1 Einleitung

Derzeit findet in Gesellschaft und Wirtschaft aufgrund neuer digitaler Technologien ein Transformationsprozess statt. Im Zuge dieser fortschreitenden Digitalisierung und zur Optimierung wesentlicher Systemeigenschaften erfolgt zunehmend die Integration intelligenter Hard- und Software in technische Systeme [32], welche als Innovationstreiber fungieren. Zusammen mit Sensorik und Aktorik entsteht aus der eingesetzten Informationsverarbeitung ein typisches eingebettetes mechatronisches System [143]. Den Kern dieser zunehmend intelligenten Systeme bilden Steuergeräte (Embedded Control Units – ECU) und die darauf implementierten hochwertigen Reglerfunktionen einschließlich der Signalverarbeitung [5]. Nur durch den Einsatz solcher eingebetteten mechatronischen Systeme können die aktuellen und stetig steigenden Anforderungen an Funktionalität, Sicherheit und Vernetzungsgrad bei immer kürzeren Entwicklungszeiten bewältigt werden [199].

Neben dem technologischen Fortschritt liegt ein weiterer gesellschaftlicher Schwerpunkt im Wandel zur ökologischen Nachhaltigkeit. Der Einsatz digitaler Schlüsseltechnologien ist ein wesentlicher Baustein zur Erreichung dieses Ziels [187]. In diesem Zusammenhang kann eine beschleunigte Digitalisierung einen unmittelbaren Beitrag zur Erreichung der Klimaziele für das Jahr 2030 leisten [19]. Besonders der Einsatz von Vernetzung und künstlicher Intelligenz (KI) ermöglicht neue Funktionalitäten [199]. Der stetig steigende Funktionsumfang und Vernetzungsgrad führen jedoch auch zu hochgradig komplexer Software für Steuerungs- und Regelsysteme sowie Software-intensiven cyber-physischen Systemen (CPS) [143]. Zukünftig werden ECU daher eine Schlüsselstellung bei der Generierung kundenwahrnehmbarer Funktionen einnehmen. Die Entwicklungssystematik, Validierung und Verifikation sowie digitale Durchgängigkeit sind dabei wesentliche Einflussfaktoren auf die Zukunft der Entwicklung [49]. Aus diesen Faktoren ergibt sich die Motivation, Problemstellung und der weitere Aufbau der vorliegenden Arbeit.

1.1 Motivation und Problemstellung

Der Einsatz eingebetteter mechatronischer Systeme erstreckt sich heute auf eine Vielzahl von Anwendungsbereichen. Dazu zählen Fahrzeugkomponenten wie Batteriemanagementsysteme (BMS), welche die Sicherheit der Batterien kontinuierlich überwachen, deren Leistungsfähigkeit abschätzen und prognostizieren, sowie autonome und vernetzte CPS [199]. Obgleich die Komplexität kontinuierlich zunimmt, sind Hersteller gefordert, Produkte in kürzerer Zeit und zu niedrigen Kosten zur Serienreife zu bringen, um im harten Wettbewerb zu bestehen.

Folglich beinhaltet der erste Entwurf der Soft- und Hardware häufig Fehler [261]. Werden diese Fehler nicht durch aufwendiges Testen eliminiert, führt dies zu folgenreichen Fehlfunktionen. So haben durchschnittlich 20 % der Rückrufe von Kraftfahrzeugen in den Vereinigten Staaten von Amerika ihre Ursache in der Software[1] – Tendenz steigend.

Da mechatronische Komponenten unmittelbar mit Aspekten wie der Zuverlässigkeit und Sicherheit eines Systems assoziiert werden, ist eine effektive Softwareentwicklung unabdingbar [211]. Die Automatisierung sowie die physikbasierte Simulation eines technischen Systems bilden dabei die essenziellen Grundvoraussetzungen für eine solche Entwicklung [67]. Zur Unterstützung werden diverse Computer Aided Engineering (CAE) basierte Entwicklungswerkzeuge eingesetzt. Angestrebt wird es, Fehler frühestmöglich auf ein Minimum zu reduzieren [160]. Besonders fehleranfällig und zeitaufwändig sind manuelle Tätigkeiten [193]. Die Entwicklung eines Systems kann daher durch den Einsatz einer durchgängigen, hoch automatisierten Entwicklungsplattform signifikant optimiert werden. Ganzheitliche, modellbasierte Ansätze wie das Rapid Control Prototyping (RCP) weisen daher ein vielversprechendes Potenzial auf [130].

