zur Verfügung [55].

Am Lehrstuhl von Prof. Slomka an der Universität Ulm wird aktuell noch ein weiteres Verfahren zur approximativen Timing-Analyse entwickelt. Dieses basiert auch auf dem Ereignisstrommodell von Gresser [44]. Im Vergleich zu dem ersten Verfahren wird beim

zweiten Verfahren nicht der Real-Time Calculus verwendet, sondern auf die klassischen Analysetechniken zurückgegriffen (z.B. [127]).

Weitere Arbeiten

Im Umfeld der Timing-Bewertungen gibt es noch weitere Arbeiten, die im Folgenden kurz vorgestellt werden. Dabei richtet sich der Hauptfokus auf Ansätze, welche für die Bewertung von Fragestellungen im automotive Umfeld von Interesse sind.

An den Universitäten Uppsala (Schweden) und Aalborg (Dänemark) wurde der sogenannte *Uppaal* Ansatz entwickelt, welcher die Möglichkeit der Modellierung, Simulation und Verifikation von Echtzeitsystemen auf der Basis von zeitgesteuerten Automaten (*Timed Automata*) bietet [134], [49]. Die Modellierung der Systeme erfolgt über nicht-deterministische Prozesse mit endlichen Kontrollstrukturen und realen Uhrenwerten. Die Kommunikation wird über Kanäle oder gemeinsame Variablen modelliert. Uppaal besteht aus drei Teilen: Einer Beschreibungssprache, einem Simulator und einem Model-Checker. Durch die Möglichkeit der exakten Abbildung von Datenabhängigkeiten können Systeme in Uppaal korrekt nachmodelliert werden. Der Einsatz von Uppaal für große Systeme ist allerdings kritisch zu bewerten, da die Vielzahl an Systemzuständen, welche in einem automotive System eintreten zu berücksichtigen sind und sehr schnell zu einer *State Space Explosion* führen (siehe hierzu auch [118]).

Im *TIMMO*-Projekt (Timing Model), welches von der Europäischen Union gefördert wird, geht es um die Definition einer Timing-Beschreibungssprache *TADL (Timing Augmented Description Language)* sowie um die Entwicklung einer Bewertungsmethodik für E/E-Systeme im Kraftfahrzeug. In diesem Zusammenhang werden auch das Zusammenspiel und der Austausch von Timing-Informationen zwischen OEM und Zulieferer näher beleuchtet.

In einem weiteren von der Europäischen Union geförderten Projekt mit dem Namen *INTEREST* wird an der Standardisierung der Austauschformate zwischen den Modellierungs- und Bewertungswerkzeugen gearbeitet.

Weiterhin wird das Timing-Thema im Kontext von AUTOSAR stetig vorangetrieben. In der AUTOSAR-Timing-Gruppe wird u.a. der Ausbau der AUTOSAR-Beschreibungen um Timing-Attribute erweitert [15]. Wichtige Arbeiten entstanden unter anderem bei der BMW Car IT. Hier wird beispielsweise an der Entwicklung einer gemeinsa-

men Entwicklungsplattform *AUTOSAR Tool Platform (ARTOP)* für AUTOSAR-basierte Systeme gearbeitet [48]. Ferner wurden Ansätze für die Berücksichtigung des Timing-Verhaltens von AUTOSAR-basierten Systemen aufgezeigt [108], [107].

Der Beitrag von Reichelt et. al. diskutiert die Anforderungen und Auswirkungen hinsichtlich des Timing-Verhaltens bei der Entwicklung von FlexRay-basierten Systemen [95]. Ferrari et. al. beschreibt in [36] die Einflüsse von AUTOSAR auf das Zeitverhalten und die Speicheranforderungen.

4.4. Einordnung und Abgrenzung

In den vorangegangenen Abschnitten wurde der aktuelle Stand der Technik für die Verfahren vorgestellt, welche eine Timing-Bewertung ermöglichen. Viele der Verfahren wurden bisher nur anhand kleinerer Beispiele aus dem Automotive-Umfeld evaluiert (z.B. [46], [50]). Eine vollständige Integration der Verfahren in den Entwicklungsprozess von E/E-Architekturen im Kraftfahrzeug wurde bisher nicht aufgezeigt. Aus diesem Grund ist eines der Ziele dieser Arbeit eine Methodik für die durchgängige Integration der Timing-Bewertungsverfahren in den existierenden Entwicklungsprozess für E/E-Architekturen im Kraftfahrzeug vorzustellen. Mittels der Timing-Bewertungsverfahren können die in Abschnitt 3.4 beschriebenen Kenngrößen untersucht werden.

In allen bisherigen Projekten und Arbeiten stand immer die Beschreibung des Timing-Verhaltens im Vordergrund, die konkrete Modellierung wurde jedoch nur oberflächlich diskutiert. Um die Genauigkeit der Bewertungsverfahren zu steigern und die Überschätzung bei den analytischen Verfahren zu minimieren, werden in dieser Arbeit Regeln abgeleitet, welche eine vollständige Modellierung des Timing-Verhaltens von automotive Vernetzungsarchitekturen und Gateway-Systemen ermöglichen (Kapitel 6). Ferner werden die hierfür notwendigen Timing-Attribute identifiziert (Kapitel 2 und Kapitel 5).

Verschiedene Arbeiten haben sich in den letzten Jahren mit der Optimierung der Offsets für den CAN-Bus beschäftigt. *Grenier et. al.* beschreiben einen Algorithmus zur Offsetoptimierung in [43]. Dieser wurde von *Schillp* noch verfeinert (siehe [109]). Weiterhin bietet die Firma Symtavision ein Optimierungsverfahren für Offsets an [116]. Diese Verfahren führen die Optimierung immer für jeweils einen CAN-Bus aus. Weiterhin werden je nach Verfahren nicht alle notwendigen Parameter bei der Berechnung mit einbezogen, obwohl diese für eine korrekte Berechnung wichtig sind. Ein Beispiel

ist die Periode des COM-Tasks der Steuergeräte. Da die Offsets direkt vom ComTask abhängen, können nur Offsets mit einem Vielfachen der Periode des COM-Tasks T_{com} vergeben werden. In der vorliegenden Arbeit wird ein Algorithmus vorgestellt, welcher erstens alle relevanten Parameter bei der Vergabe der Offsets berücksichtigt und zweitens die Offsets global berechnet, d.h. für alle CAN-Busse einer Vernetzungsarchitektur. Dadurch ist es möglich, zusätzlich zur Minimierung der Antwortzeiten der Botschaften auf den einzelnen CAN-Bussen, die Routing-Last der Gateways besser zu verteilen.

Für die automatische Generierung der Annotationen für die WCET-Analyse wurden von *Ringler* und *Montag et. al.* erste Konzepte vorgestellt [102], [80] und [81]. Diese schlagen u.a. vor, mittels formaler Code-Analyse (z.B. Polyspace der Firma Mathworks [75]) die Annotationen zu ermitteln. Durch die Einführung der AUTOSAR-Systembeschreibungen besteht nun die Möglichkeit einen großen Teil der Annotationen aus diesen automatisiert zu extrahieren. Hierfür wird in dieser Arbeit ein Konzept vorgestellt. Weiterhin wird für das SymTA/S-Verfahren ein Konzept für die vollständige Modellierung von AUTOSAR-basierten Gateway-Steuergeräten entwickelt.

Der Fokus für die Integration der Timing-Bewertungsverfahren in den E/E-Entwicklungsprozess liegt dabei auf den formalen Analyseverfahren (Kapitel 7). Diese kamen bisher für die Bewertung von Vernetzungsarchitekturen und Gateway-Systeme nicht zum Einsatz. Bei dem Entwurf und der Entwicklung zukünftiger Vernetzungsarchitekturen und Gateway-Systeme wird die sichere Bestimmung der Zeitanforderungen immer wichtiger. Gründe hierfür sind die Zunahme an verteilten Funktionen und die steigende Anzahl an sicherheitskritischen Systemen. Die meisten der beschriebenen Ansätze und Methoden sind allgemein gültig und können auch bei Simulation und Tests angewendet werden.

5. Verfahren zur Extraktion von Timing-Informationen

In Kapitel 2 wurden die notwendigen Informationen beschrieben, die für eine aussagekräftige Timing-Bewertung notwendig sind. Insbesondere für die Bewertung von CAN-Bussen sowie für die Untersuchung von Ende-zu-Ende-Latenzzeiten über Busgrenzen hinweg sind Informationen über das Timing-Verhalten von Steuergeräten von zentraler Bedeutung. Durch die in Kapitel 3 beschriebenen verteilten Verantwortlichkeiten zwischen OEM und Zulieferern stehen dem OEM nicht immer alle Details über die einzelnen Systeme einer Vernetzungsarchitektur zur Verfügung. Weiterhin müssen bei reaktiven Systemen realistische Szenarien zu Grunde gelegt werden, um eine Über- bzw. Unterschätzung zu vermeiden. Als Abhilfe für die Unvollständigkeit an Timing-Informationen wird im Folgenden ein Verfahren beschrieben, welches die Extraktion dieser Informationen auf der Basis von Messungen, den sogenannten *Loggingdaten*, der Buskommunikation ermöglicht. Die Loggingdaten können z.B. an existierenden Brettaufbauten oder Fahrzeugen aufgezeichnet werden. Die Extraktion von Timing-Informationen bietet mehrere Vorteile:

1. Die Extraktion liefert eine detaillierte Übersicht über das Timing-Verhalten von aktuellen und sich in der Entwicklung befindlichen Vernetzungsarchitekturen.

2. Die gewonnenen Daten können für eine Verfeinerung der bisherigen Annahmen, insbesondere in der frühen Entwicklungsphase, für die Timing-Bewertung verwendet werden.

3. Durch die gewonnenen Erfahrungen sind für zukünftige Vernetzungsarchitekturen verfeinerte Timing-Anforderungen ableitbar, die in die Lastenhefte einfließen können.

4. Weiterhin sind diese Daten als erweiterte Grundlage für eine verbesserte Testabdeckung in der Integrationsphase verwendbar.

Zusätzlich zu den Attributen, die direkt aus den Spezifikationen abgeleitet werden können, sind folgende Informationen für eine aussagekräftige Timing-Bewertung notwendig: Jitter, Drift und Offsettabellen der Steuergeräte, Jitter der einzelnen Botschaften sowie dynamische Aktivierungen von spontanen Botschaften. Weiterhin sind verschiedene Modi, in denen sich die Kommunikation eines Fahrzeuges befinden kann, zu berücksichtigen. Diese Betriebsmodi sind zu großen Teilen aus Spezifikationen ableitbar. Insbesondere während des Fahrbetriebes treten jedoch verschiedenste dynamische Lastsituationen auf, die in bisherigen Spezifikationen nicht oder nur unzureichend beschrieben sind bzw. sich generell nicht formal beschreiben lassen, da zu viele unsichere Faktoren eine Rolle spielen (z.B. durch die Interaktionen des Fahrers und der Passagiere).

Abbildung 5.1 zeigt exemplarisch einige solcher Betriebsmodi und Betriebszustände die während der Entwicklung, im Betrieb oder in der Werkstatt auftreten können. In Abschnitt 5.1.3 wird ein Verfahren beschrieben, welches aus Fahrzeugmessungen typische Lastsituationen der CAN-Busse extrahiert. Für die Umsetzung wurde teilweise auf *Data Mining-Methoden* zurückgegriffen, wie sie z.B. in [74] zu finden sind.

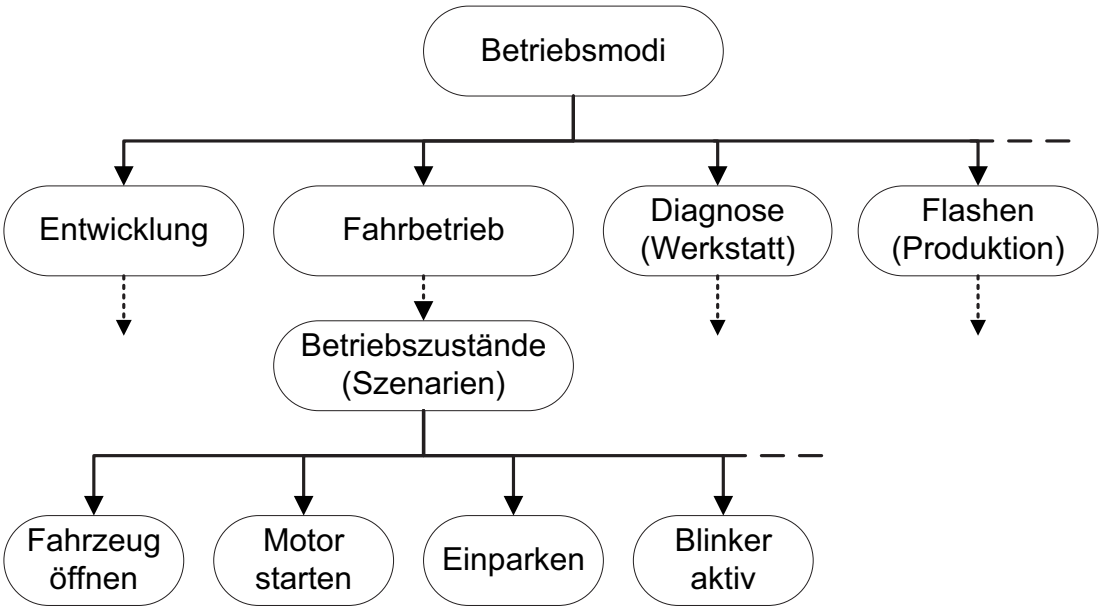


Abb. 5.1.: Verschiedene Betriebsmodi und -zustände eines Fahrzeuges

5.1. Extraktionsverfahren

Die Grundlage für die Extraktion von Timing-Informationen ist ein Loggingdatensatz, der über eine Messung am Komponenten-Teststand *(Hardware-in-the-loop, HiL)* oder direkt im Fahrzeug aufgezeichnet werden kann. Ein solcher Loggingdatensatz Y enthält Tupels y von Botschaften m_k und deren Auftrittszeitpunkt t_y. M_Y ist dabei die Menge der Botschaften, die in Y auftraten:

$$Y = \{y = (m_k, t_y) | m_k \in M_Y, t_{Y,start} \leq t_y \leq t_{Y,end}\} \quad [5.1]$$

Der Startzeitpunkt des Loggingdatensatzes wird mit $t_{Y,start}$ und dessen Ende mit $t_{Y,end}$ annotiert. Das Tupel $y = (m_k, t_y)$ beschreibt genau einen bestimmten Auftrittszeitpunkt t_y von m_k.

Die einzelnen Schritte für die Datenextraktion sind in Abbildung 5.2 dargestellt. Ein Loggingdatensatz sowie die Kommunikationsmatrizen und weitere Informationen aus der Spezifikation dienen als Eingangsdaten für die Datenextraktion. Über eine Importschnittstelle werden die Informationen miteinander kombiniert, d.h. die Angaben aus der Spezifikation werden mit den Loggingdaten fusioniert. Über den sogenannten *Calculator* werden die allgemeinen Timing-Attribute extrahiert (z.B. Jitter, Drift, Offsettabellen). Die Generierung der Szenarien ist im linken Pfad aufgezeigt. Nach einem Filterungsschritt werden die Botschaften des Loggingdatensatzes anhand deren Sendeverhalten in zwei Mengen aufgeteilt: Die M_{static} und $M_{dynamic}$, mit $M_{static}, M_{dynamic} \subseteq M$. Die Menge M_{static} enthält alle Botschaften, die rein zyklisches Auftrittsverhalten haben. Die Menge $_{dynamic}$ umfasst alle Botschaften, die nicht streng periodisch innerhalb des Loggingdatensatzes auftraten. Auf der Basis der dynamischen Botschaften erfolgt die Extraktion der Szenarien über den *Szenario Extraktor*. In der Export Schnittstelle werden die Szenarien mit den Botschaften der Menge M_{static} sowie den allgemeinen Timing-Attributen zusammengefasst und stehen für eine Timing-Bewertung zur Verfügung. Die allgemeinen Timing-Attribute können auch separat für die Verfeinerung von Timing-Modellen verwendet werden.

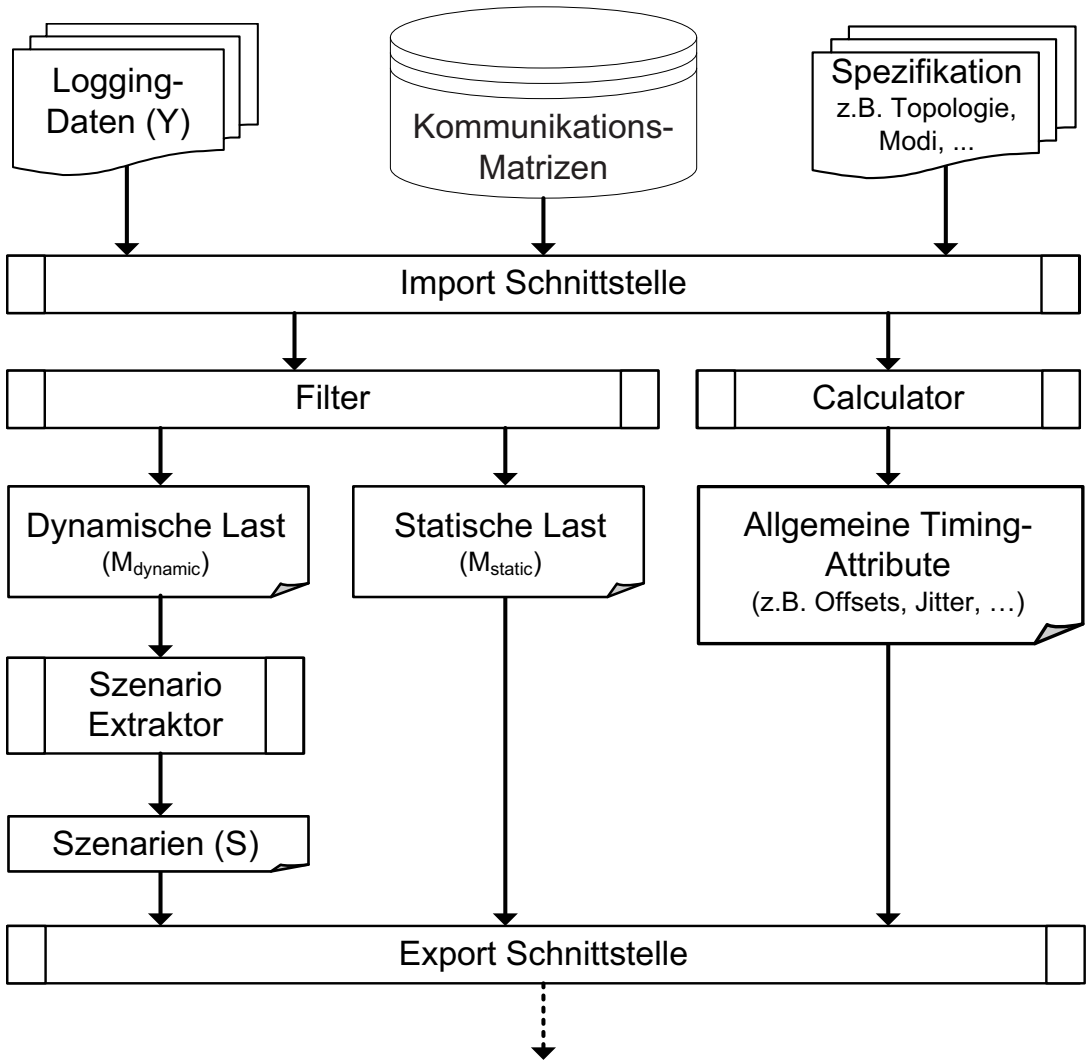


Abb. 5.2.: Übersicht über die einzelnen Schritte, die für die Extraktion von Timing-Informationen notwendig sind [132]

5.1.1. Sende- und Übertragungsjitter

Für den Jitter sind zwei verschiedene Werte von Interesse. Erstens der Sendejitter J_{send} eines Steuergerätes, dieser entsteht durch Steuergeräte-interne Schedulingeffekte (siehe Abbildung 5.3). Der Sendejitter wird durch das Jittern des Sendetasks *(COM-Task)* verursacht und ist somit für alle Botschaften eines Steuergerätes gleich. Zweitens der Übertragungsjitter $J_{trans,k}$, der durch die Busarbitrierung entsteht (siehe Abbildung 5.4). Dieser ist bei jeder Botschaft unterschiedlich.

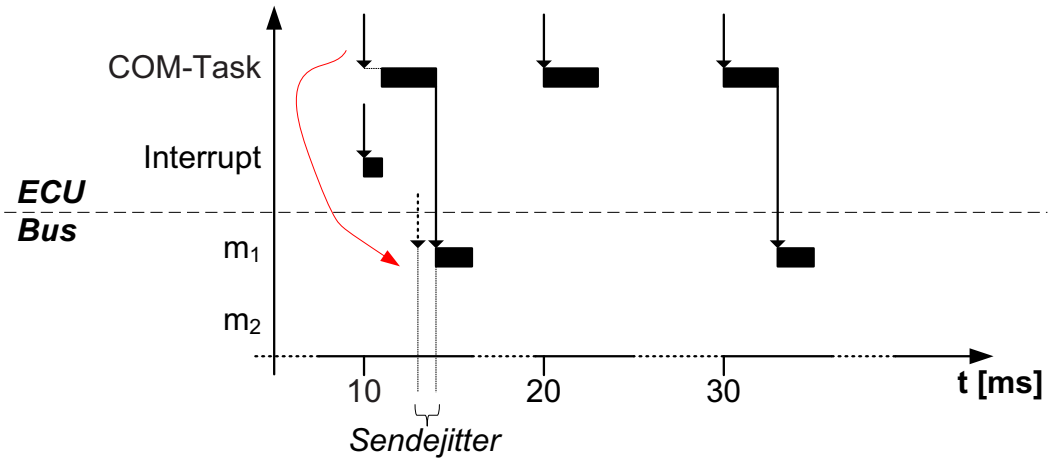


Abb. 5.3.: Beispiel für den Sendejitter J_{send}

.

Der Sendejitter J_{send} ist über die Beobachtung der periodischen Botschaften $m_k \in M_{cyc}$ aus einem Loggingdatensatz Y extrahierbar. M_{cyc} beinhaltet alle Botschaften eines Loggingdatensatzes Y. Dabei werden nur die Sendezeitpunkte $t_{send,k}$ einer Botschaft für die Extraktion herangezogen, bei denen keine andere Botschaft mit höherer Priorität den Bus belegt, d.h. bei $t_{send,k} : hp(m_k) = \emptyset$.

Der Übertragungsjitter $J_{trans,k}$ lässt sich über den Empfangszeitpunkt einer Botschaft gewinnen. Der Übertragungsjitter beinhaltet auch den Sendejitter.

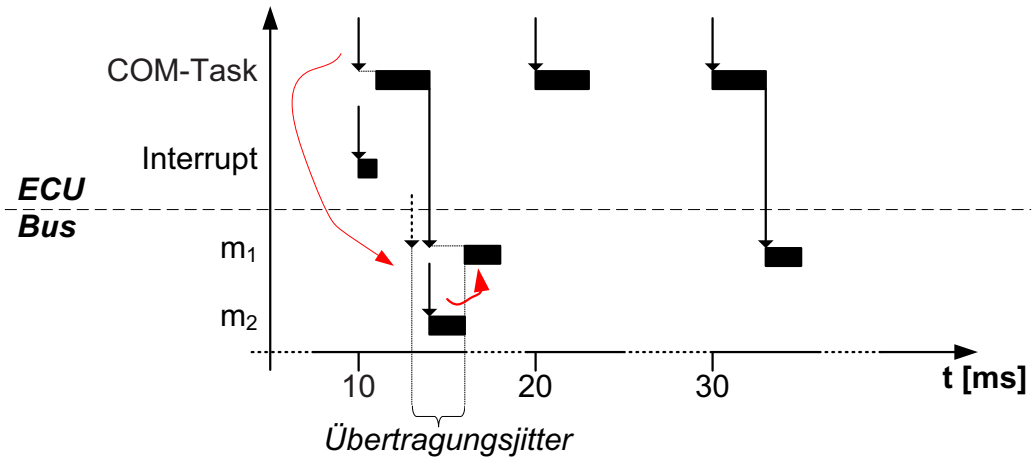


Abb. 5.4.: Beispiel für den Übertragungsjitter $J_{trans,k}$

.

5.1.2. Offsettabellen der Steuergeräte

Die Offsettabellen der Steuergeräte spielen bei der Bewertung von CAN-Bussen eine zentrale Rolle (siehe Abschnitt 4.2). Diese Informationen werden aktuell getrennt für jedes Steuergerät vergeben. Die Vergabe der Offsets wird in den meisten Fällen beim Zulieferer durchgeführt. Dem OEM stehen die Werte oft jedoch nicht zur Verfügung. Um eine exakte Timing-Bewertung aktueller Vernetzungsarchitekturen durchführen zu können, wurde in dieser Arbeit ein Algorithmus entwickelt, welcher aus einem Loggingdatensatz Y die Offsettabellen der Steuergeräte rekonstruiert.

Der Algorithmus (siehe Algorithmus 1) benötigt folgende Eingangsdaten: 1. Einen Loggingdatensatz Y, 2. Die Spezifikation K des CAN-Busses (K-Matrix) und 3. Die Anzahl an Samples $NbOfSamples \in \mathbb{N}$, welche für die Extraktion herangezogen werden sollen. Über die Anzahl der Samples wird angegeben, wie oft die Offsetwerte innerhalb des Loggingdatensatzes ermittelt werden. Je höher die Anzahl der Samples desto exakter wird die Berechnung. 4. Die Zykluszeiten $ComTaskCycle$ der *ComTasks* der Steuergeräte. Als Ausgangsdaten liefert der Algorithmus die Offsets $t_{off,m}$ der einzelnen Botschaften m.

Im ersten Schritt des Algorithmus wird über die Funktion `GetEcus` eine Liste *ecuList* aus der K-Matrix K erzeugt. Im Folgenden wird über alle Elemente der *ecuList* iteriert. Zu Beginn wird innerhalb der Schleife eine Liste *msgList* erzeugt. Diese enthält alle Botschaften der aktuellen *ecu*, für die potentiell Offsets vergeben werden können (siehe hierzu [138] und [10]). Über die Funktion `Init` in Zeile X wird Liste $tempOffset$ mit 0 initialisiert. Im nächsten Schritt wird aus der Liste *msgList* die Botschaft mit der höchsten Priorität ermittelt und der Variablen *hpId* übergeben (Zeile 8 bis Zeile 13). Anschließend wird mit der Extraktion der Offsets begonnen (Zeile 14 bis Zeile 26) und abhängig von der vorgegebenen Anzahl an Samples $NbOfSamples$ mehrmals durchlaufen. Zuerst wird der Zeitpunkt sp des Auftritts der Botschaft mit der höchsten Priorität im Loggingdatensatz Y abhängig von der aktuellen Samplezahl über die Funktion `GetFirst` ermittelt. Beispielsweise wird bei $NbOfSamples = 2$ im ersten Durchlauf der erste Auftritt der höchstprioren Botschaft verwendet und im zweiten Durchlauf entsprechend der zweite Auftritt. Der Offset der höchstprioren Botschaft wird auf Null gesetzt (Zeile 17). Innerhalb der Schleife (Zeile 19 bis Zeile 25) erfolgt die Berechnung der Offsets für alle Botschaften $m \in msgList$, mit Ausnahme der höchstprioren. Innerhalb der Funktion

`CalcOffset` wird der Auftrittszeitpunkt der aktuell gewählten Botschaft m nach dem Startzeitpunkt sp im Loggingdatensatz Y ermittelt und entsprechend der Offset berechnet. In Zeile 27 bis 32 werden die finalen Offsets berechnet. Hierzu werden für jede Botschaft der Liste *msgList* die aufsummierten Offsetwerte durch die Anzahl der Samples $NbOfSamples$ geteilt. Über die Funktion `Correct` werden Jittereffekte eliminiert und auf die vorgegebene Schrittweite $ComTaskCycle$ angepasst. Anschließend werden die ermittelten Offsets ausgegeben.

In Abbildung 5.5 ist ein Vergleich zwischen den original vergebenen Offsets (generiert) der Botschaften eines Steuergerätes und den extrahierten Werten auf der Basis eines Loggingdatensatzes. Die Schrittweite beträgt $ComTaskCycle = 10ms$. Als Anzahl an Samples wurden $NbOfSamples = 10$ verwendet. Die generierten Offsets sind in *blau* dargestellt. Die extrahierten Werte sind in *gelb* ohne Verwendung der Funktion `Correct` aufgetragen und in *rot* mit den korrigierten Werten. Die Ergebnisse zeigen deutlich, dass der Algorithmus sehr exakt die originalen Offsets rekonstruiert. Nur in einem Fall bei der Botschaft mit dem Identifier 907 wurde der Wert nicht korrekt rekonstruiert und um eine Schrittweite ($10ms$) verfehlt. Eine solche Abweichung entsteht, wenn eine Botschaft innerhalb des Loggingdatensatzes aufgrund von Bursts sehr stark jittert. Durch die Erhöhung der Anzahl an Samples können solche Abweichungen eliminiert werden.

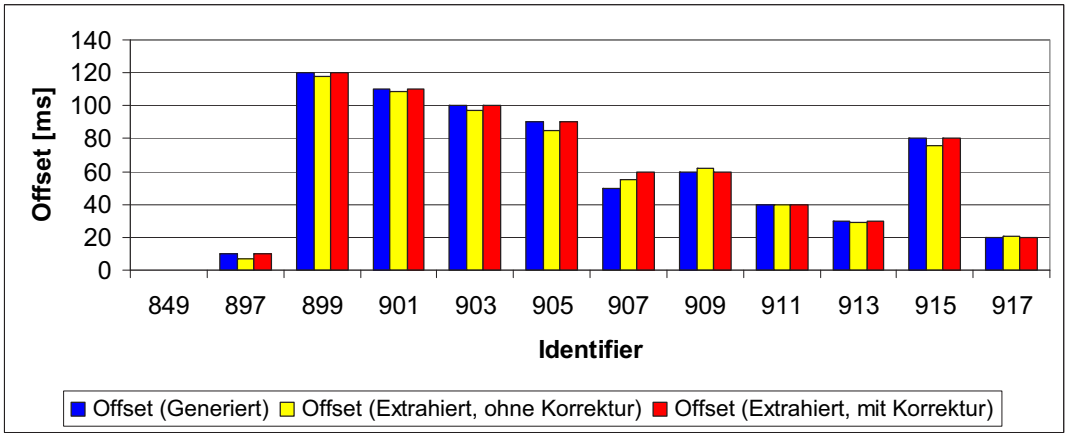


Abb. 5.5.: Vergleich zwischen den generierten Offsets eines Steuergerätes und den extrahierten Offsets (mit und ohne Korrektur)

Algorithm 5.1: Extraktion der Offsettabellen von Steuergeräten eines CAN-Busses

input : Y, K, $NbOfSamples$, $ComTaskCycle$

output: $t_{off,m}$

```
1  ecuList ← GetEcus (K) //Get all ECUs of the CAN-Bus;
2  forall the ecu ∈ ecuList do
3      msgList ← GetMsg (K, ecu);
4      Init (tempOffset, 0) ;
5      //Search m ∈ msgList with highest priority;
6      hpId ← 2047  //Here lowest priority of Basic CAN;
7      forall the m ∈ msgList do
8          if hpId > GetId (m) then
9              hpId ← GetId (m);
10         end
11     end
12     //Iterate over all samples;
13     for t ← 1 to NbOfSamples, t++ do
14         //Search first occurence of hpId in Y and set startpoint sp;
15         sp ← GetFirst (hpId, Y, t);
16         tempOffset[hpId] ← 0;
17         //Calculate offset for all other messages of the ecu;
18         forall the m ∈ msgList do
19             if GetId (m) ≠ hpId then
20                 temp ← CalcOffset (Y, sp, m);
21                 tempOffset(GetId (m)) ← tempOffset(GetId (m)) + temp;
22             end
23         end
24     end
25     //Generate final offsets;
26     forall the m ∈ msgList do
27         temp ← tempOffset(GetId (m))/NbOfSamples;
28         temp ← Correct (temp, ComTaskCycle) ;
29         t_off,m ← temp;
30     end
31 end
```

5.1.3. Extraktion von Betriebsszenarien

Für die Extraktion von Betriebsszenarien sind mehrere Schritte notwendig. Ausgehend von dem Import eines Loggingdatensatzes Y und der dazugehörigen Kommunikationsmatrizen sowie weiterer Informationen (z.B. Netzwerkspezifikation, Expertenwissen, etc.) können auf der Basis dieser Daten Betriebsszenarien extrahiert werden. In Abbildung 5.2 sind im linken Teil des Ablaufdiagramms die notwendigen Schritte dargestellt. Die aus dem Import ermittelten Botschaften $m_k \in M_Y$ werden über einen *Filter* in zwei Untermengen aufgeteilt:

1. Die Botschaften, welche streng periodisch innerhalb des Loggingdatensatzes Y auftreten, werden der Menge $M_{static} \subseteq M_Y$ übergeben.

2. Die Menge $M_{dynamic} \subseteq M_Y$ enthält alle Botschaften, die nicht streng periodisch im gesamten Loggingdatensatz Y auftreten.

Auf der Basis der Menge $M_{dynamic}$ extrahiert der *Szenario-Extraktor* die Betriebsszenarien. Die generierten Betriebsszenarien sind Untermengen $M_i \in M_{dynamic}$. Im Anschluss an die Generierung erfolgt die Ergänzung der Betriebsszenarien mit der statischen Grundlast resultierend aus M_{static}. Weiterhin werden die allgemeinen Timing-Informationen (siehe die Unterabschnitte 5.1.2 und 5.1.1) übergeben. Basierend auf diesen Daten kann dann eine Timing-Bewertung durchgeführt werden.

Filterung der Loggingdaten

Die Aufteilung der Botschaften in die beiden Mengen M_{static} und $M_{dynamic}$ erfolgt im Filterungsschritt. Dabei werden die Botschaften anhand des Sendetyps (siehe Abschnitt 3.1.3) und des Auftrittsverhaltens innerhalb eines Loggingdatensatzes klassifiziert. Exemplarisch werden die Daimler-Sendetypen verwendet. Prinzipiell können auch andere Sendetypen, unter Berücksichtigung deren zeitlichen Verhaltens, verwendet werden In Abbildung 5.6 ist die Aufteilung der Botschaften in $M_{dynamic}$ detailliert dargestellt. $M_{dynamic}$ wird in weitere Untermengen M_{dual}, M_{csx} und $M_{spontan}$ abhängig vom Sendetyp der Botschaft aufgeteilt ($M_{dynamic} = M_{dual} \cup M_{csx} \cup M_{spontan}$):

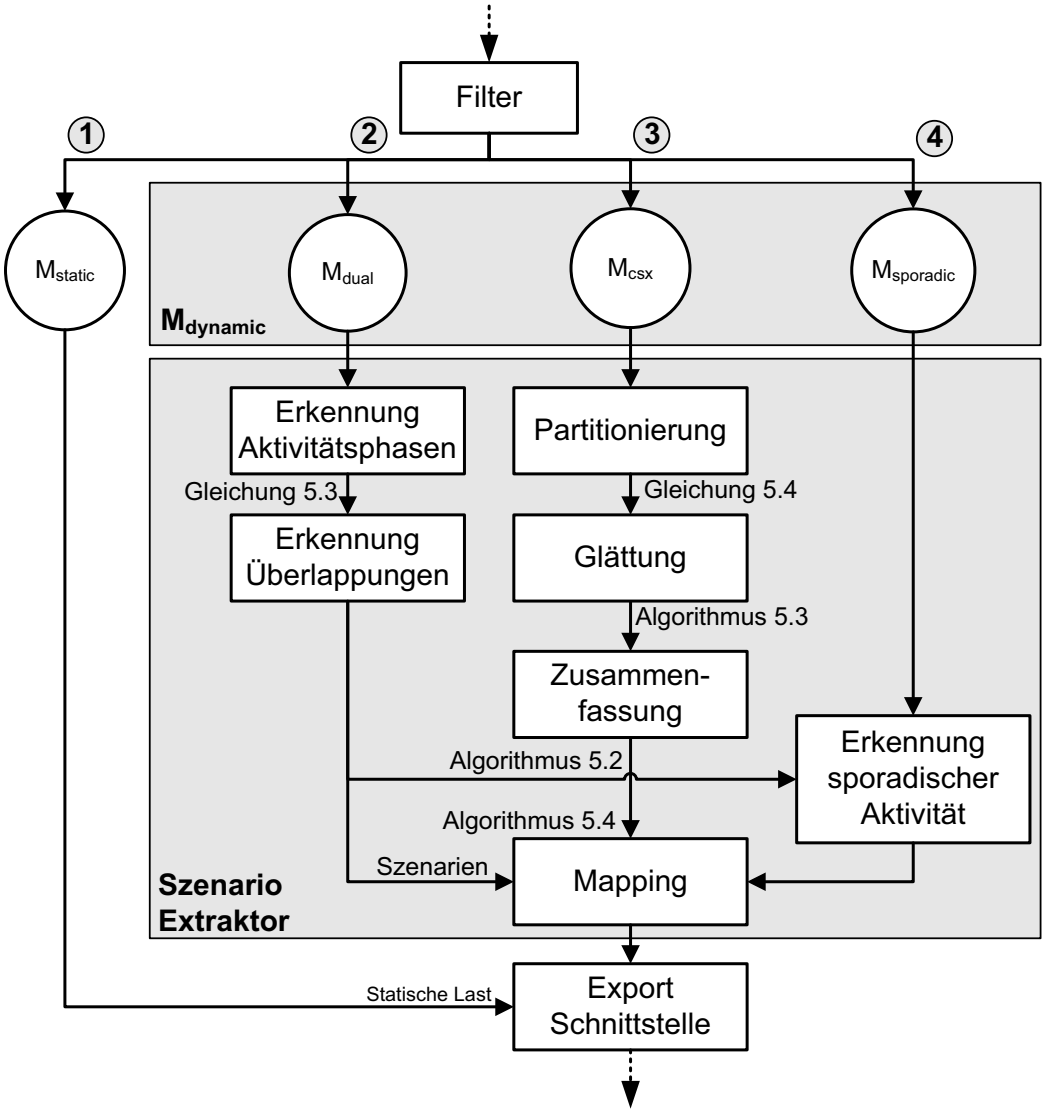


Abb. 5.6.: Die einzelnen Schritte des *Szenario Extraktors*, inklusive der darin eingesetzten Algorithmen

1. *cyclicX:* Eine Botschaft mit diesem Sendetyp wird immer zu M_{static} hinzugefügt.

2. *spontanX:* Botschaften dieses Sendetyps werden der Menge M_{static} übergeben, wenn diese innerhalb des Loggingdatensatzes durchgehend periodisch auftreten, andernfalls fallen sie der Menge $M_{spontan}$ zu.

3. *cyclicIfActiveX:* Ist eine Botschaft von diesem Sendetyp und wird diese im kompletten Loggingdatensatz ausschließlich zyklisch übertragen, erfolgt deren Zuwei-

sung zu der Menge M_{static}. Tritt die Botschaft nur teilweise zyklisch auf, wird diese der Menge M_{dual} übergeben.

4. *cyclicAndSpontanWithDelay:* Botschaften dieses Sendetyps werden bei rein zyklischem Auftreten der Menge M_{static} übergeben, andernfalls der Menge M_{csx}.

5. *cyclicIfActiveFast:* Wird eine Botschaft des Sendetyps *cyclicIfActiveFast* im kompletten Loggingdatensatz immer nur mit einer Periode übertragen, erfolgt die Übergabe an die Menge M_{static}. Ist die zweite (schnelle) Periode auch im Loggingdatensatz vorhanden, wird die Botschaft der Menge M_{dual} übergeben.

6. *cyclicWithRepeatOnDemand:*Botschaften vom Sendetyp *cyclicWithRepeatOnDemand* werden bei rein zyklischem Auftreten M_{static} übergeben, ansonsten der Menge M_{csx}.

Wie in Abbildung 5.6 zu sehen ist, existieren insgesamt vier Mengen an Botschaften nach dem Filterungsschritt (M_{static}, M_{dual}, M_{csx} und $M_{spontan}$). Die Generierung der Betriebsszenarien erfolgt auf der Basis der Menge $M_{dynamic}$. Die Botschaften der Menge M_{static} werden nach der Extraktion den einzelnen Szenarien wieder hinzugefügt.

Erkennung der Betriebsszenarien

Beginnend mit dem zweiten Pfad von links in Abbildung 5.6, werden auf Basis der Menge M_{dual} die Betriebsszenarien in zwei Schritten angelegt. Die Botschaften der Menge M_{dual} haben bei Aktivität zyklisches Auftrittsverhalten, es kann dabei mehrere sogenannter *Aktivitätsintervalle* innerhalb eines Loggingdatensatzes geben. Jedes dieser Aktivitätsintervalle $i_{k,x}$ hat einen festen Startzeitpunkt $t_{x,start}$ und einen Endzeitpunkt $t_{x,end}$. Beim Durchgehen eines kompletten Loggingdatensatzes ist das Ergebnis eine Menge an Intervallen I_{dual} mit $i_{k,x} \in I_{dual}$:

$$I_{dual} = \{i_{k,x} = [t_{x,start}, t_{x,end}] | t_{Y,start} \leq t_{x,start} < t_{x,end} \leq t_{Y,end}, x \in \mathbb{N}\} \quad [5.2]$$

Weiterhin gilt:

$$\forall i_{k,x} = [t_{x,start} und t_{x,end}], i_{k,y} = [t_{y,start}, t_{y,end}] : t_{x,end} + \tau_k < t_{y,start} \quad [5.3]$$

τ_k ist dabei die Periode der Botschaft m_k. Ein Intervall $i_{k,x}$ gilt genau für eine Botschaft m_k. Innerhalb des Intervalls wird die Botschaft permanent zyklisch übertragen. Die Intervalle $i_{k,x}$ der einzelnen Botschaften m_k können sich dabei überlappen. Ein Beispiel für eine Überlappung von Intervallen ist in Abbildung 5.8 dargestellt. Die Intervalle der Botschaften m_1, m_9 und m_{17} überlappen sich im Bereich t_0 bis t_1 und bilden somit das Szenario s_1. Auf der Basis der sich überlappenden Intervalle $i_{k,x}$, beinhaltet ein Szenario alle Botschaften $m_k \in M_{dual}$, deren Intervalle sich überlappen. Die Bestimmung und Zusammenfassung der zeitlichen Überlappung von Intervallen wird über einen Art *Sweepline-Algorithmus* realisiert [62]. Ausgehend vom Beginn des Loggingdatensatzes Y am Startzeitpunkt $t_{Y,start}$ wird solange ein Zeitfenster aufgezogen bis eine Stelle erkannt wird, während der keine Botschaften der Menge M_{dual} aktiv sind. Sobald wieder eine Botschaft der Menge M_{dual}aktiv ist, wird ein neues Fenster aufgezogen und dauert solange bis wieder eine Phase der Inaktivität festgestellt wird. Dieser Schritt wird so oft wiederholt bis das Ende des Loggingdatensatzes Y erreicht ist. In Abbildung 5.8 sind die Phasen der Inaktivität zwischen t_1 und t_2, t_3 und t_4 sowie zwischen t_5 und t_6.

