Frank Tränkle

Modellbasierte Entwicklung Mechatronischer Systeme

Weitere empfehlenswerte Titel

Modellbasierte Entwicklung Mechatronischer Systeme
Mit Funktionsmodellen und Laborprojekten für Servoantriebe
Jürgen Baur, geplant für 2022
ISBN 978-3-11-074444-6, e-ISBN (PDF) 978-3-11-074446-0,
e-ISBN (EPUB) 978-3-11-074448-4

Toleranzdesign
im Maschinen- und Fahrzeugbau
Bernd Klein, 2021
ISBN 978-3-11-072070-9, e-ISBN (PDF) 978-3-11-072072-3,
e-ISBN (EPUB) 978-3-11-072075-4

Maschinenelemente
Hubert Hinzen
Maschinenelemente 1, 2017
ISBN 978-3-11-054082-6, e-ISBN (PDF) 978-3-11-054087-1,
e-ISBN (EPUB) 978-3-11-054104-5
Maschinenelemente 2:
Lager, Welle-Nabe-Verbindungen, Getriebe, 2018
ISBN 978-3-11-059707-3, e-ISBN (PDF) 978-3-11-059708-0,
e-ISBN (EPUB) 978-3-11-059758-5
Maschinenelemente 3: Verspannung, Schlupf und
Wirkungsgrad, Bremsen, Kupplungen, Antriebe, 2020
ISBN 978-3-11-064546-0, e-ISBN (PDF) 978-3-11-064707-5,
e-ISBN (EPUB) 978-3-11-064714-3

Mechatronische Netzwerke
Praxis und Anwendungen
Jörg Grabow, 2018
ISBN 978-3-11-047084-0, e-ISBN (PDF) 978-3-11-047085-7,
e-ISBN (EPUB) 978-3-11-047095-6

Frank Tränkle

Modellbasierte Entwicklung Mechatronischer Systeme

mit Software- und Simulationsbeispielen für Autonomes Fahren

DE GRUYTER
OLDENBOURG

Autor

Prof. Dr.-Ing. Frank Tränkle
Fakultät Mechanik und Elektronik
Hochschule Heilbronn
Max-Planck-Str. 39
74081 Heilbronn

ISBN 978-3-11-072346-5
e-ISBN (PDF) 978-3-11-072352-6
e-ISBN (EPUB) 978-3-11-072355-7

Library of Congress Control Number: 2021940890

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.dnb.de abrufbar.

Druck und Bindung: CPI books GmbH, Leck
Coverabbildung: Frank Tränkle

www.degruyter.com

Vorwort

Die modellbasierte Entwicklung mechatronischer und cyber-physischer Systeme ist eine Erfolgsgeschichte. Ohne den Einsatz modellbasierter Methoden wären die Innovationen der letzten beiden Jahrzehnte im Automobilbereich und in der Industrieautomatisierung nicht möglich gewesen. Die modellbasierte Entwicklung setzt mathematische Modelle zur Beschreibung technischer Prozesse und Systeme ein, um Entwicklungsumfänge durch modellbasierten Entwurf, Analyse und Computersimulation in frühe Phasen der Systementwicklung zu verlagern. Dadurch wird eine Kostenreduktion in der Systementwicklung bei einhergehender Erhöhung der Qualität, Sicherheit und Verfügbarkeit erzielt.

Bereits seit Mitte der 1990er Jahre wird die modellbasierte Entwicklung von mir mitgestaltet und angewendet. Ich entwickelte in meiner Doktorarbeit an der Universität Stuttgart das Modellierungswerkzeug PROMOT und die objektorientierte Modellierungssprache MDL zur Modellierung und Simulation dynamischer, verfahrenstechnischer Prozesse. Während meiner beruflichen Tätigkeit bei ETAS und GIGATRONIK setzte ich die modellbasierte Softwareentwicklung mit Hilfe von ASCET und MATLAB®/Simulink® zur Entwicklung von Echtzeit- und Embedded-Systemen im Automobilbereich ein. An der Hochschule Heilbronn und an der Hochschulföderation Südwest unterrichte ich den Einsatz modellbasierter Methoden in der Entwicklung von Regelungsfunktionen im Fachgebiet Automotive-Systems-Engineering.

Dieses Lehrbuch bündelt meine Erfahrungen aus Industrie, Lehre und Forschung. Es behandelt Grundlagen der modellbasierten Entwicklung und deren Anwendung in der Embedded-Software-Entwicklung für mechatronische Systeme. Weiterhin werden Modellierungsrichtlinien vermittelt, die die modellbasierte Entwicklung vereinfachen und die Entwicklungsergebnisse verbessern. Als Anwendungsbeispiel wird das Laborsystem Mini-Auto-Drive (MAD) für autonomes Fahren verwendet. In MAD fahren batterieelektrische Modellfahrzeuge autonom auf einer frei konfigurierbaren Fahrbahn. In den einzelnen Buchkapiteln werden Bewegungsregelungsfunktionen für autonomes Fahren schrittweise und durchgängig entwickelt. Durchgängigkeit bedeutet, dass Leser*innen dieses Lehrbuchs die modellbasierte Entwicklung mit allen Entwicklungsschritten von der Anforderungsspezifikation bis hin zum Testen der Funktionen auf dem Laborsystem kennenlernen und durchführen. Jede Funktion wird dabei mit Hilfe regelungstechnischer Methoden entworfen, modellbasiert entwickelt, in Fahrdynamiksimulationen getestet und als Embedded-Software implementiert und in Betrieb genommen.

Mein Kollege und Freund Jürgen Baur an der Hochschule Aalen und ich starteten vor einigen Jahren mit der Erstellung neuartiger Lehrbücher in den Fachgebieten des Automotive-Systems-Engineering und der Industrieautomatisierung, die sowohl die Grundlagen als auch die durchgängige Anwendung der modellbasierten Entwicklung anhand von Anwendungsbeispielen vermitteln. Während das vorliegende Lehrbuch

https://doi.org/10.1515/9783110723526-202

die Embedded-Software-Entwicklung für autonomes Fahren im Fachgebiet des Automotive-Systems-Engineering als Anwendungsbeispiel behandelt, wendet das Lehrbuch von Jürgen Baur modellbasierte Methoden zur Entwicklung von Servoachsantrieben in der Industrieautomatisierung an. Die modellbasierte Entwicklung in beiden Fachgebieten Automotive-Systems-Engineering und Industrieautomatisierung beruht auf denselben Grundlagen und Konzepten. In beiden Lehrbüchern wird einheitlich MATLAB/Simulink als Toolkette für die modellbasierte Entwicklung verwendet.

