Eine strukturierte Modellbibliothek für Analysen an Antriebssträngen von Elektrofahrzeugen
A Structured Model Library for the Analysis of Electric-Vehicle Drivetrains

Dipl.-Ing. Dipl.-Inf. Simon Niedzwiedz, Prof. Dr.-Ing. Stephan Frei,
Technische Universität Dortmund, AG Bordsysteme, Dortmund, Germany

Kurzfassung

Im diesem Beitrag soll ein Konzept für die Entwicklung strukturierter Bibliotheken für Simulationsmodelle vorgestellt werden. Die Hauptziele sind dabei die Sicherstellung des verlässlichen Zusammenspiels der enthaltenen Modelle miteinander und die Vorgabe, dass eine Modellbibliothek einerseits flexibel und in der Verwaltung komfortabel sein soll, andererseits aber eine nachvollziehbare und eindeutige Ordnung definieren muss. Gerade in größeren oder verteilten Arbeitsgruppen sind die genannten Eigenschaften wichtig, um eine Modellbibliothek langfristig und effizient nutzen zu können. Für die Wiederverwendbarkeit und die kontinuierliche Weiterentwicklung ist eine konsequente Strukturierung unverzichtbar. Das entwickelte Konzept wird hier exemplarisch an einer Modellbibliothek für Analysen an Antriebssträngen für Elektrofahrzeuge erläutert.

Abstract

This contribution presents a concept for developing structured simulation model libraries. The main objectives are securing a reliable functionality of the contained models among each other and meeting the requirements that a model library has to be flexible and easy to manage on one side, while it has to define a comprehensible and distinct order on the other side. These qualities are especially important in bigger or distributed working groups, for ensuring a long-term and efficient usability. For model reusability and continuous further development a consistent structuring is essential. This concept will be exemplified through a structured model library for the analysis of electric-vehicle drivetrains.

1 Einleitung

Fragestellungen zur Entwicklung von elektrischen und elektronischen Fahrzeugkomponenten werden immer häufiger durch Simulationen beantwortet. Für Einzelkomponenten müssen dabei oft unterschiedliche Analysemodelle, angepasst an die jeweils spezielle Fragestellung, entwickelt werden, während der Modellkern oder das umgebende Gesamtsystem jedoch unverändert bleiben. Daher ist es sinnvoll, ausgehend von wohldefinierten Basismodellen, unterschiedliche Funktionsmodelle abzuleiten oder diese Modelle sukzessive zu erweitern, wodurch wertvolle Entwicklungszeit gespart werden kann. Bei simulatorischen Untersuchungen von einzelnen Komponenten stellt dies im Allgemeinen kaum Probleme dar. Bei komplexen Gesamtsystemen aus verschiedenen Einzelkomponenten, kann dies jedoch schnell zu Inkompatibilitäten und einer äußerst unübersichtlichen Modellbibliothek führen. Wodurch der eigentliche Modellierungs- und Verwaltungsaufwand wiederum deutlich erhöht wird.
Ziel bei der Entwicklung einer Modellbibliothek ist daher nicht nur die triviale Sicherstellung der zuverlässigen und korrekten Funktion der Einzelmodelle. Ebenso wichtig ist die Gewährleistung der verlässlichen Kompatibilität und Interoperabilität der Komponenten innerhalb größerer Gesamtmodelle. Diese Anforderungen dürfen auch dann nicht verletzt werden, wenn Modelle von Einzelkomponenten durch weiterentwickelte oder detailliertere Modellvarianten ersetzt werden, oder gänzlich neue Komponenten hinzugefügt werden.
Es ist daher angebracht, diesen Anforderungen frühzeitig und präventiv zu begegnen, indem passende Konzepte aus der Softwaretechnik – insbesondere der objektorientierten Programmierung oder Protokollentwicklung – auf die Simulations- und Modellierungsprozesse übertragen werden, mit dem Ziel ein höchstmögliches Maß an Interoperabilität und Wiederverwendbarkeit vorhandener und verifizierter Modelle zu ermöglichen. Verschiedene Ideen hierzu lieferten [1], [2], [3], [4] und [5].
Eine mögliche Herangehensweise, die aktuell beim Aufbau einer strukturierten Modellbibliothek für simulationsbasierte Analysen an Elektrofahrzeugen eingesetzt wird, soll in diesem Beitrag präsentiert und exemplarisch erläutert werden. Als konkretes Anwendungsbeispiel für die Methodik dient dabei abschließend das Modell eines stark vereinfachten elektrischen Antriebsstrangs.