Im dritten Pfad von Links in Abbildung 5.6 werden die Botschaften $m_k \in M_{csx}$ ausgewertet. Diese Botschaften haben sowohl ein zyklisches als auch ein spontanes Auftrittsverhalten. Um die unterschiedlichen Aktivierungen zu erkennen, wird deren Auftrittsfrequenz ermittelt. Hierfür wird der Loggingdatensatz Y auf alle Botschaften $m_k \in M_{csx}$ analysiert. Im ersten Schritt der Analyse erfolgt die Unterteilung von Y in gleichgroße Intervalle $i_{k,x} \in I_{csx}$. I_{csx} ist dabei die Menge an Intervallen der Botschaften aus M_{csx}. Die Intervalle $i_{k,x}$ haben die Größe der Zykluszeit τ_k der jeweiligen Botschaft m_k:

$$
\begin{aligned}
I_{csx} = & \{ i_{k,x} = [t_{x,start}, t_{y,end}] | t_{Y,start} \leq t_{x,start} < t_{x,end} \leq t_{Y,end} \} : \\
& \text{i) } t_{x,start} = x \cdot \tau_k \wedge t_{x,end} = (x+1) \cdot \tau_k, x \in \mathbb{N} \\
& \text{ii) } 0 \leq x \leq \left\lfloor \frac{t_{Y,end} - t_{Y,start}}{\tau_k} \right\rfloor \\
& \text{iii) } \forall t \text{ mit } t_{x,start} \leq t < t_{x,end} : \exists y = (m_k, t) : m_k \in M_{csx}
\end{aligned}
\qquad [5.4]
$$

In jedem Intervall $i_{k,x}$ kann eine Botschaft $m_k \in M_{csx}$ genau einmal auf der Basis deren Periode und mehrmals spontan gesendet werden. D.h. in einem Intervall $i_{k,x}$ einer Botschaft m_k ist die Auftrittshäufigkeit $\chi_{k,x}$ der Botschaft wie folgt definiert:

$$\chi_{k,x}(i_{k,x}) = |\{(m_k,t)|t \in i_{k,x}\}| \quad [5.5]$$

Im zweiten Schritt werden alle adjazenten Intervalle $i_{k,x}$ und $i_{k,x+1} \in I_{csx}$ zusammengefasst, welche die gleiche Auftrittshäufigkeit $\chi_{k,x} = \chi_{k,x+1}$ haben. Dieser Aggregationsschritt ist in Algorithmus 2 dargestellt. Aufgrund von Jittereffekten können immer nur sehr wenige Intervalle zusammengefasst werden. Diese Jittereffekte sind über einen nicht-linearen Filter eliminierbar. Hierfür kommt ein Verfahren aus der Bildbearbeitung zum Einsatz. Mit diesem sogenannten *Glättungsverfahren* sind die Jittereffekte kompensierbar und es kann dadurch die Zusammenfassung der Intervalle verbessert werden.

Algorithm 5.2: Zusammenfassung der Intervalle mit gleichem $\chi_{k,x}$

input : Botschaften der Menge M_{csx}
output: Intervalle I_{csx}

```
1 forall the m_k ∈ M_csx do
2     for x ← 1, x ≤ |I_csx|, x++ do
3         if χ_{k,x} = χ_{k,x+1} then
4             i_{k,x+1} ← (t_0(i_{k,x}), t_1(i_{k,x+1}));
5             i_{k,x} ← ∅;
6         end
7     end
8 end
```

In Gleichung 5.5 wird die Auftrittshäufigkeit für eine Botschaft m_k innerhalb eines Intervalls $i_{k,x}$ berechnet. Auf dieser Basis kann eine Sequenz $H_k \in \mathbb{N}$ erzeugt werden, welche die Auftrittshäufigkeiten $\chi_{k,x}(i_{k,x})$ umfasst. Mit dieser Sequenz H_k und einer ungeraden Zahl $w \in \mathbb{N}$, welche die Fenstergröße des Filters angibt, kann der Glättungsalgorithmus (siehe Algorithmus 5.3) ausgeführt werden. Ein Beispiel für die Vorgehensweise des Algorithmus ist in Abbildung 5.7 aufgezeigt. Die Sequenz H_K enthält 14 Elemente. In dem Beispiel wird ein Fenster der Größe $f = 3$ über die Sequenz geschoben, dabei wird immer die größte Zahl innerhalb des Fensters übernommen. Daraus resultiert eine

neue Sequenz Z_k. Wichtig ist dabei, dass diese Art der Approximation nicht zu einer Unterschätzung bei der nachgelagerten Timing-Bewertung führt. Da immer der größte Wert aus einem Fenster übernommen wird.

Algorithm 5.3: Glättungsalgorithmus für die Eliminierung von Jittereffekten

input : Sequenz H_k, Fenstergröße f

output: Geglättete Sequenz Z_k

```
1 mask ← ⌊w/2⌋;
2 for x ← 1, x ≤ (|H_k| − mask), x++ do
3 |   z_{k,x} ← 0;
4 |   for y ← −mask, y ≤ mask, y++ do
5 |   |   z_{k,x} ← max{z_{k,x}, h_{k,x+y}};
6 |   end
7 end
```

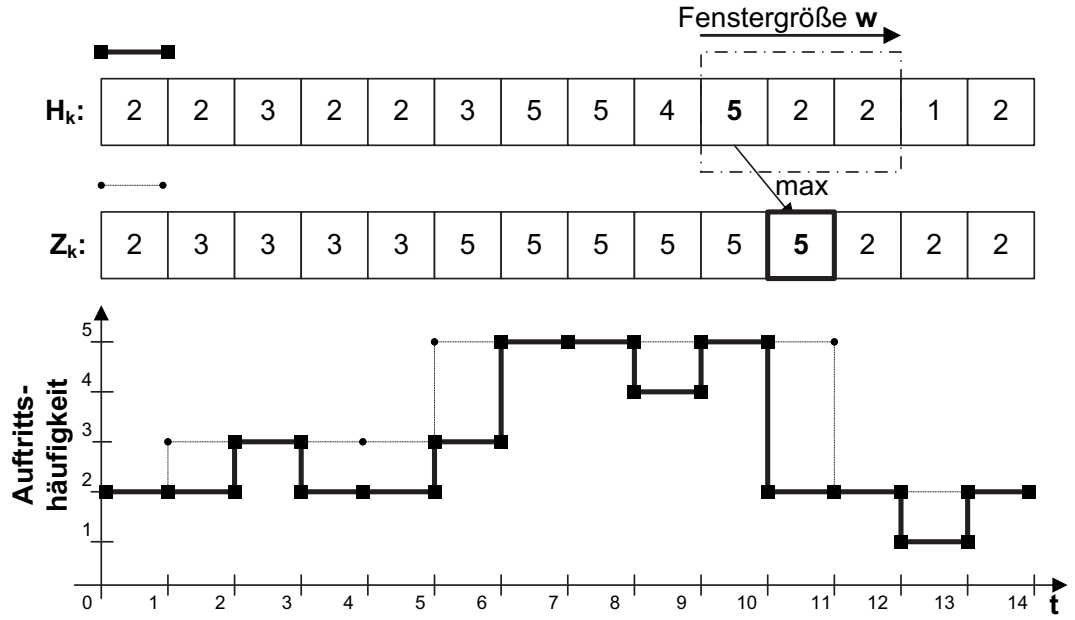


Abb. 5.7.: Beispiel für den Glättungsalgorithmus, um den Einfluss der Jitter-Effekte bei der Zusammenfassung der I_{csx} Intervalle zu reduzieren

Die Auswertung der spontanen Botschaften $m_k \in M_{spontan}$ ist im vierten Pfad von links in Abbildung 5.6 aufgezeigt. Die Menge D enthält die Aktivierungen $d_{k,i} \in D$ einer Botschaft m_k innerhalb eines Betriebsszenarios $s_i \in S$:

$$d_{k,i} = \begin{cases} 1, \text{ if } \exists(m_k, t) : t_{i,start} \leq t \leq t_{i,end} \\ 0, \text{ else} \end{cases} \quad [5.6]$$

Ist eine Botschaft $m_k \in M_{spontan}$ innerhalb eines Betriebsszenarios s_i aktiv, so wird die Botschaft als aktiv markiert und deren Mindestsendeabstand $t_{min,k}$ wird als Zykluszeit übernommen. In Abbildung 5.8 ist dieser Schritt beispielhaft dargestellt. Die Botschaft m_{31} ist nur in den Betriebsszenarien s_1 und s_2 aktiv. In den Betriebsszenarien s_3 und s_4 ist die Botschaft m_{31} nicht enthalten. Weiterhin wird für die spontanen Botschaften für jedes Szenario deren Auftrittshäufigkeit bestimmt. Eine solche Häufigkeitsverteilung ist in Tabelle 5.1 aufgeführt.

Prioritäten der spontanen	**Auftrittsmatrix für $m_{spontan}$ der Szenarien s_i**											
Botschaften $m_{spontan}$	1	2	3	4	5	6	7	8	9	10	11	12
0x23	3	4	-	3	-	-	-	8	-	3	-	-
0xdf	-	-	-	-	-	-	-	-	-	-	-	-
0x34	-	-	5	20	19	15	16	9	-	-	-	8
0x3	4	6	7	8	9	4	4	4	-	7	-	-

Tab. 5.1.: Häufigkeiten der spontanen Botschaften innerhalb von Szenarien

Im letzten Schritt (siehe Abbildung 5.6) der Szenario-Extraktion erfolgt die Abbildung der einzelnen Intervalle auf die Betriebsszenarien. Dieses Vorgehen ist in Algorithmus 5.4 aufgezeigt. Dabei werden die Intervalle I_{csx} aus Pfad 3 und die spontanen Botschaften $M_{spontan}$ auf die Betriebsszenarien $s_i \in S$ aus Pfad 2 abgebildet. Das Ergebnis ist eine Menge an Botschaften M_i für jedes Betriebsszenario s_i mit $M_i \subseteq M_{dynamic}$. Dieser Abbildungsschritt ist beispielhaft in Abbildung 5.8 dargestellt. Die Botschaften m_1, m_9 und $m_{17} \in M_{dual}$, die Botschaften m_7, $m_{41} \in M_{csx}$ sowie die spontanen Botschaft

$m_{31} \in M_{spontan}$ sind alle auf Betriebsszenario s_1 abgebildet (siehe oberer Abschnitt in Abbildung 5.8).

Vor dem Export werden die Botschaftsmengen M_i der Szenarien $s_i \in S$ mit den Botschaften der statischen Last M_{static} zusammengefasst. Weiterhin werden die allgemeinen Timing-Informationen mit exportiert.

Algorithm 5.4: Mapping von $i_{k,x} \in I_{csx}$, $i_{k,x} \in I_{dual}$ und $d_{k,i} \in D$ auf $s_i \in S$ sowie Generierung der Menge an Botschaften M_i

input : I_{csx}, I_{dual}, D, s_i, M
output: M_i

forall the $s_i \in S$ **do**
  $M_i \leftarrow \emptyset$;
  forall the $i_{k,y} \in I_{Dual}$ **do**
    if $t_{i,start} \leq t_0(i_{k,y})$ *and* $t_{y,end} \leq t_{m-n}(i_{k,y})$ **then**
      $M_i \leftarrow M_i \cap m_k$;
    end
  end
  forall the $i_{k,y} \in I_{csx}$ **do**
    if $t_{i,start} \leq t_0(i_{k,y}) < t_{i,end}$ *or* $t_{i,start} < t_1(i_{k,y}) \leq t_{i,end}$ **then**
      $M_i \leftarrow M_i \cap m_k$;
    end
  end
  forall the $d_{k,i} \in D$ **do**
    if $d_{k,i} = 1$ **then**
      $M_i \leftarrow M_i \cap m_k$;
    end
  end
  output M_i;
end

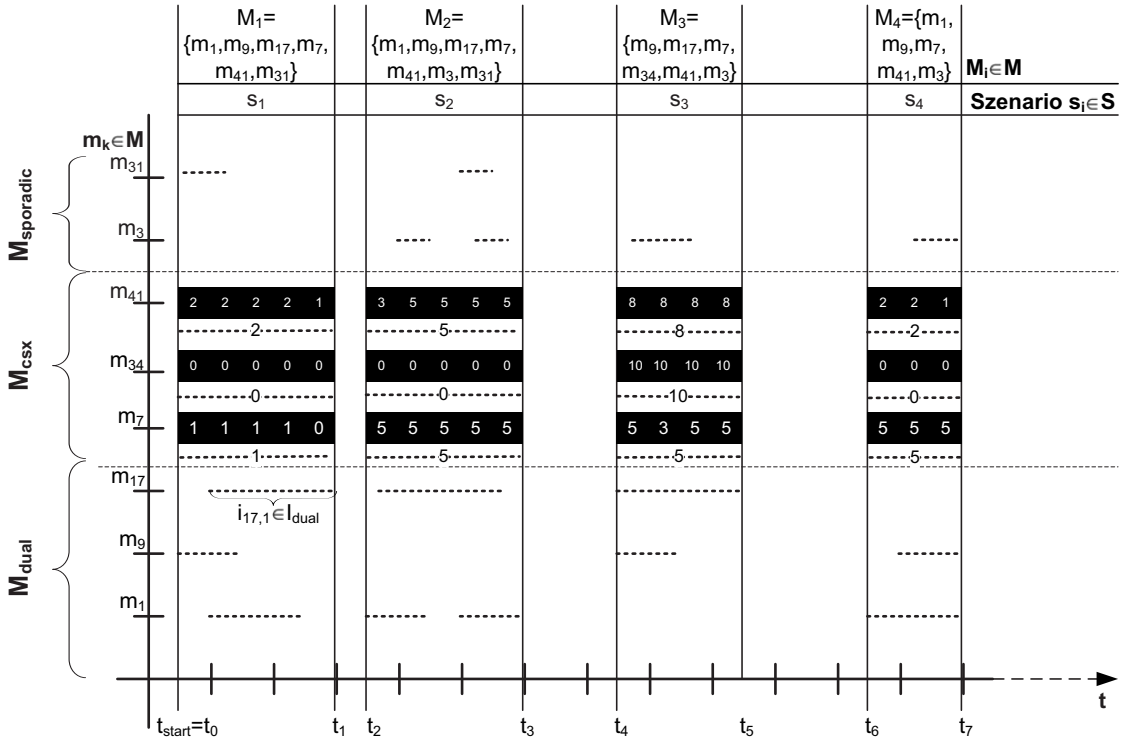


Abb. 5.8.: Mapping von M_{dual}, M_{csx} und $M_{spontan}$, die gemeinsam in einem Szenario $s_i \in S$ aktiv sind

5.2. Bewertung des Verfahrens

Die beschriebenen Verfahren wurden prototypisch im Rahmen der vorliegenden Arbeit, u.a. durch studentische Arbeiten, in einem Werkzeug, dem sogenannten *CAT - CAN Analyze Toolkit* umgesetzt [20],[66],[51]. Abbildung 5.9 zeigt eine Übersicht über das Werkzeug. Es können Loggingdatensätze, die mehrere *Giga Byte* groß sind, importiert werden. Es besteht die Möglichkeit, die importierten Daten direkt im Werkzeug zu untersuchen. Hierfür stehen verschiedene *Viewer* zur Verfügung. Weiterhin sind die ermittelten Informationen auch exportierbar.

Ein weiteres Einsatzszenario für das Verfahren ist die Auswertung von Messungen während der Integrationstests. Ein Beispiel hierfür ist in Abbildung 5.10 aufgezeigt. Während des gesamten Testlaufes waren bei der Aktivität der Botschaften m_1 bis m_4, die Botschaften m_5 und m_6 nicht aktiv. Bei den Messungen wurde für die Botschaft m_4 eine maximale Antwortzeit von $R_4 = 0.8ms$ ermittelt. Eine nachträgliche Analyse lieferte

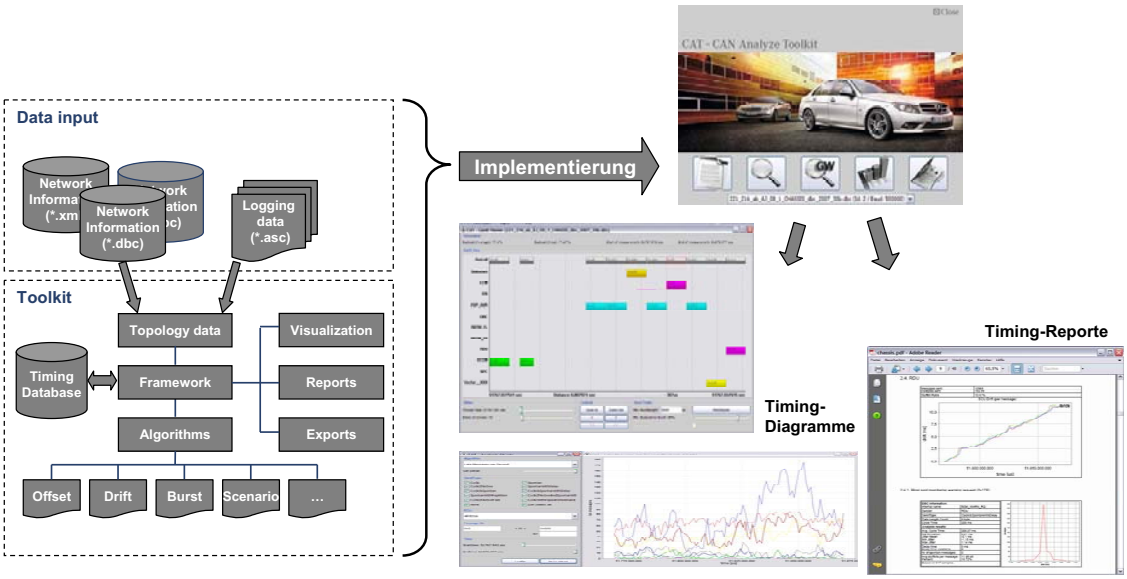


Abb. 5.9.: Realisierung der Verfahren im CAN Analyse Toolkit [20]

eine maximale Antwortzeit von $R_4 = 1.8ms$. Das Beispiel zeigt, dass durch eine Kombination aus Testing und Analyse eine erweiterte Abdeckung des Timing-Verhaltens erreicht werden kann. Es können die in einer Messung möglichen, aber nicht aufgetretenen *Corner Cases* sicher ermittelt werden. Für die sichere Bestimmung der globalen oberen Schranke sind entsprechend alle relevanten Botschaften zu berücksichtigen.

Durch diese Art der Informationsrückgewinnung konnte die Kenntnis über den aktuellen Stand der CAN-Busse signifikant gesteigert werden. Weiterhin wurde die Genauigkeit der Timing-Bewertungsverfahren durch die zusätzlichen Informationen stark erhöht. Dies betrifft zum einen die Modellierung in der frühen Entwicklungsphase einer E/E-Architektur. Hierfür können die gewonnenen Timing-Daten als zusätzliche Information (Erfahrungswerte) verwendet werden. Zum anderen kann das Verfahren zur Timing-Auswertung von langen Loggingdatensätzen zum Einsatz kommen, die bei Testfahrten aufgezeichnet wurden. Die dadurch ermittelten Ergebnisse geben einen sehr guten Aufschluss über die dynamischen Lastzustände auf den CAN-Bussen, während des normalen Fahrbetriebes. In Abschnitt 8.3 wird das Extraktionsverfahren anhand eines Beispiels aus der Praxis evaluiert.

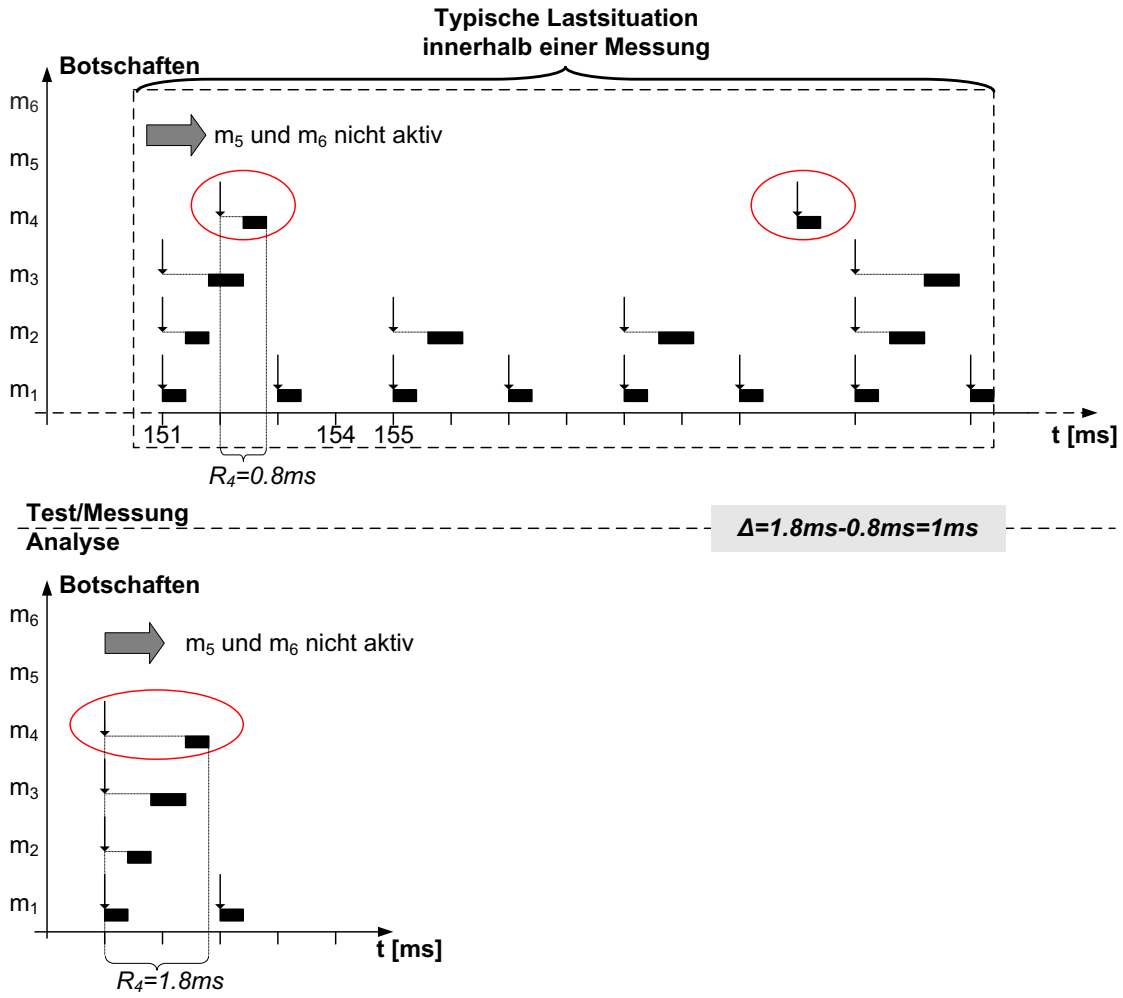


Abb. 5.10.: Beispiel für eine vollständige Testabdeckung auf der Basis eines Loggingdatensatzes

6. Modellierungsregeln zur exakten Timing-Bewertung

In Kapitel 3 wurden die Anforderungen sowie die Notwendigkeit für eine Bewertung des Timing-Verhaltens von Vernetzungsarchitekturen verdeutlicht und die hierfür relevanten Kenngrößen vorgestellt. Bei der Betrachtung der existierenden Timing-Bewertungsverfahren in Kapitel 4 konnte keines der existierenden formalen Verfahren direkt eine effiziente und brauchbare Lösung bieten.

Im den folgenden Abschnitten werden Regeln vorgestellt, welche eine exakte Modellierung des Timing-Verhaltens von Vernetzungsarchitekturen und Gateways (CAN u. FlexRay) ermöglichen. Ausgehend von den formulierten Kenngrößen in Kapitel 3 werden Modellierungsregeln und Umsetzungsvorschriften entwickelt, die eine exakte Abbildung des Zeitverhaltens auf die in Kapitel 4 vorgestellten Verfahren möglich machen. Im ersten Abschnitt werden Modellierungsregeln für den CAN-Bus und FlexRay diskutiert. Im Anschluss daran erfolgt die Ableitung von Regeln, die für alle Kommunikationssysteme Gültigkeit haben. Weiterhin wird ein Konzept für ein Analyse-Modell eines Gateway-Steuergerätes entwickelt, welches eine umfassende Timing-Bewertung für Steuergeräte ermöglicht, die sowohl Routing- als auch Applikationsaufgaben ausführen.

6.1. Modellierungsregeln für das Timing-Verhalten des CAN-Bus

Der CAN-Bus ist ein ereignis-gesteuertes Kommunikationssystem. In den aktuell verfügbaren Konfigurationsdaten, die in Form von sogenannten Kommunikationsmatrizen vorliegen, sind nicht genug Informationen vorhanden, auf deren Basis eine aussagekräftige Timing-Bewertung durchgeführt werden kann. Um diese Lücke zu schließen, besteht die Möglichkeit ein großer Teil der Informationen über das in Kapitel 5 vorgestellte Verfahren generiert werden. Dies erfordert jedoch, dass bereits erste Prototypen des Systems vorliegen, außerdem besteht die Möglichkeit, die Informationen aus existierenden Systemspezifikationen abzuleiten; teilweise ist das Timing-Verhalten implizit

oder explizit darin beschrieben. Auf der Basis dieser Informationen können Regeln abgeleitet werden, die eine exakte Modellierung des Kommunikationsverhaltens ermöglichen. Diese Regeln umfassen die Sendetypen, die Stuffbits und die Modellierung der Offsettabellen der Steuergeräte.

6.1.1. Abbildung der Sendetypen

Ein wichtiger Punkt für die exakte Modellierung des Kommunikationsverhaltens ist die korrekte Abbildung der CAN-Sendetypen auf das Ereignismodell. Je nach eingesetztem Software-Stand (OSEK oder AUTOSAR) sind unterschiedliche Sendetypen für die CAN-Botschaften definiert. Weiterhin können sich die Definitionen von OEM zu OEM unterscheiden. In Tabelle 6.1 sind die Sendetypen des bei der Daimler AG verwendeten Kommunikationsstandards für CAN aufgeführt (siehe auch Tabelle 3.1 in Abschnitt 3.1.3) und deren Abbildung auf das Standard Ereignis Modell (siehe Abschnitt 4.3.2). Eine Erweiterung mit anderen oder weiteren Sendetypen sowie die Anpassung auf ein anderes Ereignismodell ist ohne großen Aufwand möglich.

Tab. 6.1.: CAN-Sendetypen, die bei der Daimler AG verwendet werden und deren Abbildung auf das Standard Ereignis Modell

Sendetyp	**Abbildung auf Ereignis Modell**
zyklisch *(cyclicX)*	P oder $P+J$
spontan *(spontanous)*	A oder $A+J$
bei aktiver Funktion (BAF) *(cyclicIfActive)*	P oder $P+J$
zyklisch und spontan (csx) *(cyclicAndSpontanousWithDelay)*	(P oder $P+J$) und (A oder $A+J$)
schnell *(cyclicIfActiveFast)*	P oder $P+J$
Keine *(none)*	Inaktiv

6.1.2. Korrekte Berücksichtigung der Stuffbits

Die Stuffbits $n_{stuff,k}$ einer CAN-Botschaft m_k haben großen Einfluss auf deren Übertragungsdauer C_k. Die Anzahl der Stuffbits ist abhängig von der Payload p_k, dem Header und dem CRC-Feld (siehe Abschnitt 3.1.3). Maximal sind 24 Bits bei einer *Standard* CAN-Botschaft mit einer Payload von $p_k = 8[Byte]$ möglich. In die Timing-Bewertung können die Stuffbits auf unterschiedliche Weise einfließen:

- Die Stuffbits werden nicht berücksichtigt: $n_{stuff,k} = 0$
- Die mittlere Anzahl der Stuffbits wird aus einer Fahrzeugmessung extrahiert
- Auf Basis der *Default-Werte* aus der K-Matrix erfolgt die Bestimmung der Worst-Case Stuffbits einer Botschaft
- Die Stuffbits werden gemittelt berücksichtigt:
  $n_{stuff,k} = \left\lfloor \frac{34+8 \cdot p_k}{4 \cdot 2} \right\rfloor$
- Es wird für jede Botschaft deren Worst-Case Stuffbits laut der Formel 3.1 berechnet

In Abbildung 6.1 ist der Einfluss der Stuffbitvarianten auf die Übertragungszeit einer Botschaft mit $p_k = 8[Byte]$ dargestellt (CAN-Bus mit $B = 500kBit/s$).

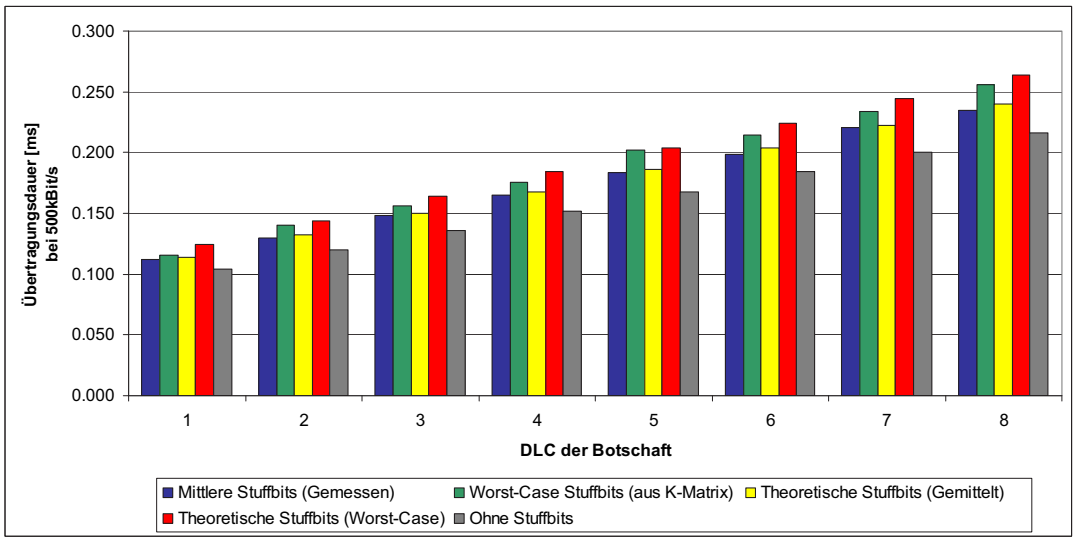


Abb. 6.1.: Einfluss der Stuffbits auf die Übertragungszeit einer CAN-Botschaft

Abbildung 6.2 zeigt die Anzahl von Stuffbits für eine Menge M an Botschaften, die innerhalb eines Loggingdatensatzes Y aufgetreten sind. Die Stuffbits wurden auf vier verschiedenen Varianten ermittelt. Die verwendete Messung hatte eine Länge von 20 Minuten. Die *blauen* Linie zeigt die gemittelten Stuffbits, die über die Auswertung des Loggingdatensatzes gewonnen wurden (Variante 1). Anhand der *grünen* Linie sind die berechneten Stuffbits auf Basis der K-Matrix aufgetragen (Variante 2). Die *gelbe* Linie zeigt die per Formel gemittelten Stuffbits (Variante 3). Die *rote* Linie stellt die theoretischen Worst-Case Stuffbits dar (Variante 4).

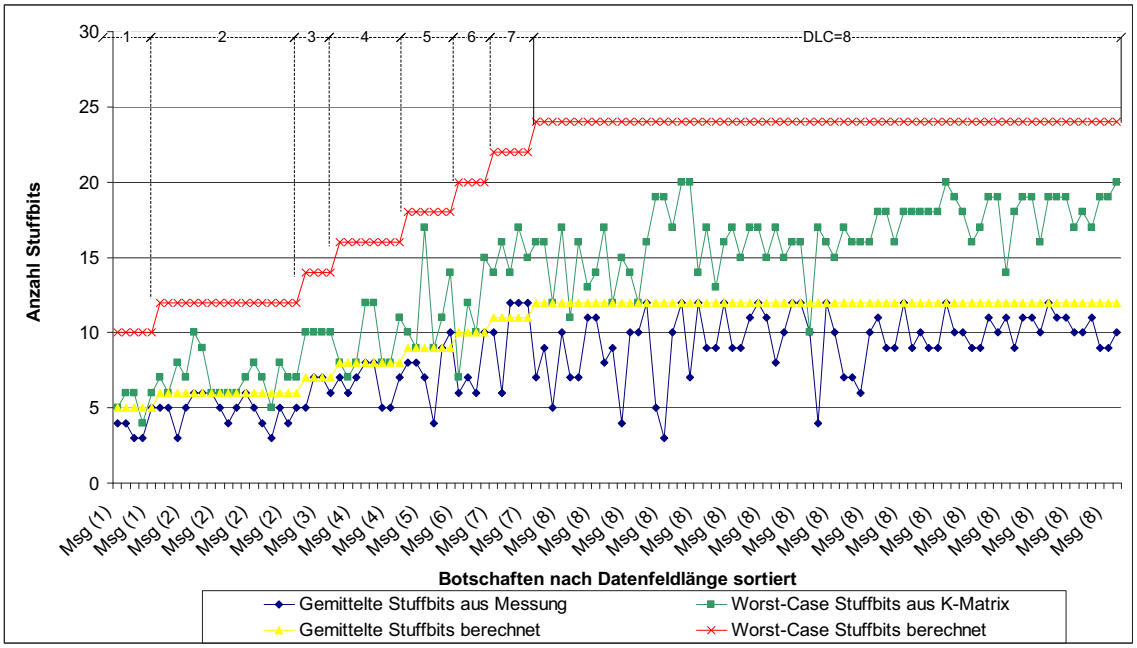


Abb. 6.2.: Vergleich zwischen Stuffbits aus einer Messung ermittelt, berechnet auf Basis K-Matrix und via Formel

Durch die gezielte Auswahl der Varianten kann eine Überschätzung oder Unterschätzung reduziert bzw. vermieden werden. Bei der Untersuchung der Aufstartphase eines Busses sind die Variante 2 oder Variante 4 zu verwenden. Beim Systemstart senden die Steuergeräte ihre Botschaften mit sogenannten *Defaultwerten*, z.B. $0x00$ oder $0xFF$ für eine 1 Byte Botschaft. Durch die gleichen Bitwerte erhöht sich die Anzahl der Stuffbits.

Bei der Analyse des *Normalbetriebes* liefert Variante 1 oder Variante 3 die genauesten Ergebnisse. Wie aus Abbildung 6.2 ersichtlich ist, stellen die gemittelten Stuffbits eine sehr gute Approximation der real aufgetretenen mittleren Stuffbits dar. Nur in drei Fällen werden die gemessenen Stuffbits von der Approximation um ein Bit unterschätzt. Bei einer Übertragungsgeschwindigkeit von $V = 500kBit/s$ ergibt sich ein Fehler von 2μ. Dieser wird aber durch die Überschätzung der meisten anderen Botschaften kompensiert.

6.1.3. Verfahren zur Vergabe von Offsets

Der CAN-Bus ist ein ereignis-orientiertes Kommunikationssysteme. Das Senden der Botschaften von den einzelnen Steuergeräten auf einem CAN-Bus erfolgt asynchron. Dies führt dazu, dass in einem ungünstigen Fall bei allen Steuergeräten des Busses gleichzeitig mehrere Botschaften zum Senden anstehen. Dies führt zu sogenannten *Botschafts-Bursts*, d.h. der CAN-Bus ist für längere Zeit zu 100% ausgelastet. Um solche Bursts zu reduzieren, besteht die Möglichkeit über die Vergabe von Offsets, für die zyklischen Botschaften der einzelnen Steuergeräte das Versenden zu entzerren. Durch den Offset entstehen zwischen dem Senden der einzelnen Botschaften eines Steuergerätes freie Zwischenräume auf dem Bus. Diese Zwischenräume können von anderen Steuergeräten für den Versand deren Botschaften genutzt werden. Sowohl in der aktuell verwendeten Basis-Software bei Daimler als auch bei der AUTOSAR Basis-Software, können diese Offsets für zyklische Botschaften vergeben werden (siehe hierzu in folgenden Spezifikationen des Kommunikationsverhaltens [138], [10]).

Die bisher verfügbaren Optimierungsverfahren sind, aufgrund der fehlenden Parametrierbarkeit, nicht praxistauglich (siehe Abschnitt 4.4). Aktuell findet die Vergabe der Offsets manuell statt oder wird aus dem verwendeten Konfigurationswerkzeug (z.B. *GENy* von Vector Informatik) für die Basis-Software generiert. Diese Generierung basiert auf einem sehr einfachen inkrementellen Vergabeverfahren. Weiterhin werden die Offsets lokal für jedes Steuergerät von den Zulieferern vergeben.

Bei der Vergabe der Offsets sind einige Parameter zu beachten:

- Die Startzeit T_{start}, diese gibt vor nach welcher Zeit ein Steuergerät alle seine zyklischen Botschaften mindestens einmal nach dem Hochfahren des CAN-Busses versendet haben muss.

- Die Zykluszeit T_{com} des COM-Tasks der Steuergeräte, dieser ist für die Schrittweite der Offsets notwendig. Beispielsweise können bei einer Zykluszeit $T_{com} = 10ms$ die Offsetwert nur ein Vielfaches von $10ms$ annehmen.

In dieser Arbeit wurde ein Algorithmus entwickelt, welcher eine optimierte Vergabe der Offsets über Busgrenzen hinweg ermöglicht. Der Vorteil dieses Vorgehens ist eine verbesserte Ausbalancierung der Offsets über die gesamten CAN-Busse einer Vernetzungsarchitektur. Es werden dabei sowohl die Antwortzeiten der Botschaften auf den einzelnen CAN-Bussen reduziert als auch die Routing-Last der Gateways zeitlich besser verteilt. Insbesondere in der Startupphase können so hohe Routinglasten an den Gateways reduziert werden.

Bei dem Algorithmus handelt es sich um eine Heuristik. Diese Art der Optimierung der Offsets aller zyklischer CAN-Botschaften einer Vernetzungsarchitektur liefert nicht in jedem Fall auch ein lokales Optimum, d.h. bei der Optimierung der Offsets für jeden einzelnen Bus getrennt können in manchen Fällen bessere Ergebnisse erzielt werden. Nachteil ist jedoch dann, dass die Routing-Last an den Gateways nicht geglättet ist. Je nach Anforderung entsprechend kann eine Optimierung der Offsets für jeden einzelnen CAN-Bus (lokal) durchgeführt werden oder die Offsets unter Berücksichtigung der gesamten Vernetzungsarchitektur (global) vergeben wenden.

Als Eingangsdaten für den Algorithmus sind folgende Parameter notwendig: Die Periode der COM-Tasks T_{com}, die Startzeit T_{start}, die K-Matrizen K der relevanten CAN-Busse sowie die Anzahl der Busse $NbOfBusses$. Als Ausgabe liefert der Algorithmus die Offsets für die periodischen Botschaften der Steuergeräte.

Zuerst wird die Anzahl an Intervallen bestimmt. Diese ergibt sich aus der Startzeit T_{start} dividiert durch die ComTask-Periode T_{Com}. Weiterhin wird eine *Hash-Liste distr* angelegt. In dieser Liste werden für alle Busse die Offsetverteilung hinterlegt. Die Berechnung der Offsets startet dann im darauffolgenden Schritt. Es wird über alle relevanten CAN-Busse iteriert. Zuerst erfolgt die Bestimmung der Steuergeräteanzahl des aktuellen Busses über die Funktion `getNbOfEcus`. Daraufhin wird über alle Steuergeräte des Busses iteriert und die Anzahl an zyklischen Botschaften $nbOfMsg$ bestimmt sowie eine Liste *msgList* dieser Botschaften erstellt. Dies ist über die Funktionen `getNbOfMessages` und `getMessages` umgesetzt.

Im nächsten Schritt erfolgt die Prüfung, ob die aktuelle ECU periodische Botschaften besitzt. Ist dies der Fall wird die Schrittweite *stepSize* berechnet. Diese gibt an in welchen Abständen die Offsets initial für die Botschaften eines Steuergerätes vergeben werden. Dann erfolgt die Iteration über die Liste der Botschaften *msgList*.

Innerhalb der Schleife wird überprüft, ob die aktuelle Botschaft von einem anderen Bus in den aktuell relevanten Bus geroutet wird, d.h. das Quellsteuergerät befindet sich an einem anderen Bus. Ist dies der Fall wird kein Offset vergeben. Dies wird über die Funktion `isInRoute` abgefragt. Liefert die Funktion *False* zurück, d.h. die aktuelle Botschaft ist keine geroutete Botschaft, erfolgt die Berechnung des Offsets. Für die Berechnung wird die initiale Schrittweite *stepSize*, die Anzahl an Intervallen *nbOfIntervals* und die Verteilung *distr* der Funktion `calcOffset` übergeben.

Auf der Basis dieser Werte wird eine optimierte Verteilung der Offsets realisiert. Der Offsetwert wird so gewählt, dass die Offsetverteilung über die Startzeit T_{start} optimal ausbalanciert ist. Anschließend wird der Offset der aktuellen Botschaft ausgegeben und der Verteilung *distr* über die Funktion `updateDistr` übergeben. Im nächsten Schritt wird über die Funktion `isOurRoute` geprüft, ob es sich um einen Botschaft handelt, die in andere CAN-Netze weitergeleitet wird. Ist dies der Fall wird über die Funktion `addOff` der Offset den entsprechenden Sendebotschaften der Gateways übergeben und die Hash-Liste *distr* entsprechend ergänzt.