Derzeit kann diese RCP basierte Funktionsentwicklung ausschließlich durch sehr kostspielige High-Cost CAE-Werkzeuge – beispielsweise mit Matlab / Simulink[2] und dSPACE[3] – realisiert werden, wie in Abbildung 1.1 zusammengefasst. Durch umfangreiche, kostenintensive Modellbibliotheken wird hier die Modellbildung beliebiger technischer und cyber-physischer Systeme mit unterschiedlichsten Modellierungstiefen unterstützt. Kostenpflichtige Erweiterungen für Simulink bieten Analyse- und Synthesemöglichkeiten für jegliche Art von Systemen und ermöglichen so Model-in-the-Loop (MiL)-Simulationen. Anschließend kann über den Simulink Coder automatisiert Programmcode für verschiedene Zielsprachen generiert und durch Software-in-the-Loop (SiL)-Simulationen getestet werden. Über das dSPACE-RTI[4] erfolgt eine automatisierte Echtzeitrealisierung auf verschiedenen leistungsstarken Zielhardwaresystemen – wie beispielsweise dem modularen, flexibel erweiterbaren dSPACE Scalexio. Das Mensch-Maschine-Interface (MMI) dSPACE ControlDesk ermöglicht umfangreiche, flexibel konfigurierbare Onlineexperimente für Hardware-in-the-Loop (HiL)-Simulationen. Eine ausführliche Darstellung der Features[5] erfolgt in Abschnitt 2.2.

---

[1] Die Erhebung erfolgte in Anlehnung an [40] auf Basis von Daten der National Highway Traffic Safety Administration Rückrufdatenbank [150] für die Jahre 2018 bis 2023.

[2] Simulink ist eine Erweiterung der Numerik-Software Matlab und dient der Modellierung unter anderem technischer und physikalischer Systeme [9]. Bei der Software handelt es sich um ein Produkt der Firma The MathWorks, Inc.

[3] Produkte der Firma dSPACE GmbH mit Hauptsitz in Paderborn.

[4] RTI steht für Real-Time-Interface.

[5] Ein Feature ist ein ausgeprägter, für Benutzende sichtbarer, Aspekt eines Systems [209].