Unterschiede bestehen in den Zielplattformen zur Regelung und Steuerung der Systeme in Echtzeit. So werden die Softwarefunktionen für autonomes Fahren auf Hochleistungsrechnern implementiert und ausgeführt. Im Laborsystem MAD werden dazu ein Echtzeit-Linux-Computer und das Robot-Operating-System (ROS) als Middleware eingesetzt. Die automatisierte Codegenerierung für die Embedded-Software des Linux-Computers erfolgt mit Hilfe des Embedded-Coders®. Embedded-Coder ist ein Zusatzprodukt zu MATLAB/Simulink, das automatisiert C/C++-Code aus Simulink-Modellen generiert. Diese Form der modellbasierten Softwareentwicklung wird auch als modellgetriebene Softwareentwicklung bezeichnet.

Da neben dem Einsatz der modellgetriebenen Softwareentwicklung mit MATLAB/Simulink die Anwendung der general-purpose Programmiersprache C++ in der Softwareentwicklung für autonomes Fahren sehr verbreitet ist, behandelt dieses Lehrbuch weiterhin die modellbasierte Softwareentwicklung mit C++. Die Leser*innen können entscheiden, ob sie die Regelungsfunktionen modellgetrieben mit MATLAB/Simulink oder modellbasiert mit C++ entwickeln.

Zielgruppen des Lehrbuchs sind Student*innen in Bachelor- oder Masterstudiengängen und berufserfahrene Softwareentwickler*innen, die sich grundlegend in die modellgetriebene oder modellbasierte Embedded-Software-Entwicklung mit MATLAB/Simulink oder C++ einarbeiten und Robot-Operating-System (ROS) unter Linux zur Steuerung und Regelung autonomer Systeme anwenden möchten. Weiterhin ist dieses Lehrbuch sehr gut als Manuskript für Vorlesungen und Labore geeignet, da es kapitelweise die Grundlagen der modellbasierten Softwareentwicklung und die schrittweise Entwicklung der Bewegungsregelungsfunktionen für autonomes Fahren im Rahmen von Laborübungen behandelt.

Heilbronn, im Juni 2021 Frank Tränkle

Inhalt

Abbildungsverzeichnis

https://doi.org/10.1515/9783110723526-204

Tabellenverzeichnis

https://doi.org/10.1515/9783110723526-205

1 Einleitung

Die *modellbasierte Entwicklung* ist eine Methode zur Entwicklung softwareintensiver, mechatronischer und cyber-physischer Prozesse und Systeme. Sie wird seit über 20 Jahren in verschiedenen Anwendungsbereichen eingesetzt und hat sich in der Entwicklung von Steuerungs- und Regelungsfunktionen auf breiter Basis durchgesetzt, vor allen Dingen in der Verfahrenstechnik, in Aerospace und in Automotive-Systems-Engineering [1]. Aufgrund dieser Erfolgsgeschichte setzt sich die modellbasierte Entwicklung auch in der Industrieautomatisierung und anderen Bereichen immer mehr durch. Nur durch Einsatz modellbasierter Entwicklungsmethoden und dabei unterstützender Modellierungs- und Simulationsumgebungen ist die Realisierung hoch komplexer mechatronischer Systeme oder Fahrzeugsysteme bei stetig steigender Variantenvielfalt und immer kürzer werdenden Entwicklungszyklen möglich. Innovative Produkte für die Automotive Megatrends Elektromobilität, vernetztes Fahren und autonomes Fahren wären ohne Einsatz der modellbasierten Entwicklung nicht realisierbar.

Die modellbasierte Entwicklung setzt mathematische Modelle zur Beschreibung technischer Prozesse und Systeme ein, um Entwicklungsumfänge durch modellbasierten Entwurf, Analyse und Computersimulation in frühe Phasen der Systementwicklung zu verlagern. Dadurch wird eine Kostenreduktion in der Systementwicklung bei einhergehender Erhöhung der Qualität, Sicherheit und Verfügbarkeit erzielt.

Der Schwerpunkt dieses Lehrbuchs liegt auf der *modellbasierten Softwareentwicklung* als Teildisziplin der modellbasierten Entwicklung. Das Produkt der modellbasierten Softwareentwicklung ist Embedded-Software, die den wesentlichen Teil der Funktionalität in heutigen und zukünftigen Systemen realisiert und auf elektronischen Steuergeräten implementiert wird. Allgemein setzt die modellbasierte Softwareentwicklung formale, simulationsfähige oder auch nicht ausführbare *Funktions- und Softwaremodelle* für den Entwurf der Embedded-Softwarefunktionen ein.

Als Anwendungsbeispiel behandelt dieses Lehrbuch die modellbasierte Softwareentwicklung von Funktionen der Bewegungsregelung (engl. Motion-Control) für autonomes Fahren. Die Methoden der modellbasierten Softwareentwicklung werden im buchbegleitenden Laborprojekt Mini-Auto-Drive (MAD) [2] angewendet und praktisch vertieft. Die Leser*innen können sich dabei für die Modellierungs- und Simulationsumgebung MATLAB®/Simulink® [3] oder alternativ für die general-purpose Programmiersprache C++ [4, 5] entscheiden. Simulink und C++ ermöglichen beide ein hohes Abstraktionsniveau in der Softwareentwicklung und werden in der Entwicklung von Softwarefunktionen für automatisiertes und autonomes Fahren auf breiter Basis sowohl in der Industrie als auch an Hochschulen und Forschungsinstituten eingesetzt.