Bei der Modellierung des CAN-Busses ist die Berücksichtigung der Offsets sehr wichtig. Da diese großen Einfluss auf die ermittelten maximalen Antwortzeiten haben. Bei unbekannten Offsettabellen können diese entweder über den vorgestellten Algorithmus generiert werden oder aber über das in Abschnitt 5.1.2 vorgestellte Extraktionsverfahren aus existierenden CAN-Bussen ermittelt werden. Ein weiterer wichtiger Punkt bei der Modellierung von CAN-Bussen ist die Berücksichtigung der korrekten Offsets der gerouteten Botschaften. Hierzu ist immer der Offset der Botschaft des eigentlichen Quellsteuergerätes zu verwenden. Verfeinernd kann noch die Latenzzeit des Routings L_{rout} hinzu addiert werden. Weiterhin können noch die Aufstartzeiten der Steuergeräte sowohl bei der Vergabe als auch bei der Modellierung mitberücksichtigt werden.

Algorithm 6.1: Algorithmus zur Generierung von Offsets

input : T_{start}, T_{com}, K, $NbOfBusses$

output: $T_{off,k}$

```
1  //Calculate number of intervals;
2  nbOfInterval ← T_start / T_Com;
3  //Initialize Hash-List;
4  initDistr(distr, NbOfBusses);
5  //Start offset calculation for the messages of all busses;
6  for i ← 1 to NbOfBusses, i++ do
7      nbOfEcus ← getNbOfEcus(K[i]);
8      //Iterate over each ECU;
9      for j ← 1 to nbOfEcus, j++ do
10         msgList ← getMessages(K[i]);
11         nbOfMsg ← getNbOfMessages(K[i]);
12         //The actual ECU has periodic messages;
13         if msgList ≠ ∅ then
14             stepSize ← T_start / nbOfMsg − (T_start / nbOfMsg) Modulo 10;
15             //Iterate over all periodic messages of an ECU;
16             for k ← 1 to nbOfMsg, k++ do
17                 //Check if it is a routing message;
18                 if isInRoute(msgList[k]) = False then
19                     t_off,k ← calcOffset(stepSize, nbOfInterval, distr);
20                     updateDistr(t_off,k, distr, i);
21                     //Add the calculated offset of routing messages to the other busses;
22                     if isOutRoute(msgList[k]) = True then
23                         addOff(msgList[k], K, t_off,k, distr);
24                     end
25                 end
26             end
27         end
28     end
29 end
```

6.2. Modellierungsregeln für das Timing-Verhalten von FlexRay

Bei der Bewertung der reinen FlexRay-Kommunikation sind wenig Timinig-Effekte zu beobachten. Erst bei der Berücksichtigung des Sendeverhaltens bzw. des Steuergeräteverhaltens in Verbindung mit dem FlexRay-Schedule müssen gezielt die Parameter der beteiligten Systeme berücksichtigt werden. Die möglichen Effekte wurden schon in verschiedenen Publikationen erörtert, u.a. in [47], [79], [98] und [97]. Im Folgenden werden die wichtigsten Faktoren und Effekte im Hinblick auf eine korrekte Timing-Modellierung diskutiert.

Im Gegensatz zum CAN-Bus, bei dem das Steuergerät einfach seine zu sendende Botschaft in den Ausgangspuffer des Buscontrollers legt, sind beim FlexRay für eine schnelle Übertragung der aktuellen Daten einige Abhängigkeiten zu berücksichtigen. Einen solchen *asynchronen Übergang* zwischen Steuergerät und Kommunikationsmedium wie es bei CAN-Netzen der Fall ist, gibt es bei FlexRay nicht. Durch die Abhängigkeiten des Steuergeräte- und des FlexRay-Schedules können die zeitlichen Verzögerungen sehr groß werden und ein Vielfaches der Zykluszeit des FlexRay-Schedules betragen. In Abbildung 6.3 und 6.4 sind die Effekte verdeutlicht.

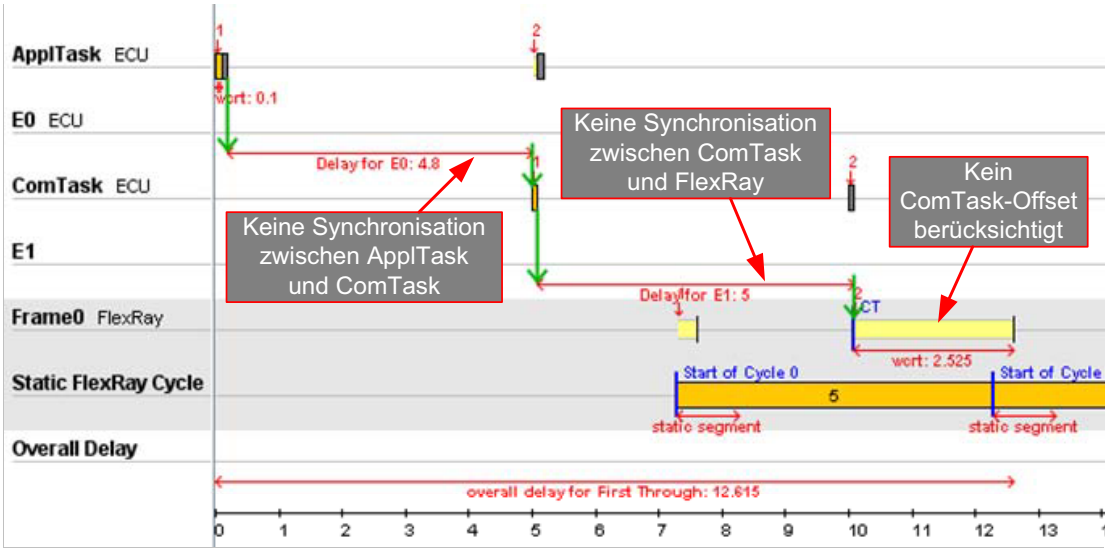


Abb. 6.3.: Beispiel für Timing-Effekte beim Übergang zwischen ECU und dem FlexRay-Bus, ohne Berücksichtigung der Abhängigkeiten und ohne optimierte Offsets

Abbildung 6.3 wurden keine Abhängigkeiten berücksichtigt (keine Synchronisation zwischen den Tasks sowie mit dem FlexRay-Schedule) und die relevanten Attribute (Offsets der Tasks) wurden willkürlich gesetzt. Es resultiert eine maximale Antwortzeit $R = 12.6ms$. Im Modell, welches in Abbildung 6.4 dargestellt ist, wurden sämtliche Abhängigkeiten berücksichtigt und die Offsets optimal gesetzt. Die maximale Antwortzeit beträgt hier $R = 1.3ms$.

Die Beispiele zeigt deutlich, dass für die Modellierung dieser *synchronen* Systeme, die Konfigurationsdaten der Steuergeräte eine zentrale Rolle spielen, um aussagekräftige Ergebnisse zu ermitteln. Aus diesem Grund müssen der *COM-Task-Offset* der Steuergeräte sowie deren interne Task-Struktur (Schedule) möglichst vollständig bekannt sein. Weiterhin sollte insbesondere bei Gateway-Steuergeräten die Konfiguration der *FlexRay-Jobs* zur Verfügung stehen.

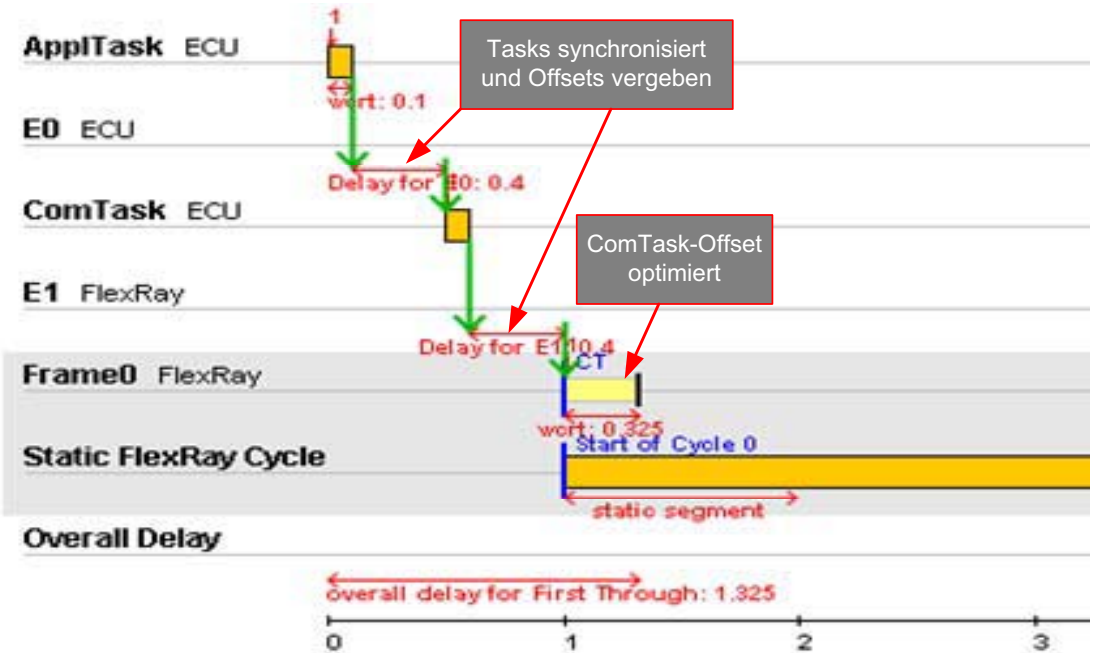


Abb. 6.4.: Beispiel für Timing-Effekte beim Übergang zwischen ECU und dem FlexRay-Bus, mit Berücksichtigung der Abhängigkeiten und optimierten Offsets

Bei der Modellierung der Übergänge zwischen CAN und FlexRay ist es wichtig die Synchronisationseffekte im Gateway-Modell korrekt zu erfassen. Bei einfachen Gate-

way-Modellen, d.h. das Gateway wird als reines Verzögerungselement im Gesamtmodell berücksichtigt, sind zusätzlich zu den Routinglatenzzeiten noch die entstehenden zeitlichen Verzögerungen am Übergang zwischen CAN und FlexRay zu berücksichtigen. Hierzu ist der Zyklus der FlexRay-Botschaft als zusätzliche Verzögerungszeit im Gateway mit einzupflegen.

6.3. Modellierungsregeln für Kommunikationssysteme

In den vorangegangenen Abschnitten wurden Modellierungsregeln beschrieben, die speziell für die Bewertung von CAN-Bussen oder FlexRay gelten. Im Folgenden werden Regeln abgeleitet, die für alle automotive Kommunikationssysteme gültig sind. Die folgenden Regeln umfassen die Modellierung der Transportprotokolle, die Berücksichtigung der dynamischen Kommunikation, die korrekte Modellierung der Kommunikation in der frühen Entwicklungsphase sowie die Ableitung von Bewertungsszenarien für eine detailliertere Bewertung der Kommunikation.

6.3.1. Modellierung von Transportprotokollen

Transportprotokolle (TP) werden zur Übertragung von Datenblöcken eingesetzt, die größer als die natürliche Größe des *Data Link Layers* sind. Die natürliche Größe bei CAN sind 8 Bytes bei FlexRay 254 Bytes. Das Transportprotokoll ist innerhalb der OSI-Schichten drei und vier realisiert. Zusätzlich zur Seg- und Desegmentierung erlauben Transportprotokolle auch eine Flusskontrolle, d.h. es erfolgt die Steuerung des zeitlichen Abstandes der Datenblöcke, um den Empfänger nicht zu überlasten. Ferner kann darüber die gesamte Kommunikation zeitlich überwacht werden. Eine ausführliche Beschreibung der Transportprotokolle ist u.a. in [144], [9] und [12] zu finden.

Bei der Modellierung von *Transportprotokollbotschaften (TP-Botschaft)* ist zwischen *Single Frame*-Botschaften und *Multi Frame*-Botschaften zu unterscheiden. Single Frames werden spontan versendet. Ein Mindestsendeabstand T_{min} gibt es nicht. Daher kann eine TP-Botschaft dieses Typs als spontan modelliert werden. Da kein Mindestsendeabstand T_{min} spezifiziert ist, muss ein Mindestsendeabstand aus der Funktionsspezifikation abgeleitet werden oder es sind Aktivierungsobergrenzen für die Botschaft zu definieren. Bei Multi Frames ist der Kontrollfluss bei der Modellierung zu berücksichtigen, da sonst unrealistische Lastzustände entstehen können. Abbildung 6.5 zeigt die Ablaufkontrolle.

Je nach gewähltem Analyseverfahren kann der Kontrollfluss direkt modelliert werden (z.B. bei UPPAAL [49]). Ist eine direkte Modellierung des Kontrollflusses nicht möglich (z.B. bei SymTA/S [50]), gilt es über *Work-Arounds* das entsprechende Verhalten abzubilden.

Über die Modellierung von synchronisierten Ereignispattern (siehe Abschnitt 4.3.2) kann der Kontrollfluss implizit berücksichtigt werden. Beim FlexRay-TP ist diese Art der Umsetzung direkt möglich. Beim CAN-TP sind zwei Schritte notwendig, um die obere Schranke (maximale Antwortzeiten der TP-Botschaften) sicher bestimmen zu können. Im ersten Schritt ist die maximale Antwortzeit R_{rq} der *Request*-Botschaft m_{rq} zu bestimmen. Im zweiten Schritt erfolgt dann die Bestimmung des gesamten Pfades. Dabei wird für die *Response*-Botschaften als Offset mindestens die ermittelte Antwortzeit R_{rq} der Request-Botschaft gesetzt. Dadurch wird das gegenseitige Blockieren der Request- und Response-Botschaften ausgeschlossen.

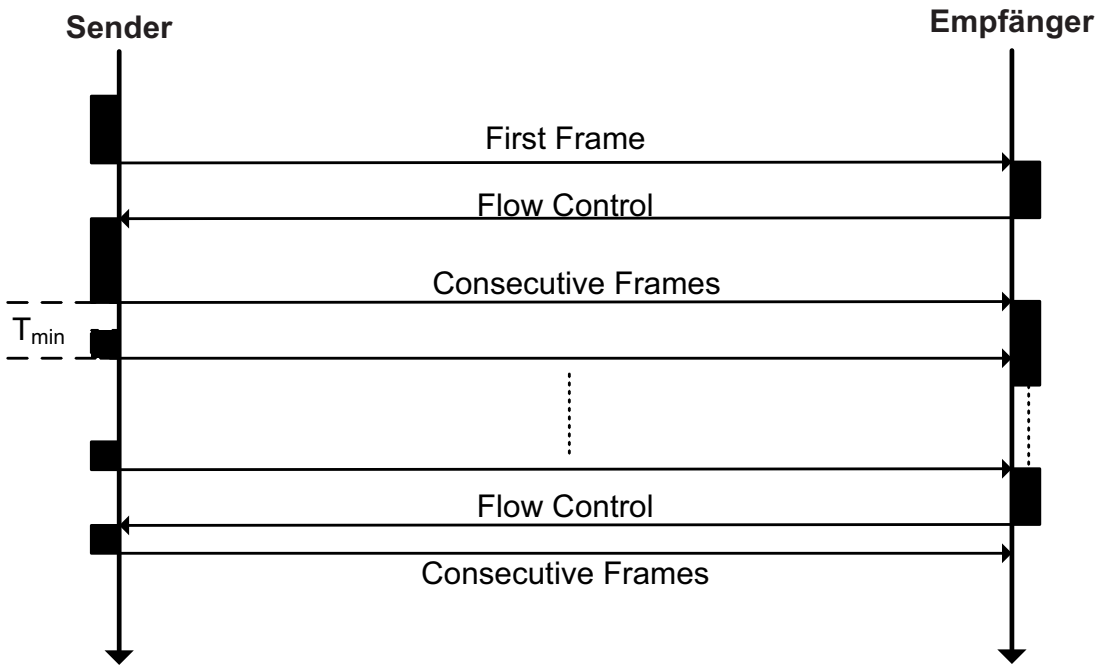


Abb. 6.5.: Flusskontrolle bei Transportprotokollen für *Multi Frame*-Botschaften [144]

6.3.2. Modellierung des Netzwerkmanagements

Bei der Modellierung des Timing-Verhaltens des Netzwerkmanagements (NM) gilt es zwischen dem *OSEK NM* und *AUTOSAR NM* zu unterscheiden. Da sich deren logisches Verhalten auch direkt auf das Kommunikationsverhalten auswirkt.

OSEK NM

Beim OSEK NM wird bei aktivem Bus ein logischer Ring aufgebaut. Dabei senden alle Busteilnehmer eine sogenannte *NM-Botschaft*, welche den Status jedes einzelnen Busteilnehmers enthält. Beim Start des Busses sendet jeder Knoten spontan die NM-Botschaft mit *Alive-Kennung*. Sobald der logische Ring etabliert ist, sendet jeder Knoten zyklisch die NM-Botschaft mit *Ring-Kennung*. Die Zykluszeit T_{ring} hängt von der Größe des Ringes ab. Ein Knoten empfängt die NM-Botschaft seines logischen Vorgängers und wartet dann eine Zeit T_{typ} ab bevor er dann seine NM-Botschaft sendet.

Die Modellierung des OSEK-NM-Verhaltens kann wie folgt umgesetzt werden. Dabei ist zwischen der Startupphase eines Netzwerkes und dem eingeschwungenen Zustand (Ringbetrieb) zu unterscheiden. Für die Analyse der Startupphase eines Netzwerkes sind die NM-Botschaften aller Steuergeräte des Netzwerkes mit deren Mindestsendeabstand T_{min} zu modellieren. Im Ringbetrieb sind zwei Modellierungsvarianten denkbar:

1. Jede NM-Botschaft wird zyklisch mit einer Zykluszeit T_{ring} modelliert. Alle NM-Trigger werden unter Berücksichtigung eines Offsets $T_{off,i}$ miteinander synchronisiert. n_i ist die Nummer des jeweiligen Steuergerätes ECU_i. M_{ecu} ist die Menge aller Steuergeräte des Busses. Der Offset $T_{off,i}$ berechnet sich wie folgt:

$$T_{off,i} = \sum_{i=1}^{n_i} T_{typ} \cdot n_i \text{ mit } 0 \leq n_i \leq |M_{ecu}| \qquad [6.1]$$

2. Es wird die NM-Botschaft eines Knotens ausgewählt und zyklisch mit T_{typ} versendet.

Die erste Variante berücksichtigt alle NM-Botschaften eines Netzes, hat jedoch den Nachteil, dass bei einer Antwortzeit $R_i > T_{typ}$ für eine NM-Botschaft m_k, schon deren logischer Nachfolger aktiviert wird, was nicht dem realen OSEK-NM-Verhalten entspricht. Bei der zweiten Variante kann diese Ungenauigkeit nicht auftreten.

AUTOSAR NM

Im Gegensatz zum OSEK NM wird beim AUTOSAR NM kein logischer Ring aufgebaut, d.h. das Netzwerkmanagement weist ein dezentrales und direktes Verhalten auf. Wird von einem Netzwerkteilnehmer der Bus benötigt, sendet dieser zyklisch mit T_{typ} die NM-Botschaft, andernfalls wird keine NM-Botschaft versendet. Die Modellierung kann wie folgt umgesetzt werden: Jede NM-Botschaft wird zyklisch mit T_{typ} modelliert. Alle NM-Trigger werden miteinander synchronisiert und sofern diese bekannt sind, die Verzögerungszeit als Offset mit berücksichtigt. Sofern die *Bus Load Reduction* (siehe hierzu [13] verwendet wird, darf diese nicht mit berücksichtigt werden, da im schlimmsten Fall davon auszugehen ist, dass sich die Netzteilnehmer noch in der Phase befinden, während die Bus Load Reduction noch nicht wirksam ist und alle Steuergeräte ihre NM-Botschaften versenden.

6.3.3. Modellierung der dynamischen Kommunikation

Die korrekte Modellierung der dynamischen (spontanen) Kommunikation, insbesondere für die Timing-Bewertung von CAN-Bussen, ist ein wesentlicher Faktor, der sich direkt auf die Qualität der Ergebnisse auswirkt. In den aktuellen K-Matrizen wird für spontane Botschaften nur ein Mindestsendeabstand T_{min} spezifiziert. Bei der Verwendung dieses Wertes als Untergrenze für die maximale Auftrittsperiode einer spontanen Botschaft, liefern die Bewertungsverfahren bei der Berücksichtigung aller Botschaften als Ergebnis eine Buslast $> 100\%$. Für eine korrekte Abbildung der spontanen Kommunikation sind verschiedene Varianten möglich:

- Gezielte Modellierung der Betriebsmodi soweit diese sich aus den aktuellen Spezifikationen ableiten lassen.

- Berücksichtigung von Betriebszuständen, die mit dem in Kapitel 5 vorgestellten Verfahren ermittelt werden können.

- Ermittlung der Auftrittshäufigkeiten von spontanen Botschaften und Modellierung dieser mittels entsprechender Funktionen (Zufallsverteilung, Gaußverteilung, etc.), siehe hierzu z.B. ChronSim [54].

- Begrenzung der spontanen Aktivierungen über das sogenannte *Aktivierungslimit (Activation Restrictions)*, siehe hierzu z.B. SymTA/S [115]. Das Aktivierungslimit kann über die Auswertung von Traces bestimmt werden. Hierfür ist die maximale Anzahl an quasi gleichzeitig aufgetretenen spontanen Botschaften zu ermitteln.

6.3.4. Modellierung des Jitters

Relevant für die Modellierung ist der Sendejitter des Steuergerätes J_{send}. Bei bereits existierenden Steuergeräten kann der Wert für den Jitter des Com-Tasks über das in Abschnitt 5.1.1 vorgestellte Verfahren ermittelt werden oder der Wert wird direkt beim Zulieferer des Steuergerätes abgefragt. Liegt das Steuergerät nicht vor, kann dieser auch geschätzt werden. Je nach Mächtigkeit des Steuergerätes liegt der Jitter zwischen $0.5ms$ und $3ms$. Weiterhin besteht die Möglichkeit, bei existierender Konfiguration des Betriebssystems, den Jitter des *ComTasks* über zwei verschiedene Varianten zu ermitteln. In der ersten Variante wird ein komplettes Schedulingmodell des Steuergerätes erstellt und der Jitter J_{send} berechnet. Die Werte der Ausführungszeiten C_k der Tasks w_k sind über das in Abschnitt 4.1 beschriebene Verfahren zu berechnen oder es werden mit Annahmen getroffen bzw. Erfahrungswerte eingesetzt. Die zweite Variante zeigt einen schnellen Weg auf, um einen nicht ganz exakten Wert für den Sendejitter zu ermitteln. Hierfür werden die Ausführungszeiten C_k aller Tasks und Interrupt-Serviceroutinen $w_k \in W_{Ecu}$ des Steuergerätes berücksichtigt, die eine höhere Priorität als der ComTask w_{com} besitzen:

$$J_{send} = \sum_{w_k \in hp(w_{com})} C_k \qquad [6.2]$$

6.3.5. Modellierungsregeln für die frühe Entwurfsphase

In der frühen Entwurfsphase von Vernetzungsarchitekturen besteht immer die Herausforderung auf der Basis von wenigen Informationen Aussagen über die Qualität einer Architekturvariante zu treffen. Oftmals wird für die initiale Modellierung auf bereits existierenden Daten aufgesetzt, die dann schrittweise mit neuen Werten ersetzt oder verfeinert werden. Ein solches Vorgehen ist auch für die Timing-Bewertung von Ver-

netzungsarchitekturen möglich. Ein solcher Weg wurde für die Timing-Bewertung von Steuergeräten schon aufgezeigt [57], [101].

Bei der Timing-Bewertung von Kommunikationssystemen in der frühen Entwurfsphase von E/E-Architekturen sind einige Punkte zu beachten [130]. Die einzelnen Schritte, die für die Definition und die Konfiguration eines Kommunikationssystems notwendig sind, stellt Abbildung 6.6 am Beispiel des Architekturwerkzeuges *PREEvision* dar [6].

Im ersten Schritt weist der E/E-Architekt die Funktionen den entsprechenden Komponenten zu. Der zweite und dritte Schritt kann werkzeuggestützt durchgeführt werden. Dabei erfolgt das Routing der Signale und das Mapping der Signale und PDUs auf die Botschaften. Die Botschaften sind entweder schon vordefiniert oder es werden entsprechend neue erzeugt. Im vierten Schritt sind für die Timing-Bewertung notwendige Konfigurationen zu tätigen. Bei den CAN-Botschaft ist eine passende Priorität zu vergeben. Bei FlexRay gilt es einen sinnvollen Schedule zu erstellen.

6.3.6. Definition von Bewertungsszenarien

Um die verschiedenen Betriebszustände eines Fahrzeuges in der Timing-Bewertung zu berücksichtigen sowie obere Grenzwerte für die zyklische Grundlast und die theoretisch mögliche Maximallast ermitteln zu können, ist es notwendig verschiedene Bewertungsszenarien zu definieren. Einige der Szenarien lassen sich direkt aus existierenden Spezifikationen ableiten (siehe Kapitel 5). Für repräsentative Betriebszustände, die während des Fahrbetriebes auftreten, sind die aktuell vorliegenden Informationen in den Spezifikationen nicht ausreichend. Aus diesem Grund kann hier auf das in Kapitel 5 vorgestellte Verfahren zurück gegriffen werden. Tabelle 6.2 zeigt exemplarisch einige mögliche Bewertungsszenarien am Beispiel eines CAN-Busses.

Die ersten beiden Spalten (Grundlast und maximale Last) bieten die Möglichkeit, Aussagen über die zyklische Grundlast des Busses zu erhalten sowie die maximale Obergrenze zu ermitteln. Über die Szenarien können weitere Betriebsmodi, wie Diagnosebetrieb, Flashen in der Fabrik und dynamische Lastzustände berücksichtigt werden. Je nach Anforderung sind dann die Parameter entsprechend zu setzen.

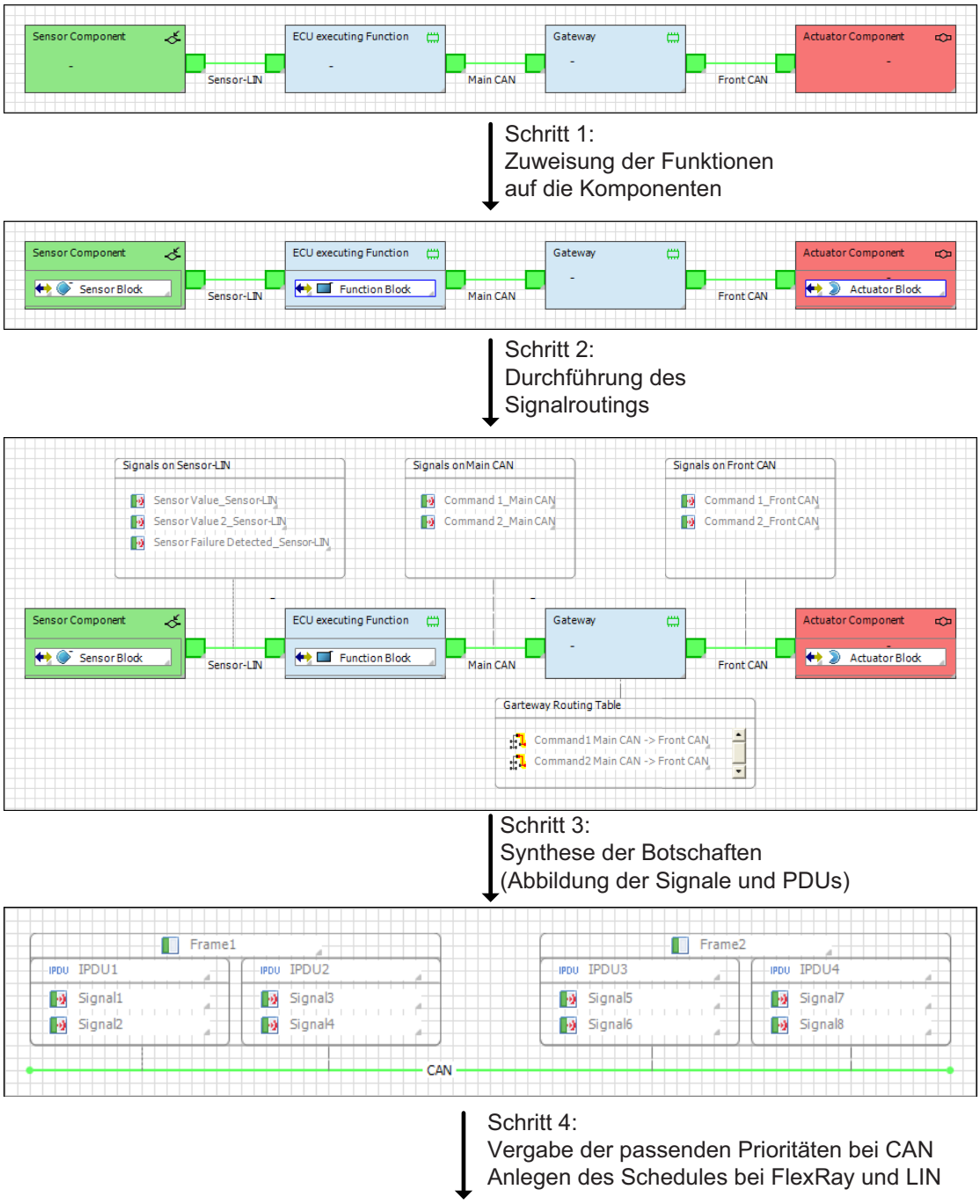


Abb. 6.6.: Schrittweises Vorgehen bei der Definition und Konfiguration eines Kommunikationssystems während der Entwurfsphase von E/E-Architekturen [131]

Tab. 6.2.: Definition von Bewertungsszenarien für eine verfeinerte Timing-Bewertung der Kommunikation

Eigenschaften	**Grundlast**	**Max. Last**	**Szenarien**
Offsets	berücksichtigt	berücksichtigt	berücksichtigt
Jitter	berücksichtigt	berücksichtigt	berücksichtigt
Stuffbits	berücksichtigt	berücksichtigt	berücksichtigt
Sendetyp: cyclicX	berücksichtigt	berücksichtigt	berücksichtigt
Sendetyp: csX	nur zyklischer Anteil berücksichtigt	berücksichtigt mit Aktivierungslimit	abhängig vom Szenario
Sendtyp: BAF	nicht berücksichtigt	berücksichtigt	abhängig vom Szenario
Sendetyp: schnell	langsame Periode	schnelle Periode	abhängig vom Szenario
Sendetyp: csxOnDemand	nicht berücksichtigt	berücksichtigt	abhängig vom Szenario
Sendetyp: spontan	nicht berücksichtigt	berücksichtigt mit Aktivierungslimit	abhängig vom Szenario
Netzwerkmanagement	berücksichtigt	berücksichtigt	berücksichtigt
Diagnosebotschaften	nicht berücksichtigt	nicht berücksichtigt	abhängig vom Szenario
Transportprotokolle	nicht berücksichtigt	berücksichtigt	abhängig vom Szenario

6.3.7. Bewertung der Regeln

Durch die Anwendung der Modellierungsregeln für die Bewertung der Timing-Verhaltens von Vernetzungsarchitekturen ist eine signifikante Verbesserung der Ergebnisqualität möglich, dies wird anhand von Evaluierungsbeispielen in Kapitel 8 im Detail aufgezeigt. Mit der Berücksichtigung des Timing-Verhaltens der verschiedenen Dienste, wie Netzwerkmanagement und Transportprotokolle, kann das reale Kommunikationsverhalten genauer nachgebildet werden. Die zusätzliche Beachtung des Überganges zwischen den Steuergeräten und dem FlexRay-Bus liefert ein exaktes Bild des Timing-Verhaltens. Insbesondere bei den CAN-Bussen wird die Bewertungsgenauigkeit durch die Anwendung der Modellierungsregeln wesentlich gesteigert.

Ein Beispiel für die Steigerung der Bewertungsgenauigkeit ist in Abbildung 6.7 anhand eines CAN-Busses aufgezeigt. Ohne die Berücksichtigung der Timing-Attribute (Offset, Jitter, etc.) und den Modellierungsregeln kommt es zu einer großen Überschätzung (blaue Linie). Durch die Anwendung der Modellierungsregeln und der Verwendung der Timing-Attribute ist eine exakte Timing-Bewertung möglich (rote Linie). Zusätzlich sind in dem Diagramm auch noch die gemessenen Worst-Case Werte abgebildet (gelbe Linie).

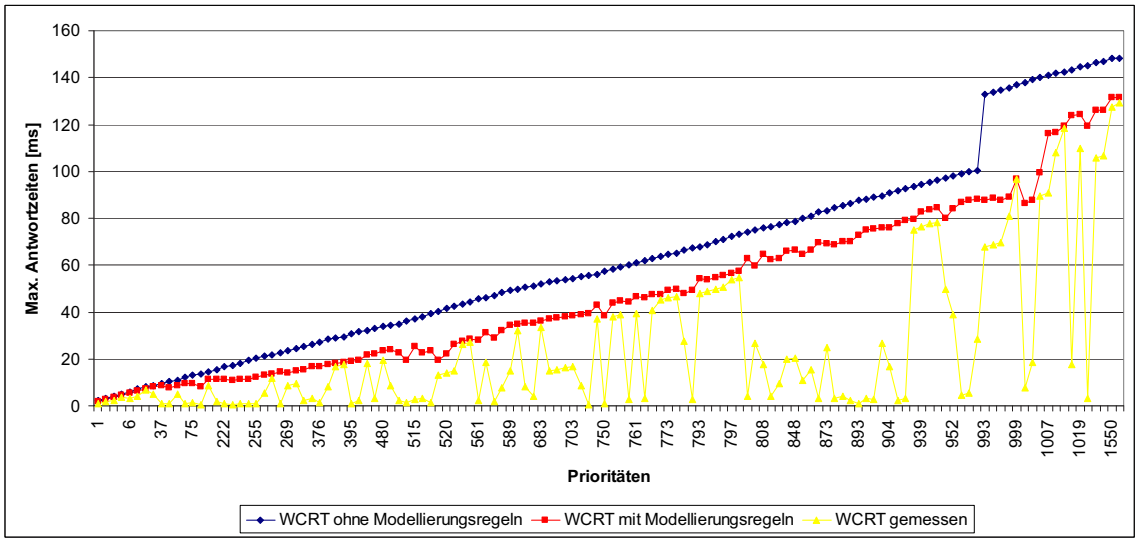


Abb. 6.7.: Vergleich Timing-Bewertung eines CAN-Busses anhand der maximalen Antwortzeiten mit (blau) und ohne (rot) Modellierungsregeln sowie per Messung (gelb) ermittelte Werte

Der Vergleich zwischen roter und gelber Linie zeigt deutlich die erzielte Steigerung der Genauigkeit. Die Ergebnisse, welche unter Berücksichtigung der Modellierungsregeln ermittelt wurden, spiegeln wesentlich exakter den realen Worst-Case wieder. Im Mittel beträgt die Verbesserung der Ergebnisse 15ms, im maximalen Fall sind es 52ms bei CAN-Priorität 1003.

Den Mehrwert des in Abschnitt 6.1.3 vorgestellten Algorithmus zu optimierten Vergabe von Offsets für CAN-Steuergeräte zeigt Abbildung 6.8. Die Startzeit beträgt $T_{start} = 330ms$ und die Periode des COM-Tasks liegt bei $T_{com} = 10ms$. In *blau* ist eine lokale Vergabe der Offsets dargestellt. Diese wurde über das Genierungswerkzeug *GENy* von Vector-Informatik durchgeführt. In *orange* ist die globale Optimierung der Offsets dargestellt. Bei der lokalen Optimierung ist eine deutliche Anhäufung der Botschaften im Zeitraum zwischen 0ms und 100ms zu erkennen. Bei der globalen Optimierung sind die Werte dagegen gleichmäßig für über die gesamte Startzeit verteilt. Diese Verteilung bringt den Vorteil, dass in der Aufstartphase die Gateway-Steuergeräte nicht mit langen Bursts auf allen Bussen rechnen müssen.

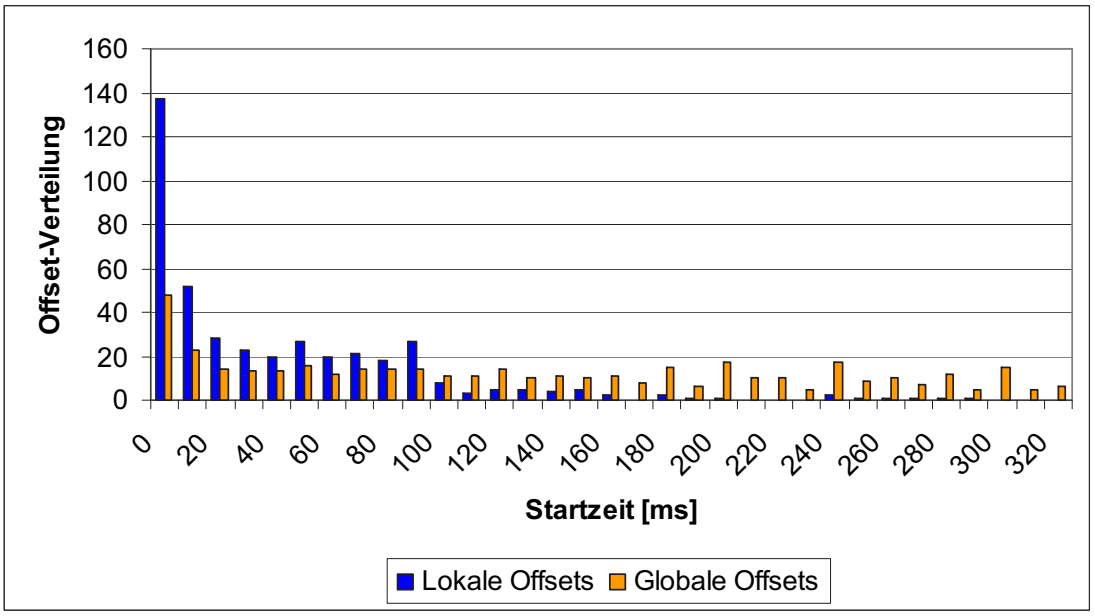


Abb. 6.8.: Vergleich der Offsetwerte anhand der Verteilung über die Startzeit: Lokale Vergabe (blau) und globale Vergabe (orange)

Die resultierenden Antwortzeiten für einen CAN-Bus sind in Abbildung 6.9 aufgezeigt. Die Verbesserung durch die Optimierung ist deutlich zu erkennen. Nur bei zwei Botschaften sind die Antwortzeiten der lokalen Vergabe besser. Da es sich bei dem Optimierungsalgorithmus, um eine Heuristik handelt, können solche Effekte nicht ausgeschlossen werden. Die vielen Auswertungen haben jedoch gezeigt, dass bei allen verwendeten Systemen eine deutliche Verbesserung erzielt werden konnte.

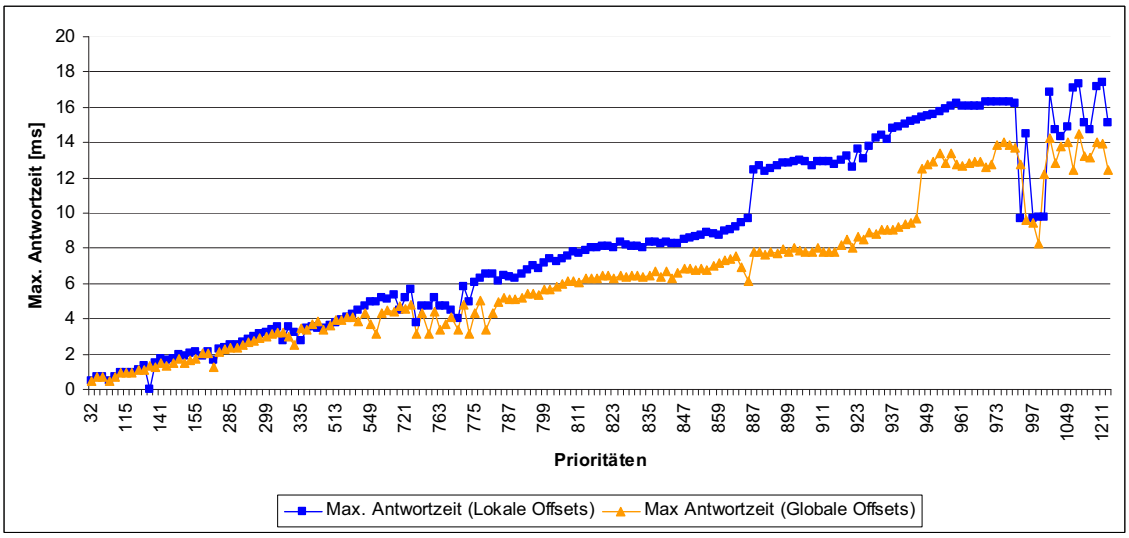


Abb. 6.9.: Vergleich der resultierenden maximalen Antwortzeiten: Lokale Vergabe (blau) und globale Vergabe (orange)

6.4. Modellierungsregeln für das Timing-Verhalten von Gateways

Ein Steuergerät mit Routing-Aufgaben (Gateway) stellt einen Spezialfall dar. Je nach Integrationsumfang sind zusätzlich zu den Routing-Aufgaben auch noch Applikationen auf dem Steuergerät integriert. Ein solches System kann in unterschiedlichen Granularitätsstufen in einer Timing-Bewertung berücksichtigt werden. In Abbildung 6.10 sind drei unterschiedliche Modellierungsgranularitäten für das System aufgezeigt. Das linke Modell zeigt ein einfaches Gateway-Modell. Hier wird für die interne Routingzeit ein fester Wert angenommen. Weiterhin kann das Scheduling mit berücksichtigt werden, d.h. bei mehreren angeschlossenen Bussen ist so eine Abschätzung über die Dimensionierung der Eingangs- und Ausgangspuffer möglich und es kann die Verzögerung durch

gleichzeitig eintreffende Botschaften mit berücksichtigt werden. Im mittleren Modell erfolgt noch die Unterscheidung zwischen *PDU*- und *Signal*-Routing. Für beide Wege kann eine Routingzeit definiert werden. Über die Routingtabelle wird für jede eingehende Botschaft die entsprechende Funktion ausgewählt, dadurch ist eine genauere Last- und Latenzzeit-Abschätzung möglich als mit dem linken Modell. Im rechten Modell wird der komplette Software-Stack des Gateway-Steuergeräts nachgebildet. Auf diese Weise können einerseits die Routingzeiten sicher bestimmt werden und andererseits sind der Einfluss sowie das Zusammenspiel zwischen Routingtasks und Applikationstasks exakt bewertbar. Je nach Anforderung ist zwischen den drei Modellen auszuwählen. Der Modellierungsaufwand steigt von links nach rechts.