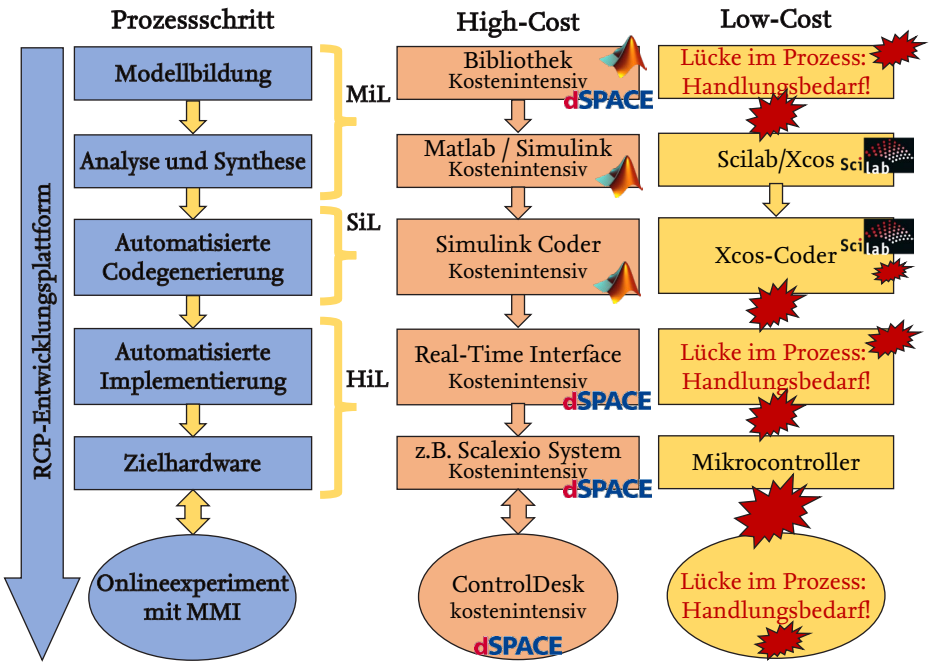

Abbildung 1.1: RCP-Prozess mit durchgängiger CAE-Entwicklungsplattform und Lücken des Low-Cost-Ansatzes in Anlehnung an [253].

Die Anschaffungskosten für verfügbare durchgängige Werkzeugketten sind erheblich und stellen insbesondere für Akteure mit begrenztem Budget, wie Forschungseinrichtungen oder kleine und mittelständische Unternehmen (KMU), eine große Herausforderung dar [251]. Darüber hinaus sind diese Werkzeugketten oft umfangreich und in der Bedienung komplex. Bei einem begrenzten Einsatz der Entwicklungsplattform, etwa für Systeme mit wenigen Zuständen und geringem Funktionsumfang, ist das Verhältnis von Kosten zu genutzter Leistung häufig unausgewogen.

Hinderungsgründe bei der Einführung einer modellbasierten Funktionsentwicklung sind vor allem finanzielle Beschränkungen, die Anzahl und Qualifikation der Mitarbeiter, die Organisationsstruktur sowie das Beharren auf etablierte Entwicklungsprozesse [163]. In einer Umfrage gaben zudem 26 % der Befragten an, dass das Fehlen passgenauer Entwicklungswerkzeuge ein weiteres Hindernis bei der Einführung modellbasierter Methoden darstellt [89]. Vorhandene Werkzeuge sind entweder zu komplex in der Anwendung oder zu kostenintensiv in der Anschaffung. Auch die Auswahl geeigneter Werkzeuge stellt aufgrund fehlender Zeit und Erfahrung eine erhebliche Herausforderung dar. Es besteht daher ein dringender Bedarf an einer kostengünstigen, durchgängigen und hoch automatisierten Entwicklungsplattform. Zur Steigerung der Akzeptanz und Übersichtlichkeit muss zudem die RCP-Systematik nahtlos in eine vertraute Umgebung integriert sein.

Eine solche Plattform nach dem Low-Cost-Ansatz mit Open-Source-Software[6] wie beispielsweise Scilab / Xcos und einem Mikrocontroller als Zielhardware für Echtzeitsimulationen ist aktuell mit vielen Lücken behaftet [257]. Das Prinzip ist in Abbildung 1.1 als Low-Cost-Ansatz dargestellt. Aufgrund der vorhandenen Lücken sind zahlreiche manuelle Arbeiten wie händische Programmierung notwendig. Dies führt zu hohem Zeitaufwand und zufälligen Fehlerquellen. Die Reproduzierbarkeit möglicher Fehler ist dabei nicht gegeben, was deren Ermittlung und Beseitigung erschwert. Hierdurch sind viele komplexe Tests der fertigen Software notwendig und es entstehen langwierige Iterationszyklen. Daher können solche kostengünstigen Kombinationen mit den vorhandenen Lücken aktuell nicht durchgängig im Entwicklungsprozess eingesetzt werden.

In dieser Arbeit wird ein Ansatz für die durchgängige Funktionsentwicklung mit Open-Source-Software, genannt *Low-Cost Rapid Control Prototyping Entwicklungsplattform (LoRra)*, entworfen. LoRra soll dabei die RCP-Systematik nahtlos in die Plattform integrieren und so eine hoch automatisierte, kostengünstige Funktionsentwicklung ermöglichen.