In der modellbasierten Softwareentwicklung von Steuerungs- und Regelungsfunktionen kommen im Allgemeinen *formale, simulationsfähige Modelle* zum Einsatz,

https://doi.org/10.1515/9783110723526-001

die die Embedded-Software durch Differentialgleichungen, Differenzengleichungen, Übertragungsfunktionen, Signalflusspläne und endliche Zustandsautomaten basierend auf den Entwurfskonzepten der Steuerungstechnik, der Regelungstechnik und der digitalen Signalverarbeitung beschreiben. Diese Modelle werden zum einen dazu genutzt, die Softwarefunktionen frühzeitig in Simulationen durch Einsatz von MATLAB/Simulink oder C++ zu testen. Zum anderen generieren Codegeneratoren, wie z.B. Embedded-Coder®, aus MATLAB/Simulink/Stateflow®-Modellen automatisiert Embedded-C/C++-Code für elektronische Steuergeräte.

Während des Entwurfs und Tests der Softwarefunktionen kommen darüber hinaus dynamische Umgebungsmodelle zur Beschreibung des physikalischen, deterministischen oder stochastischen Prozess- und Systemverhaltens zum Einsatz. Diese basieren allgemein auf gewöhnlichen und partiellen Differentialgleichungen, endlichen Zustandsautomaten und algebraischen Gleichungen. Gesamtsystemmodelle, die aus den Funktions- und Softwaremodellen der Embedded-Software sowie den Umgebungsmodellen aufgebaut sind, werden zur Validierung und Verifikation der Softwarefunktionen durch Computersimulation in MATLAB/Simulink/Stateflow oder in C++ eingesetzt.

Über den Lehrbuchinhalt hinaus werden in der modellbasierten Softwareentwicklung auch UML- und SysML-Modelle für den objektorientierten Entwurf und den Systementwurf verwendet [6]. UML- und SysML-Modelle werden jedoch nicht im Rahmen dieses Lehrbuchs behandelt, da diese nur bedingt für den signalflussbasierten Entwurf von Regelungs- und Signalverarbeitungsfunktionen oder für die Autocodegenerierung entsprechender Softwarefunktionen geeignet sind.

1.1 Lehrbuchinhalte

Dieses Lehrbuch behandelt die modellbasierte Softwareentwicklung der Bewegungsregelung (engl. Motion-Control) für autonomes Fahren. Als Beispiel wird die Embedded-Software für folgende *automatisierte Fahrfunktionen* als Komponenten der Bewegungsregelung entwickelt:

- Geschwindigkeitsregelung,
- Longitudinalpositionsregelung,
- Bahnfolgeregelung.

Dabei kommen frequenzkennlinien- und zustandsraumbasierte Reglerentwurfsmethoden sowie der Vorsteuerungsentwurf aus der Regelungstechnik zum Einsatz. Die Regelungsfunktionen werden zunächst zeitkontinuierlich entworfen und dann im nächsten Schritt durch zeitdiskrete Differenzengleichungen approximiert. Diese zeitdiskrete Formulierung ist eine Voraussetzung für die Implementierung von Regelungsfunktionen als Embedded-Software auf Digitalrechner. Für den Entwurf und den Test der Regelungsfunktionen werden darüber hinaus Fahrdynamikmodelle hergeleitet und in Computersimulationen eingesetzt.

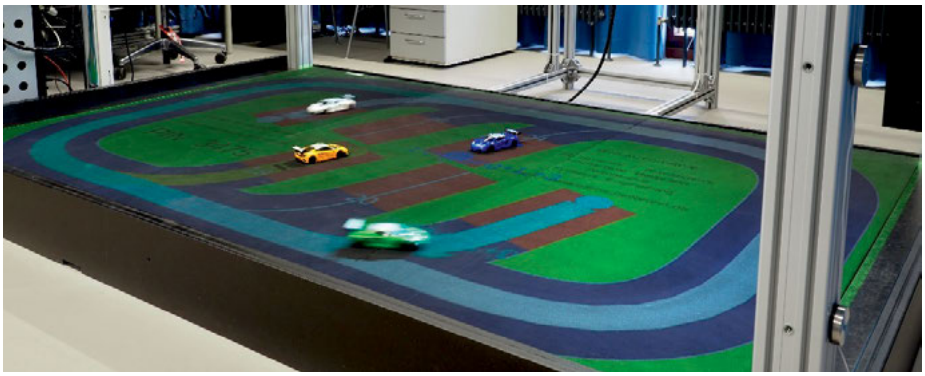

Abb. 1: Laborprojekt Mini-Auto-Drive mit autonomen Modellfahrzeugen

Im buchbegleitenden Laborprojekt Mini-Auto-Drive (MAD) wird das theoretische und praktische Wissen schrittweise angewendet und vertieft. Das Laborsystem MAD ist eine *Miniplant*[1] im Maßstab 1:24 zur Entwicklung automatisierter Fahrfunktionen. In MAD fahren batterieelektrische Modellfahrzeuge autonom auf der in Abb. 1 dargestellten horizontalen, ebenen Fahrbahn mit einer Fläche von $2{,}70m$ auf $1{,}80m$. Weitere Fotos und Videos zu MAD sind unter `https://asert.hs-heilbronn.de/` zu finden.

Die Leser*innen setzen in diesem Laborprojekt entweder die Modellierungs- und Simulationsumgebung MATLAB/Simulink [3] oder alternativ die general-purpose, objektorientierte Programmiersprache C++ [4] zur modellbasierten Softwareentwicklung der Fahrfunktionen ein. Die Leser*innen können sich dabei

- für die *modellgetriebene Softwareentwicklung* mit MATLAB/Simulink
- oder die *modellbasierte Softwareentwicklung* mit Modern C++ ab Version C++14

1 *Miniplants* sind Anlagen im Labormaßstab, mit welchen durch Rapid-Control-Prototyping neue technische Prozesse und Systeme entwickelt und erprobt werden.

entscheiden, wobei in beiden Fällen die MATLAB-Control-System-Toolbox™ für den Reglerentwurf zum Einsatz kommt. Zielsystem ist ein Echtzeit-Linux-Computer als elektronisches Steuergerät, der die Fahrzeuge über Bluetooth-Low-Energy (BLE) fernsteuert. Die Softwarefunktionen werden als Software-Komponenten in der Middleware Robot-Operating-System (ROS) [7] implementiert. Die *modellgetriebene Softwareentwicklung* ist eine Variante der modellbasierten Softwareentwicklung, wobei bei Verwendung von MATLAB/Simulink der C++-Code der Software-Komponenten automatisiert mit Hilfe der MATLAB/Simulink-Toolbox Embedded-Coder® generiert wird.