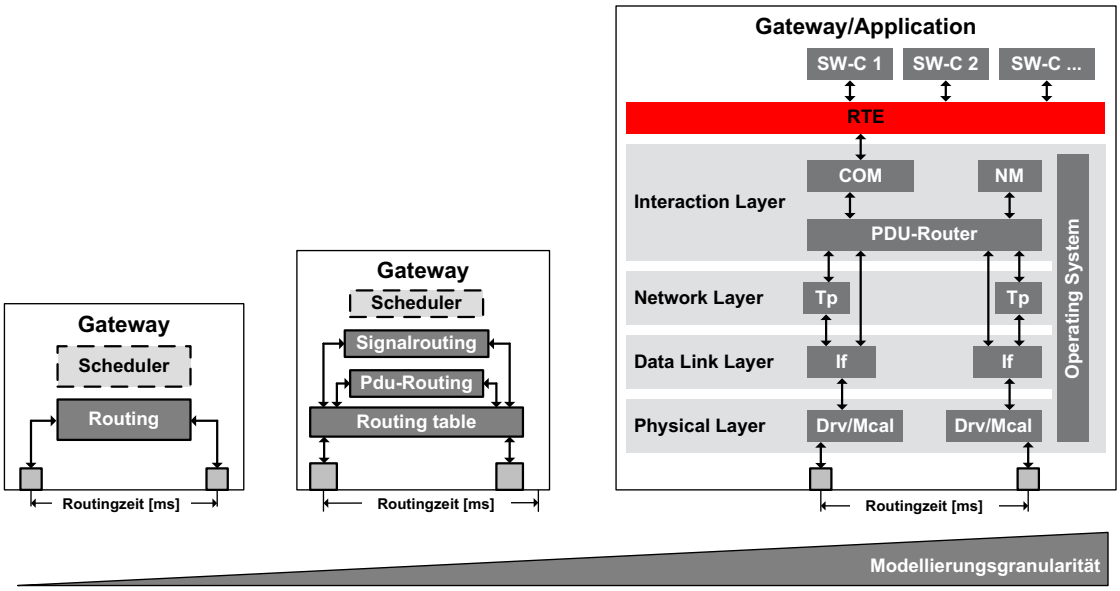


Abb. 6.10.: Granularitäten für die Modellierung von Steuergeräten mit Gateway-Anteilen

Für die vollständige Modellierung eines Gateway-Steuergerätes (rechtes Modell) sind verschiedene Schritte notwendig:

1. Bestimmung der Ausführungszeiten C_i der einzelnen Tasks und Funktionen des Gateway-Steuergeräts.

2. Modellierung des Schedulingmodells (Übernahme der Konfiguration des Betriebssystems) und Annotation der Ausführungszeiten.

3. Modellierung der an das Gateway angeschlossenen Busse.

4. Bewertung des Systems unter der Berücksichtigung des Kommunikationsverhaltens der angeschlossenen Busse.

Für die Ermittlung der Ausführungszeiten sowie für die vollständige Modellierung des Schedulingmodells werden in den nächsten Abschnitten Konzepte und Regeln vorgestellt, welche eine exakte Bewertung eines Gateway-Steuergeräts ermöglichen. Ferner kann damit gezielt die Integration von Gateway- und Applikationsaufgaben auf einem Steuergerät (einer CPU) ausgelegt und abgesichert werden. Weiterhin sind die Anforderungen und Auswirkungen der Konfiguration der Kommunikationssysteme (K-Matrizen, Schedules, etc.) auf ein Gateway-Steuergerät sicher bewertbar.

Ein Großteil der im Folgenden diskutierten Modellierungsregeln und Annotationen sind allgemein gültig und können auch für die Bewertung von reinen Applikationssteuergeräten angewendet werden.

6.4.1. Regeln zur Generierung von Annotationen

Die Ausführungszeiten können auf unterschiedliche Weise bestimmt werden (siehe hierzu Abschnitt 4.1). Hierzu gibt es verschiedene Möglichkeiten:

1. Die Messung der Zeiten direkt auf der Hardware

2. Die Bestimmung der Werte mit einem entsprechenden Simulationsmodell

3. Die Berechnung der einzelnen Ausführungszeiten mittels einer statischen Code-Analyse (WCET-Analyse)

Im Folgenden wird für das in Punkt 3 beschriebene Verfahren ein Konzept vorgestellt, der es ermöglicht aus AUTOSAR-Systembeschreibungen, einen Großteil der notwendigen Annotationen für die WCET-Analyse regelbasiert (automatisch) zu generieren.

Abbildung 6.11 zeigt den Ablauf sowie die notwendigen Eingangsdaten, die für eine WCET-Analyse notwendig sind. Nach der Kompilierung des Codes kann das *Binary* dem Analysewerkzeug übergeben werden. Um eine Analyse durchführen zu können, sind noch Annotationen notwendig, welche u.a. die Schleifengrenzen vorgeben und

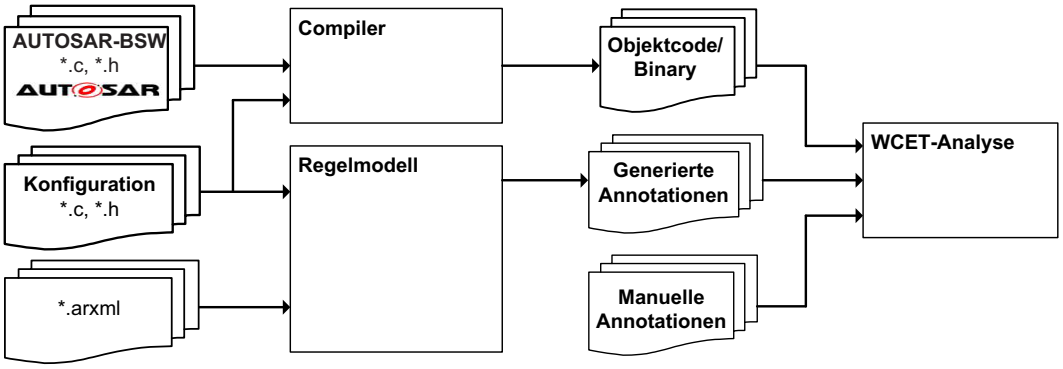


Abb. 6.11.: Daten und Ablauf der Generierung der Annotationen für die statische Code-Analyse

die Zeiger auf die Speicherbereiche setzen (siehe hierzu auch Abschnitt 4.1.2). Diese Informationen können zu großen Teilen über ein Regelmodell aus der AUTOSAR-Systembeschreibung des Steuergerätes abgeleitet werden (*Generierte Annotationen*). Die zusätzlich notwendigen Annotationen sind manuell hinzuzufügen (*Manuelle Annotationen*). Zu den Annotationen im einzelnen zählen:

Generierte Annotationen

1. Globale Einstellungen, z.B. Taktfrequenz der CPU, verwendeter Compiler, etc.
2. Viele Schleifen sind abhängig von der Systemkonfiguration. Die Werte für die Schleifenbegrenzung lassen sich direkt oder indirekt (regelbasiert) aus dem ECU-Extract ermitteln.
3. Einige Code-Segmente (siehe Punkt 2 bei den manuellen Annotationen), die nicht auf dem WCET-Pfad liegen, lassen sich auch automatisiert ermitteln.

Manuelle Annotationen

1. Viele Funktionen innerhalb der AUTOSAR-Basissoftware werden nicht direkt über den statischen Funktionsnamen aufgerufen, sondern über Funktionspointer. Kann die Speicheradresse vom WCET-Werkzeug nicht automatisch aufgelöst werden, müssen hierfür die entsprechenden Speicheradressen zugewiesen werden.
2. Abhängig welcher Betriebsmodi zu untersuchen ist, dürfen bestimmte Funktionen nicht im WCET-Pfad berücksichtigt werden. Zum Beispiel bei der Untersuchung

des Normalbetriebes findet kein Runterfahren des Betriebssystems statt. Die entsprechenden Code-Segmente sind bei der Analyse zu deaktivieren *(auszuschneiden)*.

In Tabelle 6.3 sind exemplarisch einige Parameter des AUTOSAR-ECU-Extractes aufgelistet, welche für die Ableitung der Schleifengrenzen des FlexRay-Interfaces (FrIf) notwendig sind

Tab. 6.3.: Exemplarische Auflistung einer AUTOSAR-Parameter

Funktion	**Parameter im ECU-Extract**	**Beschreibung**
FrCopyWordsInJob	FrIf/FrIfConfig/FrIfCluster/ FrIfController/ /FrIfFrameTriggerings/ FrIfLSduLength	Summe aller PDU Größen in Words (4 Bytes) aller Frames, die im aktuellen Job verschickt werden.
ListFramesInJob	FrIf/FrIfConfig/FrIfCluster/ FrIfJobList/FrIfJob/ FrIfStartSlot und FrIfEndSlot	Liste aller Frames, die im aktuellen Job versendet bzw. empfangen werden.
MaxPdusToGwInJob	FrIf/FrIfConfig/Name/ FrIfPDUDirection/FRIfRx	Maximale Anzahl der PDUs im aktuellen Job, die an das PDU-Gateway weitergegeben werden.

Die Umsetzung des Regelmodells ist in Abbildung 6.12 dargestellt. Als Eingangsdaten dienen der AUTOSAR-ECU-Extract des Gateway-Steuergerätes sowie die globalen Systemeinstellung und eine Beschreibung, welche die Struktur der Annotationsdateien

festlegt. Da einige Parameter, welche für die Ableitung der Annotationen notwendig sind, erst im Konfigurationsschritt des AUTOSAR-Software-Stacks festgelegt werden, sind diese Angaben aus den Konfigurationsdateien zu extrahieren. Die Eingangsschnittstelle sowie das Eingangsmodell basiert auf dem *Open Source Framework ARTOP*. Durch die Verwendung von ARTOP ist die Eingangsschnittstelle sowie das Datenmodell bei einem neuen AUTOSAR Release nicht manuell anzupassen, sondern es kann das entsprechende Update von ARTOP integriert werden. Für die Bereitstellung der zusätzlichen Konfigurationsdaten ist eine Erweiterung des Datenmodells notwendig. Die Befüllung erfolgt über einen Parser für *C-Dateien*. Über den Generator werden die einzelnen Annotationen auf der Basis der vorliegenden Daten generiert. Der *Annotationsparser* erzeugt die entsprechenden Annotationsdateien.

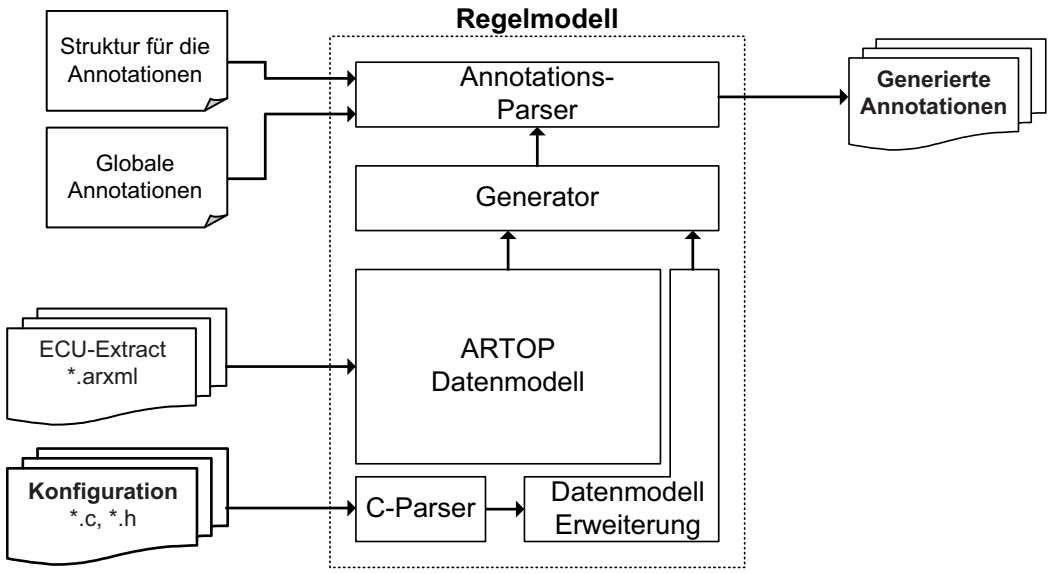


Abb. 6.12.: Umsetzung des Regelmodells für die automatische Generierung der Annotationen

In Abschnitt 8.4 wird das vorgestellte Konzept beispielhaft anhand eines Prototyps für ein Gateway-Steuergerät evaluiert. Dabei werden u.a. die Verbesserungen aufgezeigt, welche durch die einzelnen Annotationsschritte (globale, manuelle und generierte Annotationen) erreicht werden können.

6.4.2. Vollständiges Gateway-Modell für die Timing-Analyse

Um das Scheduling-Verhalten von Gateway-Steuergeräten zukünftig im Detail untersuchen zu können, wurde gemeinsam mit der Firma Symtavision ein Konzept für die vollständige Abbildung des Timing-Verhaltens von Gateway-Steuergeräten in einem Timing-Modell erarbeitet. Ziel der verfeinerten Bewertungsmöglichkeiten ist es, folgende Punkte detailliert untersuchen zu können:

- Auswertung der max. Routingzeiten für jede geroutete Botschaft und jedes Signal.
- Bewertung des Einflusses der Routing-Tasks auf die Applikations-Tasks.
- Bestimmung der notwendigen Puffergrößen im Gateway-Steuergerät.
- Vergleichsmöglichkeit zwischen verschiedenen Abarbeitungsoptionen.
- Bewertung der Einflüsse und Anforderungen, welche sich aus der Konfiguration der Kommunikationssysteme ergeben.

Ein wichtiger Punkt sind die möglichen Abarbeitungsoptionen, diese haben direkten Einfluss auf die Systemperformance und die Routingzeiten eines Gateways. Insgesamt gibt es drei Abarbeitungsoptionen: 1) Interrupt-Modus, 2) Task-Modus und 3) Polling-Modus. Für das Routing von und in Richtung FlexRay wird ausschließlich der Task-Modus verwendet. Für das CAN-CAN-Routing kann je nach Anforderung die eine oder die andere Abarbeitungsoption sinnvoller sein. Anhand der Modellierungselemente von SymTA/S [8] und eines CAN-CAN-Routingpfades werden die Abarbeitungsoptionen im Folgenden dargestellt [128].

Abarbeitungsoption *Interrupt-Modus*

Im Interruptbetrieb (siehe Abbildung 6.13) werden die empfangenen Botschaften (1) direkt innerhalb der Interruptserviceroutine an den entsprechenden Zielbus weitergeleitet (2). Bei dieser Umsetzung kann das Routing nicht durch einen anderen Task oder Interrupt verzögert werden. Bei reinen Gateway-Steuergeräten kann diese Option gut eingesetzt werden. Hat das Steuergerät zusätzlich zum Routing auch noch Applikationsaufgaben abzuarbeiten, kann es bei hoher Routinglast zur großen Verzögerungen der Applikationstasks kommen.

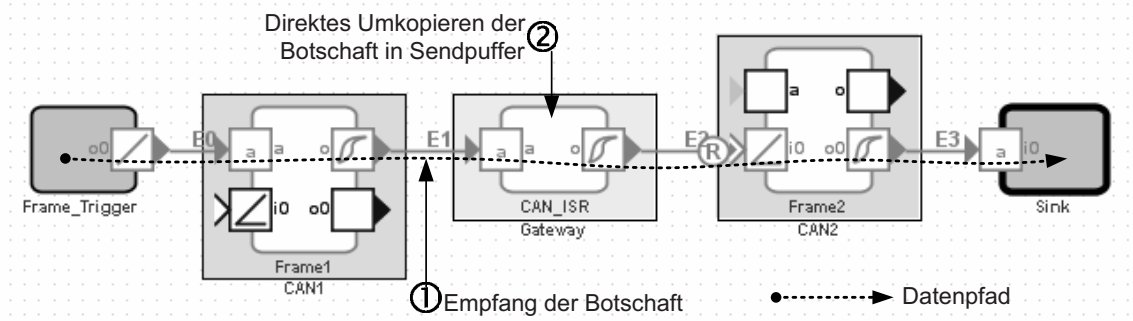


Abb. 6.13.: Beispielmodell für den ISR-Modus

Abarbeitungsoption *Task-Modus*

Im Task-Modus (siehe Abbildung 6.14) kopiert die Interruptserviceroutine die empfangenen Botschaften nur in den internen Speicher (1). Das eigentliche Routing wird von einem zyklisch aufgerufenen Task durchgeführt (2), der die Botschaft aus dem Zwischenspeicher in den Sendepuffer des Zielbusses schreibt (3). Bei dieser Option können Verzögerungen auftreten, bedingt durch den unterbrechbaren *COM_Task*. Bei gleichzeitig aktiven Applikationstasks erhalten diese bei einer entsprechenden Priorisierung der Tasks mehr CPU-Zeit.

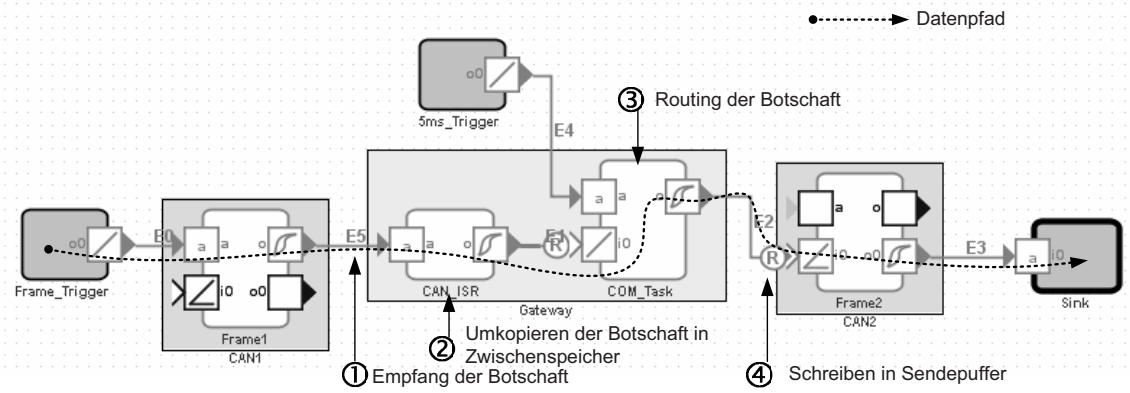


Abb. 6.14.: Beispielmodell für den Task-Modus

Abarbeitungsoption *Polling-Modus*

Im Polling-Modus (siehe Abbildung 6.15) meldet der Controller den Empfang einer Botschaft nicht mittels eines Interrupts. Die Empfangsregister des Gateways werden über einen Pollingtask zyklisch ausgelesen und die empfangenen Botschaften in den Zwischenspeicher kopiert (1). Das Routing wird zyklisch durchgeführt (3). Die Routinglatenz ist sehr stabil. Bedingt durch die Zykluszeiten von Polling- und COM-Tasks sind die Routingzeiten entsprechend länger als beim ISR-Modus.

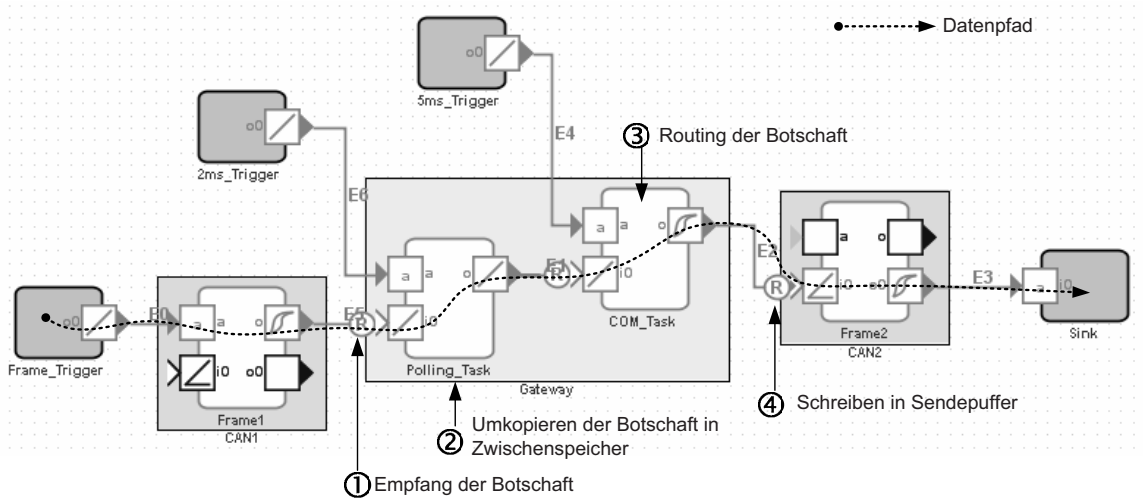


Abb. 6.15.: Beispielmodell für den Polling-Modus

Das zu entwickelnde Analyse-Modell für Gateways soll dabei die Mechanismen von OSEK und AUTOSAR unterstützen und die Möglichkeit bieten die Abarbeitungsoptionen im Detail modellieren zu können. Abbildung 6.16 zeigt die prinzipielle Umsetzung des Modells. Die Grundlage für die Realisierung bildet der AUTOSAR-Softwarestack, wie in Abbildung 3.7 dargestellt. Wichtige Modellierungselemente für das Gateway-Timing-Modell sind:

- Die Interruptservice-Routinen, z.B. für die Modellierung des Empfangs- und Sendeverhaltens für die CAN-Kommunikation.
- Die FlexRay-Jobs, zur korrekten Abbildung des FlexRay-Interfaces.

- Die Gateway-Tasks, für die Modellierung der Frame- bzw. PDU-Routing und Singal-Routing Funktionen.
- Die Speicher zwischen den Elemente, um die Hardware-seitigen Empfangs- und Sendepuffer sowie die Software-Queues korrekt dimensionieren zu können.

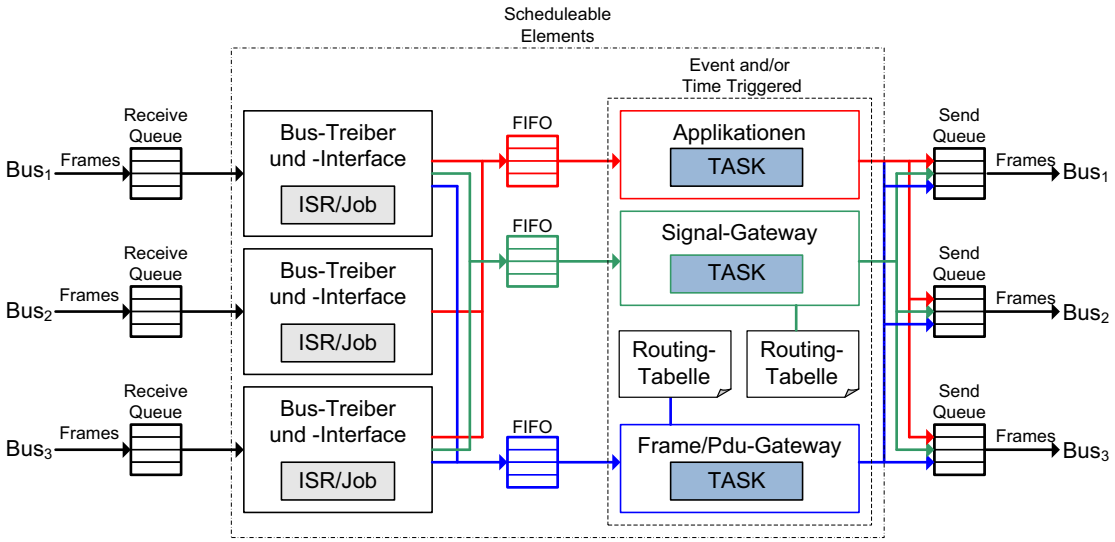


Abb. 6.16.: Konzept für die Umsetzung eines Gateway-Analyse-Modells

Die resultierende Erweiterung des Datenmodells von SymTA/S ist in Abbildung 6.17 aufgezeigt. Für die notwendigen Modellierungselemente sind die entsprechenden Klassen definiert, welche eine vollständige Abbildung eines AUTOSAR-basierten Gateways ermöglichen.

Für die Analyse eines Gateway-Modells wird das Modell über eine Modelltransformation auf das bereits existierende Analysemodell abgebildet. Auf der Basis des transformierten Modells kann dann eine Timing-Analyse erfolgen. Die resultierenden Ergebnisse werden dann den einzelnen Gateway-Modellelementen übergeben und stehen zur Auswertung zur Verfügung.

Für eine weitere Verfeinerung kann in einem nächsten Schritt der Kontrollfluss noch detaillierter in der Analyse berücksichtigt werden. In diesem Kontext laufen aktuell an der Universität Braunschweig einige Arbeiten [105], [59].

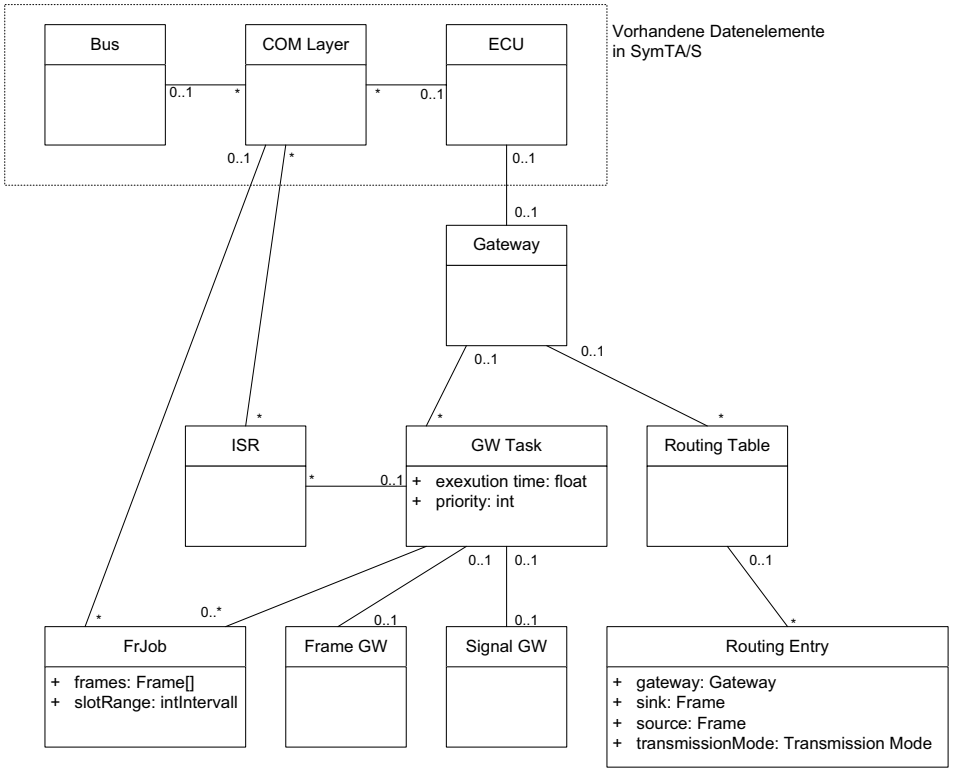


Abb. 6.17.: UML-Diagramm des Gateway-Datenmodells und die Integration in das existierende Datenmodell von SymTA/S [76]

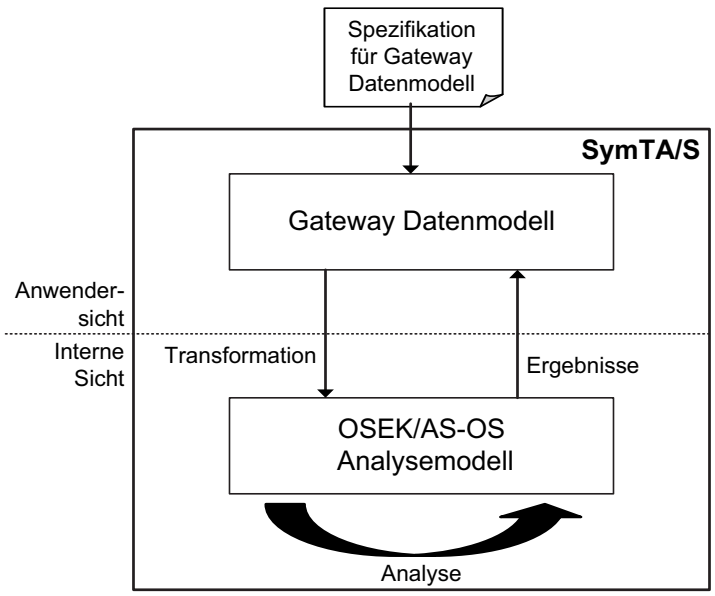


Abb. 6.18.: Konzept für die Umsetzung der Gateway-Analyse innerhalb von SymTA/S [76]

6.4.3. Bewertung der Regeln und Konzepte

Im Zuge der zunehmenden Hochintegration von Steuergeräten erfolgt immer öfter die Umsetzung von Gateway- und Applikationsfunktionalität gemeinsam auf einer CPU. Um hierfür die Auslegung eines solchen Systems in der Entwurfsphase zu unterstützen sowie während der Integrationsphase eine vollständige Absicherung durchführen zu können, liefert das vorgestellte Konzept einen großen Mehrwert. Mittels des vollständigen Gateway-Modells und den exakt bestimmbaren Ausführungszeiten sind gezielt die Auswirkungen der Routinglast auf die Applikationstasks bewertbar. Ein Beispiel hierfür ist in Abbildung 6.19 dargestellt. Das Gateway-Steuergerät koppelt vier CAN- und einen FlexRay-Bus. Die eintreffenden Botschaften werden über die entsprechenden ISR-Routinen abgearbeitet. Weiterhin sind vier Tasks für die Abarbeitung von Services und Applikationen aktiv. Die durchschnittliche CPU-Last beträgt 39%. Die Untersuchung zeigt, dass trotz einer akzeptablen CPU-Last es in bestimmten Fällen, z.B. bei kurzzeitigen Bursts auf allen Bussen, zu Deadline Überschreitungen bei bestimmten Tasks kommen kann (hier: *Task_2ms*).

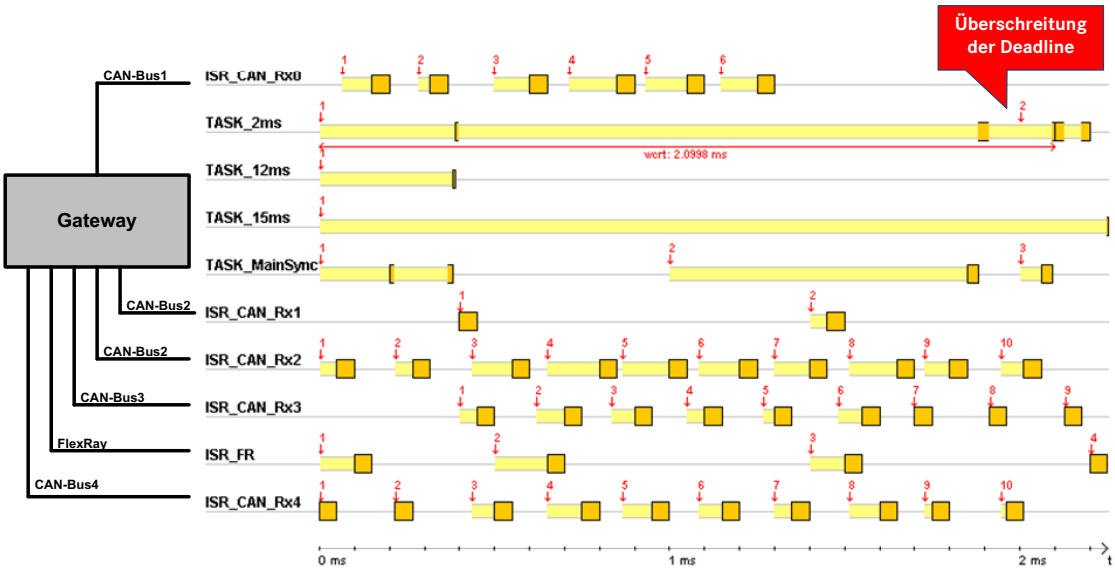


Abb. 6.19.: Beispiel für eine Mehrfachaktivierung, bedingt durch ein hohes Botschaftsaufkommen und eine damit verbundene Interruptlast [128]

An einem weiteren Beispiel soll anhand eines Vergleichs zwischen den verschiedenen Abarbeitungsoptionen (siehe hierzu Abschnitt 6.4.2) auf der Basis eines simulierten Gateway-Modells die Eigenschaften aufgezeigt werden. In Abbildung 6.20 ist das Routingverhalten eines Gateways in Abhängigkeit der drei Abarbeitungsoptionen dargestellt.

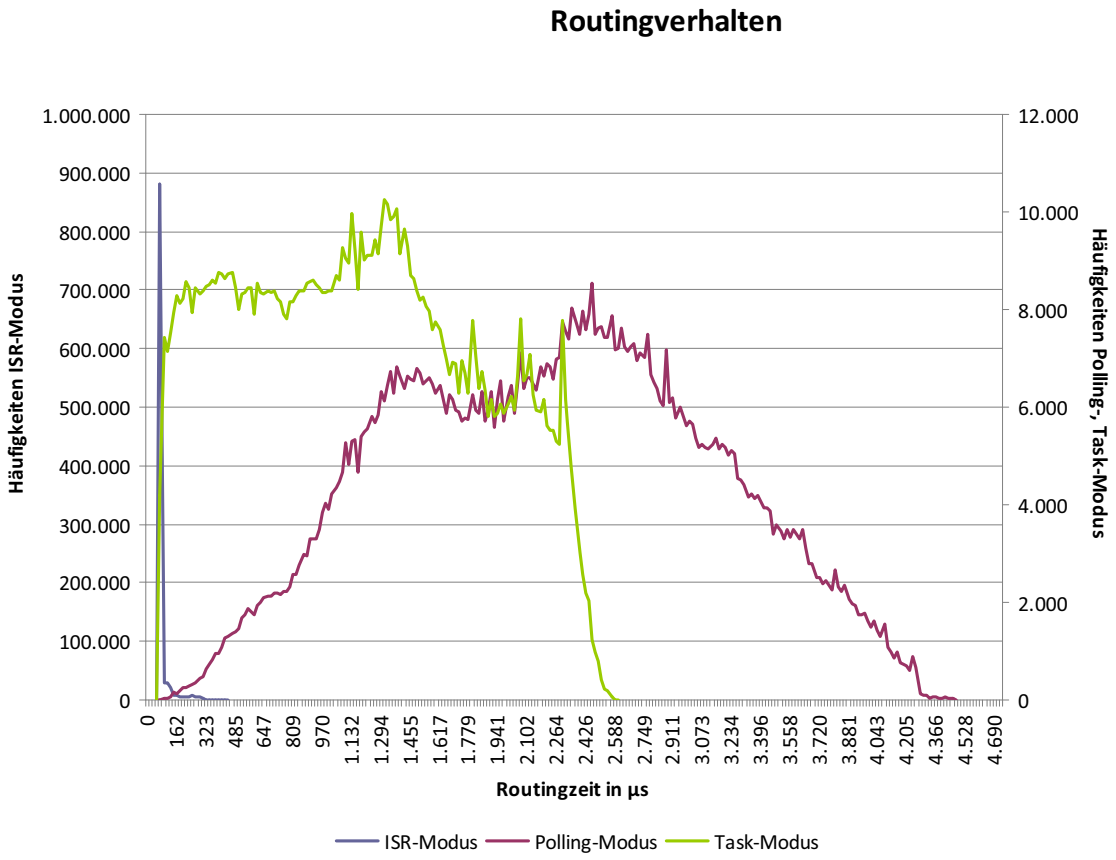


Abb. 6.20.: Vergleich der verschiedenen Abarbeitungsoptionen (ISR-, Task- und Polling-Modus) anhand einer Simulation [113]

Die Routingzeiten im ISR-Modell sind stabil und bewegen sich im Vergleich zu den anderen Modellen in einem engen Zeitbereich. Diese Option hat die kürzeste mittlere Routingzeit mit $70\mu s$, da die Routingzeit ausschließlich von der Ausführungszeit der Interrupt-Service-Routine abhängt. Die Routingzeit im Polling-Modus hängt von ihrem Auftreten in Relation zu dem Polling- und dem Routing-Task ab. Als maximale Routing-

zeit wurden $4610\mu s$ ermittelt. Demnach wurde eine geroutete Botschaft im *Worst-Case* kurz nach einem ausgeführten Polling-Task auf den entsprechenden CAN-Bus geschrieben und dann erst nach $2ms$ gepollt. Anschließend wurde diese nach $2,5ms$ verarbeitet und weitergeleitet. Während der Taskausführung ist dieser durch eine Verdrängung, zum Beispiel durch einen FlexRay-Job oder ISR, verlängert worden. Die minimale Routingzeit von $59\mu s$ kann, unter der Annahme dass der Polling-Task mit dem Task des PDU-Routers aktiviert wurde, auftreten.

Der Verlauf der Kurve über die Verteilung der Routingzeiten beim Task-Modell hat Ähnlichkeiten mit dem ISR- und dem Polling-Modell. Als Mischform beider Modelle besitzen die Botschaften im Task-Modell eine kürzere mittlere Routingzeit von $1,2ms$ gegenüber dem Polling-Modell und sind nicht so weit gestreut. Der Maximalwert setzt sich zusammen aus dem maximalen Offset resultierend aus der zyklischen Aktivierung des PDU-Routers, der Routingzeit und der Ausführung der ISR sowie weiteren Verzögerungszeiten verursacht durch Blockierungs- oder Verdrängungseffekten. Für das Task-Modell wurde eine Routingzeit im *Worst-Case* von $2,6ms$ ermittelt [113].

Die beiden Beispiele zeigen deutlich, dass durch die gezielte Timing-Bewertung mithilfe von Gateway-Modellen die kritischen Systemzustände identifiziert, untersucht und abgesichert werden können. Weiterhin liefern die Modelle ein detailliertes Feedback auf die Anforderungen bzw. Auswirkungen der Buskonfigurationen.

7. Methodik für eine durchgängige Timing-Bewertung

Um die in Kapitel 4 vorgestellten Verfahren in den existierenden E/E-Entwicklungsprozess (siehe hierzu Kapitel 3) zu integrieren, müssen die hierfür notwendigen Informationen identifiziert werden. Diese sind über standardisierte Schnittstellen und Datenformate zwischen den einzelnen Entwicklungs- und Bewertungswerkzeugen auszutauschen. Im Folgenden wird eine Methodik vorgestellt, die eine effiziente und durchgängige Timing-Bewertung von Vernetzungsarchitekturen ermöglicht.

Abbildung 7.1 zeigt wie sich die Timing-Bewertung in den existierenden E/E-Entwicklungsprozess eingliedert. Die in Abschnitt 3.4 definierten Kenngrößen bilden dabei die Grundlage für die Timing-Bewertung. Über die folgenden Schritte kann das Timing-Verhalten innerhalb des E/E-Entwicklungsprozesses bewertet werden:

1. In der Entwurfsphase können über Timing-Abschätzungen für verschiedene Architekturvarianten auf der Basis der Kriterien und Randbedingungen durchgeführt werden. Anhand der ermittelten Kenngrößen (siehe Kapitel 3.4) sind erste Aussagen über das Timing-Verhalten möglich.

2. Auf der Basis der gewonnenen Erkenntnisse aus Punkt 1 kann dann in der Spezifikationsphase die Definition der Timing-Anforderungen und -Randbedingungen für die einzelnen Systeme erfolgen. Diese Angaben können als weitere Anforderungen dann in die Lastenhefte einfließen. Hieraus ergeben sich dann die funktionalen Anforderungen an die Einzelkomponenten in einem verteilten System.

3. In der Implementierungsphase kann auf der Basis der Anforderungen eine Timing-Auslegung der Systeme erfolgen.

4. Im rechten Ast des V-Modells erfolgt die Timing-Absicherung. Erst auf Komponentenebene und dann auf Systemebene, anhand der aus Punkt 1 und Punkt 2 abgeleiteten Testfälle und Bewertungskriterien.

5. Über die während der Testphase durchgeführten Messungen können die Bewertungskriterien und Anforderungen weiter verfeinert werden. Zusätzlich können die Informationen als Eingangsdaten (Frontloading) für die Entwurfsphase der nächsten Architekturgeneration dienen.

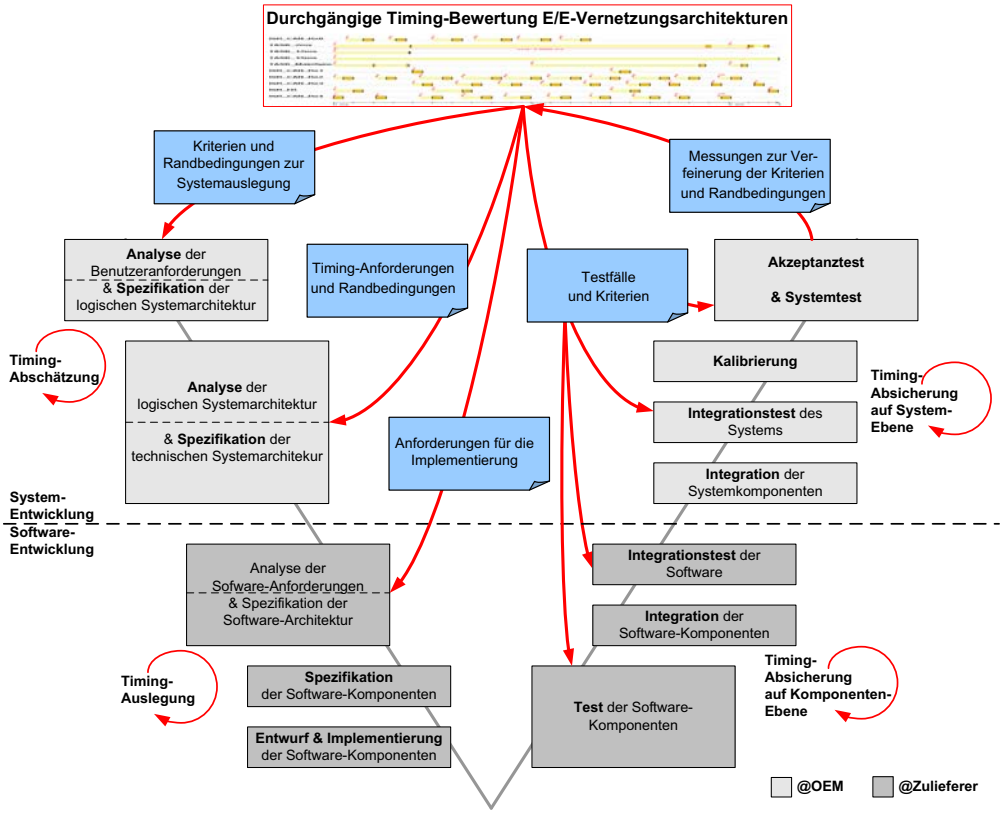


Abb. 7.1.: Durchgängige Timing-Bewertung für den Entwurf, die Auslegung und Absicherung von Vernetzungsarchitekturen im Kraftfahrzeug

Um die in Abbildung 7.1 Timing-Bewertungen durchgängig durchführen zu können wird im Folgenden eine Methodik diskutiert sowie ein Konzept für eine Werkzeugkette vorgestellt, die sich optimal in den existierenden E/E-Entwicklungsprozess eingliedert.