Im Rahmen des EU-geförderten Forschungsprojektes *Low-Cost Rapid Control Prototyping-System mit Open-Source-Plattform für die Funktionsentwicklung von eingebetteten mechatronischen Systemen* wurde ein Lösungsansatz entwickelt, der Scilab / Xcos und einen Mikrocontroller als Echtzeithardware verwendet. Dieser soll es zukünftig allen Anwendern ermöglichen, mechatronische Funktionen auf verschiedenen Systemebenen zielgerichtet, wirtschaftlich und methodisch zu entwickeln und zu erforschen. Auch ein Einsatz in der Lehre ist denkbar. Die Ergebnisse werden im Niedersächsischen Zukunftslabor Mobilität (vgl. [114]) weitergehend optimiert und verifiziert. Sie bilden die Grundlage für diese Arbeit.

1.2 Ziele und Aufbau der Arbeit

Im weiteren Verlauf erfolgt eine Diskussion der in dieser Arbeit behandelten Problemstellungen und Ziele. Zudem wird der prinzipielle Weg zur Lösung dieser Problemstellungen erarbeitet und erläutert. Des Weiteren findet eine kurze Abgrenzung des Kontextes statt. Die Betrachtung des Themas erfolgt aus der Perspektive von Personen, welche für die Entwicklung von mechatronischen Funktionen zuständig sind. In der Regel handelt es sich dabei um Ingenieure mit vertieften Kenntnissen in den Bereichen Regelungstechnik und Systemtheorie. Hinsichtlich der konkreten Umsetzung einer entworfenen Funktion auf einem Steuergerät besteht lediglich ein begrenztes Maß an Fachwissen. Dies betrifft insbesondere hardwarenahe Ansteuerungen sowie Wechselwirkungen mit anderen implementierten Funktionen.

[6] Open-Source-Software bezeichnet in diesem Kontext Software, deren Lizenzbedingungen eine kostenfreie Verwendung sowie Zugriff auf den Quelltext erlauben [157].

Aufbauend auf dieser Perspektive widmen sich die folgenden Diskussionen den Werkzeugen zur Entwicklung von Softwarefunktionen für eingebettete mechatronische Systeme, insbesondere solchen mit regelungstechnischen Aufgaben[7]. Hierbei tritt die Bezeichnung Funktion kontextabhängig sowohl als *Baustein zur Abarbeitung einer bestimmten Aufgabe in einem eingebetteten mechatronischen System* oder als *Fähigkeit einer Entwicklungsplattform* auf. Sofern nicht abweichend angegeben, wird die Formulierung *Funktion* für den mit der Plattform zu entwickelnden Systembestandteil verwendet. Die Fähigkeit der Plattform wird als Feature (siehe Fußnote 5) bezeichnet. Auch der Bezeichner *Methode* tritt sowohl im Rahmen der Funktionsentwicklung als auch bei der Plattformentwicklung auf. Die entsprechende Verwendung ergibt sich jeweils aus dem Kontext. Der Begriff Entwicklungsplattform wird in Anlehnung an Mutijarsa et al. [149] wie folgt definiert: *Eine Entwicklungsplattform besteht aus verschiedenen Soft- und Hardwarekomponenten*[8]*. Sie basiert auf kohärenten Entwurfsphilosophien und integriert mehrere einzelne Entwicklungswerkzeuge zu einer höherwertigen, emergenten Funktionalität. Durch Bereitstellung wiederverwendbarer Datenquellen, Werkzeuge und Prozesse werden Nutzende bei der Entwicklung unterstützt.*

Aufbauend auf diesem Verständnis liegt der Fokus des anwendungsorientierten Forschungsansatzes darauf, mit einer kostengünstigen Entwicklungsplattform systematisch und durchgängig Funktionen nach dem modellbasierten RCP-Prozess (vgl. Abschnitt 2.1.2) zu entwerfen und prototypisch zu realisieren. Hieraus ergeben sich folgende Arbeitshypothesen:

1. Der modellbasierte RCP-Prozess lässt sich mittels kostengünstiger Entwicklungsplattform durchgängig realisieren.
   - Sämtliche notwendigen Prozessschritte werden unterstützt.
   - Der Übergang zwischen den Prozessschritten erfolgt hoch automatisiert ohne manuelle Tätigkeiten.
   - Die Ergebnisse sind konsistent und reproduzierbar.
2. Die Entwicklungsplattform ist nicht auf vorgegebene Anwendungen und Domänen beschränkt.