Beide Ansätze führen zu demselben Ergebnis, unterscheiden sich aber in der Vorgehensweise und im Abstraktionsgrad. Während in MATLAB/Simulink Signalflusspläne für Funktions-, Software- und Umgebungsmodelle direkt eingegeben und simuliert werden können, ist bei Verwendung von C++ eine Abbildung der Signalflusspläne in Datenobjekte und Programmflüsse notwendig. Deshalb und wegen der notwendigen Einbindung der Numerik-Bibliothek Boost-Odeint[2] [8] ist der Arbeitsaufwand in der modellbasierten Softwareentwicklung mit C++ als höher einzustufen als bei der modellgetriebenen Softwareentwicklung mit MATLAB/Simulink.

1.1.1 Lernziele

- Die Leser*innen kennen die Softwarearchitektur für automatisierte Fahrfunktionen.
- Die Leser*innen können die modellbasierte und modellgetriebene Softwareentwicklung anwenden.
- Die Leser*innen können automatisierte Fahrfunktionen in MATLAB/Simulink oder C++ modellieren bzw. programmieren.
- Die Leser*innen können die Fahrdynamik und die Dynamik elektrischer Antriebssysteme der MAD-Fahrzeuge entweder in MATLAB/Simulink oder in C++ modellieren und simulieren.
- Die Leser*innen können Sollbahnkurven mit Hilfe von Streckenabschnitten und kubischen Splines erstellen.
- Die Leser*innen können Geschwindigkeits-, Positions- und Bahnfolgeregler entwerfen, modellieren und testen.
- Die Leser*innen können Softwarefunktionen für autonomes Fahren in MATLAB/Simulink oder C++ modellieren bzw. programmieren und in Betrieb nehmen.

2 Boost ist eine weit verbreitete C++-Bibliothek, die eine große Zahl an Routinen für I/O, Prozessmanagement, Datenstrukturen, Numerik usw. enthält.

- Die Leser*innen können Simulationsmodelle, Geschwindigkeits-, Positions- und Bahnfolgeregler in verschiedenen Fahrszenarien, wie z.B. für Parksysteme, Kreuzungen, Kreisverkehre und Rennstrecken, anwenden.
- Die Leser*innen können die erforderlichen Ergebnisse in Teamarbeit erzielen.

1.1.2 Softwareentwicklungsumgebungen

Im Laborprojekt MAD wird ROS (Robot-Operating-System) unter Linux als Middleware und Bedienumgebung zur Entwicklung und Ausführung der automatisierten Fahrfunktionen eingesetzt. Die ROS-Softwarekomponenten für die Fahrdynamiksimulation und automatisierten Fahrfunktionen werden entweder in MATLAB/Simulink modelliert oder in C++ programmiert. Für den Reglerentwurf wird in beiden Fällen die MATLAB-Control-System-Toolbox angewendet.

Im Fall von MATLAB/Simulink generiert die MATLAB/Simulink-Toolbox Embedded-Coder den C/C++-Code für ROS. Im Fall von C++ werden die Werkzeugkette GNU C/C++ sowie die integrierte Entwicklungsumgebung (engl. Integrated-Development-Environment) QT-Creator angewendet.

1.1.3 Vorausgesetzte Kenntnisse

Dieses Lehrbuch setzt folgende Kenntnisse voraus, die in Lehrveranstaltungen bis zum 4. Semester eines ingenieur- oder naturwissenschaftlichen Bachelorstudiengangs erlangt werden:

- Signale und Systeme: Beschreibung und Analyse von linearen, dynamischen Übertragungsgliedern im Zeit- und im Frequenzbereich,
- Simulationstechnik: Erstellung und Simulation dynamischer Zustandsraummodelle,
- Regelungstechnik: Frequenzkennlinien-basierter Entwurf von PID-Reglern und zeitdiskreten Reglern,
- Anwendung von MATLAB/Simulink einschließlich der Control-System-Toolbox,
- Programmiererfahrung in der objektorientierten Programmiersprache C++,
- selbständige und vernetzte Lösung komplexer Problemstellungen in der Regelungstechnik.

Zur Einarbeitung in MATLAB/Simulink steht auf der Webseite von MathWorks [3] Schulungsmaterial zur Verfügung. Online-Kurse für MATLAB/Simulink sind unter `https://matlabacademy.mathworks.com/` zu finden.

1.2 Automatisiertes und autonomes Fahren

Abb. 2 zeigt einen signifikanten Rückgang an tödlichen Unfällen auf deutschen Straßen im Zeitraum von 1990 bis 2010. Dieser großartige Rückgang wurde durch die Einführung von passiven und aktiven Sicherheitssystemen und Verbesserungen in der Infrastruktur erreicht. Jedoch stagniert die Zahl an tödlichen Unfällen seit 2010. Nur durch eine konsequente Einführung von *Advanced-Driver-Assistance-Systems* (ADAS) und des *automatisierten Fahrens* kann die Zahl an Verkehrstoten zukünftig weiter reduziert werden. Aktuell verfügbare ADAS, beispielsweise Abstandsregel- und Spurhaltesysteme,

- unterstützen den Fahrer in den primären Fahrfunktionen,
- reduzieren den Stress für den Fahrer,
- informieren den Fahrer über das Fahrzeugumfeld,
- warnen den Fahrer vor Gefahren,
- verbessern Komfort und Sicherheit.