Weiterhin wird ein Konzept vorgestellt, welches die Generierung von sogenannten *Routing-Testpattern* ermöglicht. Dieses Konzept gliedert sich direkt in die vorgestellte Werkzeugkette ein. Den Routing-Testpattern liegt folgender Gedanke zu Grunde: Am Komponenten-HiL des OEMs liegen die Steuergeräte meist als Black-Boxen vor. Bei

den Tests der applikativen Umfänge wurde das Routing bisher nur über eine einfache Restbussimulation stimuliert. Hierfür wurden die als zyklisch definierten Botschaften verwendet. Für die dynamischen Anteile erfolgte keine Berücksichtigung. Dieses Vorgehen verursacht jedoch nur eine kleine bis mittlere Routing-Grundlast während eines Tests der Applikation. Die gezielte Erzeugung von Routinglast gleichzeitig zu funktionalen Tests wurde bisher nicht angewendet.

7.1. Methodik und Werkzeugkette

Die Werkzeugkette ist in abstrakter Form in Abbildung 7.2 aufgezeigt. Als Datenquellen für die Timing-Informationen kommen im aktuellen Entwicklungsprozess verschiedene Werkzeuge in Frage. In der frühen Phase werden meist Architekturmodellierungswerkzeuge verwendet, während in der Entwicklungsphase meist auf einer Entwicklungsdatenbank gearbeitet wird. Um einen effizienten Datenaustausch zu ermöglichen, wurde auf den Standardaustauschformaten aufgesetzt (z.B. AUTOSAR und FIBEX).

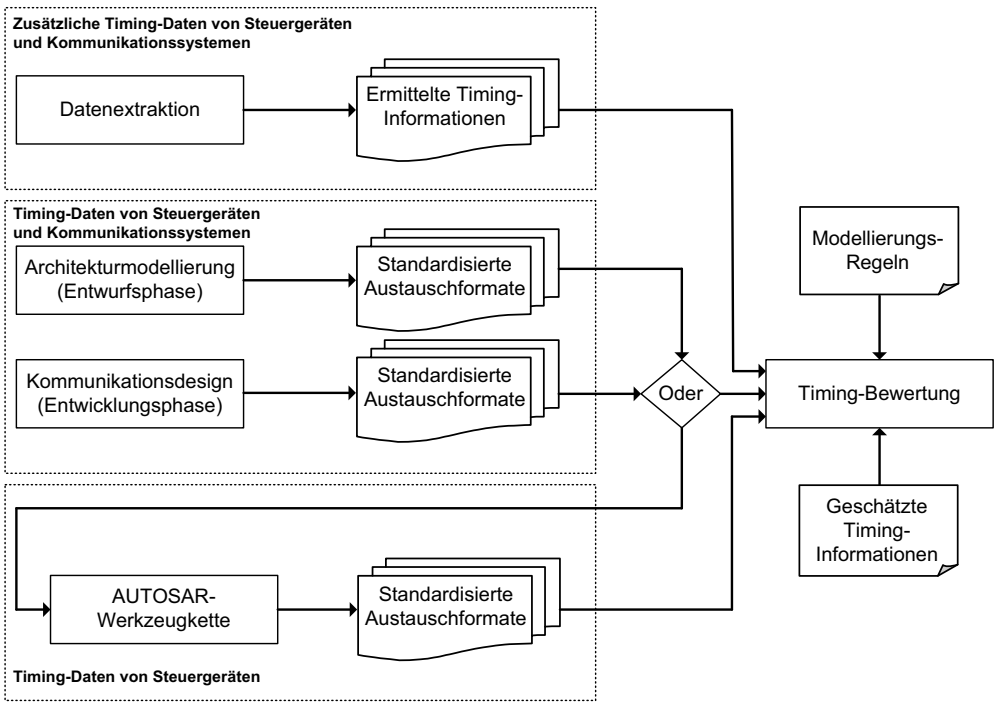


Abb. 7.2.: Konzept der Werkzeugkette für die durchgängige Timing-Bewertung im Entwicklungsprozess von E/E-Architekturen im Kraftfahrzeug

Zusätzlich zu den Daten aus den Architekturwerkzeugen sind noch weitere Informationen für die Timing-Bewertung notwendig. Diese stehen insbesondere in der frühen Entwurfsphase noch nicht zur Verfügung. Über das in Kapitel 5 aufgezeigte Verfahren sind diese Daten, mittels einer Datenextraktion aus bereits existierenden Vernetzungsarchitekturen ermittelbar und können als initiale Werte beim Entwurf von E/E-Architekturen verwendet werden.

Tab. 7.1.: Beispiel für die Umsetzung der Sendetypen *(Transmission Types)* zwischen PREEvision und SymTA/S anhand des FIBEX-Standards [130]

PREEvision	**SymTA/S**	**FIBEX Timing Types**
zyklisch *(cyclicX)*	periodic	cyclic 1: set cyclic 2: unset final repetition: unset debounce: : unset
sofort *cyclicAndSpontanWithDelay*	mixed	cyclic 1: set cyclic 2: unset final repetition: unset debounce: : set
BAF *(cyclicIfAcive)*	direct	cyclic 1: set cyclic 2: unset final repetition: set debounce: : unset
spontan *(spontan)*	direct	cyclic 1: unset cyclic 2: unset final repetition: unset debounce: : unset
schnell *(cyclicIfAcitveFast)*	periodic	cyclic 1: set cyclic 2: set final repetition: unset debounce: : unset

Weiterhin können die Werte auch geschätzt werden. In einer späteren Entwicklungsphase sind diese Annahmen durch die realen Werte ersetzbar. Ferner sind für eine exakte Bewertung die Modellierungsregeln aus Kapitel 6 anzuwenden.

Durch den Einsatz von standardisierten Schnittstellen ist eine Erweiterung des Datenaustausches zwischen den Werkzeugen ohne großen Aufwand möglich. Ein wichtiger Punkt ist die Transformation der Ereignismodelle. Anhand des FIBEX-Standards ist eine exemplarische Umsetzung zwischen den Werkzeugen PREEvision und SymTA/S in Tabelle 7.1 aufgezeigt. Als Beispiele kommen die Daimler-Sendetypen zum Einsatz (siehe Abschnitt 3.1.3).

Eine weitere Optimierung des Datenaustausches zwischen den beteiligten Werkzeugen kann durch den Einsatz der AUTOSAR-Timing-Beschreibungen [15] erreicht werden. Diese sind seit dem AUTOSAR-Release 4.0 Bestandteil der Austauschformate. Für die Verwendung der Timing-Attribute sind jedoch noch in den Datenmodellen der verwendeten Architekturentwicklungswerkzeugen die entsprechenden Attribute zu integrieren und zu pflegen.

7.2. Konzept zur Generierung von Routing-Testpattern

Das im Folgenden vorgestellte Konzept ermöglicht unter Verwendung von statischen Timing-Analyseverfahren und eines *Routing-Pattern-Generators* die Generierung von Routing-Testpattern für den Komponenten-HiL. Diese Testpattern erzeugen gezielt an Steuergeräten mit Gateway-Anteilen eine maximale Routinglast. In Verbindung mit Tests der Applikationen kann so eine genauere Absicherung solcher Systeme durchgeführt werden. Weiterhin kann dabei das korrekte Routingverhalten (z.B. kein Botschaftsverlust, kein signifikanter Anstieg der Routinglatenzzeiten) bei aktiven Applikationen sichergestellt werden. Während einer hohen Belastung des Systems durch Routing ist der Nachweis möglich, dass auch in diesem Fall die applikativen Aufgaben des Steuergerätes korrekt ausgeführt werden. In Abbildung 7.3 ist ein solcher Fall dargestellt.

Über die Routing-Testpattern werden gezielt auf allen CAN-Bussen synchron Bursts von zu routenden Botschaften auf den Bus gelegt, welche die Routinglast an einem Gateway-Steuergerät erhöhen. Ist an das Gateway auch ein FlexRay-Bus gekoppelt, erfolgt die Synchronisation der CAN-Bursts auf den FlexRay-Job, welcher die längste Ausführungszeit besitzt. In einem FlexRay-Job wird das Schreiben und das Lesen der

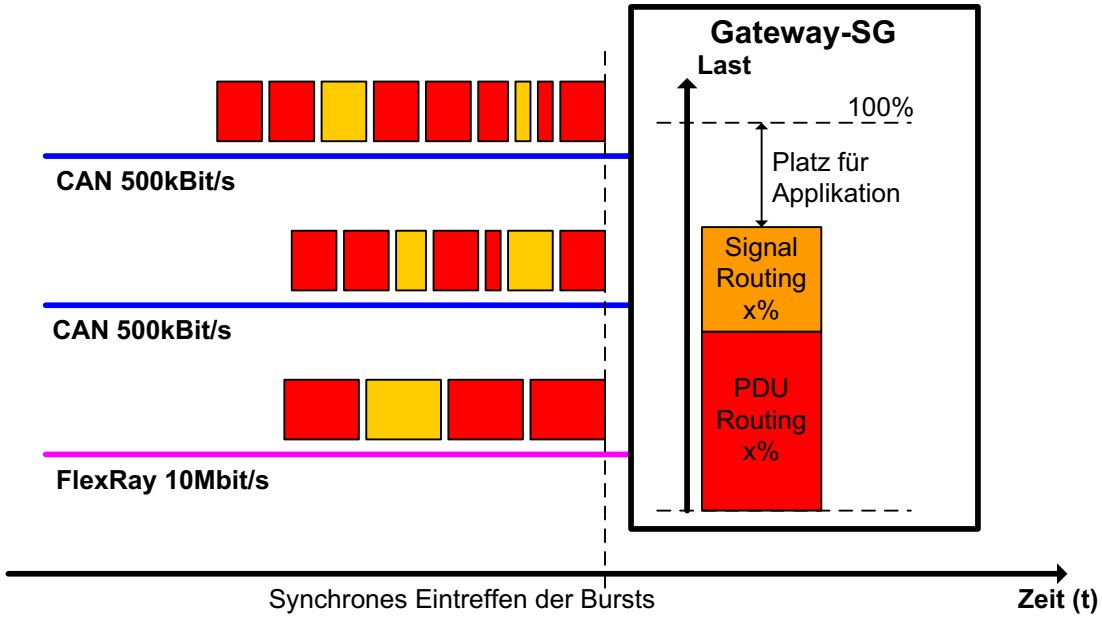


Abb. 7.3.: Synchrones Eintreffen von Bursts auf den CAN-Bussen und Aktivierung des FrJobs mit der längsten Ausführungszeit

Botschaften aus dem *Message RAM* des FlexRay-Controllers realisiert (siehe hierzu die AUTOSAR FlexRay-Schnittstellen-Spezifikation [11]).

Das Konzept für die Generierung ist in Abbildung 7.4 aufgezeigt. Die schlimmsten Bursts auf den CAN-Bussen können über eines der in Kapitel 4 vorgestellten Analyseverfahren berechnet werden. Als Eingangsdaten für die Berechnung werden die K-Matrizen sowie die Offsettabellen und die Jitter der Steuergeräte benötigt. Diese Werte sind entweder vom Zulieferer bereit zu stellen oder es erfolgt die Ermittlung der Werte über das in Kapitel 5 vorgestellte Extraktionsverfahren. Als weitere Information wird die Liste an Botschaften/Signalen benötigt, welche später auf dem Komponenten-HiL für die Prüfung der Applikationen verwendet werden. Diese Botschaften/Signale sind bei der Berechnung der Bursts zu deaktivieren und dürfen nicht mit berücksichtigt werden.

Anschließend wird für jede Botschaft $m_k \in M$ der K-Matrizen die maximale Antwortzeit bestimmt. Die Reihenfolge der Botschaften, die für die Verzögerung einer Botschaft m_k verantwortlich sind, werden als Burst b_k gespeichert. Die Menge B_M enthält alle Bursts der Botschaften einer K-Matrix. Alle ermittelten Bursts $b_k \in B_M$ sowie die Routinginformationen und die K-Matrizen werden im nächsten Schritt an den

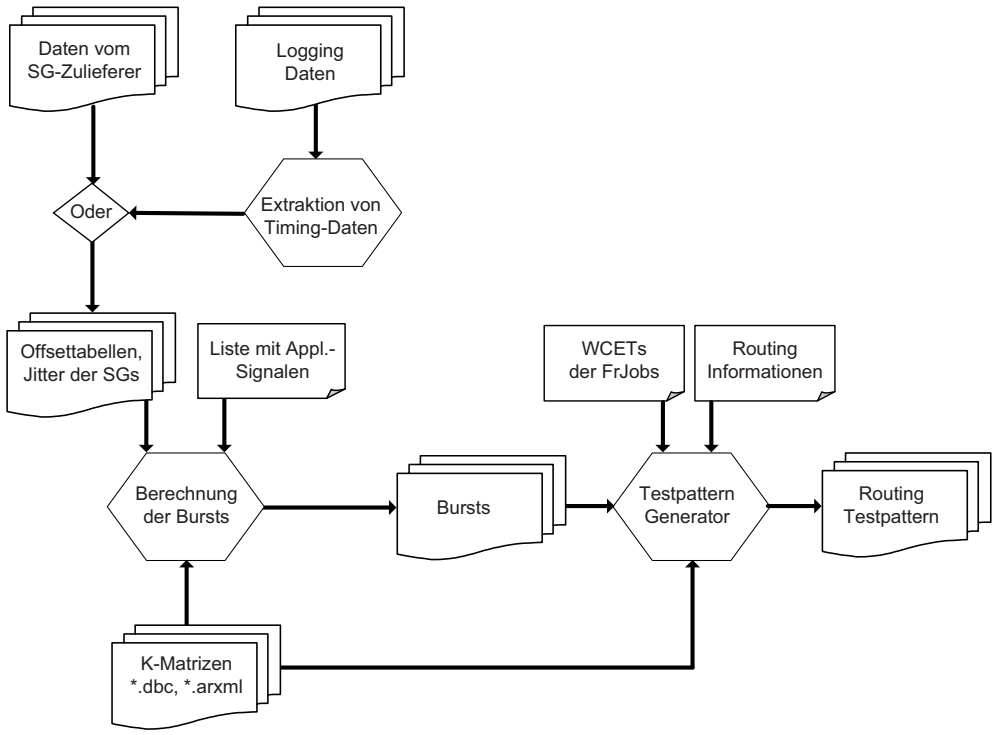


Abb. 7.4.: Werkzeugkette zur Erstellung von Routing-Testpattern für den Komponenten-HiL

Pattern-Generator übergeben. Der Pattern-Generator gewichtet die einzelnen Bursts b_k anhand einer Gewichtungsfunktion 7.1 und den Routinginformationen. Dabei wird über die Gewichtungsfaktoren (n_{pdu}, n_{com} und n_{none}) der Burst b_{wc} ermittelt, welcher für das Gateway die schlimmste Last erzeugt. Die Gewichtungsfaktoren geben an, welcher Routingvorgang am meisten CPU-Zeit belegt. Anschließend werden die schlimmsten Bursts der einzelnen Busse synchronisiert und in einer abspielbaren Loggingdatei gespeichert. Ist auch ein FlexRay-Bus an das Gateway gekoppelt, wird für die Synchronisation der CAN-Bursts der Zeitpunkt ausgewählt, bei dem der längste FrJob zur Ausführung kommt.

$$
\begin{aligned}
b_{wc} = & \max_{b_k \in B}\left(\#PduR_{b_k} \cdot n_{pdu} + \#ComR_{b_k} \cdot n_{com} + \#NoneR_{b_k} \cdot n_{none}\right) \text{ mit} \\
& n_{pdu}, n_{com} \text{ und } n_{none} \in \mathbb{N} \\
& n_{pdu} : \text{Gewichtungsfaktor für PDU-/Botschafts-Routing} \\
& n_{com} : \text{Gewichtungsfaktor für COM-/Signal-Routing} \\
& n_{none} : \text{Gewichtungsfaktor für Botschaften, die nicht geroutet werden}
\end{aligned}
\qquad [7.1]
$$

7.3. Bewertung der Methodik und Konzepte

7.3.1. Bewertung der Methodik

Die vorgestellte Methodik für eine durchgängige Bewertung des Timing-Verhaltens von Vernetzungsarchitekturen und Gateway-Systemen wurde im Rahmen dieser Dissertation in der Forschung/Vorentwicklung der Daimler AG erarbeitet. Die umgesetzte prototypische Werkzeugkette ist in Abbildung 7.5 dargestellt. In der frühen Entwurfsphase werden die Vernetzungsarchitekturen im E/E-Architektur-Werkzeug *PREEvision* der Firma Aquintos modelliert [6]. Über ein standardisiertes Austauschformat (Fibex oder AUTOSAR) sind die für eine Timing-Bewertung notwendigen Artefakte exportierbar. In der Entwicklungsphase kommt das Daimler-interne *Communication Design Framework XDIS* zum Einsatz. Als Exportformate steht hier neben FIBEX und AUTOSAR auch das DBC-Format zur Verfügung. Für die Bestimmung der Ausführungszeiten der Funktionen und Tasks, der relevanten Gateway-Steuergeräte, wird über eine gängige AUTOSAR-Werkzeugkette (hier: *MICROSAR* von der Firma Vector-Informatik [139]) der Software-Stack konfiguriert und kann dann kompiliert werden (hier: *GHS-Compiler* der Firma Greenhills [42]). Die anschließende WCET-Analyse (hier: *aiT* der Firma Angewandte Informatik AbsInt [1]) erhält als Input das Binary des Steuergeräts sowie die generierten (siehe Abschnitt 6.4.1) und manuell erstellten Annotationen. Für die Verifikation der Ausführungszeiten kann parallel dazu eine direkte Messung auf der Hardware erfolgen. Die ermittelten Ausführungszeiten sowie die exportierten Vernetzungsdaten aus PREEvision oder XDIS dienen als Eingangsdaten für das Timing-Analyse-Werkzeug (hier: *SymTA/S* der Firma Symtavision [116]). Für die exakte Modellierung und Analyse werden noch die Modellierungsregeln angewandt. Weiterhin können fehlende Timing-Informationen als Schätzwerte oder ermittelt aus der Datenextraktion (siehe Kapitel 5) eingefügt werden.

Über die in Abbildung 7.5 beschriebene Werkzeugkette ist eine durchgängige Bewertung des Timing-Verhaltens in den verschiedenen Phasen des Entwicklungsprozesses von Vernetzungsarchitekturen und Gateway-Systemen möglich. Im folgenden Kapitel 8 werden verschiedene Beispielsysteme mithilfe des Regelmodells und der entwickelten Werkzeugkette evaluiert, um die Methodik zu validieren.

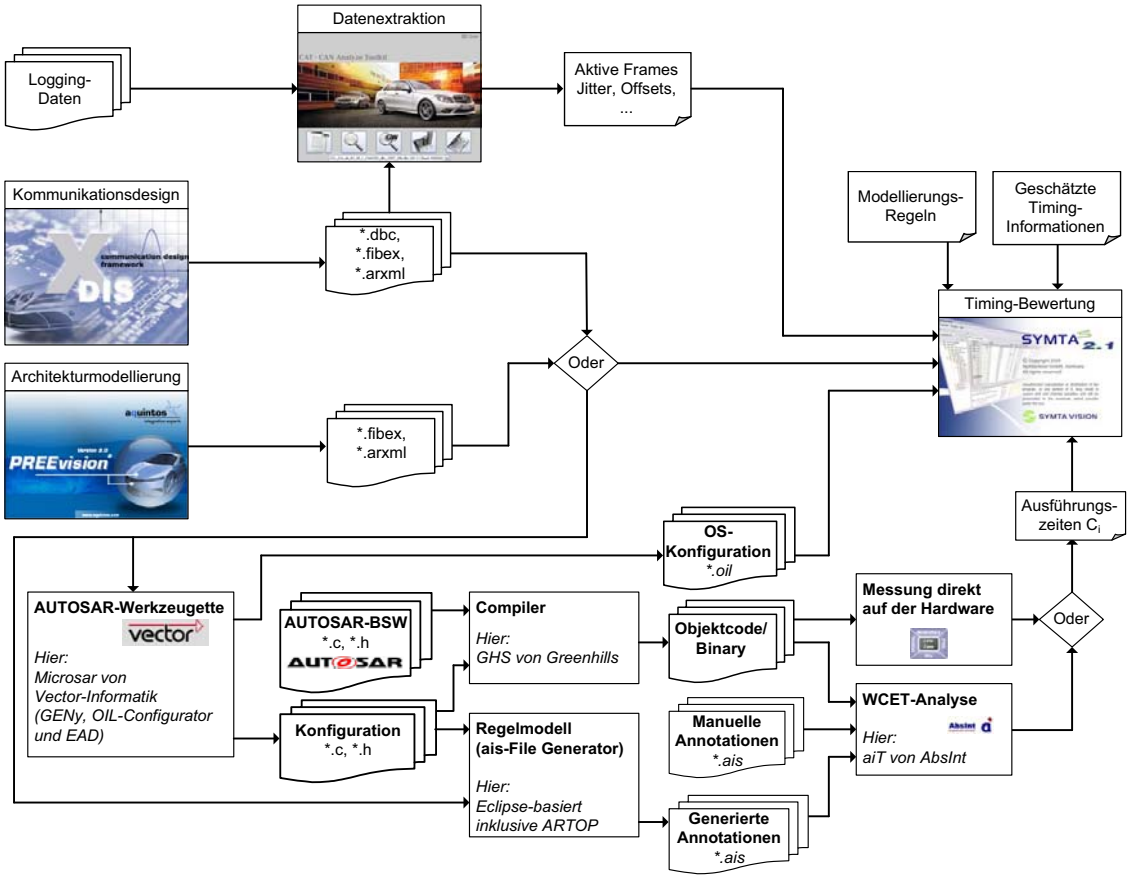


Abb. 7.5.: Umgesetzte Werkzeugkette für die durchgängige Timing-Bewertung im Entwicklungsprozess von E/E-Architekturen [128]

7.3.2. Bewertung des Konzeptes für die Routing-Testpattern

Der Mehrwert des Konzeptes liegt in der gezielten Erzeugung von Routinglast am Steuergerät mit Gateway-Anteilen, während der Durchführung von Tests der Applikation am Komponenten-HiL des OEMs. Dabei liegt das Steuergerät meist als Black-Box vor, d.h. das detaillierte interne Verhalten des Systems ist nicht bekannt. Über die aufgezeigte Vorgehensweise kann abgesichert werden, dass auch bei hohen Routinglasten die Applikationen auf dem Steuergerät korrekt ausgeführt werden, ohne dabei eine detaillierte Kenntnis über die interne Systemkonfiguration haben zu müssen. Der vorgestellte Generierungsablauf lässt sich direkt in den aktuellen Entwicklungsprozess eingliedern. In Abbildung 7.6 ist die Umsetzung für den Serienprüfstand des zentralen Gateway-Steuergerätes des PKW-Bereichs bei der Daimler AG aufgezeigt. Die ersten

Tests wurden bereits damit durchgeführt. Durch eine Instrumentierung der Steuergeräte Software konnte der Nachweis erbracht werden, dass sich die Antwortzeiten der Tasks durch die Routing-Testpattern stark erhöhen. Mit der gezielten Herbeiführung eines definierten Lastzustandes sind die Tests und deren Ergebnisse sicher reproduzierbar. Ein Beispiel für die Erhöhung der Antwortzeiten der Tasks des Steuergerätes ist der Task `ComTask`. Bei der Verwendung einer Restbussimulation lag die maximale Antwortzeit bei $T_{com} = 45\mu s$. Mit verwendeten Routing-Testpattern konnte die maximale Antwortzeit reproduzierbar mit $T_{com} = 168\mu s$ gemessen werden.

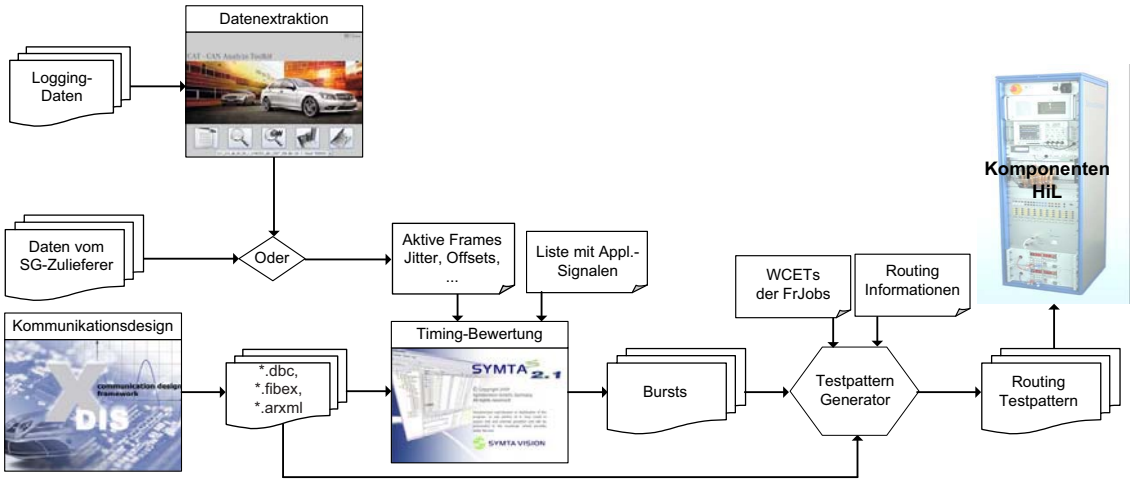


Abb. 7.6.: Realisierte Werkzeugkette zur Erstellung von Routing-Testpattern für den Komponenten-HiL

Durch den Einsatz der Routing-Testpattern am Komponentenprüfstand konnte die Testabdeckung signifikant gesteigert werden. Zukünftig kann bei vorliegenden Konfigurationsdaten (Gateway-Steuergerät als White-Box) sowie den notwendigen Timing-Informationen (Ausführungszeiten) über das in Abschnitt 6.4.2 entwickelte Gateway-Modell, der schlimmste Lastfall exakt bestimmt werden. Dieses kann dann als Grundlage für die Generierung der Routing-Testpattern dienen. Weiterhin gliedert sich das vorgestellte Konzept für die Generierung von Routing-Testpattern direkt in die Werkzeugkette für die Timing-Bewertung ein.

8. Evaluierung

In den vorangegangenen Kapiteln wurde ein Verfahren zur Extraktion von Timing-Informationen aus existierenden automotive Vernetzungsarchitekturen vorgestellt sowie Modellierungsregeln für eine verfeinerte Timing-Bewertung beschrieben. Die entwickelten Konzepte und Verfahren werden im Folgenden anhand von Beispielen aus der Praxis evaluiert. Weiterhin wird anhand der Beispiele die Tauglichkeit der entwickelten durchgängigen Timing-Bewertungsmethodik aufgezeigt. Bei den Beispielen handelt es sich um Vernetzungsarchitekturen und Gateways, die sich aktuell bei der Daimler AG für die nächste Fahrzeuggeneration in der Entwicklung befinden. Im Abschnitt 8.1 werden einige Timing-Bewertungsverfahren anhand eines Topologieausschnittes verglichen. Der Abschnitt 8.2 behandelt die Timing-Bewertung eines CAN-Busses. Im Abschnitt 8.4 wird ein Beispiel für die Extraktion von Betriebsszenarien diskutiert. In Abschnitt 8.4 erfolgt die Evaluierung eines AUTOSAR-basierten Gateways. Der Abschnitt 8.5 dieses Kapitels widmet sich der Bewertung einer Vernetzungsarchitektur anhand eines Ende-zu-Ende-Pfad-Beispiels.

Für alle Evaluierungsbeispiele wird zuerst ein kurzer Überblick über das System gegeben und die eingesetzte Werkzeugkette vorgestellt. Im Anschluss werden dann die erzielten Ergebnisse im Detail diskutiert.

8.1. Vergleich der Timing-Bewertungsverfahren

Um einen Überblick über den Stand der Technik der aktuellen Timing-Bewertungsverfahren zu bekommen, wurde gemeinsam mit der Universität Ulm eine Studie durchgeführt. Ziel war es, anhand eines realitätsnahen Evaluierungsbeispiels aus dem Automotive-Bereich einige der Verfahren einander gegenüberzustellen. Die folgenden Werkzeuge und Verfahren wurden miteinander verglichen. Ein Simulationswerkzeug: *ChronSim* der Firma Inchron [55] sowie drei Timing-Analyseverfahren: *SymTA/S* der Firma Symtavision [116], *EFIXES* der Universität Ulm [135] und *RTC* der Eidgenössischen

Technischen Hochschule Zürich [31]. Die Beschreibung der Verfahren ist in Kapitel 4 zu finden. Die analytischen Verfahren EFIXES und RTC-MPA sowie das Simulationswerkzeug ChronSim wurden an der Universität Ulm evaluiert. Die Modellierung des Beispiels mit dem Analyse-Werkzeug SymTA/S wurde im Rahmen dieser Arbeit durchgeführt.

8.1.1. Übersicht

Als Beispiel für die Evaluierung wurde ein Teilausschnitt einer aktuellen Vernetzungsarchitektur von Mercedes-Benz verwendet. Dieser Ausschnitt umfasste einen CAN-Bus mit 500kBit/s Übertragungsgeschwindigkeit und einen FlexRay-Bus mit 10MBit/s Übertragungsgeschwindigkeit. Am CAN-Bus sind 9 Steuergeräte und ein Gateway gekoppelt. Es werden 85 Botschaften versendet. Der FlexRay hat 6 Steuergeräte und ein Gateway. In Summe werden 28 Botschaften verschickt. Abbildung 8.1 zeigt den Teilausschnitt der Vernetzungsarchitektur. Diese wurde einem aktuellen Fahrzeug von Mercedes-Benz entnommen.

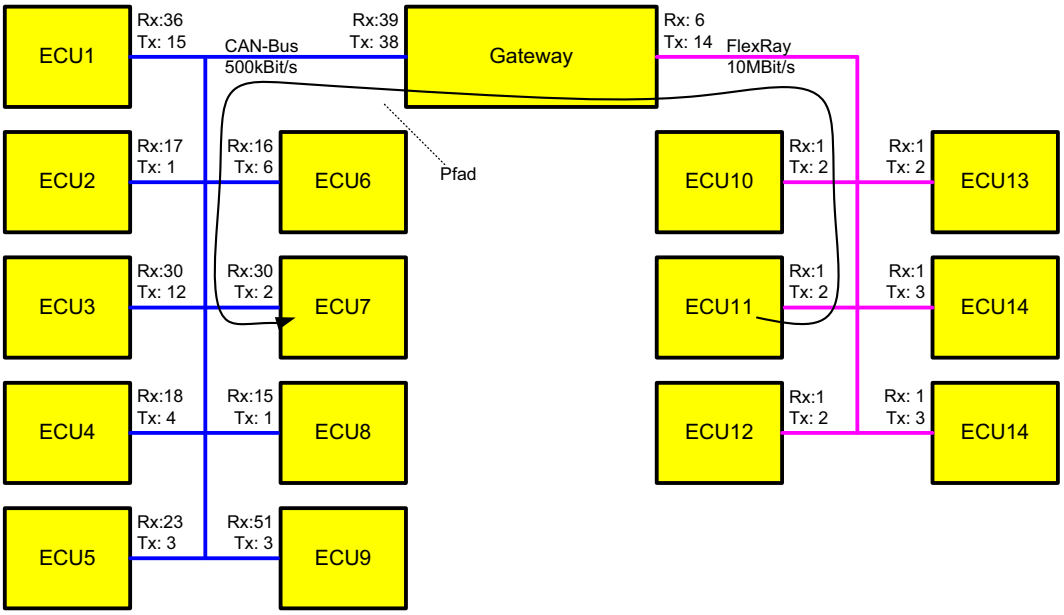


Abb. 8.1.: Ausschnitt der Vernetzungsarchitektur, welcher für den Vergleich der Verfahren verwendet wurde, inklusive der untersuchten Pfade: Pfad1 (*ECU*7 nach *ECU*11 und Pfad2 (*ECU*11 nach *ECU*7)

Im Fokus der Studie standen die Timing-Bewertung des CAN-Busses sowie die Ende-zu-Ende-Latenzzeiten (Reaktionszeiten) ausgewählter Pfade. Pfad1 geht von $ECU7$ nach $ECU11$ und Pfad2 verläuft von $ECU11$ nach $ECU7$. In Abbildung 8.2 ist die interne Taskstruktur von $ECU7$ dargestellt. Abbildung 8.3 zeigt die interne Taskstruktur von $ECU11$. Das *Gateway* ist mit Verzögerungselementen modelliert. Als Routinglatenzzeit wurde $L_{rout} = 1ms$ angenommen. Die Synchronisationseffekte am Übergang zwischen CAN und FlexRay werden bei der Simulation sowie der Analyse mit berücksichtigt.

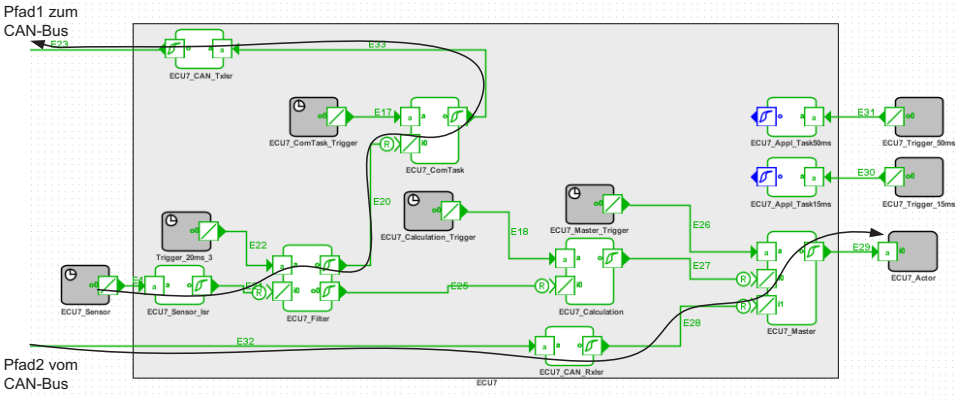


Abb. 8.2.: Modell der $ECU7$ in SymTA/S mit den einzelnen Tasks

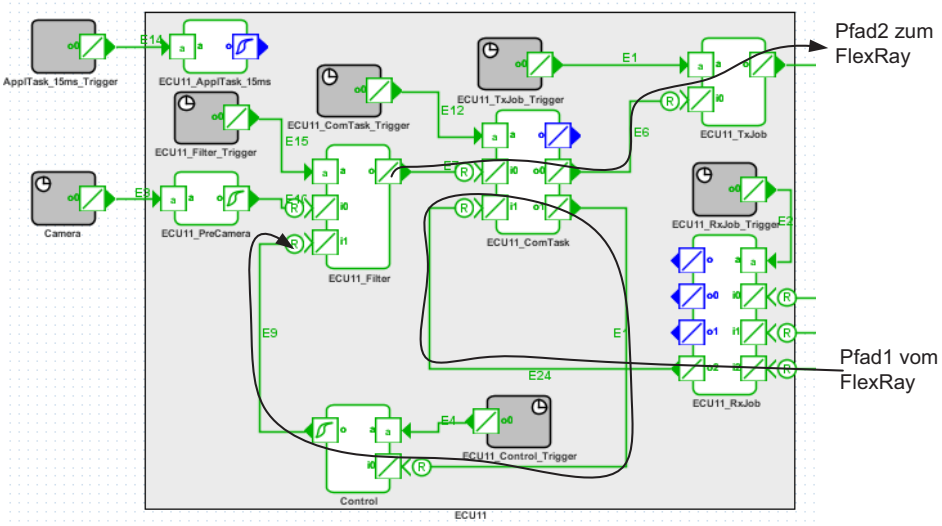


Abb. 8.3.: Modell der $ECU11$ in SymTA/S mit den einzelnen Tasks

8.1.2. Verwendete Werkzeugkette

Das verwendete Modell wurde aus dem Architekturwerkzeug *PREEvision* exportiert und in die ausgewählten Timing-Werkzeuge importiert. Je nach Werkzeug stand eine Standardimportschnittstelle zur Verfügung oder es mussten die Daten manuell angelegt bzw. Skript-basiert eingelesen werden. Das Werkzeug *ChronSim* ist ein Echtzeitsimulator für verteilte eingebettete Systeme. Es können damit sowohl einzelne Steuergeräte und Busse als auch komplette Vernetzungsarchitekturen simuliert werden. Hierfür stehen entsprechende Bibliotheken zur Verfügung (z.B. OSEK, FlexRay, CAN). *SymTA/S* ist ein Timing-Analyse-Werkzeug, welches auf den in Abschnitt 4.3.2 Verfahren basiert. Es stellt wie *ChronSim* alle notwendigen Bibliotheken für Automotive Systeme zur Verfügung. Das Werkzeug *EFIXES* wird am Lehrstuhl von Prof. Slomka an der Universität Ulm entwickelt [64], [63]. Der *RTC* ging aus Forschungsarbeiten am Lehrstuhl von Prof. Thiele hervor, diese wurden in Abschnitt 4.3.2 erläutert.

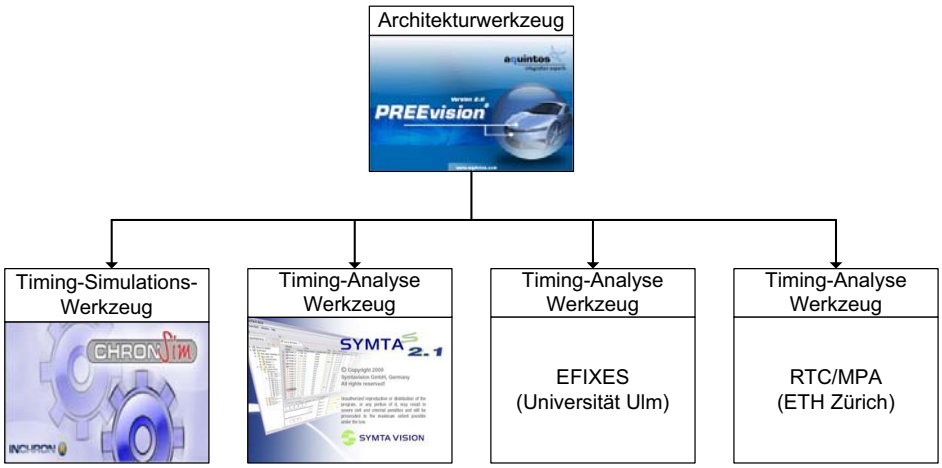


Abb. 8.4.: Vernetzungsarchitektur, welche für den Vergleich der Verfahren verwendet wurde

8.1.3. Diskussion der Ergebnisse

Für die Ergebnisse in ChronSim wurde $1min.$ simuliert. In allen vier Verfahren wurde für den CAN-Bus eine Buslast von 43% ermittelt. Die maximalen Antwortzeiten der Botschaften unterscheiden sich dagegen teilweise erheblich. Dies wird in Abbildung 8.5 deutlich.

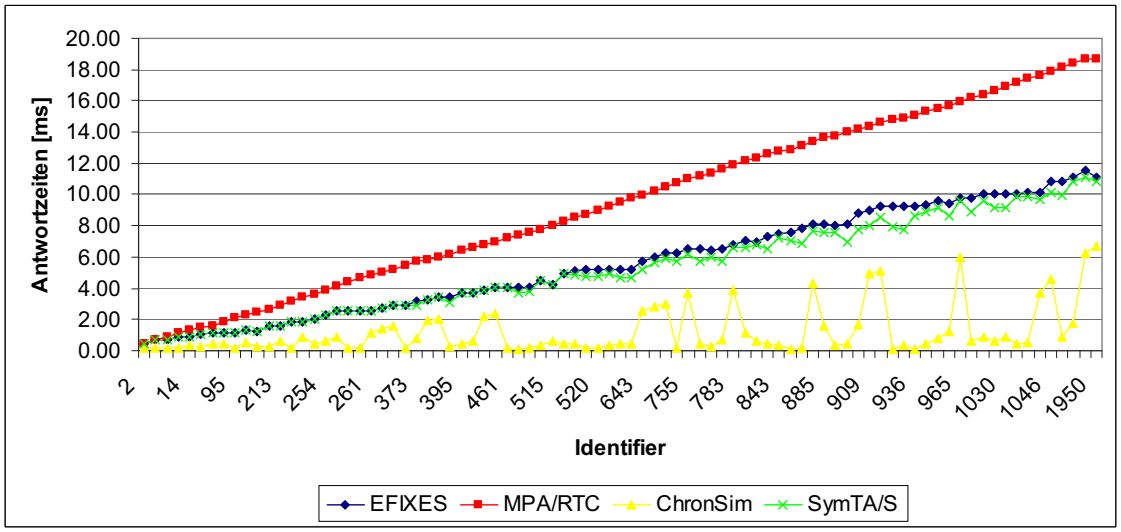


Abb. 8.5.: Vergleich der ermittelten Antwortzeiten des CAN-Busses

Bei den Ergebnissen des RTC-Verfahrens ist die Abweichung am größten. Grund hierfür ist, dass beim RTC die Offsets der Botschaften nicht bei der Berechnung berücksichtigt werden. Die ermittelten Werte von EFIXES und SymTA/S liegen sehr dicht beieinander. An einigen Stellen liegen die Ergebnisse von EFIXES oberhalb von SymTA/S. Der Grund ist das approximative Berechnungsmodell von EFIXES, welches in manchen Fällen überschätzt. SymTA/S dagegen liefert die exakten Werte. Die Ergebnisse der Simulation liegen unterhalb der analysierten Werte und weichen unterschiedlich stark von der Analyse ab. Dieser Effekt liegt darin begründet, dass während der Simulation nicht zwangsläufig der kritische Zustand für jede Botschaft auf dem Bus herbeigeführt wird. Zur Verdeutlichung der Ergebnisse sind in Abbildung 8.6 noch die relativen Antwortzeiten der Botschaften im Verhältnis zu deren Deadline dargestellt.