Um diese Hypothesen zu belegen, erfolgt in dieser Arbeit die Konzeption und exemplarische Realisierung der LoRra-Entwicklungsplattform mit Scilab / Xcos als Simulationswerkzeug. Abbildung 1.2 illustriert den forcierten weiteren Aufbau der Arbeit anhand des RCP-Prozesses. Mittels systematischer Untersuchungen sowie anhand einer Pilotanwendung wird nachgewiesen, dass die oben genannten Hypothesen erfüllt werden können.

[7] Funktionsentwicklung für überwiegend kontinuierliche Systeme mit dem Ziel, das Systemverhalten durch Anwendung von regelungstechnischen Methoden gezielt zu beeinflussen.

[8] Software sind sämtliche Computerprogramme, Prozeduren und Daten, welche den Betrieb eines Computersystems betreffen. Hardware bezeichnet die physischen Komponenten eines Computersystems [100].

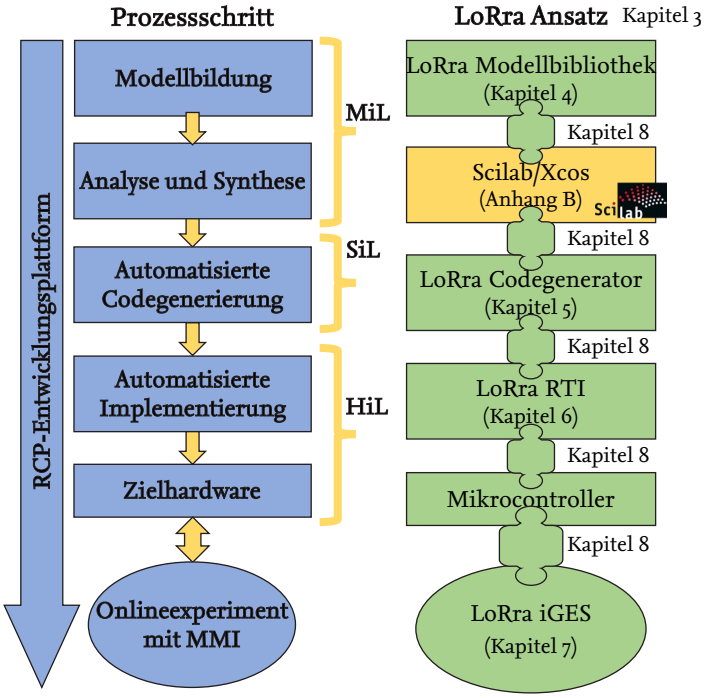

Abbildung 1.2: Weiterer Aufbau der Arbeit anhand des RCP-Prozesses.

Kapitel 2 fasst den aktuellen Stand des Wissens zum Themenkomplex zusammen. Es bietet eine Übersicht bestehender Ansätze zur modellbasierten Funktionsentwicklung und stellt die in dieser Arbeit verwendete wissenschaftliche RCP-Entwicklungsmethodik vor. Anschließend erfolgt eine Diskussion ausgewählter Plattformen zur modellbasierten Funktionsentwicklung sowie eine Betrachtung spezifischer Werkzeuge zur Durchführung einzelner Aufgaben im RCP-Entwicklungsprozess.

In Kapitel 3 erfolgt die Konzeption der neuartigen LoRra-Entwicklungsplattform. Zunächst werden Nutzungsszenarien analysiert und mittels Anforderungsmanagement wird ein hierarchisches Anforderungsmodell auf Systemebene abgeleitet. Das in dieser Arbeit als Vorgehensmodell bei der Plattformentwicklung eingesetzte Spiralmodell wird eingeführt und ein Systemkonzept der LoRra-Plattform erarbeitet.

Die LoRra-Modellbibliothek zum ganzheitlichen Datenmanagement ist Inhalt von Kapitel 4. Anhand der wesentlichen Nutzungsszenarien erfolgt eine Anforderungsanalyse sowie die Konzeption der Modellbibliothek. Kernbestandteil sind das Versions- und Konfigurationsmanagement für sämtliche Ergebnisse des Entwicklungsprozesses. Anschließend erfolgen der Entwurf zentraler Datenstrukturen und Features sowie die Realisierung als eigenständig einsetzbares Werkzeug.