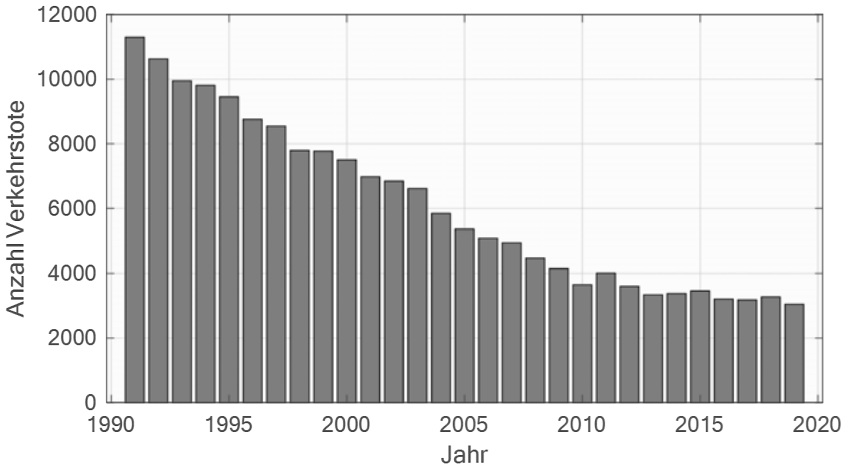

Abb. 2: Anzahl an Verkehrstoten im deutschen Straßenverkehr [9]

Die sogenannte *Vision Zero* sagt eine unfallfreie Zukunft voraus [10]. Die Ziele des *automatisierten Fahrens* sind:

- Reduktion der Unfallhäufigkeit,
- Reduktion der Unfallschwere.

Hoch- und vollautomatisiert fahrende Robotertaxis auf Sonderfahrspuren sind heute bereits im Betrieb. Fahrzeughersteller bieten Funktionen für das *teilautomatisierte*

Fahren auf Autobahnen an. Softwarefunktionen im teil-, hoch- und vollautomatisierten Fahren übernehmen zeitweise oder vollständig die Fahrfunktion und führen das Fahrzeug autonom. Die wesentlichen Merkmale von Systemen für *autonomes Fahren* sind:

- Das Fahrzeug übernimmt selbst die Fahrzeugführung einschließlich der Planung und der Entscheidungsfindung.
- Im Personentransport ist der Fahrer nicht länger der Fahrer sondern ein Passagier.
- Im Gütertransport transportiert das Fahrzeug Güter ohne Fahrer oder Passagiere.
- Das autonome Fahren reduziert den Energieverbrauch und Abgase durch optimale Planung und Regelung der Fahrdynamik und des Antriebsstrangs.
- Das autonome Fahren erleichtert und ermöglicht das Fahren für ältere und kranke Menschen.
- Das autonome Fahren führt zu einem Ausbau alternativer Betreibermodelle wie z.B. Car-Sharing und -Leasing.
- Das autonome Fahren erhöht die Betriebszeiten von Fahrzeugen um mehrere Größenordnungen.

Die amerikanische Society-of-Automotive-Engineers (SAE International) veröffentlichte die *SAE-Automation-Levels* zur Kategorisierung von Systemen des automatisierten Fahrens [11]. In den SAE-Levels 3 bis 5 übernimmt das automatisierte Fahrsystem alle Fahrfunktionen in einigen oder sogar in allen Fahrsituationen.

Teilautomatisierung (Level 3) implementiert eine „fußfreie“ (engl. „feet-free“) und gleichzeitig „handfreie“ (engl. „hands-free“) Bedienung des Fahrzeugs, wobei der Fahrer auf Aufforderung durch das System hin oder bei Gefahren die Fahrzeugkontrolle jederzeit wieder übernehmen muss.

Hochautomatisierung und *Vollautomatisierung* (Levels 4 und 5) implementieren darüber hinaus eine „gehirnfreie“ (engl. „brains-free“) Bedienung. D.h. der Fahrer kann während der Fahrt als Passagier schlafen oder sogar das Fahrzeug verlassen. Das Fahrzeug kann auch ohne Passagiere einen Gütertransport oder ein Automated-Valet-Parking durchführen. Es beherrscht selbständig sicherheitskritische Situationen ohne Fahrereingriff, d.h. *autonom*. Während Vollautomatisierung in Level 5 überall verfügbar ist, ist sind Operation-Design-Domains (ODD) der Hochautomatisierung in Level 4 auf abgegrenzte Szenarien oder Situationen, z.B. automatisiertes Parken oder Autobahnfahrten, begrenzt.

1.3 Kapitelübersicht

Jedes der in Abb. 3 dargestellten Kapitel vermittelt zunächst den theoretischen und methodischen Inhalt für die modellbasierte Softwareentwicklung der Regelungsfunktionen. Ab Kapitel 5 enthalten alle Kapitel am Ende Laboraufgaben als Teil des

buchbegleitenden Laborprojekts Mini-Auto-Drive (MAD). Nach erfolgreicher Lösung dieser Laboraufgaben und entsprechender modellbasierter Entwicklung der Regelungsfunktionen fährt das MAD-Fahrzeug autonom auf einer konfigurierbaren Fahrbahnkarte (siehe auch Fotos und Videos unter `https://asert.hs-heilbronn.de/`).