Die Ergebnisse für die Ende-zu-Ende-Latenzzeiten der Pfade sind in Abbildung 8.7 dargestellt. Als Pfadsemantik wurde die Reaktionszeit gewählt (siehe hierzu Abschnitt 3.4). In SymTA/S werden die Pfadsemantiken in der Analyse automatisch berücksichtigt. In ChronSim kann für die Pfade jeweils eine entsprechende Wirkkette angelegt werden. Bei EFIXES und RTC wurden die Pfade manuell anhand der Einzelergebnisse berechnet. Bei beiden Pfaden liegen die Ergebnisse von EFIXES und SymTA/S sehr dicht beieinander. Die ermittelten Werte von EFIXES sind etwas höher als die von SymTA/S berechneten. Der Grund hierfür ist die Approximation, die der Methodik von EFI-

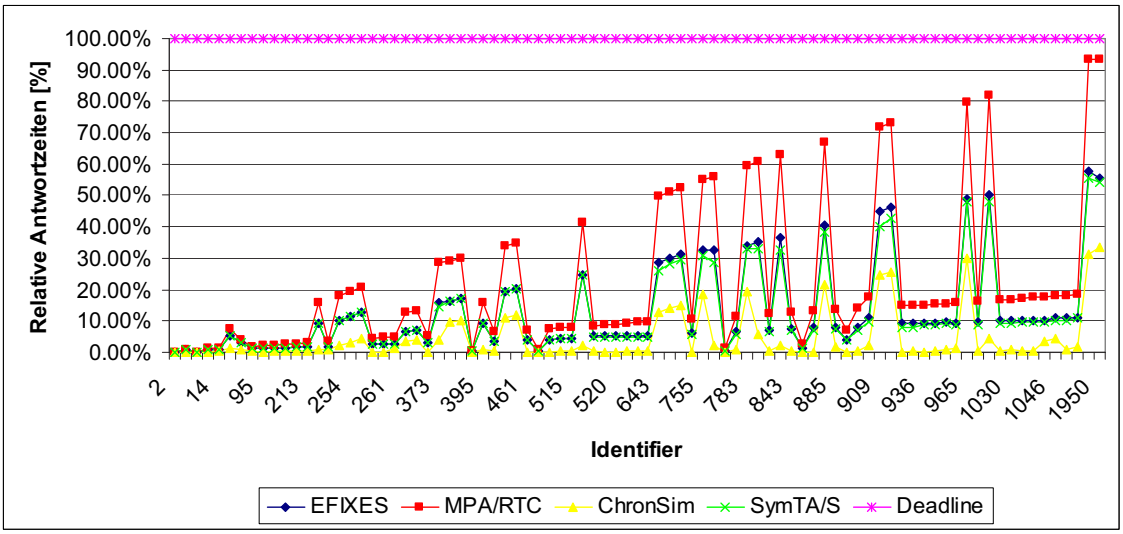


Abb. 8.6.: Vergleich der resultierenden relativen Antwortzeiten in [%] des CAN-Busses

XES zu Grunde liegt. Die Ergebnisse des RTC-Verfahrens liegen oberhalb der ermittelten Werte von EFIXES und SymTA/S. Ursache hierfür sind die höheren Antwortzeiten der CAN-Botschaften, welche mit dem RTC-Verfahren berechnet wurden. Steuergeräteintern liegen die Werte bei allen drei analytischen Verfahren sehr nah beieinander. Bei der Simulation mit ChronSim konnte für Pfad2 der *Worst-Case* fast exakt ermittelt werden. Für den Pfad1 dagegen wurde die maximale Reaktionszeit nicht ermittelt. Der Wert der Simulation für Pfad1 ist 29.433*ms*. Der reale *Best-Case* liegt bei 2.311*ms*, der *Worst-Case* basierend auf dem Ergebnis von SymTA/S hat den Wert 46.219*ms*.

Das Ergebnis macht einen wichtigen Punkt sehr deutlich. Sowohl die analytischen Verfahren als auch bei der Simulation liegt inhärent immer folgende Herausforderung zu Grunde. Bei den analytischen Verfahren ist nicht immer sichergestellt, dass alle relevanten Kontexte bei der Analyse berücksichtigt wurden. Bei der Simulation ist es oftmals schwierig die exakten Eigenschaften und Parameter im Modell zu hinterlegen, welche sicher zum Auftreten des *Worst-Case* innerhalb eines Simulationslaufes führen. Aus diesem Grund ist es hilfreich sowohl Ergebnisse basierend auf analytischen Verfahren als auch von Simulation oder Messung zu ermitteln, um die Qualität der Werte einschätzen zu können.

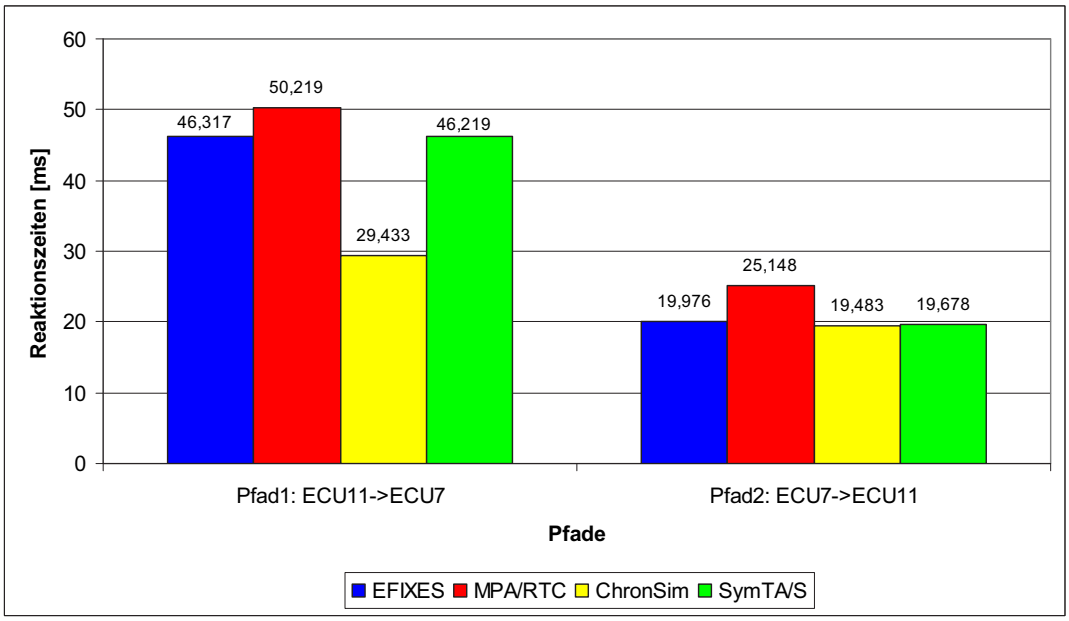


Abb. 8.7.: Vergleich Latenzzeiten der Ende-zu-Ende-Pfade anhand der Reaktionszeiten

8.2. Timing-Bewertung eines CAN-Busses

Für die korrekte Abbildung des Timing-Verhaltens von Kommunikationssystemen wurden in Kapitel 6 Modellierungsregeln diskutiert. An einem Praxisbeispiel wird im Folgenden die exakte Modellierung des Timing-Verhaltens eines CAN-Busses evaluiert. Weiterhin erfolgt der Vergleich eines Busses anhand von zwei Baureihengenerationen.

8.2.1. Übersicht

Als Beispiel dient ein CAN-Bus zweier Baureihen derselben Fahrzeugklasse. Die eine Variante *Variante 1* basiert auf der aktuellen Baureihe, d.h. alle Daten sind verfügbar und es können Messungen im Fahrzeug vorgenommen werden. Für die zweite Variante *Variante 2* wird eine Baureihe verwendet, die sich aktuell in der Entwicklung befindet. In Abbildung 8.8 ist der relevante Topologie-Ausschnitt dargestellt. Die Variante 1 umfasst 26 Steuergeräte und 1 Gateway sowie 160 Applikationsbotschaften. Die Variante 2 enthält 24 Steuergeräte und 1 Gateway sowie 183 Applikationsbotschaften. In beiden Varianten wird der CAN-Bus mit einer Übertragungsgeschwindigkeit von $125 kBit/s$ betrieben.

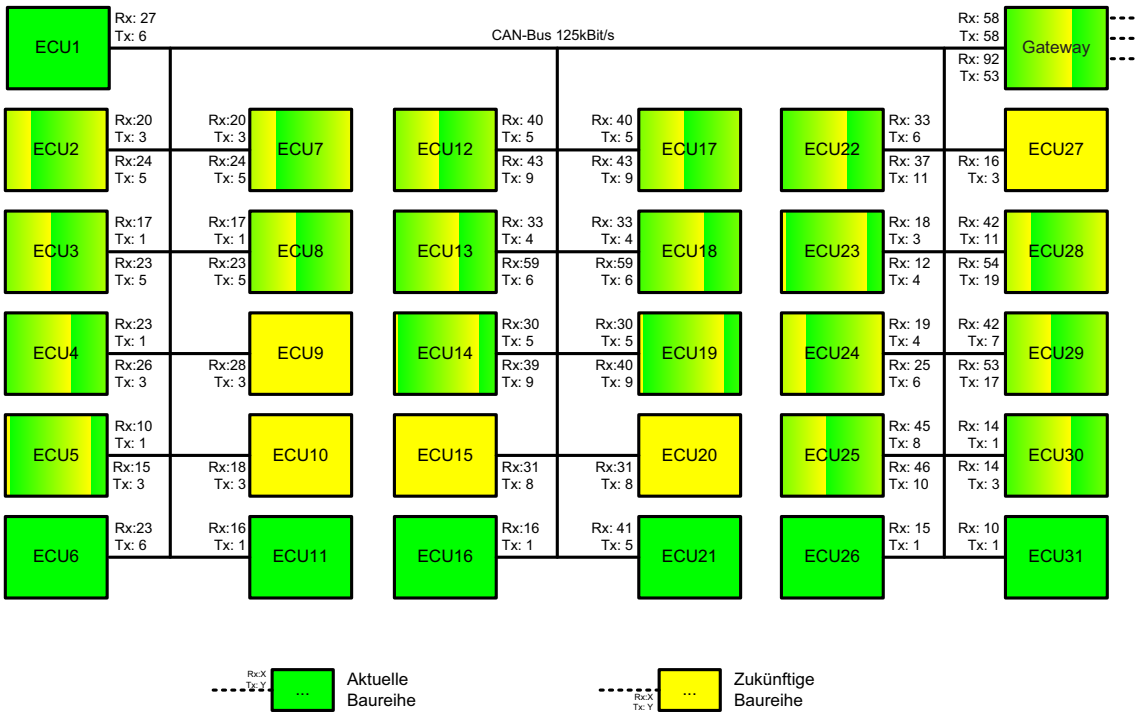


Abb. 8.8.: Topologie-Ausschnitt mit den beiden Konfigurationen des CAN-Busses (125kBit/s), aktuelle Baureihe (grün) und zukünftige Baureihe (gelb)

8.2.2. Verwendete Werkzeugkette

In Abbildung 8.9 ist die verwendete Werkzeugkette für die vergleichende Evaluierung des CAN-Busses darstellt. Die Analyse des CAN-Busses der aktuellen Baureihe erfolgte auf Basis der vorhandenen K-Matrizen (Export aus dem Daimler-internen *Communication Design Framework XDIS*). Zusätzlich wurden aus Messungen die Offsettabellen der Steuergeräte sowie die Jitter extrahiert und in das Analysemodell importiert. Für die Analyse der zukünftigen Baureihe wurde die Buskonfiguration wie in Abschnitt 6.3.5 beschrieben und im Architekturwerkzeug *PREEvision* modelliert. Die Generierung der Offsettabellen erfolgte auf zwei verschiedene Arten.

1. Für jedes Steuergerät wurde eine lokale Offsettabelle erzeugt *Variante 2a*.

2. Die Offsettabellen wurden global mit dem in Abschnitt 6.1.3 beschriebenen Algorithmus generiert *Variante 2b*. Als Parameter wurden folgende Werte verwendet: *Startzeit* $T_{Start} = 320ms$, *ComTaskCycle* $T_{Com} = 10ms$ und Anzahl Busse $NbOf$-

Busses = 4, diese ergibt sich über die Anzahl der am Gateway angeschlossenen Busse.

Als Jitter wurde für jedes Steuergerät $J = 0.5ms$ angenommen. Die spontanen Aktivierungen wurden auf 5 begrenzt. Weiterhin erfolgte die Anwendung der Modellierungsregeln für alle Analysemodelle.

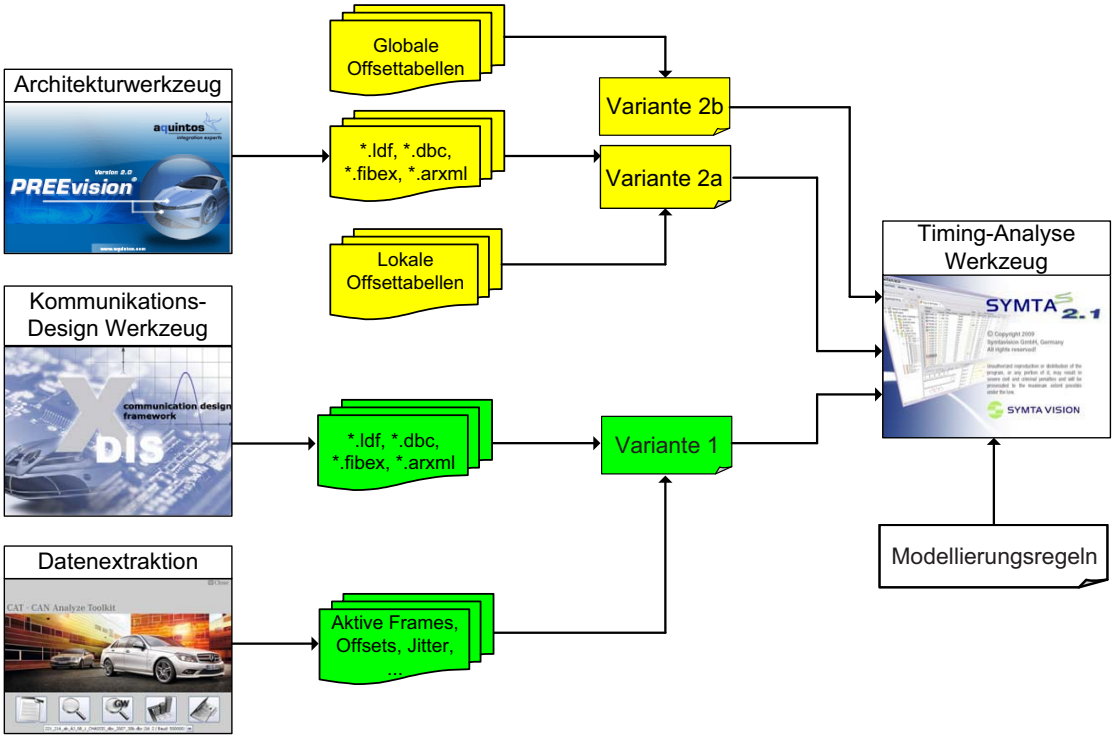


Abb. 8.9.: Verwendete Werkzeugkette für die CAN-Bus-Analyse und den Vergleich des Kommunikationsverhaltens zwischen zwei Baureihengenerationen

8.2.3. Diskussion der Ergebnisse

Für die aktuelle Baureihe (Variante 1) wurde eine zyklische Buslast von 38% berechnet. Die zyklische Buslast bei der neuen Baureihe (Variante 2a/2b) liegt bei 42%. Trotz der höheren Anzahl an Applikationsbotschaften in der neuen Baureihe $(+23)$ fällt der Anstieg der Buslast verhälnismäßig gering aus. Dies liegt darin begründet, dass bei vielen bestehenden Botschaften in der neuen Baureihe die Zykluszeit reduziert wurde.

In Abbildung 8.10 sind die resultierenden maximalen Antwortzeiten der Botschaften dargestellt. Das Resultat für die aktuelle Baureihe zeigt die Linie mit *grün*. Die Ergebnisse für die beiden Offset-Varianten der neuen Baureihe stellt die Linie mit *blau* für die Variante 2a und die Linie mit *rot* für die Variante 2b dar.

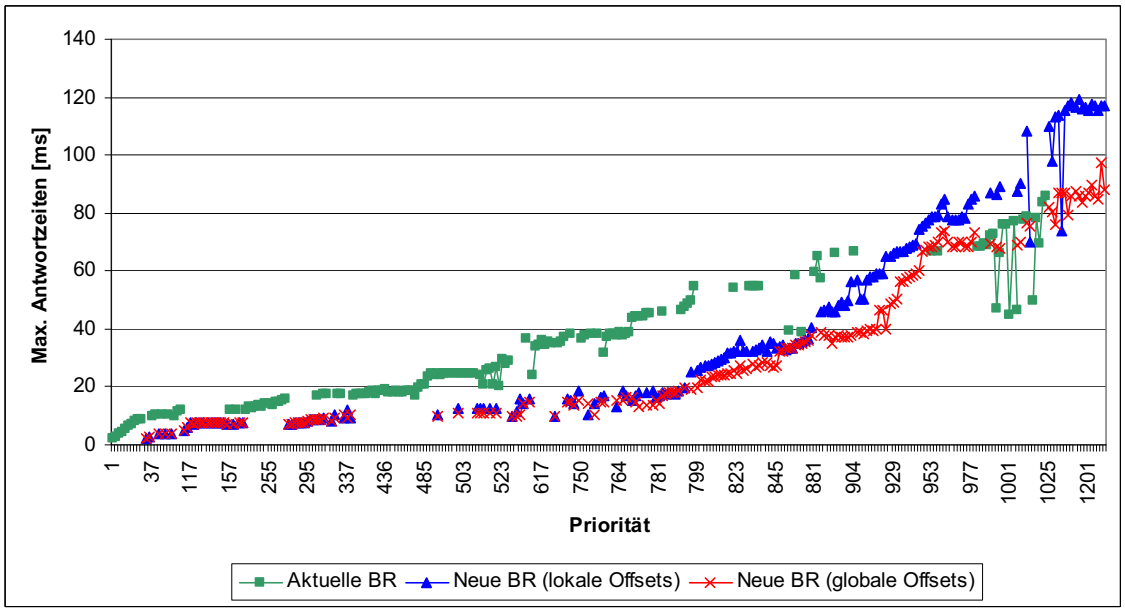


Abb. 8.10.: Vergleich der berechneten maximalen Antwortzeiten für die aktuelle Baureihe (grün) sowie die neue Baureihe mit lokal (blau) und global (rot) vergebenen Offsets

Die maximale Verbesserung der Antwortzeit zwischen der Variante 1 und Variante 2b liegt bei 31*ms* sowie im Mittel bei 15*ms*. Durch die globale Optimierung des Offsets konnte eine Verbesserung von 38*ms* maximal und 8*ms* im Mittel erzielt werden. Die Ergebnisse zeigen die deutliche Verbesserung, welche durch die Vergabe von globalen Offsets erreicht werden kann.

Weiterhin werden in Abbildung 8.11 die Antwortzeiten (blau) den Deadlines der Botschaften mit Zykluszeiten bis 50*ms* (Variante 2b) gegenübergestellt. Keine der Botschaften überschreitet deren Deadline und auch der Abstand ist noch hinreichend groß, für spätere Erweiterungen. In Abbildung 8.12 sind noch die relativen Deadlines der Botschaften für die Variante 2b aufgeführt. Der schlimmste Fall liegt bei 56% der Zykluszeit für die Botschaft mit Priorität 501. Im Mittel liegt die relative Antwortzeit bei 13%.

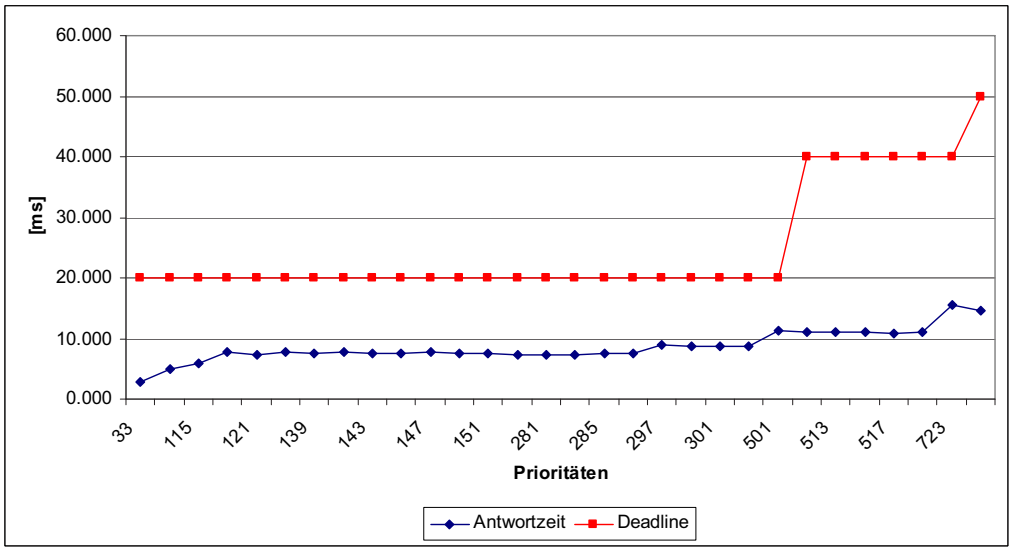


Abb. 8.11.: Gegenüberstellung der Antwortzeiten (blau) der Botschaften und deren Deadlines (rot); neue Baureihe mit globalen Offsets

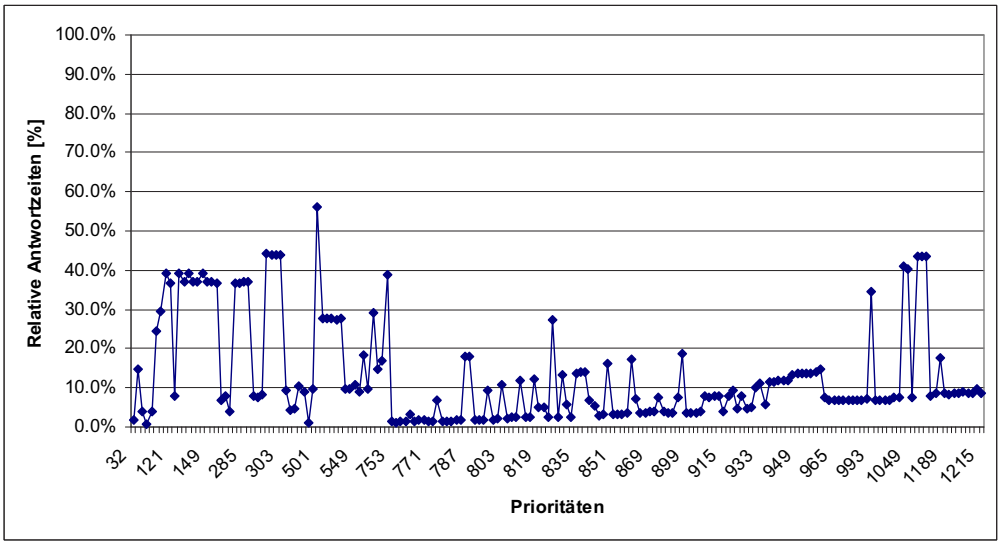


Abb. 8.12.: Relative Antwortzeiten der Botschaften; neue Baureihe mit globalen Offsets

8.3. Timing-Bewertung eines CAN-Busses via Szenarien

In Kapitel 5 wurde ein Verfahren vorgestellt, welches u.a. die Möglichkeit bietet, auf Basis von Fahrzeugmessungen (Loggingdaten) Betriebsszenarien zu extrahieren. Eine solche Datenextraktion wird im Folgenden anhand eines aktuellen CAN-Busses einer Baureihe von Mercedes-Benz diskutiert.

8.3.1. Übersicht

Als Beispiel dient ein Loggingdatensatz der Messung eines *Innenraum-CANs*. Über diesen Bus sind die Innenraumsteuergeräte miteinander vernetzt. Der Bus wird mit einer Übertragungsrate von $B = 125 kBit/s$ betrieben. Es sind 22 Steuergeräte an den Bus gekoppelt. Insgesamt werden 116 Botschaften zwischen den Steuergeräte ausgetauscht.

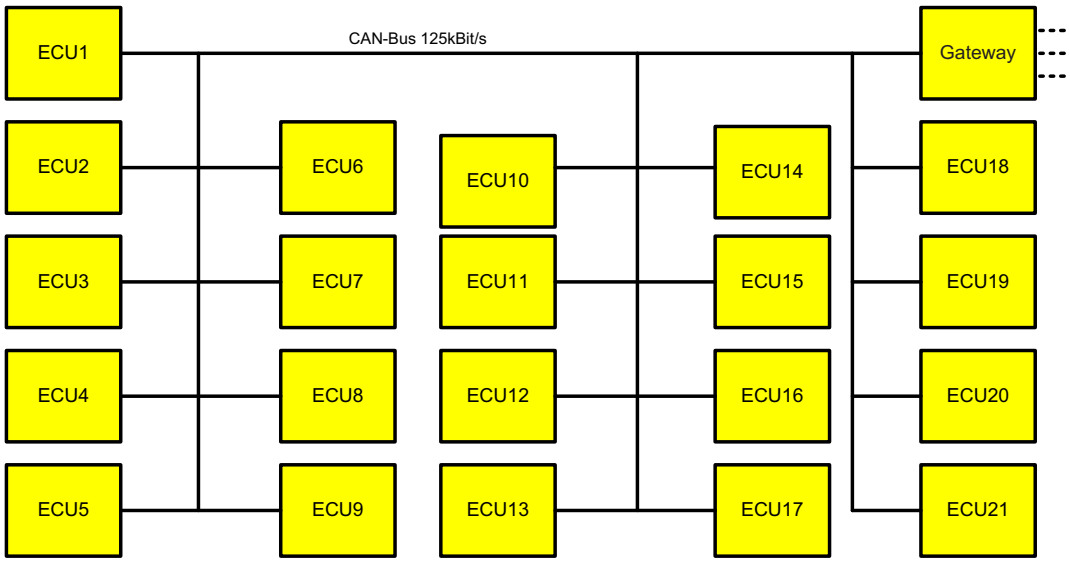


Abb. 8.13.: Topologie-Ausschnitt mit dem analysierten CAN-Bus einer aktuellen Baureihe von Mercedes-Benz Cars

8.3.2. Verwendete Werkzeugkette

Die verwendete Werkzeugkette ist in Abbildung 8.14 dargestellt. Der Loggingdatensatz sowie die K-Matrizen aus *XDIS* dienen als Input für das Werkzeug zur Datenextraktion.

Die extrahierten Szenarien und Timing-Attribute werden inklusive der K-Matrizen in das Timing-Analyse-Werkzeug *SymTA/S* importiert und analysiert.

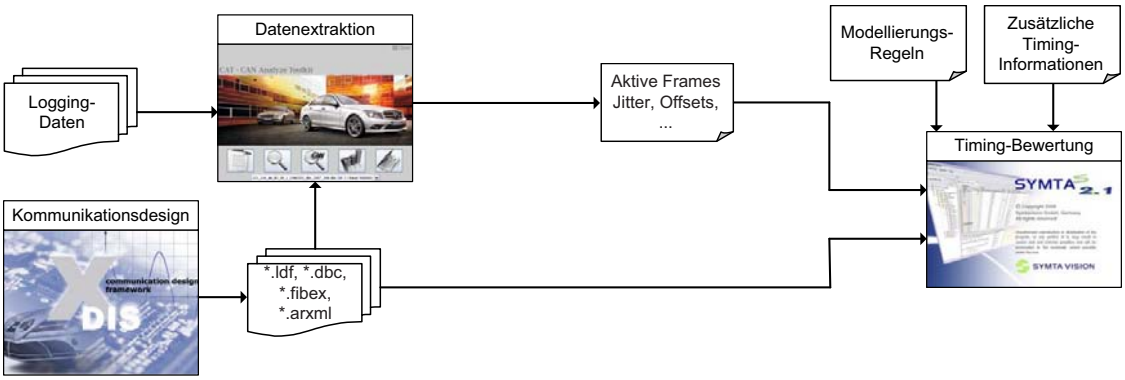


Abb. 8.14.: Verwendete Werkzeugkette für die Szenarien-Extraktion

Den Ablauf der Analyse zeigt Abbildung 8.15. Im ersten Schritt werden die Topologiedaten importiert und die Einstellungen des Busses übernommen. Im zweiten Schritt werden die Offsets und Jitter übergeben und die Einstellungen für Netzwerkmanagement und Diagnose gesetzt. Im dritten Schritt wird die globale Analyse ausgeführt, um die globalen maximalen Antwortzeiten zu ermitteln. In Schritt vier wird die Konfiguration für die Szenarien übergeben und die notwendigen Einstellungen gesetzt. Im letzten Schritt erfolgt die Analyse der einzelnen Szenarien und der Export der Ergebnisse.

8.3.3. Diskussion der Ergebnisse

In Abbildung 8.16 ist die ermittelte Buslast aufgeführt. Bei der Analyse des *globalen Worst-Case* ergab sich ein Wert von 54.7%. Dabei wurden alle periodischen Botschaften mit deren schnellsten Zykluszeit berücksichtigt. Aus dem Loggingdatensatz wurden insgesamt vier Betriebsszenarien extrahiert. Die hierfür ermittelten Buslasten liegen zwischen 36.7% bei *Schn3* und 53.4% bei *Scn1*.

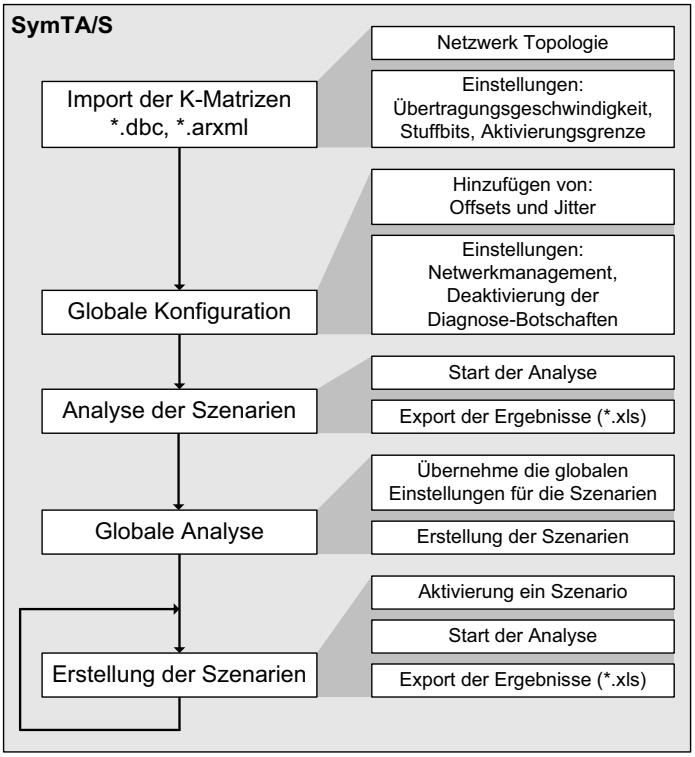


Abb. 8.15.: Ablauf der Analyse der Szenarien mit SymTA/S

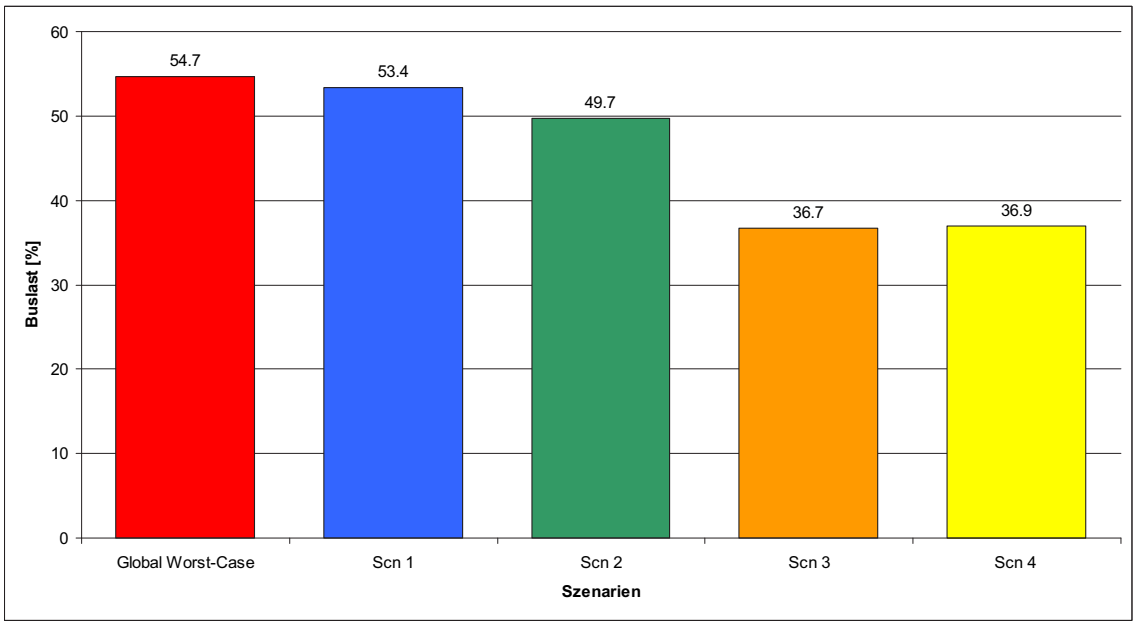


Abb. 8.16.: Buslast auf dem untersuchten CAN-Bus: Globaler Worst-Case (rot) und die berücksichtigten Szenarien *(Scn1 bis Scn4)*

Die Ermittlung der Buslasten für die einzelnen Szenarien ist ein Mehrwert des Verfahrens. Ein weiterer wird bei der Berechnung der maximalen Antwortzeiten der einzelnen Botschaften deutlich deutlich. In Abbildung 8.17 sind die maximalen Antwortzeiten für die Szenarien *(Scn1 bis Scn4)* sowie für den *Globalen Worst-Case* aufgezeigt. Im ersten Drittel (Priorität 1 bis 37) sind noch keine großen Unterschiede erkennbar, aber gerade im niederen Prioritätsbereich weicht der theoretische schlimmste Wert stark von den aus der Messung extrahierten Werten ab.

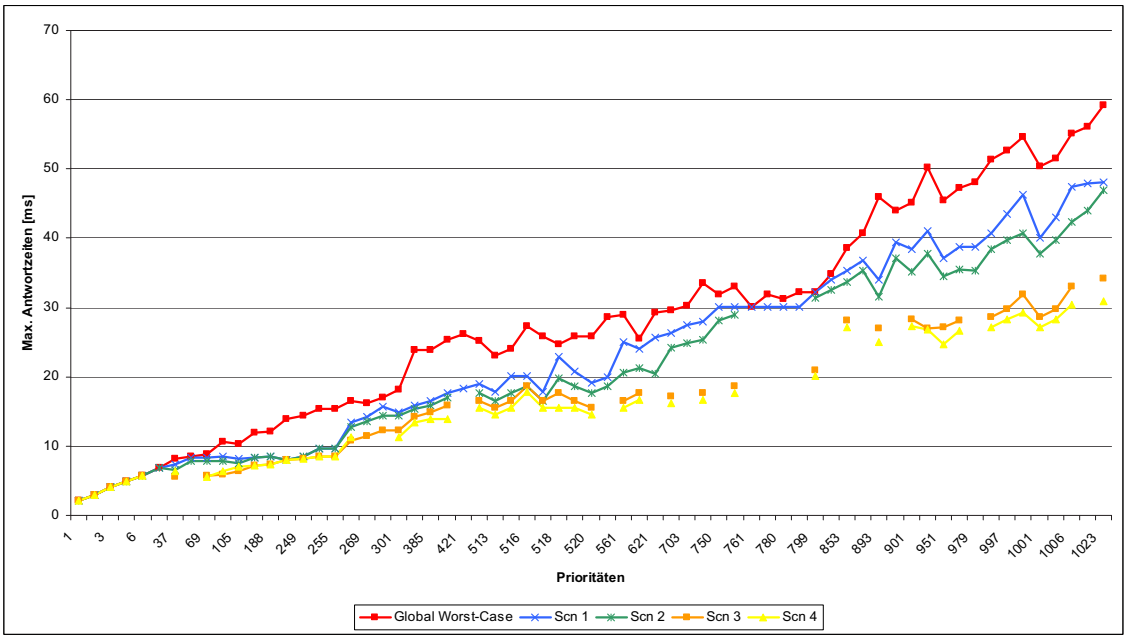


Abb. 8.17.: Maximale Antwortzeiten: Globaler Worst-Case und die Maximalwerte der Szenarien (Scn1 bis Scn4)

In Abbildung 8.18 wird der *Globale Worst-Case* mit den Maximalwerten aller vier Szenarien und den maximalen Antwortzeiten, welche direkt aus der Messung ermittelt wurden, verglichen. Dabei zeigt sich, dass die oberere Schranke der Szenarien die gemessenen Werte umfasst. Weiterhin wird hier der Nachteil einer reinen messbasierten Auswertung deutlich, da für viele Botschaften nicht der schlimmste Fall eingetreten ist. Dieser kann erst sicher über eine Timing-Analyse ermittelt werden.

Die Ergebnisse dieser Evaluierung haben gezeigt, dass die Ermittlung des *Global Worst-Cases* sehr wichtig ist, um die vollständige Absicherung einer Vernetzung auf

Basis von Fahrzeugmessungen zu gewährleisten. Über die Extraktion und Analyse der Szenarien kann ein besseres Bild über das tatsächliche Verhalten eines CAN-Busses gegeben werden. Dadurch lassen sich mögliche Überschätzungen der Analyseverfahren gezielter ermitteln und reduzieren. Bei der Auslegung der Vernetzung besteht so die Möglichkeit eine bessere Einschätzung über den Ist-Zustand zu treffen und gezielter Optimierungen durchführen.

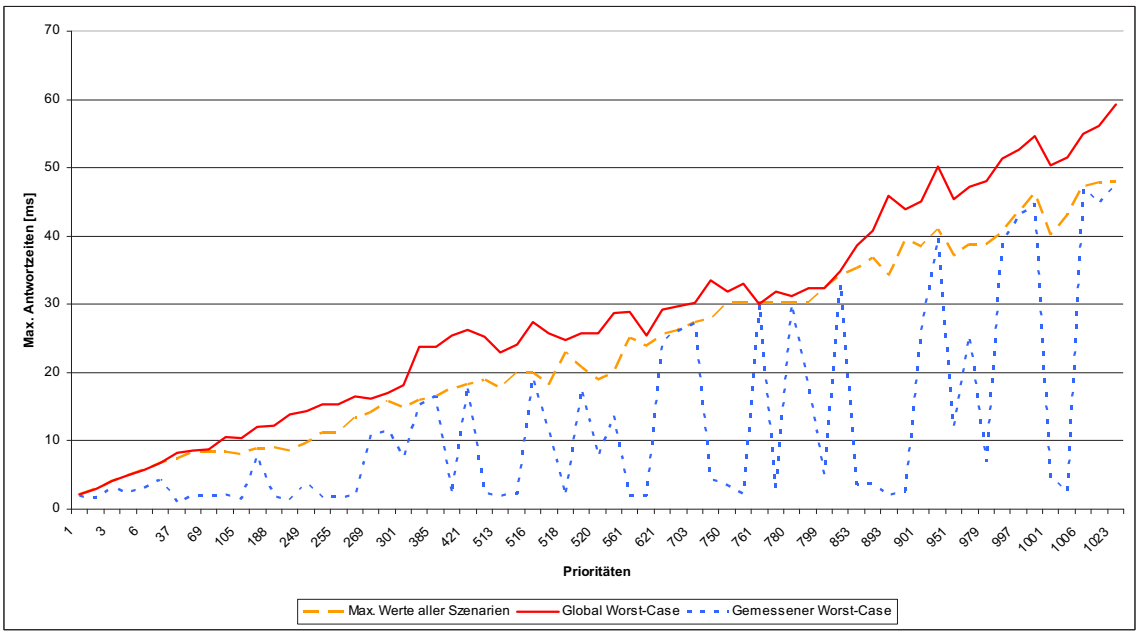


Abb. 8.18.: Vergleich des globalen Worst-Case mit den Maximalwerten der Szenarien (blau) sowie der gemessenen Werte

8.4. Timing-Bewertung eines AUTOSAR-basierten Gateways

Das in Abschnitt 6.4.1 eingeführte Regelmodell zur Ableitung von Annotation für eine WCET-Analyse, wird im Folgenden anhand eines konkreten AUTOSAR-basierten Gateways evaluiert. Um die Qualität der Analyseergebnisse bewerten zu können, wurden parallel dazu Vergleichsmessungen auf der realen Hardware durchgeführt. Ziel ist es, zum einen ein Delta bezüglich der gemessene WCET versus berechnete WCET zu erhalten und zum anderen soll der Anteil der automatischen generieren Annotation ins Verhältnis zu den implementierungsabhängigen Annotationen gestellt werden. In den

folgenden Abschnitten erfolgt die Vorstellung der Ergebnisse aus Analyse und Messung sowie eine darauf aufsetzende Interpretation der Resultate.

8.4.1. Übersicht

Als Beispiel für die Evaluierung dient ein AUTOSAR-Gateway auf Basis eines 32-Bit Microkontrollers der Firma NEC *V850-PHO3*. Details zu dem Mikrocontroller sind in [84] und [85] zu finden. Als Software-Stack kommt die AUTOSAR-Basissoftware *Micorsar* der Firma Vector-Informatik zum Einsatz (siehe [137] und [139]). Die untersuchte Version des Gateway-Prototypen ist mit 80MHz getaktet und verfügt über eine CAN- und eine FlexRay-Schnittstelle. In Abbildung 8.19 ist das AUTOSAR-Gateway dargestellt.