Im Rahmen der automatisierten Codegenerierung wird in Kapitel 5 der Entwurf einer Modell-zu-Text-Transformation für Blockschaltbild basierte Xcos-Modelle präsentiert. Die Generierung von Code erfolgt automatisch auf Basis von Graphen-Transformationen der Modelltopologie sowie einer gesonderten Be-

trachtung des Verhaltens einzelner Blöcke. Im Rahmen der Bewertung des generierten Codes wird zudem eine Diskussion der Ergebnisse im Vergleich mit denen anderer Codegeneratoren anhand ausgewählter Metriken durchgeführt.

Kapitel 6 enthält Untersuchungen zur automatisierten Echtzeitimplementierung. Zur flexiblen Nutzung mit verschiedenen Hardwaresystemen wird eine Schichtenarchitektur mit standardisierten Schnittstellen zur Hardwareabstraktion eingesetzt. Es erfolgen der Entwurf einer exemplarischen Echtzeithardware zur Behandlung der in Kapitel 9 forcierten Pilotanwendung sowie von spezialisierten Funktionsblöcken zur modellbasierten Konfiguration und Anbindung der Hardwareschnittstellen.

Für die Durchführung von Mess- und Kalibrieraufgaben im Rahmen von Online-Experimenten wird in Kapitel 7 das grafische MMI iGES[9] eingeführt. iGES ist modular aufgebaut und ermöglicht das einfache Erstellen und flexible Konfigurieren grafischer Experimentieroberflächen. Zudem ist eine Echtzeitkommunikation für Online-Experimente integriert.

In Kapitel 8 werden sämtliche Bestandteile zur neuartigen LoRra-Entwicklungsplattform mit ihren emergenten Eigenschaften integriert. Dies ermöglicht die Realisierung eines durchgängigen Entwicklungsprozesses mit nahtloser Unterstützung der RCP-Systematik. Anschließend erfolgt in Kapitel 9 die durchgängige Entwicklung einer Pilotanwendung mit LoRra zur grundlegenden Verifikation der Plattform. Kern ist die Auslegung einer Trajektorienfolgeregelung für ein autonomes Fahrzeug. Hierfür wird zunächst eine Drehzahlregelung der Antriebsmotoren als innere Kaskade und anschließend die Folgeregelung entwickelt.

Die Arbeit schließt in Kapitel 10 mit einer Zusammenfassung sowie einem Ausblick auf weiterführende Forschungs- und Entwicklungstätigkeiten.

[9] iGES steht für integrierte Grafikunterstützte Experimentierumgebung.

2 Stand des Wissens

In der aktuellen Forschung ist die Entwicklung von Funktionen für eingebettete Systeme, wie in Abschnitt 1.1 dargelegt, von großer Bedeutung. Die Arbeiten erstrecken sich von abstrakten Ansätzen zum modellbasierten Systems Engineering (vgl. [140]) bis hin zur optimierten Realisierung eingebetteter Systeme, etwa durch die automatisierte Parallelisierung von Funktionen für Mehrkernsysteme (siehe z. B. [138]).

Die vorliegende Arbeit verfolgt das Ziel, die kostengünstige Funktionsentwicklung für eingebettete mechatronische Systeme durchgängig und systematisch zu ermöglichen. Eine grundlegende Voraussetzung hierfür ist der Einsatz von Modellen sowie einer ganzheitlichen Entwicklungsmethodik. In diesem Kontext wurden bereits verschiedene Forschungs- und Lösungsansätze präsentiert, die in der vorliegenden Arbeit weiterentwickelt und ergänzt werden.