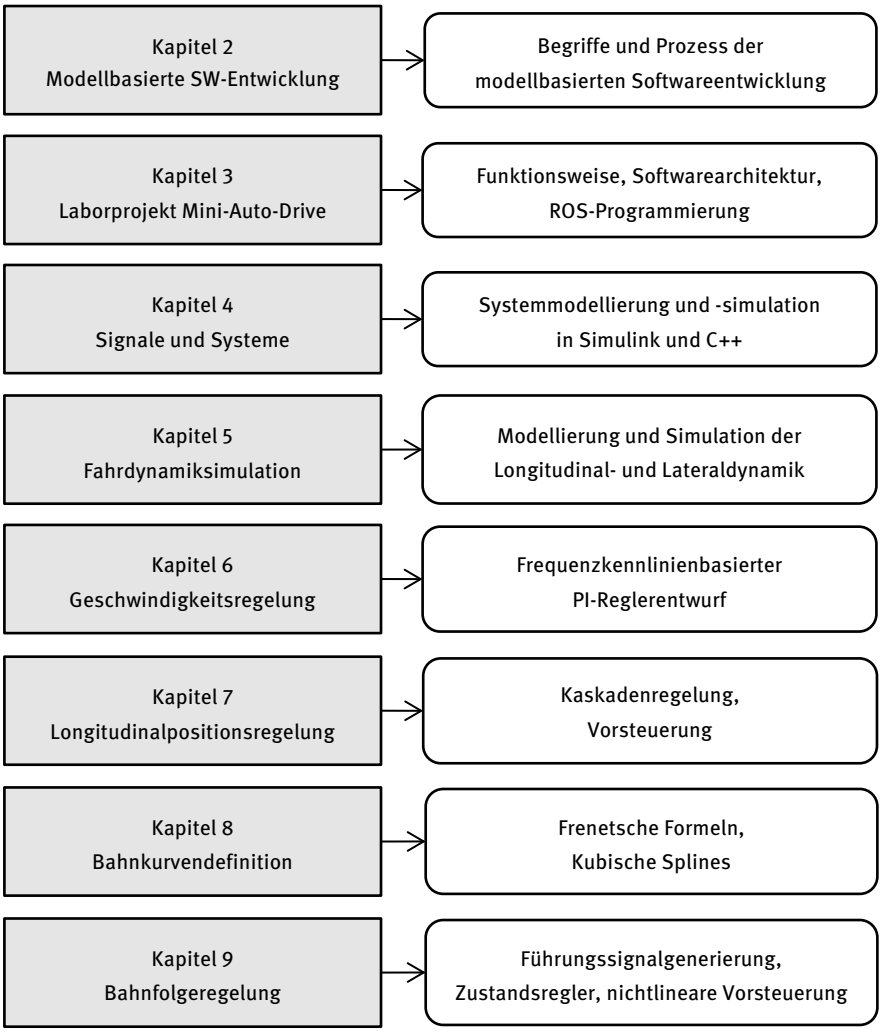

Abb. 3: Gliederung der Lehr- und Laborinhalte

Kapitel 2 definiert die technischen Begriffe der modellbasierten Entwicklung und stellt den Prozess der modellbasierten Softwareentwicklung vor. Anhand von Beispielen werden in den beiden einführenden Kapiteln 3 und 4 die Programmierung von ROS-Software-Komponenten sowie die Modellierung und Simulation dynamischer, zeitkontinuierlicher und zeitdiskreter Systeme grundlegend behandelt. Darüber hinaus beschreibt Kapitel 3 die Funktionsweise und Softwarearchitektur von MAD und definiert den Inhalt des Laborprojekts.

Im Rahmen des Kapitels 5 erfolgt zunächst die Modellierung und Simulation der Fahrdynamik, so dass in den nachfolgenden Kapiteln die Regelungsfunktionen modellbasiert entwickelt und simulativ getestet werden können. Kapitel 6 behandelt zunächst die Geschwindigkeitsreglung des Fahrzeugs. Darauf aufbauend wird in Kapitel 7 die Longitudinalpositionsregelung entwickelt, die beim Parken oder beim Halten an einer Kreuzung zur Anwendung kommt. Während in Kapitel 6 und 7 ausschließlich das Motorsignal für die Beschleunigung und Verzögerung des Fahrzeugs gestellt wird, wird in Kapitel 8 und 9 die Bahnfolgeregelung entwickelt, die den Lenkwinkel stellt.

2 Modellbasierte Softwareentwicklung

Innovationen in mechatronischen Systemen und Fahrzeugsystemen werden hauptsächlich realisiert durch softwareintensive Steuerungs- und Regelungssysteme. Die modellbasierte Softwareentwicklung ist eine sehr geeignete Methode in der Realisierung aktueller und zukünftiger technischer Innovationen. Nur durch Einsatz der modellbasierten Entwicklungsmethode und von Modellierungs- und Simulationsumgebungen ist die Realisierung hoch komplexer mechatronischer Systeme und Fahrzeugsysteme bei steigender Variantenvielfalt und immer kürzer werdenden Entwicklungszyklen möglich. Damit stellt die modellbasierte Softwareentwicklung einen wichtigen Wegbereiter der Innovationen in Automotive-Systems-Engineering der letzten 20 Jahre und in der Zukunft dar.

Modellbasierte Softwareentwicklung bedeutet allgemein, dass bei der Entwicklung der Embedded-Software elektronischer Steuergeräte Modelle für Systeme, Systemkomponenten und Software eingesetzt werden. Der Begriff *modellbasierte Softwareentwicklung* ist dabei abzugrenzen vom Begriff der *modellgetriebenen Softwareentwicklung*, die ein Teilgebiet der modellbasierten Softwareentwicklung darstellt. In der modellgetriebenen Softwareentwicklung generieren Codegeneratoren automatisiert Embedded-Software aus formalen Modellen. Die englischen Begriffe für modellbasierte und modellgetriebene Softwareentwicklung sind *Model-Based Software-Engineering (MBE)* bzw. *Model-Driven Software-Engineering (MDE)*.

In der MBE werden *Modelle* eingesetzt zur Beschreibung, Simulation, und Analyse von

- Steuergerätesoftware,
- Steuergerätehardware,
- Computernetzwerken,
- Regelstrecken (mechatronische Grundsysteme mit Systemkomponenten),
- Sensoren und Aktuatoren,
- Bediener bzw. Fahrer,
- Systemumfeld.

In der MBE werden Entwicklungsumfänge in frühe Phasen der Systementwicklung verlagert, in welchen die zu entwickelnde Software und das System simuliert werden. Durch diese Simulationen und zusätzliche Reviews werden Fehler in den Anforderungen oder in der Software frühzeitig noch vor der Integration ins reale Zielsystem erkannt. Dies spart Kosten und Zeit in der Software- und Systementwicklung.

Die folgenden Abschnitte behandeln die Prozesse der modellbasierten und modellgetriebenen Entwicklung von Embedded-Software für mechatronische Systeme und Fahrzeugsysteme. Insbesondere werden die Begriffe *Funktionsmodell*, *Softwaremodell* und *Umgebungsmodell* definiert. Des Weiteren wird gezeigt, in welchen

https://doi.org/10.1515/9783110723526-002

Prozessschritten Modelle der Embedded-Software und die Embedded-Software selbst validiert und verifiziert werden.