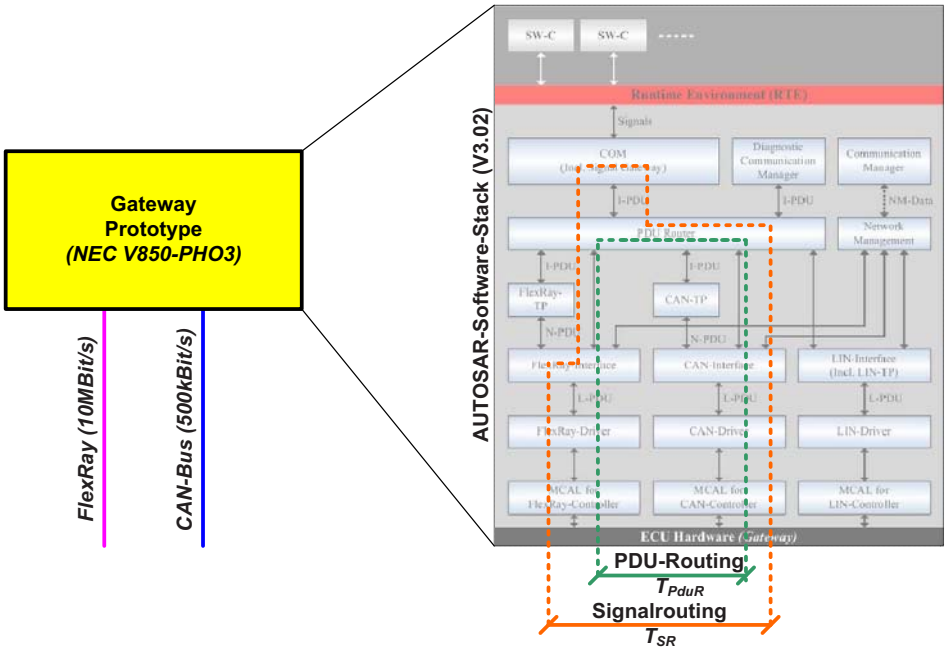


Abb. 8.19.: Gateway-Prototpye auf Basis eines *V850-PHO3* von NEC

In Tabelle 8.1sind die analysierten bzw. vermessenen Funktionen aufgeführt, welche im Folgenden näher diskutiert werden. Weiterhin erfolgte auch die Untersuchung der CAN-Interrupt-Routinen sowie der COM-Funktionen. Auf diese wird jedoch im Folgenden nicht weiter eingegangen.

Tab. 8.1.: Analysierte bzw. vermessene Funktionen des Gateways

Funktionen	**Beschreibung**
`TxJob1`	Sendejob für die Frames der Slots 1-39 im statischen Segment
`TxJob2`	Sendejob für die Frames der Slots 40-88 im statischen Segment
`TxJob3`	Sendejob für die Frames der Slots 89-331 im dynamischen Segment
`RxJob1`	Empfangsjob für die Frames der Slots 1-39 im statischen Segment, inkl. *Tx-Confirmation*
`RxJob2`	Empfangsjob für die Frames der Slots 40-88 im statischen Segment
`RxJob3`	Empfangsjob für die Frames der Slots 89-331 im dynamischen Segment, inkl. *Tx-Confirmation*

8.4.2. Verwendete Werkzeugkette

In Abbildung 8.20 ist die verwendete Werkzeugkette dargestellt. Ausgehend von einer AUTOSAR-Gateway-Konfiguration *(ECU-Extract.arxml)* wurde über die AUTOSAR-Werkzeugkette der Konfigurationscode für das Gateway generiert.Die Konfigurationsdateien wurden zusammen mit den AUTOSAR-BSW-Dateien compiliert. Das generierte *Binary* konnte dann zum einen für die Messungen direkt auf den Prototypen geladen werden, zum anderen bildete das Binary auch die Grundlage für die WCET-Analyse. Die Annotationen wurden über das Regelmodell erzeugt sowie manuell vorgegeben. Das *ECU-Extract.arxml* und die Konfigurationsdateien dienten als Eingabe für das Regelmodell. Auf Basis dieser Informationen wurden die Annotationen generiert. Zu Beginn der Messung erfolgte auch die Verifikation der analysierten Ausführungspfade, um sicherzustellen, dass nur die im Normalbetrieb aktiven Codestücke berücksichtigt wurden. Nach erfolgreicher Messung und Analyse konnten die Ergebnisse ausgewertet werden.

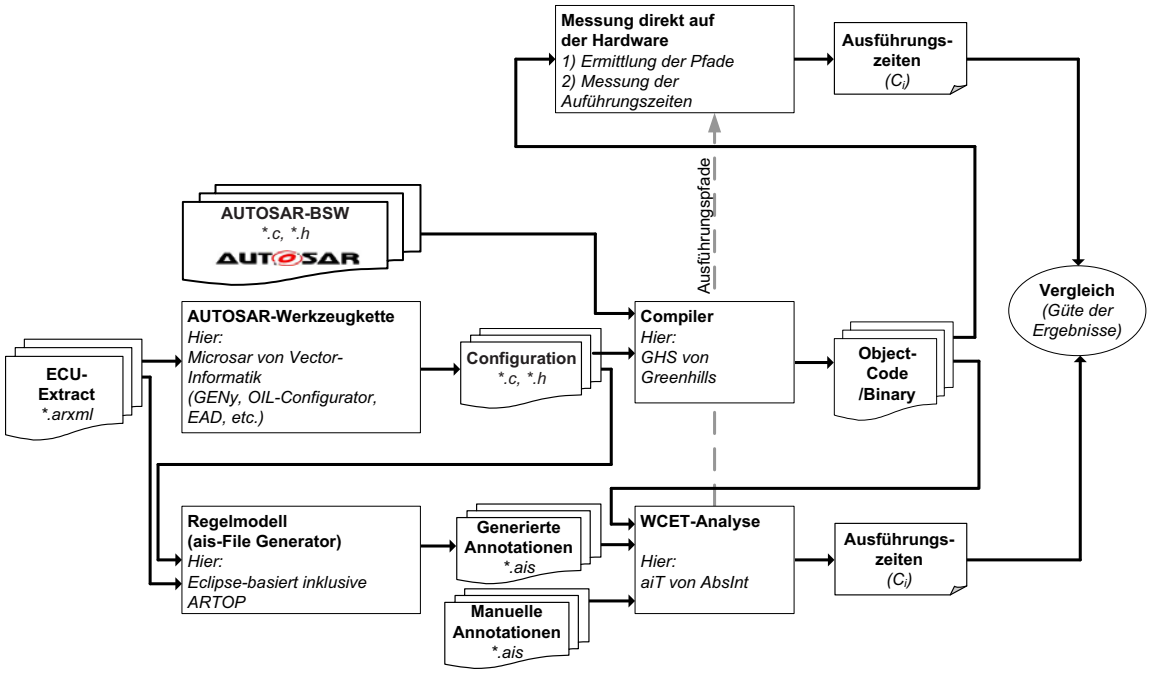


Abb. 8.20.: Für die Evaluierung verwendete Werkzeugekette

8.4.3. Analyse der Ausführungszeiten mit aiT

Für die Analyse der Ausführungszeiten kam das statische Analysetool *aiT* der Firma AbsInt zum Einsatz [1]. Die Annotationen, welche für eine korrekte Code-Analyse notwendig sind, wurden über die in Abschnitt 6.4.1 beschriebenen Regeln generiert. Um die Steigerung der Analysegenauigkeit durch die Modellierungsregeln und die manuellen Annotationen zu verdeutlichen, wurden die FlexRay-Jobs in drei Schritten analysiert:

1. Es wurden nur generische Annotationen und Schleifenbegrenzungen verwendet: $WCET_{gen}$.

2. Zusätzlich zu den Annotationen aus Schritt 1, kamen die generierten Annotationen auf der Basis der AUTOSAR-Systembeschreibungen hinzu: $WCET_{not}$.

3. Die Annotationen aus Schritt 1 und Schritt 2 wurden noch durch die manuellen, implementierungsspezifischen Annotationen ergänzt: $WCET_{all}$.

In Tabelle 8.2 sind die ermittelten Ausführungszeiten für die einzelnen FlexRay-Jobs angegeben. Seitens der Firma AbsInt wird eine Überschätzung der Ausführungszeiten von $20-35\%$ Prozent für die Modelle angegeben.

Tab. 8.2.: Analysierte Ausführungszeiten der FlexRay-Jobs des Prototypen-Gateways [133]

Job-Kontext	**Anzahl PDUs**	$WCET_{gen}[\mu s]$	$WCET_{not}[\mu s]$	$WCET_{all}[\mu s]$
TxJob1	4	196.0	116.0	94.9
TxJob2	2	131.0	74.1	58.8
TxJob3	15	4381.0	462.0	378.0
RxJob1	7	997.0	484.0	334.0
RxJob2	6	1077.0	305.0	231.0
RxJob3	10	7587.0	957.0	658.0

Die Ergebnisse zeigen deutlich, dass die Überschätzung über die verfeinerten Annotationen deutlich reduziert wird. Am größten ist die Verbesserung durch die generierten Annotationen. Für TxJob3 beträgt die Verbesserung 80% gegenüber der Analyse mit nur den generischen Annotation. Über die implementierungsspezifischen Annotationen ist eine weitere Steigerung der Genauigkeit von maximal 30% gegenüber den generierten Annotationen möglich. Um die Analyseergebnisse zu verifizieren wurden umfangreiche Messungen durchgeführt. Diese werden im Folgenden näher erläutert.

8.4.4. Messungen auf der Hardware

Um die Analyseergebnisse bewerten zu können, wurden umfangreiche Messungen auf der realen Hardware durchgeführt. Im ersten Schritt galt es den analysierten Pfad nachzuvollziehen und zu stimulieren, um dann in einem zweiten Schritt die Ausführungszeit messen zu können. Als Messaufbau wurde ein in Abschnitt 8.4.1 vorgestelltes Prototypen-System verwendet. Mittels entsprechender PC-Hardware wurde ein Restbus für CAN und FlexRay simuliert. Für das Versenden der Botschaften wurden Worst-Case

Szenarien für alle untersuchten Softwareroutinen erstellt, die die ermittelten Worst-Case-Pfade der WCET-Analyse stimulieren. Die Messungen der Ausführungszeiten der Softwareroutinen erfolgte über freie *General Purpose I/Os (GPIO)* Pins des Prozessors. Dafür wurde in den relevanten Softwarefunktionen Befehle zur Aktivierung von GPIO-Ausgängen eingefügt. Die Messung an den Ausgängen erfolgte über ein Oszilloskop. Die Verifikation über diese Messmethodik stellt nicht den effizientesten Weg dar. Mit tracing-fähiger Debug-Hardware (siehe z.B. [41]) sind Programmpfade direkt aufzeichenbar und schneller auszuwerten. Eine solche Lösung stand im Rahmen dieser Arbeit jedoch nicht zur Verfügung.

Pfadverifikation für die Messung

Für die gezielte Stimulation des Worst-Case-Pfades wurden die Schleifengrenzen und Verzweigungen der Softwareroutinen untersucht. Für alle in aiT annotierten Schleifen wurden im Quellcode Steuersignale für die GPIOs eingefügt. Im nächsten Schritt wurden während der Restbussimulation mit einem Speicheroszilloskop die Anzahl der Impulse gezählt. Somit konnte für jede Schleife und Funktion sichergestellt werden, dass die Anzahl der Iterationen bzw. Aufrufe dem ermittelten Worst-Case-Fall der Analyse entsprechen. In Abbildung 8.21 ist beispielhaft die Pfadverifizierung eines Empfangsjobs der FlexRay-Interfaces `FrIf_` dargestellt.

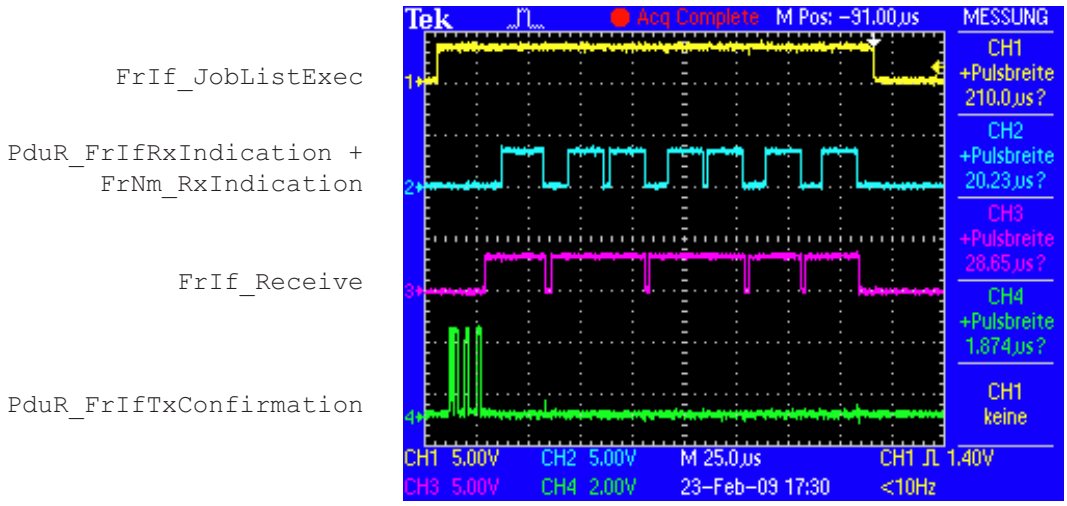


Abb. 8.21.: Verifikation des Ausführungspfades über GPIO-Operatitonen des RxJob1

Über den zweiten Kanal des Oszilloskops *CH2* wurde die Ausführungsanzahl der Benachrichtigungsfunktionen des PDU-Routers `PduR_FrIfRxIndication` und des Netzwerkmanagement-Moduls `FrNm_ RxIndication` gezählt. Dadurch konnte sichergestellt werden, dass der Worst-Case, in diesem Fall das Eintreffen von sieben PDUs, auch eingetreten ist.

Messung der Ausführungszeiten

Nach der Pfadverifikation erfolgte die Messung der Ausführungszeit der Softwareroutine. Hierzu wurden den Funktionen die Steuerbefehle, welche für die Pfadermittlung notwendig waren, wieder entfernt, um die exakte Ausführungszeit ermitteln zu können. Direkt vor dem Aufruf der zu messenden Sofwareroutine und nach deren Beendigung blieb jeweils ein GPIO-Steuerkommando eingefügt. Im nächsten Schritt wurde nun die Restbussimulation mit den bei der Pfadverifikation ermittelten Einstellungen gestartet und die Ausführungszeit konnte ermittelt werden. In Abbildung 8.22 ist die Messung am Beispiel eines FlexRay-Empfangsjobs RxJob1 dargestellt. Im vorliegenden Fall beträgt die Ausführungszeit $C_i = 193,8\mu s$.

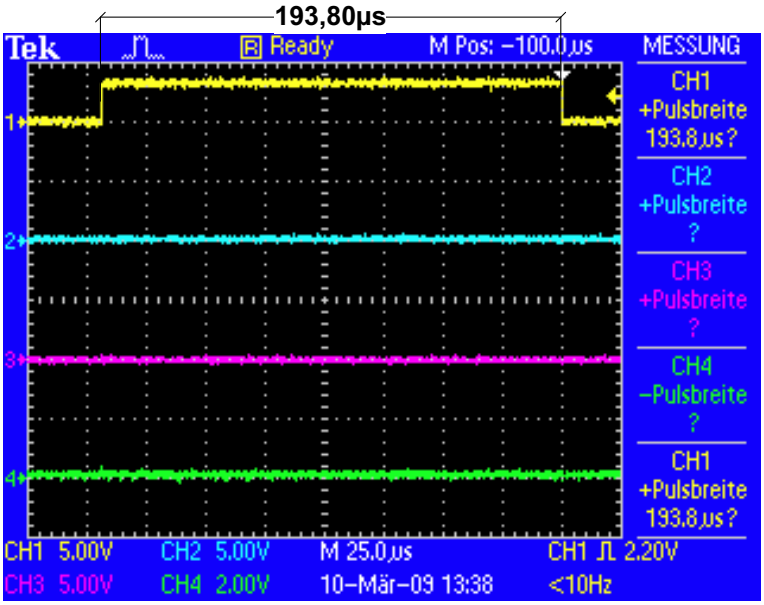


Abb. 8.22.: Beispiel für die Verifikation der Ausführungszeit *WCET*: Impulsmessung des RxJobs1

8.4.5. Diskussion der Ergebnisse

In Tabelle 8.3 sind die Ergebnisse der Analyse und die durch Messung ermittelten maximalen Ausführungszeiten gegenübergestellt. Weiterhin ist in der rechten Spalte das prozentuale Verhältnis zwischen Messung und Analyse ablesbar. Ein Verhältnis von 150% entspricht einer Überschätzung von 50% durch die WCET-Analyse gegenüber der Messung. Für die untersuchten Funktionen liegt die Überschätzung zwischen 42% und 70%.

Tab. 8.3.: WCET der FlexRay-Jobs: Vergleich von Messung und Analyse

Kontext/Funktion	**Messung** $[\mu s]$	**Analyse** $[\mu s]$	**Verhältnis A/M**
TxJob1	66.00	94.95	144%
TxJob2	41.43	58.76	142%
TxJob3	256.80	378.0	147%
RxJob1	193.40	334.0	173%
RxJob2	142.70	213.0	149%
RxJob3	388.20	658.0	170%

Das Delta zwischen der angegebenen Überschätzung von aiT und den vorliegenden Ergebnissen hat folgenden Grund: Es standen zum Zeitpunkt der Arbeit noch nicht alle Informationen über das zeitliche Verhalten des Prozessors sowie der relevanten Peripherie (RAM, Flash und Bus-Controller) der Firma AbsInt zur Verfügung, so dass an einigen Stellen im Prozessormodell noch konservative Annahmen hinterlegt waren. Diese werden, sobald die Informationen vorliegen, durch die tatsächliche Werte ersetzt. In diesem Zuge ist dann mit einer weiteren Verbesserung der Analyseergebnisse zu rechnen. Die Überschätzung sollte dann nur noch zwischen 25% und 35% liegen.

8.5. Timing-Bewertung einer Vernetzungsarchitektur

Aufgrund der zunehmenden Funktionsverteilung, insbesondere im Bereich der Fahrerassistenzfunktionen, wird die Timing-Bewertung von Ende-zu-Ende-Pfaden auf Signal- oder Botschaftsebene immer wichtiger. Im Folgenden werden ein solcher exemplarischer Pfad untersucht und die wichtigsten Eigenschaften, die zu berücksichtigen sind, herausgearbeitet.

8.5.1. Übersicht

Abbildung 8.23 zeigt den Architekturausschnitt mit den beteiligten Bussen und Steuergeräten des Ende-zu-Ende-Signalpfades. Vom Steuergerät *ECU1* wird eine Botschaft über den CAN-Bus *CAN1*, dem Gateway-Steuergerät *Central Gateway* und dem FlexRay-Bus *FlexRay* an das Steuergerät *ECU2* übertragen. Auf der *ECU2* ist der Funktionsmaster integriert, welcher eine Datenfusion verschiedener Sensorinformationen durchführt. Für die korrekte Funktionsweise des Regelalgorithmus innerhalb des Funktionsmasters ist das maximale Datenalter der einzelnen Sensorwerte wichtig.

Nach abgeschlossener Auswertung der Sensordaten schickt der Funktionsmaster eine Botschaft zum Steuergerät *ECU3*. Dieses aktiviert bei entsprechenden Datenwerten eine Funktion. Für diesen Pfad ist nicht das maximale Datenalter von Bedeutung, sondern die maximale Verzögerung, die auftreten kann (Reaktionszeit).

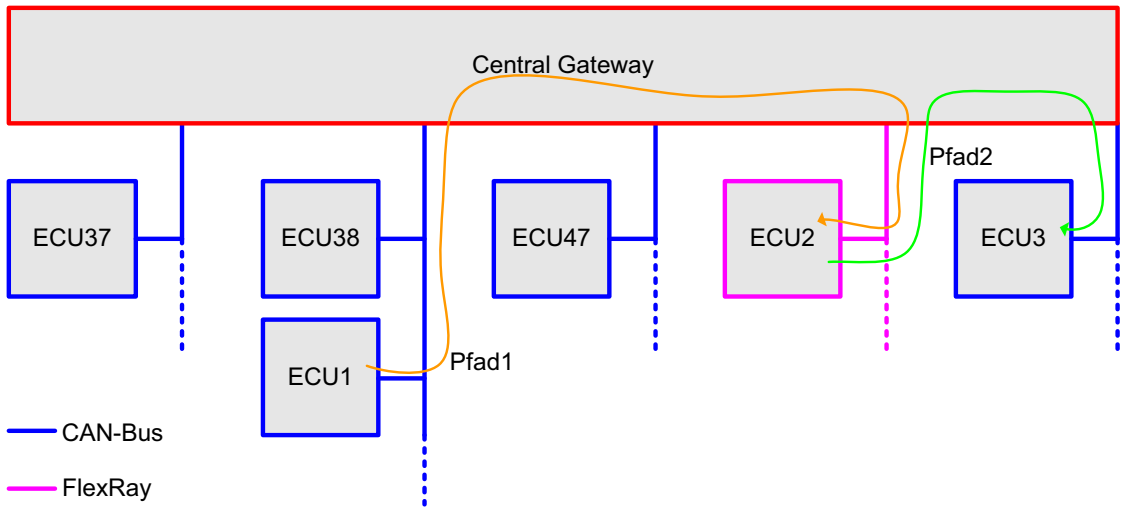


Abb. 8.23.: Architekturausschnitt mit den relevanten Ende-zu-Ende Pfaden

Nach abgeschlossener Auswertung der Sensordaten schickt der Funktionsmaster eine Botschaft zum Steuergerät *ECU3*. Dieses aktiviert bei entsprechenden Datenwerten eine Funktion. Für diesen Pfad ist nicht das maximale Datenalter von Bedeutung, sondern die maximale Verzögerung, die auftreten kann (Reaktionszeit).

8.5.2. Verwendete Werkzeugkette

Abbildung 8.24 zeigt die verwendete Werkzeugkette. Für das Evaluierungsbeispiel diente ein Modell aus der frühen Entwurfsphase. Die relevanten Informationen wurden aus dem Architekturwerkzeug *PREEvision* exportiert. Um das korrekte Timing-Verhalten zu berücksichtigen, erfolgt die Analyse der Gateway-Funktionen wie in Abschnitt 8.4 beschrieben (unterer Pfad). Die vollständige Pfadanalyse wurde mit dem Timing-Werkzeug *SymTA/S* durchgeführt.

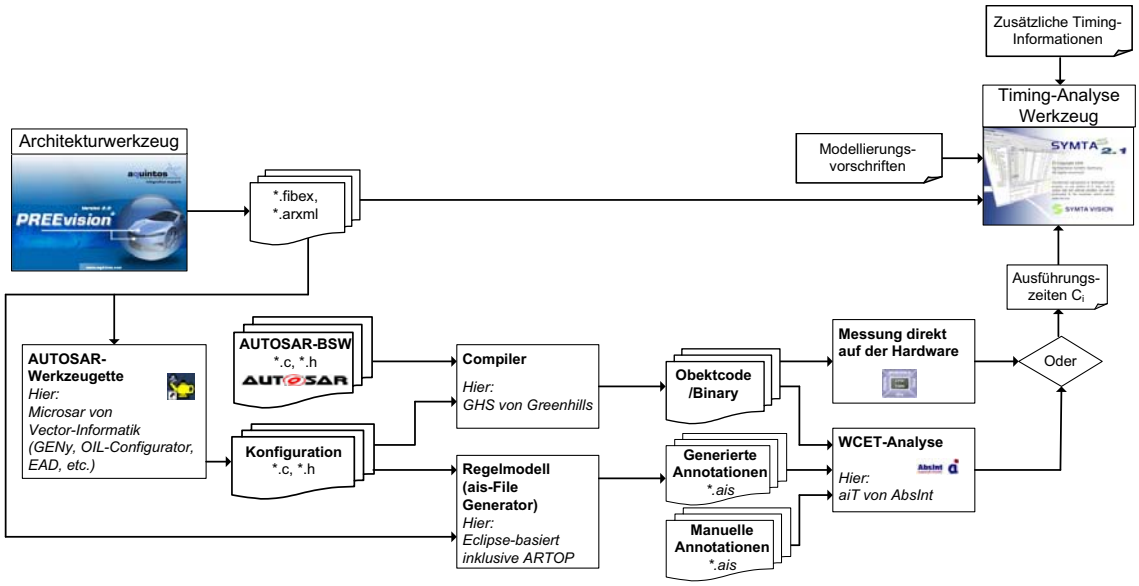


Abb. 8.24.: Werkzeugkette für die Ende-zu-Ende-Pfadanalyse in der frühen Architekturphase

8.5.3. Diskussion der Ergebnisse

Ein Ausschnitt des Timing-Modells der einzelnen Funktionsanteile der beteiligten Steuergeräte ist in Abbildung 8.25 dargestellt. Ein Sensorwert wird von *ECU1* eingelesen und innerhalb des `Appl_Task_ECU1` zyklisch alle 20*ms* verarbeitet. Über den Task

`Com_Task_ECU1` werden die Daten zyklisch alle 20*ms* über den *CAN1* an das *Central Gateway geschickt*. Dieses leitet die Daten über den *FlexRay* an die *ECU2* weiter. Der reservierte FlexRay-Slot wird alle 5*ms* übertragen. Es findet also eine Überabtastung statt. In *ECU2* werden die Daten empfangen und im `Appl_Task_ECU2` mit weiteren Informationen fusioniert. Das Ergebniss wird dann zyklisch an die *ECU3*, über den *FlexRay*, *das Central Gateway* und den *CAN2*, geschickt. Wird ein entsprechendes Datum übertragen, erfolgt direkt die Ansteuerung des Aktors von *ECU3*.

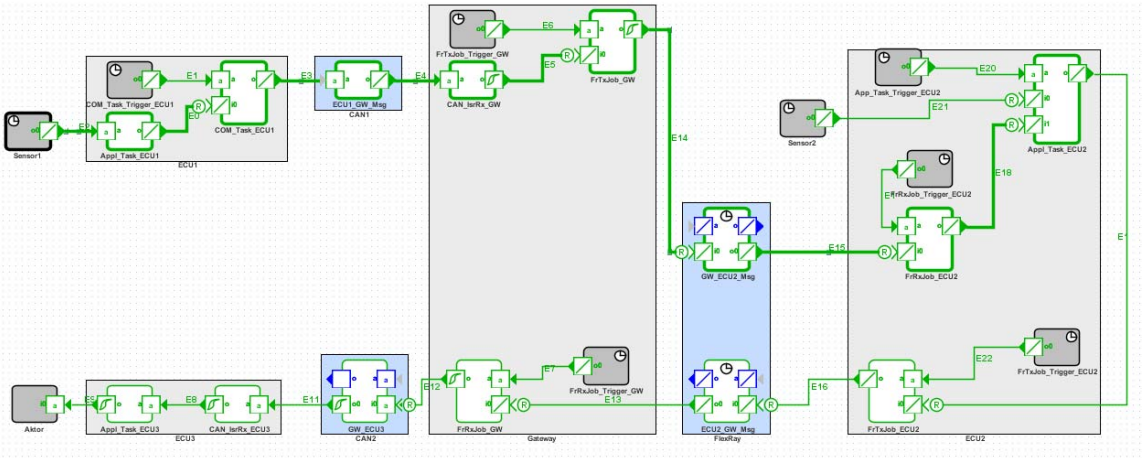


Abb. 8.25.: Modell der beteiligten Steuergeräte mit den einzelnen Tasks

Die zu bewertende Ende-zu-Ende-Pfade *Pfad1: ECU1→ CAN1 → Central Gateway → FlexRay → ECU2* und *Pfad2: ECU2→ FlexRay → Central Gateway → CAN2 → ECU3* sind in Abbildung 8.26 und Abbildung 8.27.

Für Pfad1 ist das maximale Datenalter von Bedeutung, der ermittelte Wert beträgt $L_{ma} = 30.87ms$. Aufgrund der periodischen Abfrage (alle 20*ms*) des Sensors über den `Appl_Task_ECU1` kann im besten Fall alle 20*ms* ein aktualisierter Sensorwert übertragen werden. Zusätzlich kommen die Übertragungszeiten sowie Verzögerungen durch Synchronisationseffekte hinzu.

Für Pfad2 erfolgt die Analyse für die maximale Reaktionszeit, das Ergebnis liegt bei $L_{ft} = 9.53ms$. Die größten Anteile an der Verzögerung kommen durch den ungünstigen Offset zwischen der `Com_Task_ECU2` und dem FlexRay-Schedule und die Verzögerung auf dem *CAN* zustande.

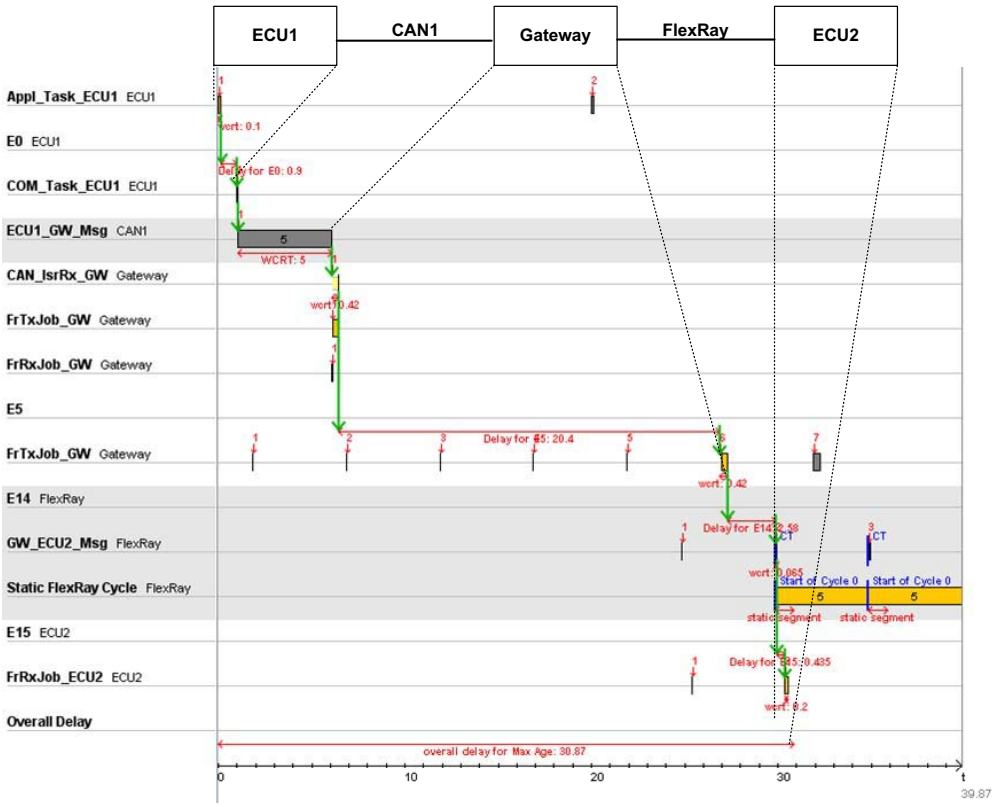


Abb. 8.26.: Analyse des maximalen Datenalters für Pfad1

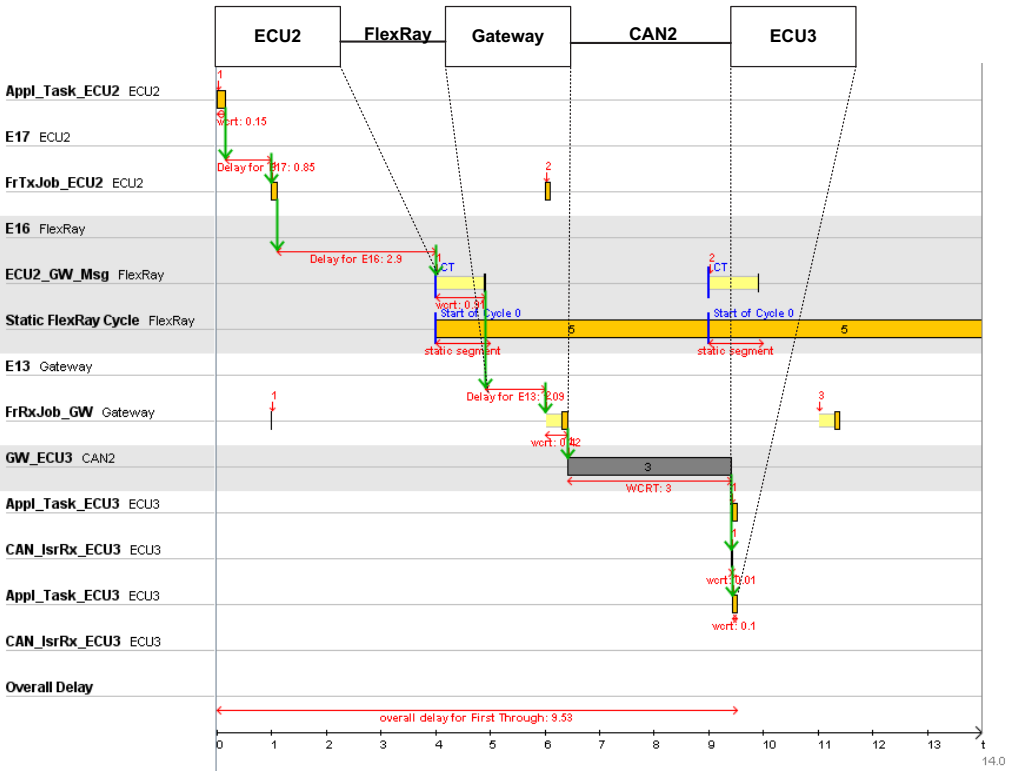


Abb. 8.27.: Analyse der Reaktionszeit für Pfad2

8.6. Zusammenfassung

Die vorgestellten Evaluierungsbeispiele zeigen deutlich den Mehrwert des entwickelten Ansatzes zur durchgängigen Timing-Bewertung von Vernetzungsarchitekturen und Gateway-Systemen im Kraftfahrzeug. Schrittweise ist die Untersuchung der einzelnen Komponenten einer Vernetzungsarchitektur möglich und mündet in der vollständigen Bewertung des Gesamtsystems.

9. Zusammenfassung und Ausblick

9.1. Zusammenfassung

Bei der Auslegung und Absicherung von aktuellen und zukünftigen Vernetzungsarchitekturen und Gateway-Systemen im Kraftfahrzeug sind in allen Phasen des Entwicklungsprozesses Fragestellungen zum Thema Ausführungs- und Latenzzeiten zu beantworten. Bedingt durch die fortschreitende Hochintegration von Steuergeräten und den zunehmenden Betrieb der Busse an deren Lastgrenzen sowie durch die neuen Anforderungen unter anderem an sicherheitskritische Systeme werden zukünftig Timing- und Ressourcenanforderungen als Entwurfskriterien immer wichtiger. Anforderungen an sicherheitskritische Systeme ergeben sich durch die Norm ISO26262, welche ab 2012 bei der Entwicklung dieser Systeme zu berücksichtigen sind.

Mit der vorgestellten Methodik ist die systematische und durchgängige Bewertung von Timing-Fragestellungen möglich, welche sich im Verlauf des Entwicklungsprozesses von Vernetzungsarchitekturen im Kraftfahrzeug ergeben. Die wesentlichen Punkte, welche den Mehrwert dieser Arbeit aufzeigen, sind:

- Die Einordnung der Timing-Bewertungsverfahren in den existierenden E/E-Entwicklungsprozess zeigt die Möglichkeiten und Potentiale auf, welche durch den gezielten Einsatz dieser Verfahren während der Entwicklung von E/E-Architekturen erreichbar sind. Weiterhin werden die offenen Punkte diskutiert, die einer direkten Verwendung der Verfahren entgegen stehen. Innerhalb dieser Arbeit erfolgt die Diskussion der aufgeführten Punkte und es werden entsprechende Lösungen entwickelt.

- Die Datenextraktion von Timing-Informationen aus bereits existierenden Vernetzungsarchitekturen liefert ein exaktes Bild zum Stand der aktuellen E/E-Systeme im Fahrzeug. Über die hierüber gewonnenen Erkenntnisse lassen sich Anforderungen ableiten, welche als Erweiterungen in zukünftige Lastenhefte einfließen kön-

nen. Dadurch ist in Zukunft eine deutliche Steigerung der Testtiefe der Integrationstests beim OEM zu erreichen. Ferner können die identifizierten Anforderungen als Input für die nächste Phase der Standardisierungsaktivitäten im Rahmen der AUTOSAR-Timing-Gruppe dienen [132].

- Die extrahierten Timing-Informationen können für eine Modellverfeinerung in der Entwurfsphase von E/E-Architekturen verwendet werden. Mittels der Wiederverwendung kann die Aussagekraft von Timing-Bewertungen während der frühen Entwicklungsphase von Vernetzungsarchitekturen signifikant gesteigert werden. Auf dieser Basis können dann die Timing-Anforderungen für die Lastenhefte abgeleitet werden [130], [129].

- Um eine exakte Abbildung des Timing-Verhaltens in den entsprechenden Bewertungsverfahren zu ermöglichen und umzusetzen, ist die Anwendung von Modellierungsregeln notwendig [131], [128]. Dadurch wird eine signifikante Steigerung der Genauigkeit der Bewertungsergebnisse erreicht. Zu den Modellierungsregeln zählen:

  1. Modellierungsregeln für die heute eingesetzten Kommunikationssysteme und Standards, insbesondere für CAN, FlexRay und AUTOSAR.

  2. Regeln zur automatischen Generierung von Annotation für die statische Code-Analyse [52], [133].

  3. Konzept für ein Gateway-Modell für eine exakte und umfassende Timing-Analyse solcher Systeme [77].

- Darstellung einer durchgängigen Bewertungsmethodik anhand der entwickelten Werkzeugkette für die Bewertung von Timing-Fragestellungen innerhalb des E/E-Entwicklungsprozesses [131].

- Konzept für die Ableitung von Routing-Testpattern auf der Basis von Timing-Analysen, um eine gezielte Berücksichtigung des Timing-Verhaltens von Steuergeräten mit Gateway-Anteilen während der Tests am Komponentenprüfstand zu ermöglichen. Die hierfür notwendigen Informationen können direkt über die aufgezeigte Werkzeugkette generiert werden.

Die in der Arbeit vorgestellten Verfahren und Konzepte wurden erfolgreich für die Auslegung der nächsten Generation der Vernetzungsarchitektur der *S-Klasse* von Mercedes-Benz Cars während der Vorentwicklungsphase eingesetzt. Aufbauend auf den Ergebnissen wird nun im nächsten Schritt die Integration der Verfahren in den Vor- und Serienentwicklungsprozess bei Mercedes-Benz Cars angegangen. Der Ausblick geht auf weitere Themen ein, die einen kontinuierlichen Ausbau und eine Verfeinerung der vorgestellten Methodik ermöglichen.

9.2. Ausblick

Im Laufe der Arbeiten entstanden noch weitere Ideen im Themengebiet der Timing-Bewertungen. Ferner wurden zusätzliche Verfeinerungsmöglichkeiten identifiziert, welche die Qualität der Methodik weiter steigern. Diese Ideen und Verfeinerungen werden im Folgenden in Anlehnung an die drei Hauptkapitel (Kapitel 5: *Verfahren zur Extraktion von Timing-Informationen*, Kapitel 6: *Modellierungsregeln zur exakten Timing-Bewertung*, Kapitel 7: *Methodik für eine durchgängige Timing-Bewertung*) diskutiert und geben einen Ausblick auf weitere Aufgaben im Themenfeld der Timing-Bewertungen. Im Hauptfokus steht dabei die vollständige Integration der Bewertungsmethodik in den Serienentwicklungsprozess.

9.2.1. Erweiterungen für die Extraktion von Timing-Informationen

Das Verfahren zur Extraktion von Timing-Informationen aus existierenden Vernetzungsarchitekturen (siehe Kapitel 5) auf der Basis erster Prototypen oder E-Fahrzeugen lässt sich noch weiter ausbauen. Im Folgenden sind einige mögliche Erweiterungen aufgeführt:

- Erweiterung des Ansatzes auf weitere Kommunikationssysteme, z.B. für FlexRay, Ethernet-IP, LIN, etc.

- Erweiterung der Schnittstellen für einen standardisierten Datenaustausch zwischen den Entwicklungswerkzeugen, z.B. Import von AUTOSAR-(System)-Beschreibungen *(*.arxml)*.

- Weitere Verfeinerung der Extraktion von Betriebsszenarien durch verbesserte Annotationsmöglichkeiten auf der Basis von Angaben aus den Lastenheften oder den Spezifikationen der einzelnen Systeme.

- Erweiterung der Auswertungen durch zusätzliche Algorithmen, z.B. Rekonstruktion der Periode des *COM-Tasks*, semantische Prüfung der Botschaften, Absicherung des Routingverhaltens von Gateways.

Über die gewonnenen Erkenntnisse lassen sich Anforderungen für die verfeinerte Spezifikation von E/E-Systemen ableiten. Weiterhin können die Erfahrungen in die Standardisierungsaktivitäten der AUTOSAR-Timing-Gruppe einfließen. Darüber hinaus besteht die Möglichkeit die existierenden Timing-Modelle zu verfeinern und zu erweitern.

9.2.2. Verfeinerung und Entwicklung weiterer Modellierungsregeln

Die Modellierungsregeln und die notwendigen Modelle für eine detaillierte Timing-Bewertung können noch erweitert und ergänzt werden. Weitere Punkte sind hier:

- Ausbau der Annotationsregeln für eine weitere Automatisierung der Timing-Bewertung von Gateway-Systemen, z.B. Berücksichtigung der gesetzten oder nichtgesetzten *Compiler-Schaltern* des Generierungstools, um die nicht verwendeten Programm-Code-Segmente aus der Analyse ausschließen zu können.

- Umsetzung bzw. Verfeinerung der Timing-Modelle für Ethernet-IP und LIN-Netzwerke.

- Erweiterte Modellierungsmöglichkeiten von höheren Protokollschichten, um eine detaillierte Bewertung der Diagnose- und Flash-Anforderungen zu ermöglichen. Hierzu zählen u.a. Modelle, welche die Layer 3 und 4 des OSI-Schichten-Modells abdecken.

- Umsetzung einer automatisierten Modellextraktion für Gateways in der frühen Entwurfsphase von Vernetzungsarchitekturen, um eine effizientere und präzisere Abschätzung der Routing- und Interruptlast durchzuführen. Weiterhin kann damit die Funktionsintegration auf Gateway-Steuergeräte besser bewertet werden.

- Ableitung von Timing-Anforderungen, die zukünftig Bestandteil der Lastenhefte sind. Ferner sollte in Zukunft ein Weg gefunden werden, der den Austausch von Informationen über das Timing-Verhalten von E/E-Systemen zwischen Zulieferer und OEM regelt. Zu den Informationen zählen z.B. Beschreibung der Betriebssystemkonfiguration (*.oil-Datei) und Ausführungszeiten der einzelnen Tasks. Auf Basis dieser Daten besteht für den OEM dann die Möglichkeit, eine exakte Timing-Absicherung auf Systemebene durchzuführen.