Im folgenden Kapitel wird der Stand der Forschung zur modellbasierten Funktionsentwicklung eingebetteter mechatronischer Systeme sowie zu den unterstützenden Werkzeugen zusammengefasst. Abschnitt 2.1 beleuchtet zunächst unterschiedliche Ansätze der modellbasierten Entwicklung und führt die im Rahmen dieser Arbeit verfolgte wissenschaftliche RCP-Entwicklungsmethodik ein. Die nachfolgenden Abschnitte 2.2 und 2.3 geben einen Überblick über verfügbare CAE-Plattformen und -Werkzeuge zur modellbasierten Funktionsentwicklung. Abschließend werden die Lücken in den gegenwärtig verfügbaren Low-Cost-Ansätzen identifiziert und diskutiert.

2.1 Modellbasierte Funktionsentwicklung

Der Markt für eingebettete Systeme ist geprägt von einem hohen Innovationsdruck, ständig sinkender Time-to-Market und einem enormen Kostendruck [164]. Infolge dieser Herausforderungen hat sich die Entwicklung vom klassischen dokumentenbasierten Ansatz hin zu einem modellbasierten Ansatz gewandelt [81]. Der gezielte Einsatz von Modellen sowie unterstützenden Werkzeugen und Bibliotheken spielt hierbei eine entscheidende Rolle. Ein zentraler Vorteil dieses Wandels liegt in der Steigerung der Effizienz bei gleichzeitiger Sicherstellung hoher Qualität [122]. So lässt sich der Entwicklungsaufwand im Vergleich zu konventionellen Methoden um bis zu 59 % reduzieren, wobei in frühen Phasen, insbesondere während der Funktionsspezifikation, bis zu 70 % der benötigten Zeit eingespart werden können [78].

Nach Kautz [109] ist ein wesentliches Ziel des Model-Based Design (MBD) die Beschreibung von Software oder Teile eines Softwaresystems durch abstrakte Modelle, um diese konstruktiv für verschiedene Aufgaben zu nutzen. Mathematische

Modelle und höherwertige Beschreibungsformen (z. B. Blockschaltbilder) dienen als ausführbare Spezifikation des Systemverhaltens. Hierdurch lassen sich interpretationsabhängige Fehler minimieren und sowohl die Verständlichkeit als auch die Transparenz innerhalb des Entwicklungsprozesses für alle Beteiligten steigern [160]. Daher arbeiten im Bereich der eingebetteten Systeme ca. 89 % der Softwareentwickelnden mit Modellen [7]. Automatisch unterstützende Techniken bei der Entwicklung – insbesondere großer eingebetteter Systeme – steigern zudem die Effektivität und führen zu qualitativ höherwertigen Ergebnissen [71].

In der Literatur werden unter dem Begriff der modellbasierten Entwicklung divergente Ansätze beschrieben. Es gibt eine Vielzahl von Begriffen (z. B. Model-Based Software Engineering, Model-Based Control Design, Model-Based Systems Engineering), die nicht einheitlich standardisiert sind [6]. In dieser Arbeit wird unabhängig von der Bezeichnung in der Ursprungsliteratur einheitlich von MBD gesprochen, wenn es um die modellbasierte Entwicklung von eingebetteten Systemen geht. RCP ist die wissenschaftliche Entwicklungsmethodik, welche dieser Arbeit zugrunde liegt und in Abschnitt 2.1.2 eingeführt wird. Andere Bezeichnungen werden kontextabhängig erläutert.

Im Folgenden sind zunächst relevante Vorarbeiten zu modellbasierter Funktionsentwicklung und RCP zusammengefasst. In diesem Kontext werden die essenziellen Forschungsarbeiten, Publikationen sowie relevante Standards und Richtlinien erörtert. Im Anschluss erfolgt eine Diskussion der wissenschaftlichen RCP-Methodik, welche dieser Arbeit als Referenz dient.

2.1.1 Vorarbeiten zu modellbasierter Funktionsentwicklung und Rapid Control Prototyping

In den letzten 20 Jahren sind hierdurch über 90 wissenschaftliche Artikel zur modellbasierten Funktionsentwicklung und RCP veröffentlicht worden, wobei in 64 % der Fälle Matlab / Simulink als Simulationswerkzeug zum Einsatz kam. Ein erkennbarer Trend im Bereich der Funktionsentwicklung ist der vermehrte Einsatz kostengünstiger digitaler Signalprozessoren (DSP)[10] [88]. Der folgende Abschnitt gibt einen Überblick über die wesentlichen Veröffentlichungen, Standards und Richtlinien.