2.1 Prozess der modellbasierten Softwareentwicklung

Die modellbasierte Softwareentwicklung (MBE) erfolgt üblicherweise im Rahmen des *V-Modells*. In diesem Entwicklungsprozessmodell wird gemäß Abb. 4 zwischen der Gesamtsystem- bzw. Fahrzeugebene und der Steuergeräteebene unterschieden. Auch in der agilen Softwareentwicklung kann das V-Modell für einzelne kurze Entwicklungszyklen einsetzt werden. Dabei wird das V-Modell mehrfach durchlaufen.

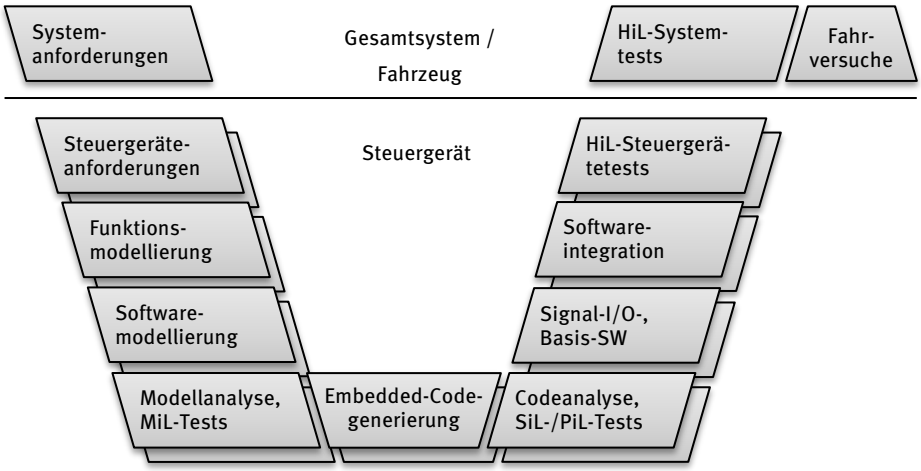

Abb. 4: V-Modell der modellbasierten Softwareentwicklung

Auf der *Gesamtsystem-* bzw. *Fahrzeugebene* werden die Anforderungen an das Gesamtsystem bzw. Fahrzeug spezifiziert und die Systemarchitektur festgelegt. Auf der rechten Seite des V-Modells erfolgen auf dieser Ebene die Integration des Gesamtsystems bzw. Fahrzeugs aus Komponenten sowie die Vernetzung elektronischer Steuergeräte. Weiterhin werden die Steuergeräte auf Systemebene getestet. Softwareparameter der Steuergeräte werden in Gesamtsystem- bzw. Fahrversuchen appliziert.

Auf der *Steuergeräteebene* erfolgt die Entwicklung der elektronischen Steuergeräte in einzelnen parallellaufenden Teilprojekten. Die Steuergeräteentwicklung umfasst die Hardware- und die Softwareentwicklung. Die Anforderungen an das jeweilige Steuergerät werden aus den Systemanforderungen konsistent hergeleitet.

Die MBE unterscheidet zwischen *Funktions-*, *Software-* und *Umgebungsmodellen*. *Funktionsmodelle* beschreiben die Applikationssoftware und sind unabhängig von der Microcontroller- oder Microprozessor-Plattform. Funktionsmodelle bestehen im Allgemeinen aus zeitkontinuierlichen Differentialgleichungen, zeitdiskreten Differenzengleichungen und endlichen Zustandsautomaten. *Softwaremodelle* werden dagegen auf die eingesetzte Microcontroller- oder Microprozessorplattform zur optimalen Nutzung der zur Verfügung stehenden Ressourcen angepasst. Daher enthalten Softwaremodelle nur Differenzengleichungen und keine Differentialgleichungen, so dass in der generierten Embedded-Software auf die ressourcenintensive Einbindung von Numerik-Bibliotheken zur Lösung von Differentialgleichungen verzichtet werden kann. *Umgebungsmodelle* beschreiben die Umgebung des elektronischen Steuergeräts einschließlich Sensoren, Aktuatoren, Fahrer bzw. Bediener und Systemumfeld. Umgebungsmodelle können dabei auch Modelle für andere elektronische Steuergeräte oder Computernetzwerke enthalten. Eine Computersimulation der Umgebungsmodelle wird daher auch als *Restbussimulation* bezeichnet. Wichtig dabei ist, dass Umgebungsmodelle die Umgebung des elektronischen Steuergeräts so genau und umfassend abbilden, dass eine modellbasierte Entwicklung der Applikationssoftware möglich ist.

In den Quality-Gates „Modellanalyse, MiL-Tests“, „Codeanalyse, SiL-/PiL-Modultests“, „HiL-Steuergerätetests“ und „HiL-Systemtests“ werden mit Hilfe von Computersimulationen die Embedded-Software und deren Funktions- und Softwaremodelle validiert und verifiziert. Diese Quality-Gates sind verpflichtend nach dem Stand der Technik und werden in Abschnitt 2.6 näher behandelt.

Im Automobilbereich erstreckt sich der gesamte Entwicklungszeitraum für ein Fahrzeug über mehrere Jahre. Dabei werden mehrere V-Zyklen iterativ durchlaufen. Durch die Verkettung der V-Zyklen entstehen *VA-Zyklen*. Im V-Teil eines VA-Zyklus erfolgt die Entwicklung der Steuergeräte inklusive der Embedded-Software. Im A-Teil erfolgen die System- oder Fahrversuche, die Applikation sowie die iterative Anpassung und Erweiterung der Systemanforderungen.