- Für einen standardisierten Austausch der Timing-Informationen zwischen OEM und Zulieferern können die in AUTOSAR ab Rel. 4.0 spezifizierten Annotationsmöglichkeiten umgesetzt werden. Fehlende Attribute oder Verfeinerungen sind in der nächsten Entwicklungsphase für das AUTOSAR Rel. 5.0 einzubringen und zu spezifizieren, z.B. die *Composition of Event-Triggering Constraints* und die erweiterten Annotationsmöglichkeiten von Ausführungszeiten (siehe hierzu [14]).

9.2.3. Integration der Werkzeugkette in den Serienentwicklungsprozess

Die in Kapitel 7 aufgezeigte Werkzeugkette kann in einem nächsten Schritt in den Serienentwicklungsprozess integriert werden. Hierfür sind die Schnittstellen zwischen den Entwicklungs- und Bewertungswerkzeugen vollständig abzugleichen. Ferner kann an der weiteren Optimierung der Abläufe gearbeitet werden, um einen möglichst hohen Grad einer automatisierten Timing-Bewertung zu erreichen.

Ferner besteht die Möglichkeit eine weitere Verfeinerung des aktuellen Generierungsprozesses für die Routing-Testpattern erfolgen. Bisher werden die maximalen Bursts auf den einzelnen Bussen, welche mit dem Gateway verbunden sind, für die Generierung der Pattern verwendet. Bedingt durch Scheduling-Annomalien (siehe hierzu z.B. in [21] und [96]) erzeugen die ermittelten Pattern nicht in jedem Fall den kritischsten Lastzustand im Gateway. Durch die Umsetzung des erweiterten Gateway-Analyse-Modells in Abschnitt 6.4.2 ist zukünftig der kritischste Lastzustand bei einem Steuergerät mit Gateway-Aufgaben sicher bestimmbar. Durch eine Koppelung der Timing-Bewertung mit formalen Verifikationsverfahren kann zusätzlich eine vollständige semantische Absicherung des Routingverhaltens anhand der Spezifikation erfolgen.

Glossar

Aktivierungszeitpunkt *(engl. Activation)*:
Der Aktivierungszeitpunkt beschreibt den Zeitpunkt an dem eine Task oder eine Botschaft zur Ausführung bzw. Übertragung ansteht.

Aktivierungsbegrenzung *(engl. Activation Restriction)*:
Über die Aktivierungsbegrenzung gibt an wie viele spontane Ereignisse maximale gleichzeitig auftreten können.

ARTOP (AUTOSAR Tool Platform):
ARTOP ist die Implementierung eine Plattform, welche die allgemeinen Funktionalitäten für Entwicklungswerkzeuge bereitstellt, um AUTOSAR-konforme Systeme sowie Steuergeräte zu entwickeln und zu konfigurieren.

Basis Software:
Die Basis Software (BSW) ist Bestandteil der AUTOSAR-Software-Architektur. Die BSW umfasst sämtliche Basis-Dienste, Treiber und das Betriebsystem. Die BSW ist über eine klar definierte Schnittstelle, die RTE, von den eigentlichen Applikationen (SW-Cs) getrennt.

Baudrate:
Die Baudrate ist ein Maß welches die Schrittgeschwindigkeit bei der Datenübertragung beschreibt. Die Baudrate definiert die Anzahl der Signaländerungen die pro Sekunde übertragen werden können. Die Einheit der Baudrate heißt *Baud*. Es ist jedoch nicht so dass die Baudrate immer gleich die Bitrate ist. Je nach Modulation und Leitungscode kann die Bitrate auch ein Vielfaches der Baudrate betragen. Dies ist dann der Fall wenn pro Zeiteinheit mehrere Bits übertragen werden.

Baureihe:
Als Baureihe werden in vielen Bereichen der Technik Geräte oder Produkte bezeichnet, die in vielfacher Ausführung in gleichartiger Weise gefertigt wurden. Diese Bezeich-

nung wird vor allem dann verwendet, wenn das gleiche Produkt vorher oder nachher oder auch gleichzeitig in ebensolchem Umfang, jedoch in deutlich abweichender Weise, gebaut wurde oder wird [142].

Binary:
Als Binary oder *Executable* werden Dateien verstanden, welche ausführbare Programme enthalten.

Blocking:
Eine Task heißt blockiert, wenn diese an der Ausführung durch einen niederprioreren Task gehindert wird.

Burst:
Ein Burst beschreibt das gebündelte Auftreten von mehreren Botschaften auf einem Bus. Die freie Zeit t auf dem Bus (Idle-Phase) zwischen den aufeinanderfolgenden Botschaften ist dabei sehr klein ($t = IFS + \varepsilon$ mit $\varepsilon \to 0$).

Ceilling:
Über sogenannte *Priority Ceiling* wird ein niederpriorer Task auf eine höhere Priorität gehoben, falls diese eine gemeinsame Ressource mit einer höherprioren Task besitzt.

Kritischer Zeitpunkt *(engl. Critical Instant)*:
Der kritische Zeitpunkt beschreibt den Moment, wenn die längste Antwortzeit einer Task oder Botschaft entsteht.

Datenfeldlänge *(engl. Payload)*:
Die Datenfeldlänge p_i auch *Payload* genannt, gibt die Anzahl der Datenbits innerhalb einer Botschaft an.

Dispatching:
Operation, wenn die zur Ausführung anstehender Task mit der höchsten Priorität, dem Prozessor zur Abarbeitung übergeben wird.

E-Fahrzeug:
Ein E-Fahrzeug ist ein Prototyp, welcher zur Absicherung der E/E-Umfänge während der Test- und Integrationsphase zum Einsatz kommt.

Executable:
Siehe *Binary*

Gateway:
Ein Steuergerät, welches die Aufgabe hat mehrere Busse miteinander zu koppeln und das Routing von Botschaften und Signalen durchzuführen.

Idle-Zustand:
Zustand einer Task, wenn diese nicht aktiviert ist.

Idle Zeit:
Zeitraum während der Prozessor keine Tasks ausführt (Leerlauf).

Interrupt:
Ein externes Signal, welches den Prozessor veranlasst, den aktuell in der Bearbeitung befindlichen Task zu unterbrechen, um einen anderen Prozess zu starten.

Interruptsperrzeiten:
Die Interruptsperrzeit gibt die Zeitdauer an, während der kein Interrupt die aktuell ausgeführte Routine (z.B. Interrupt-Serviceroutine oder Task) unterbrechen darf.

Jitter:
Die Differenz zwischen den Startzeitpunkten einer periodischen Task.

K-Matrix:
Kurzform für Kommunikationsmatrix, diese enthält die Spezifikation und Einstellungen eines Kommunikationssystems (z.B. Anzahl Steuergeräte, Botschaften, Signale, ...).

Kommunikationscontroller:
Dieser ist meist Bestandteil eines Mikrocontrollers und regelt den Zugriff auf ein Kommunikationssystem.

Kontext:
Eine bestimmte Menge an Daten, welche den Status des Prozessors zu einem bestimmten Zeitpunkt beschreiben, während der Ausführung einer Task. Typischerweise ist der Kontext einer Task die Werte, die bei deren Ausführung in den Registern des Prozessors stehen.

Kontext-Switch-Overhead:
Der Kontext-Switch-Overhead O beschreibt die Zeit, die für die Sicherung oder Wiederherstellung eines Systemzustandes benötigt wird, z.B. Sicherung der Register beim Eintreffen eines Interrupts.

Last *(engl. Load)*:
Benötigte Rechenzeit einer Task in einem bestimmten Intervall, geteilt durch die Länge des Intervalls.

Mehrfachaktivierung:
Die Mehrfachaktivierung eines Task tritt dann auf, wenn der Task nicht innerhalb seiner Deadline bzw. Periode vollständig ausgeführt wurde. D.h. die Ausführung der vorangegangenen Aktivierung wird erst nach der erneuten Aktivierung des Tasks beendet.

Mikrocontroller:
Ein Mikrocontroller ist ein Halbleiterbaustein, der neben der CPU auch noch Peripheriekomponenten (z.B. Bus-Controller, A/D-Wandler, etc.) und Speicher (RAM, Flash, etc.) enthält.

OSI-Schichtenmodell:
Das OSI-Schichten- oder Referenzmodell *(engl.Open Systems Interconnection Reference Model)* bildet die Grundlage für die Beschreibung von Kommunikationssystemen. Über insgesamt sieben Schichten wird das Verhalten und die Umsetzung der Kommunikation beschrieben. Von der physikalischen Ebene (Schicht 1) über die Transportorientierten Ebenen (Schicht 2 bis Schicht 4) bis zur Applikationsebene (Schicht 5 bis Schicht 7) wird über die einzelnen Abstraktionsebenen der Kommunikationsstack eines Systems aufgeteilt.

Overhead:
Zeitdauer die von einem Prozessor benötigt wird, um alle internen Aufgaben des Betriebssystems abzuarbeiten. Bei Kommunikationssystemen wird damit der Teil einer Botschaft verstanden, der keine Nutzdaten enthält.

Vorhersagbarkeit *(engl. Predictability)*:
Wichtige Eigenschaft eines Echtzeitsystems, welches die Konsequenzen von Scheduling-Entscheidungen voraussagt.

Prototype:
Ein Prototyp (altgr. protos = der Erste und typos = Urbild, Vorbild) stellt in der Technik ein für die jeweiligen Zwecke funktionsfähiges, oft aber auch vereinfachtes Versuchsmodell eines geplanten Produktes oder Bauteils dar. Es kann dabei nur rein äußerlich oder auch technisch dem Endprodukt entsprechen. Ein Prototyp dient oft als Vorberei-

tung einer Serienproduktion, kann aber auch als Einzelstück geplant sein, welches nur ein bestimmtes Konzept illustrieren soll. Entsprechend ist der Prototyp auch ein wesentlicher Entwicklungsschritt im Rahmen des Designs und wird nicht nur in technischen Zusammenhängen genutzt [142].

Runtime Environment:
Die Laufzeitumgebung (Runtime Environment, RTE) ist das Kenstück des Architekturkonzeptes von AUTOSAR. Die RTE ist eine Kommunikationsschicht, die auf der Basis einer Middleware die Software-Komponenten (Funktionen) der Steuergeräte von der Topologie und den Kommunikationsbeziehungen abstrahiert.

Scheduling:
Das Scheduling beschreibt die Aktivität, bei der die Ausführungsreihenfolge der Tasks auf einem Prozessor festgelegt wird. In diesem Zusammenhang wird auch oft von dem *Schedule (dem Fahrplan)* eines Systems gesprochen.

Sicherungsschicht *(engl. Data Link Layer)*:
Aufgabe der Sicherungsschicht ist es, eine zuverlässige, d.h. weitgehend fehlerfreie Übertragung zu gewährleisten und den Zugriff auf das Übertragungsmedium zu regeln.

Software-Komponenten:
Der Begriff Software-Komponente *(engl. Software-Component (SW-C))* wird innerhalb der AUTOSAR-Software-Architektur verendet. Eine SW-C kapselt einen Teil der Funktionalität einer Applikation, welche auf einem Steuergerät integriert ist. Eine SW-C ist atomar und kann nicht über mehrere Steuergeräte verteilt werden.

Synchronisation:
Mechanismus, welcher die gleichzeitige Ausführung von mehreren Tasks verhindert bzw. den gleichzeitigen Zugriff auf den Bus unterbindet.

Verfügbarkeit *(engl. Availability)*:
Umschreibung oder Maß für die korrekte Bereitstellung von Rechen- bzw. Übertragungskapazität zu einem bestimmten Zeitpunkt.

Virtual Function Bus (VFB):
Der VFB ist eines der zentralen Elemente von AUTOSAR. Über den VFB werden die Applikationen von der Infrastruktur entkoppelt. Die RTE ist die Realisierung des VFB.

Abkürzungen

API ..... Application Programming Interface
ASIC ..... Application Specific Integrated Circuit
ARTOP ..... AUTOSAR Tool Platform
AUTOSAR ..... **AUT**omotive **O**pen **S**ystem **AR**chitecture
BCET ..... Best-Case Execution Time
BCRT ..... Best-Case Response Time
BR ..... Baureihe
BSW ..... Basic Software
C2C ..... Car to Car
C2I ..... Car to Infrastructure
CAN ..... Controller Area Network
CHI ..... Controller Host Interface
COM ..... Communication
CPU ..... Central Processing Unit
CRC ..... Cyclic Redundancy Check
CSMA ..... Carrier Sense Multiple Access
DLC ..... Data Length Code
EA ..... Evolutionary Algorithm
ECU ..... Electronic Control Unit
EDF ..... Earliest-Deadline First
EMV ..... Elektromagnetische Verträglichkeit
FIBEX ..... Field Bus Exchange Format
FIFO ..... First Input First Output
FPGA ..... Field Programmable Gate Array
GPIO ..... General Purpose I/O
GW ..... Gateway

HAL ..... Hardware Abstraction Layer
HIL ..... Hardware-in-the-Loop-Simulation
HIS ..... Herstellerinitiative Software
HW ..... Hardware
ID ..... Identifier
ILP ..... Integer Linear Program
I/O ..... Input/Output
IP ..... Intellectual Property
LAN ..... Local Area Network
LIN ..... Local Interconnect Network
LLC ..... Logical Link Control
MAC ..... Media Access Control
MCAL ..... Microcontroller Abstraction Layer
MCU ..... Mikro Controller Unit
MOST ..... Media Oriented Systems Transport
NM ..... Network Management
NRZ ..... Non Return to Zero
OBD ..... On-Board Diagnosis
OEM ..... Original Equipment Manufacturer
OIL ..... OSEK Implementation Language
OS ..... Operating System
OSI ..... Open System for Interconnection
PDU ..... Protocol Data Unit
PHY ..... Physical Bus Connect
PLL ..... Phase Locked Loop
RISC ..... Reduced Instruction Set Computing
RTE ..... Runtime Environment
RTC ..... Real-Time Calculus
SG ..... Steuergerät
ST ..... Sendetyp
SW ..... Software
TADL ..... Timing Augmented Description Language
TCP/IP ..... Transmission Control Protocol/Internet Protocol

TDL ..... Timing Description Language
TDMA ..... Time Division Multiple Access
TIMMO ..... Timing Model
TT ..... Time Triggered
TTCAN ..... Time-Triggered Controller Area Network
TTP ..... Time-triggered Protocol
TP ..... Transport Protocol
VFB ..... Virtual Function Bus
XBW ..... X-By-Wire
XML ..... Extended Markup Language
WCET ..... Worst-Case Execution Time
WCRT ..... Worst-Case Response Time

Symbole

A ..... Spontanes Ereignismodell
a ..... Aktivierungszeitpunkt einer Task/Botschaft
B ..... Burst
b ..... Startzeitpunkt der Ausführung/Übertragung
C ..... Ausführungszeit oder Übertragungszeit (Computation or transmission time)
c ..... Endzeitpunkt einer Ausführung/Übertragung
D ..... Drift
d ..... Deadline einer Task/Botschaft
E ..... Ereignisstrom
e ..... Ereignis (Event)
F ..... Kapazität
f ..... Fensterbreite
G ..... Menge an Aktivierungen einer spontanen Botschaft
g ..... Aktivierung einer spontan Botschaft
H ..... Hyperperiode oder Makroperiode
I ..... Menge an Intervallen
i ..... Intervall
J ..... Jitter
J_{abs} ..... Absoluter Jitter
J_{in} ..... Eingangsjitter
J_{out} ..... Ausgangsjitter
J_{rel} ..... Relativer Jitter
K ..... K-Matrix
L ..... Latenzzeit
L_{ft} ..... Ende-zu-Ende Latenzzeit mit *First Through* Semantik
L_{ma} ..... Ende-zu-Ende Latenzzeit mit *Max. Age* Semantik

L_{route} ..... Routingzeit
M ..... Menge an Botschaften
M_{csx} ..... Menge an Botschaften mit periodischem und spontanem Auftrittsverhalten
M_{dual} ..... Menge an Botschaften, die zwei verschiedene Zykluszeiten haben
$M_{dynamic}$ ..... Menge an Botschaften mit dynamischem Auftrittsverhalten
M_{ecu} ..... Menge an Steuergeräten
$M_{periodic}$ ..... Menge an Botschaften mit periodischem Auftrittsverhalten
$M_{spontan}$ ..... Menge an Botschaften mit spontan Auftrittsverhalten
M_{static} ..... Menge an Botschaften mit statischem Auftrittsverhalten
MT ..... Macrotick
μT ..... Microtick
m ..... Botschaft (Message)
n ..... Repetition Factor
O ..... Overhead
O_{activ} ..... Overhead bevor eine Task aktiviert wird
O_{term} ..... Overhead wenn eine Task unterbrochen oder beendet wird
P ..... Periodisches Ereignismodell
p ..... Länge des Datenfeldes einer Botschaft in Bit
R ..... Antwortzeit (Response Time)
R_{rel} ..... Relative Antwortzeit
r ..... Aktivierungsbegrenzung
S ..... Menge an Betriebsszenarien
s ..... Szenario
T ..... Periode
T_{block} ..... Sperrzeit (Blocking Time)
T_{com} ..... Periode des COM-Tasks
T_{cycle} ..... Dauer eines kompletten Zykluses
T_{exe} ..... Überschreitungszeit (Exeeding Time)
T_{idle} ..... Idle-Zeit
T_{ifs} ..... Dauer des Interframe-Space bei CAN
T_{late} ..... Verspätung (Lateness)
T_{lax} ..... Laxity
T_{min} ..... Minimaler Auftritts-/Sendeabstand (Minimum distance)

T_{off} ..... Offset
T_{PduR} ..... Latenzzeit für ein PDU-Routing
T_{route} ..... Latenzzeit für ein Routing
T_{SR} ..... Latenzzeit für ein Signal-Routing
T_{start} ..... Startzeit
t ..... Zeit
U ..... Auslastung
V ..... Übertragungsgeschwindigkeit
W ..... Menge an Tasks
w ..... Task
X ..... Menge an Signalen
x ..... Signal
Y ..... Loggingdatensatz
y ..... Tuple innerhalb eines Loggingdatensatzes
Z ..... Zyklus

A. Literaturverzeichnis

[1] AbsInt - Angewandte Informatik GmbH. *www.absint.com*, 2010.

[2] K. Albers, F. Bodmann, and F. Slomka. Hierarchical Event Streams and Event Dependency Graphs: A New Computational Model for Embedded Real-Time Systems. In *IEEE Proceedings of the 18th Euromicro Conference of Real-Time Systems (ECRTS'06) in Dresden, Germany*, 2006.

[3] K. Albers, F. Bodmann, and F. Slomka. Advanced Hierarchical Event-Stream Model. In *Proceedings of the 20th Euromicro Conference on Real-Time Systems (ECRTS'08) in Prague, Czech Republic*, pages 211–220, 2008.

[4] K. Albers, S. Kollmann, F. Bodmann, and F. Slomka. Advanced Hierarchical Event-Stream Model and the Real-Time Calculus. Technical report, University of Ulm, Germany, 2008.

[5] K. Albers and F. Slomka. An Event Stream Driven Approximation for the Analysis of Real-Time Systems. In *Proceedings of the 16th Euromicro Conference (ECRTS'04) in Palma de Mallorca, Spain*, pages 187–195, 2004.

[6] Aquintos GmbH. *www.aquintos.com*, 2010.

[7] N. C. Audsley, A. Burns, M. Richardson, K. Tindell, and A. J. Wellings. Applying New Scheduling Theory to Static Priority Pre-emptive Scheduling. *Software Engineering Journal*, pages 284–292, 1993.

[8] AUTOSAR Consortium. *AUTOSAR Technical Overview (Release 3.1)*, 2008.

[9] AUTOSAR Consortium. *Specification of CAN Transport Layer (Release 3.1)*, 2008.

[10] AUTOSAR Consortium. *Specification of Communication (Release 3.1)*, 2008.

[11] AUTOSAR Consortium. *Specification of FlexRay Interface (Release 3.1)*, 2008.

[12] AUTOSAR Consortium. *Specification of FlexRay Transport Layer (Release 3.1)*, 2008.

[13] AUTOSAR Consortium. *Specification of Networkmanagement (Release 3.1*, 2008.

[14] AUTOSAR Consortium. *AUTOSAR Timing Concepts for Phase III*, 2009.

[15] AUTOSAR Consortium. *Specification of Timing Extensions (Release 4.0)*, 2009.

[16] AUTOSAR Consortium. *Specification of Operating System (Release 3.1)*, 2010.

[17] AUTOSAR Consortium. *www.autosar.org*, 2010.

[18] F. Baccelli, G. Cohen, G. Olsder, and G.-P. Quadrat. *Synchronization and Linearity*. John Wiley & Sons Ltd., 1992.

[19] H.-H. Braess and U. Seiffert, editors. *Handbuch der Kraftfahrzeugtechnik*, volume 4. Vieweg+Teubner Verlag, 2005.

[20] R. Brendle. Umsetzung einer Analyseumgebung zur Auswertung von CAN-Logging-Datensätzen aus realen Fahrzeugen. Master's thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, Deutschland, 2007.

[21] G. Buttazzo. *Hard Real-Time Computing Systems - Predictable Scheduling Algorithms and Applications*, volume 1. Springer Verlag, 2005.

[22] S. Chakraborty, S. Kuenzeli, and L. Thiele. A General Framework for Analysing System Properties in Platform-Based Embedded System Designs. In *Proceedings of the Design, Automation and Test in Europe Conference (DATE'03): Munich, Germany*, 2003.

[23] D. Chokshi and P. Bhaduri. Modeling Fixed Priority Non-Preemptive Scheduling with Real-Time Calculus. In *Proceedings of the Embedded and Real-Time Computing Systems and Applications Conference (RTCSA'08) in Kaohsiung, Taiwan*, pages 387–392, 2008.

[24] Daimler AG. *Overall Glossary of the Project Large Car Platform Project*, 2007.

[25] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien. *Real-Time Systems*, chapter Controller Area Network (CAN) Schedulability Analysis: Refuted, Revisited and Revised, pages 239–272. Springer Verlag, 2007.

[26] R. Denkelmann and K. Baron. Virtuelle Hardware: Ein zukunfsweisender Ansatz in der Steuergeräte-Enwicklung, 2008.

[27] J. M. Drake, M. G. Harbour, J. J. G. Gutiérrez, and J. C. P. Palencia. Modeling and Analysis Suite for Real Time Applications (MAST), 2002.

[28] dSpace GmbH. *www.dspace.de*, 2010.

[29] M. Däumler. Timing Analysis in Software Development. Master's thesis, Chemnitz University of Technology, Germany, 2008.

[30] E|ENOVA - Innovationsallianz Automoilelektronik. *www.eenova.de*, 2009.

[31] Eidgenössische Technische Hochschule Zürich, Schweiz. *www.ethz.ch*, 2010.

[32] Electrobit GmbH. *www.electrobit.de*, 2010.

[33] S. Esch and B. Lang. Elektronik- und Vernetzungsarchitektur mit gesteigerter Leistungsfähigkeit. *ATZ*, pages 194–198, 2008.

[34] K. Etschberger, editor. *Controller Area Network - Grundlagen, Protokolle, Bausteine, Anwendungen*, volume 2. Hanser Verlag, 2001.

[35] N. Feiertag, K. Richter, and J. Nordlander. A Compositional Framework for End-to-End Path Delay Calculation of Automotive Systems Under Different Path Semantics. In *Proceedings of the IEEE Real-Time System Symposium (RTSS), Workshop on Compositional Theory and Technology for Real-Time Embedded Systems: Barcelona, Spain*, 2008.

[36] A. Ferrari, M. Di Natale, G. Gentile, and P. Gai. Time and Memory Tradeoffs in the Implementation of AUTOSAR Components. In *Proceedings of the Design, Automation and Test in Europe Conference (DATE'09) in Munich, Germany*, 2009.

[37] FlexRay Consortium. *FlexRay Communication System - Protocol Specification (Version 3.0)*, 2010.

[38] FlexRay Consortium. *www.flexray.com*, 2010.

[39] J. Freess. *Modelle zur Beschreibung und Evaluierung von Architekturkonzepten der Elektrik und Elektronik im Kraftfahrzeugen*. PhD thesis, Universität Fridericiana Karlsruhe, 2006.

[40] E. Frickenstein. Die Zukunft der E/E Architektur in der Fahrzeugindustrie. In *Proceedings of the 13. Internationaler Fachkongress - Fortschritte in der Automobilelektronik in Ludwigsburg, Germany*, 2009.

[41] Gliwa GmbH. *T1 - Timing Measurement Tool*, 2009.

[42] Greenhills Ltd. *www.ghs.com*, 2010.

[43] M. Grenier, L. Havet, and N. Navet. Scheduling Messages with Offsets on Controller Area Network - A Major Performance Boost. Technical report, LORIA, Nancy Univerity, 2008.

[44] K. Gresser. *Echtzeitnachweis ereignisgesteuerter Realzeitsysteme*. PhD thesis, Technische Universität München, 1993.

[45] H. Grönninger. Formale Analyse eines automotive Bussystems mit SymTAS auf der Grundlage von K-Matrizen. Master's thesis, Technische Universität Carolo-Wilhelmina zu Braunschweig, Deutschland, 2005.

[46] A. Hagiescu, U. Bordoloi, S. Chakraborty, P. Sampath, V. Ganesan, and S. Ramesh. Performance Analysis of FlexRay-based ECU Networks. In *Proceedings of the 44th Design Automation Conference (DAC'07) in San Diego, USA*, pages 284–289, 2007.

[47] B. Hedenetz. *Entwurf von verteilten, fehlertoleranten Elektronikarchitekturen inchron Kraftfahrzeugen*. PhD thesis, Universität Tübingen, Deutschland, 2001.

[48] H. Heinecke, M. Rudorfer, C. Ainhauser, and O. Scheickl. Enabling of AUTOSAR System Design using Eclipse-based Tooling. In *Proceedings of the Embeded Real Time Software Conference (ERTS): Toulouse, France*, 2008.

[49] M. Hendricks and M. Verhoef. Timed Automata Based Analysis of Embedded System Architectures. In *Proceedings of the Parallel and Distributed Processing Symposium (IPDPS'06) on Rhodes Island, Greece*, 2006.

[50] R. Henia, A. Hamann, M. Jersak, K. Richter, and R. Ernst. System Level Performance Analysis - The SymTA/S Approach. *IEEE Proceedings Computers and Digital Techniques*, 2005.

[51] M. Hoffmann. Praktikumsbericht, 2008.

[52] A. Hogh-Binder. Untersuchung und Bewertung eines AUTOSAR-basierten Gateway-Systems mit Hilfe von Zeitanalyse-Werkzeugen. Master's thesis, Universität Karlsruhe, Deutschland, 2009.

[53] M. Homan. *OSEK, Betriebssystem-Standard für Automotive und Embedded Systems*. Mitp-Verlag, 2005.

[54] Inchron GmbH. *ChronSim - Echtzeitsimulator für eingebettete Systeme und Netzwerke*, 2010.

[55] Inchron GmhH. *www.inchron.de*, 2010.

[56] IXXAT GmbH. *www.ixxat.de*, 2010.

[57] T. Jablonski, C. Busse, D. Brinkema, M. Jersak, and K. Richter. Timing-Analysen als Sicherheitsnachweis. *AUTOMOTIVE*, pages 42–45, November 2008.

[58] M. Jersak. Timing-Modell und Methodik für AUTOSAR. *Elektronik Automotive*, pages 9–10, 2007.

[59] M. Jersak, R. Henia, and R. Ernst. Context-Aware Performance Analysis for Efficient Embedded System Design. In *Proceedings of the Design, Automation and Test Conference (DATE'04) in Munich, Germany*, 2004.

[60] M. Jersak, R. Kai, H. Sarnowski, and P. Gliwa. Laufzeitanalysen zur frühzeitigen Absicherung von Software. *ATZelektronik*, 1(4):52–54, Jan./Feb. 2009.

[61] D. Kerk. OSEK - Echzeitbetriebssystem für Automobile. Master's thesis, Technische Universität Carolo-Wilhelmina zu Braunschweig, Deutschland, 2003.

[62] R. Klein. *Prinzipien des Algorithmenentwurfs*. Spektrum Akademischer Verlag, 1998.

[63] S. Kollmann, K. Albers, and F. Slomka. Limiting Event Streams: A General Model to Describe Dependencies in Distributed Hard Real-Time Systems. Technical report, University of Ulm, Germany, 2010.

[64] S. Kollmann, V. Pollex, F. Slomka, M. Traub, T. Bone, and J. Becker. Comparison of Different Timing-Evaluation Methods based on a Automotive Network Topology. To be published, 2010.

[65] H. Kopetz. *Real-Time Systems - Design Principles for Distributed Embedded Applications*, volume 1. Kluwer Academic Publishers Verlag, 1997.

[66] O. Krasovyskyy. Konzeption und Umsetzung eines Verfahrens zur systematischen Auswertung der statischen und dynamischen Kommunikation zur Erzeugung von Betriebsszenarien. Master's thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, Detuschland, 2009.

[67] M. Krause, O. Bringmann, A. Hergenhan, G. Tabanoglu, and W. Rosenstiel. Timing Simulation of Interconnected AUTOSAR Software-Components. In *Proceedings of the Design, Automation and Test in Europe Conference (DATE'07) in Munich, Germany*, 2007.

[68] M. Krause, O. Bringmann, and W. Rosenstiel. Target Software Generation: An Approach for Automatic Mapping of SystemC Specifications onto Real-Time Operating Systems. *Design Automation for Embedded Systems*, 2007.

[69] W. Lawrenz. *CAN Controller Area Network - Grundlagen und Praxis*, volume 4. Hüthig Verlag, 2001.

[70] J.-Y. Le Boudec and P. Thiran. *Network Calculus*. Springer Verlag, 2004.

[71] LIN Consortium. *LIN Specification Package*, 2003.

[72] LIN Consortium. *www.lin-subbus.org*, 2010.

[73] C. L. Liu and J. W. Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment, 1973.

[74] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of Frequent Episodes in Event Sequences. *Data Mining and Knowledge Discovery*, pages 259–289, 1997.

[75] The MathWorks. *www.mathworks.com*, 2010.

[76] M. Mauritz. Lastenheft zur Gateway-Analyse, 2009.

[77] M. Mauritz. Timing-Analyse von Automotive Gateway-Systemen. Master's thesis, Technische Universität Carolo-Wilhelmina zu Braunschweig, Deutschland, 2010.

[78] A. Michailidis, U. Spieth, T. Ringler, B. Hedenetz, and S. Kowalewski. Test Front Loading in Early Stages of Automotive Software Development Based on AUTOSAR. In *Proceedings of the Design Automation and Test in Europe Conference (DATE'10) in Dresden, Germany*, 2010.

[79] R. Münzenberger, M. Dörfel, C. Diedrichs, U. Margull, and G. Wirrer. Entwurf echtzeitfähiger Steuergerätesoftware in FlexRay-Netzwerken, 2007.

[80] P. Montag, S. Goerzig, and P. Levi. Applying Static Timing Analysis to Component Architectures. In *Proceedings of the International Workshop of Software Engineering for Automotive Systems (SEAS'06): Shanghai, China*, pages 21–28, 2006.

[81] P. Montag, S. Goerzig, and P. Levi. Challenges of Timing Verification Tools in the Automotive Domain. In *Proceedings of the Second International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA'06) in Paphos, Cyprus*, pages 227–232, 2006.

[82] MOST Cooperation. *MOST - Media Oriented System Transport (Rev. 2.4)*, 2005.

[83] MOST Cooperation. *www.mostnet.de*, 2010.

[84] NEC Electronics. *Preliminary User's Manual - V850E/PHO3 - 32-bit Single-Chip Microcontroller*, 2007.

[85] NEC Electronics. *www.necel.com/index.html*, 2010.

[86] T. Nolte, H. Hansson, and C. Nordstroem. Minimizing CAN Response-Time Jitter by Message Manipulation. In *Proceedings of the 8th Real-Time and Embedded Technology and Applications Symposium in San Jose, USA*, pages 197–206, 2002.

[87] OSEK/VDX Consortium. *OSEK/VDX - Operationg System (Version 2.2.3)*, 2005.

[88] OSEK/VDX Consortium. *www.osek-vdx.org*, 2010.

[89] S. Perathoner, E. Wandelder, L. Thiele, A. Hamann, S. Schliecker, R. Henia, R. Racu, R. Ernst, and M. G. Harbour. *Design Auotmation for Embedded Systems*, chapter Influence of Different Abstractions on the Performance Analysis of Distributed Hard Real-Time Systems. Springer Verlag, 2008.

[90] T. Pop, E. Petru, and Z. Peng. Performance Estimation for Embedded Systems with Data and Control Dependencies. In *Proceedings of the eighth international workshop on Hardware/software codesign in San Diego, USA*, 2000.

[91] R. Racu, R. Ernst, M. Jersak, and K. Richter. A Virtual Platform for Architecture Integration and Optimization in Automotive Communication Networks. In *Proceedings of the SAE World Congress and Exhibition: Detroit, USA*, 2007.

[92] R. Racu, A. Hamann, and E. Rolf. *Real-Time Systems*, chapter Sensitivity analysis of complex embedded real-time systems. Springer Verlag, 2007.

[93] M. Rahmanil, J. Hillebrand, W. Hintermairl, R. Bogenberger, and E. Steinbach. A Novel Network Architecture for In-Vehicle Audio and Video Communication. In *Proceedings of the 2nd IEEE/IFIP International Workshop on Broadband Convergence Networks, (BcN '07) in Munich, Germany*, 2007.

[94] M. Rausch. *FlexRay - Grundlagen, Funktionsweise, Anwendung*, volume 1. Hanser Verlag, 2007.

[95] S. Reichelt, O. Scheickl, and G. Tabanoglu. The Influence of Real-time Constraints on the Design of FlexRay-based Systems. In *Proceeding of the Design, Automation and Test in Europe Conference (DATE'09) in Nice, France*, 2009.

[96] K. Richter. *Compositional Scheduling Analysis Using Standard Event Models*. PhD thesis, Technische Universität Carolo-Wilhelmina zu Braunschweig, 2004.

[97] K. Richter. Potential von FlexRay optimal nutzen - Teil2: Analyse und Optimierung der Echtzeit-Fähigkeit von verteilten FlexRay-Systemen. *Elektronik Automotive*, pages 42–45, 2008.

[98] K. Richter. Potential von FlexRay optimal nutzen - Teil1: Echtzeit-Fähigkeit im verteilten Regler-Entwurf. *Elektronik Automotive*, pages 42–45, Jan./Feb. 2009.

[99] K. Richter and R. Ernst. Event Model Interfaces for Heterogeneous System Analysis. In *Proceedings of Design, Automation and Test in Europe Conference (DATE'02) in Munich, Germany*, pages 506–513, 2002.

[100] K. Richter, M. Jersak, and R. Ernst. A Formal Approach to MpSocC Performance Verification. Technical report, IEEE Computer Science, 2003.

[101] K. Richter, M. Jersak, and R. Ernst. Learning Early-Stage Platform Dimensioning From Late-Stage Timing Verification. In *Proceedings of the Design, Automation and Test in Europe Conference (DATE'09) in Munich, Germany*, 2009.

[102] T. Ringler. *Entwicklung und Analyse zeitgesteuerter Systeme*. PhD thesis, Universität Stuttgart, 2002.

[103] T. Ringler, M. Simons, and R. Beck. Ein Ansatz für den werkzeuggestützten Elektrik-/Elektronikarchitekturentwurf. *ATZelektronik*, pages 52–57, 2008.

[104] Robert Bosch GmbH. *Controller Area Network (CAN) Specification - Version 2.0*, 1991.

[105] J. Rox and R. Ernst. Construction and Deconstruction of Hierarchical Event Streams with Multiple Hierarchical Layers. In *Proceedings of the Euromicro Conference of Real-Time Systems (ECRTS'08) in Prague, Czech Republic*, pages 201–210, 2008.

[106] J. Ruh. *Entwurf von fehlertoleranten Steuergeräteapplikationen in Kraftfahrzeugen unter Berücksichtigung moderner Entwicklungsmethodiken*. PhD thesis, Eberhard-Karls-Universität, Tübingen, 2005.

[107] O. Scheickl and M. Rudorfer. Automotive Real Time Development Using a Timing-augmented AUTOSAR Specification. In *Proceedings of the Embedded Real-Time Software Conference (ECRTS'08): Toulouse, France*, 2008.

[108] O. Scheickl, M. Rudorfer, C. Ainhauser, N. Feiertag, and K. Richter. How Timing Interfaces in AUTOSAR can Improve Distributed Development of Real-Time Software. In *Proceedings of the 6th Workshop of Automotive Software Engineering in Munich, Germany*, 2008.

[109] D. Schilpp. Analyse, Simulation und Optimierung der Kommunikationseigenschaften des CAN-Busses. Master's thesis, Reinhold-Würth-Hochschule der Hochschule Heilbronn/Künzelsau, Deutschland, 2008.

[110] B. Schürmann. *Rechnerverbindungsstrukturen - Bussysteme und Netzwerke*, volume 1. Vieweg+Teubner Verlag, 1997.

[111] J. Schäuffele and T. Zurawka. *Automotive Software Engineering - Grundlagen, Prozesse, Methoden und Werkzeuge*. Vieweg+Teubner Verlag, 2005.

[112] U. Seiffert and G. Rainer, editors. *Virtuelle Produktentstehung für Fahrzeug und Antrieb im KFZ*, chapter HW-/SW-Co-Simulation, pages 99–153. Springer Verlag, 2008.

[113] S. Simon. Konzeption, Modellierung und Bewertung des zeitlichen Verhaltens von Gateway-Systemen. Master's thesis, Universität Illmenau, 2010.

[114] A. Swietlik and J. Spale. Vorlesung: Echzeitbetriebssysteme an der Hochschule Furtwangen, 2008.

[115] Symtavision GmbH. *SymTA/S - Intro and Theory Manual (Version 1.4.2)*, 2009.

[116] Symtavision GmbH. *www.symtavision.com*, 2010.

[117] A. S. Tanenbaum. *Moderne Betriebssysteme*, volume 2. Pearson Studium Verlag, 2003.

[118] L. Thiele. Scalable Software for MPSoCPlatforms. In *Proceedings of the ARTIST Summer School in Autrans, France*, 2009.

[119] L. Thiele, S. Chakraborty, and M. Naedele. Real-Time Calculus for Scheduling Hard Real-Time Systems. In *Int. Symposium on Circuits and Systems (ISCAS'00) in Geneva, Switzerland*, pages 101–104, 2000.

[120] TIMING MODEL - Mastering In-Vehicle Timing Constraints. *www.timmo.org*, 2009.

[121] K. Tindell. An Extendible Approach For Analysing Fixed Priority Hard Real-Time Tasks. *Journal of Real-Time Systems*, 6, 1992.

[122] K. Tindell. Adding Time-Offsets to Schedulability Analysis. Technical report, University of York, England, 1994.

[123] K. Tindell. Analysing Real-Time Communications: Controller Area Network (CAN), 1994.

[124] K. Tindell and B. Alan. Guaranteed Message Latencies For Distributed Safety-Critical Hard Real-Time Control Networks, 1994.

[125] K. Tindell, A. Burns, and A. Wellings. Calculating Controller Area Network (CAN) Message Response Times. *Control Engineering Practice*, pages 1163–1169, 1995.

[126] K. Tindell and J. Clark. Holistic Schedulability Analysis for Real-Time Systems, 1994.

[127] K. Tindell and J. Clark. Holistic Schedulability for DISTRIBUTED Hard REAL-TIME SYSTEMS. *Microprocessing & Microprogramming*, 40:117–134, 1994.

[128] M. Traub, V. Lauer, and J. Becker. Verfahren zur Timing-Bewertung von Gateway-Systemen und Vernetzungsarchitekturen in den verschiedenen Phasen des Entwicklungsprozesses. In *Proceedings of the Elektronik im Kraftfahrzeug Konferenz in Dresden, Germany*, 2009.

[129] M. Traub, V. Lauer, B. Hedenetz, M. Conrath, M. Jersak, K. Richter, and C. Reichmann. In Search of the Best Migration Strategy from CAN to FlexRay. *Hanser FlexRay-Special*, 2009.

[130] M. Traub, V. Lauer, M. Jersak, K. Richter, J. Becker, and M. Kühl. Using Timing Analysis for Evaluating Communication Behavior and Network Topologies in an Early Design Phase of Automotive Electric/Electronic Architectures. In *Proceedings of the SAE World Congress in Detroit, USA*, 2009.

[131] M. Traub, V. Lauer, T. Weber, K. Richter, M. Jersak, and J. Becker. Timing-Analysen für die Untersuchung von Vernetzungsarchitekturen. *ATZelektronik*, pages 36–41, 2009.

[132] M. Traub, T. Streichert, O. Krasovyskyy, and J. Becker. Scenario Extraction for a Refined Timing-Analysis of Automotive Network Topologies. In *Proceedings of the Design, Automation and Test in Europe Conference (DATE'10) in Dresden, Germany*, 2010.

[133] D. Unger. Konzeption und Umsetzung eines Verfahrens zur automatischen Konfiguration von Zeitanalyse-Werkzeugen anhand von AUTOSAR-System-Beschreibungen. Master's thesis, Universität Karlsruhe, Deutschland, 2010.

[134] Universität Aalborg (Dänemark) and Universität Uppsala (Schweden). *Tool Environment for Validation and Verification of Real-Time Systems*, 2010.

[135] Universität Ulm. *www.uni-ulm.de*, 2010.

[136] VaST Systems. *www.vastsystems.com*, 2010.

[137] Vector-Informatik. *Daimler Software License Package (SLP10) - Startup Integration Step by Step*, 2009.

[138] Vector Informatik GmbH. *Specification of Daimler-Benz Communications Attributes (DBKOM*, 2003.

[139] Vector Informatik GmbH. *www.vector-informatik.de*, 2010.

[140] H. von Winner, S. Hakuli, and G. Wolf, editors. *Handbuch Fahrerassistenzsysteme Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort* . Vieweg+Teubner Verlag, 2009.

[141] G. Weißenberger. Die virtuelle Produktenstehung: Chancen, Grenzen und Risiken. In *Proceedings der 8. Euroforum Jahrestagung Elektronik-Systeme im Automobil in Ludwigsburg, Deutschland*, 2004.

[142] Wikipedia - Freie Enzyklopädie. *www.wikipedia.de*, 2010.

[143] H. Wörn and U. Brinkschulte. *Echtzeitsysteme - Grundlagen, Funktionsweisen, Anwendungen*. Springer Verlag, 2005.

[144] W. Zimmermann and R. Schmidgall. *Bussysteme in der Fahrzeugtechnik - Protokolle und Standards*, volume 3. Vieweg+Teubner Verlag, 2008.

ISBN 978-3-86644-582-6