Wesentliche Forschungen und Veröffentlichungen

Hanselmann [76] führt eine ganzheitliche Entwicklungsumgebung für automobile Anwendungen ein. Basierend auf den acht Schritten Modellbildung, Reglerauslegung, Offline-Test, Anbindung der Schnittstellen, Codegenerierung, Echtzeitsimulation, Datenauswertung und Iteration wird ein auf Matlab / Simulink und

[10] Ein DSP ist ein schneller Mikroprozessor, dessen Architektur und Befehlssatz speziell für die effiziente Implementierung digitaler Signalverarbeitungsalgorithmen entwickelt wurden [37].

einem DSP als Echtzeithardware basierender Ansatz vorgestellt. In [77] wurde diese Methode zum Einsatz für mechatronische Systeme verallgemeinert.

Gausemeier und Lückel [64] fassen die Ergebnisse des Forschungsprojektes EUMECH zusammen, wobei Schwerpunkte auf Entwicklungsmethodik, Modellbildung, Integrationsplattform und mechatronische Lösungselemente gelegt werden. Es erfolgt die Erarbeitung eines Referenzmodells zur Entwicklung mechatronischer Produkte mit den für die Entwicklung relevanten Phasen Produktfindung, Produktkonzipierung, Produktentwurf, Produktausarbeitung. Eine integrative Vorgehensweise ermöglicht eine domänenübergreifende Betrachtung. Lückel et al. [137] konkretisieren die Methode für eine integrierte, ganzheitliche, rechnergestützte Auslegung mechatronischer Systeme und stellen einen iterativen Prozess zum Entwurf von kinematischen, dynamischen und mechatronischen Funktionen vor.

Schnieder [188] beschreibt Methoden der Automatisierungstechnik, welche teilweise aus dem Schwerpunktprogramm 1016 der Deutsche Forschungsgemeinschaft (DFG): „Analyse und Synthese kontinuierlich-diskreter technischer Systeme“ (KONDISK) resultieren. Kern ist die Petrinetz-basierte Entwicklungsmethodik BASYSNET mit formal repräsentierten Grundkonzepten. Der zugehörige CAE-Werkzeugprototyp COSYNET ermöglicht den automatisierten netzbasierten Entwurf von Steuerungssystemen. Auch die Behandlung hybrider Systeme[11] ist möglich [189]. Orth [159] erweiterte das Verfahren um eine automatisierte Echtzeitrealisierung, basierend auf der Bibliothek Netlab für Matlab / Simulink und der Leittechnikplattform ACPLT als Echtzeithardware.

Als Ergebnis des DFG-Projekts „Durchgängige Entwurfsmethodik und Simulation dezentraler Steuerungselemente für mechatronische Systeme in der Automatisierungstechnik“ stellen Reichmann et al. [175] die Integrationsplattform GeneralStore vor. Basierend auf Metamodellen[12] in der Unified Modeling Language (UML)[13] werden hierbei verschiedene spezialisierte CAE-Werkzeuge integriert. Durch Einsatz verschiedener Codegeneratoren und einer auf UML Diagrammen basierten Integration der einzelnen Komponenten wird eine echtzeitfähiges Gesamtsystem geschaffen.

Abel und Bollig [2] fassen die Lehrveranstaltung RCP an der RWTH Aachen zusammen. Schwerpunkt liegt neben regelungstechnischen Grundlagen zu Modellbildung, Entwurf und Simulation auf einem durchgängigen modellbasierten Entwicklungsprozess unter Einsatz von MiL, SiL und HiL. In diesem Zusammenhang werden verschiedene RCP-Werkzeuge von The Mathworks, dSPACE, National Instruments und Modelica diskutiert.

---

[11] Hybride Systeme beinhalten sowohl zeitkontinuierliche als auch zeitdiskrete Bestandteile [75].

[12] Ein Metamodell repräsentiert auf höherer Abstraktionsebene die Konzepte nach denen ein konkreteres Modell aufgebaut ist [111].

[13] Erläuterung siehe Anhang A.