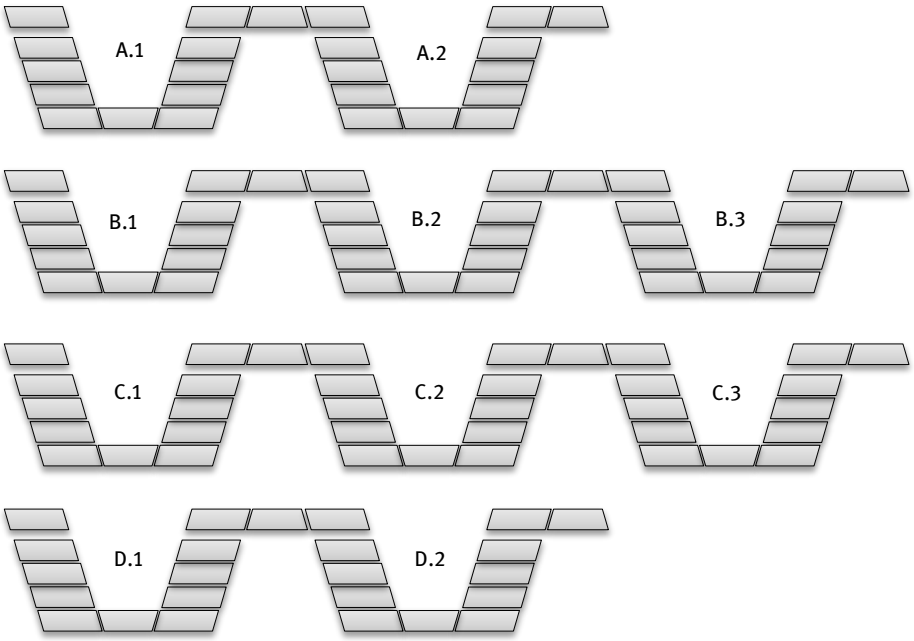

Abb. 5: VA-Zyklen der Fahrzeugentwicklung

Jeder VA-Zyklus liefert als Ergebnis einen freigegebenen Hardware- und Softwarestand der Steuergeräte. Folgende Bezeichnungen sind allgemein gebräuchlich:

- *A-Muster-Stände*: Ein A-Muster-Steuergerät verfügt über einen eingeschränkten Funktionsumfang und ist meist nicht vollständig funktionsfähig. A-Muster-Steuergeräte werden vor allen Dingen in der Vorausentwicklung und im *Rapid-Control-Prototyping* (RCP) eingesetzt. Gebräuchlich sind hierbei RCP-Steuergeräte, die über eine hohe Rechenleistung verfügen, eine Autocodegenerierung direkt für Funktionsmodelle und deren Ausführung in Echtzeit ermöglichen ohne vorherige Softwaremodellierung oder manuelle Entwicklung der Embedded-Software. RCP-Steuergeräte sind als Produkte von verschiedenen Anbietern erhältlich. Sie sind flexibel für verschiedene Anwendungen und Schnittstellen konfigurierbar.
- *B-Muster-Stände*: Ein B-Muster-Steuergerät ist ein seriennahes Steuergerät, das speziell für das Gesamtsystem bzw. das Fahrzeug entwickelt wird. Ein B-Muster-Steuergerät implementiert den vollen Funktionsumfang, wobei dieser aber nicht vollständig abgesichert ist.
- *C-Muster-Stände*: C-Muster-Stände sind Weiterentwicklungen der B-Muster-Steuergeräte. C-Muster-Stände werden in Kleinserien für Flottentests vor der Serienfreigabe hergestellt. Bei C-Mustern wird der volle Funktionsumfang abgesichert.

– *D-Muster-Stände*: D-Muster-Steuergeräte sind Seriensteuergeräte, die in hohen Stückzahlen produziert und eingesetzt werden. Bei D-Muster-Steuergeräten sind in der Regel ausschließlich eine Fehleranalyse und –behebung oder geplante Funktionserweiterungen in definierten Zyklen erlaubt.

2.2 Modellgetriebene Softwareentwicklung

Die *modellgetriebene Softwareentwicklung (MDE)* stellt eine Teildisziplin der *modellbasierten Softwareentwicklung (MBE)* dar. Codegeneratoren generieren automatisiert Embedded-Software aus Softwaremodellen. In der Mechatronik und im Automobilbereich werden hauptsächlich Embedded-C oder C++ als Zielsprachen für die generierte Embedded-Software verwendet.

Dagegen setzt die allgemeinere modellbasierten Softwareentwicklung die automatisierte Codegenerierung nicht auf alle Fälle ein. Softwaremodelle werden hier häufig als Vorlagen für eine manuelle Embedded-C/C++-Entwicklung sowie zur Systemanalyse, zum Systementwurf oder zum Testen des Embedded-Codes verwendet.

Beispielsweise erstellen Automobilhersteller Modelle, die sie an Steuergeräte-Zulieferer als Teil von Lastenheften übergeben. Die Zulieferer entwickeln dann Embedded-Code für Steuergeräte entweder mit herkömmlichen Methoden oder mit Hilfe der modellgetriebenen Softwareentwicklung.

In der modellgetriebenen Softwareentwicklung für mechatronische Systeme oder Fahrzeugsysteme werden vor allen Dingen die folgenden Modellierungs- und Simulationswerkzeuge für die Modellierung und die Simulation der Funktions- und Softwaremodelle eingesetzt:

– MATLAB/Simulink/Stateflow,
– ASCET [12].

Simulink und ASCET können AUTOSAR-Beschreibungsdateien [13] für Steuergeräte-Schnittstellen und Software-Komponenten importieren. Dies erleichtert wesentlich die Implementierung der Steuergeräte-Schnittstellen, die mehrere 1000 Signale übertragen.

In der Automobilentwicklung kommen folgende Codegeneratoren zum Einsatz, die aus Softwaremodellen in MATLAB/Simulink/Stateflow oder ASCET Embedded-C oder -C++-Code automatisiert generieren:

– Simulink-/Embedded-Coder für MATLAB-/Simulink-/Stateflow-Modelle,
– Targetlink [14] für MATLAB-/Simulink-/Stateflow-Modelle,
– ASCET-Codegenerator für ASCET-Modelle

Simulink/Stateflow und ASCET sind die „Programmiersprachen“ der Ingenieure. Durch die MBE können Ingenieure komplexe und qualitativ hochwertige Embedded-