2 Definition einer strukturierten Modellbibliothek

Ein sehr wichtiges Kriterium für eine dauerhaft funktionsfähige und benutzbare Modellbibliothek ist die Art und

Weise, wie die Bibliothek strukturiert ist. Die Struktur muss einerseits eine regelbasierte und einfach nachvollziehbare Ordnung der Modelle vorgeben, die zudem komfortabel zu verwalten ist. Andererseits darf sie jedoch kein zu starres Korsett bilden, das flexible Anpassungen und innovative Erweiterungen hinsichtlich zukünftiger Entwicklungen behindert, oder eine so komplexe Datenstruktur darstellen, dass eine zügige Orientierung erschwert wird. Darüber hinaus muss es auch möglich sein, sämtliche relevanten Daten zu Einzelmodellen schnell verfügbar zu haben und diese untereinander vergleichen zu können.

Ein weiterer sehr wichtiger Aspekt ist die Generierung eindeutiger Identifikationsmerkmale für die einzelnen Modelle. Die klassische Methode, verschiedene Modelle einfach in der Reihenfolge ihrer zeitlichen Entstehung oder mit steigender Komplexität als „*Modell M Level L*“ durchzunummerieren, stößt in größeren Bibliotheken und bei Kollaboration verschiedener Arbeitsgruppen mit unterschiedlichsten Fragestellungen an ihre Grenzen. Dies wird schnell ersichtlich, wenn bei der Weiterentwicklung eines Modells eine Verzweigung auftritt und zum gleichen Zeitpunkt zwei ähnlich komplexe Modellvarianten mit völlig unterschiedlichen Untersuchungszielen entstehen, beispielsweise eine für thermische und eine für EMV-Betrachtungen.

Eine Lösungsmöglichkeit für diese Herausforderungen bietet die folgende Herangehensweise.

2.1 Standardisierung der Modelle

Als erstes soll hier auf die Metadefinition – also die Definition, wie eine Definition zu erfolgen hat – der einzelnen Simulationsmodelle eingegangen werden, da diese die atomaren Bausteine einer Modellbibliothek darstellen. Sie können daher graphentheoretisch als Bestandteile einer Knotenmenge V betrachtet werden. Auf diese strukturierende Eigenschaft wird zwar erst in Abschnitt 2.3 näher eingegangen, allerdings sollte dieser Aspekt schon von Anfang an mit bedacht werden.

Um eine einfache Verwaltung der gesamten Einzelmodelle, den geforderten schnellen Zugriff auf die relevanten Modelldaten und die einfache Vergleichbarkeit zu gewährleisten, ist es sinnvoll, die Metadefinition der Modelle aus geeigneten standardisierten Unterstrukturen – subatomaren Bausteinen – aufzubauen (**Bild 1**). Die Metadefinition eines Modells macht dabei jedoch keinerlei Aussage über den internen Aufbau des Modells, sondern definiert lediglich alle nach außen notwendigen Aspekte. Sie ist also vergleichbar mit Klassendefinitionen, wie sie im Rahmen von UML[1]-Klassendiagrammen gemacht werden.

2.1.1 Metadaten

Die Metadaten beinhalten die Daten, die das jeweilige Modell beschreiben und in die Hierarchie einordnen. Die beschreibenden Daten bestehen aus dem Namen des Modells und einer Kurzbeschreibung seiner jeweiligen speziellen Funktion. Zusätzlich wird hier der Typ des Modells festgelegt. Ein Modell kann dabei entweder ein normal instanziierbares Modell oder ein abstraktes Modell sein, welches nicht instanziierbar ist und eigentlich nur in Verzweigungspunkten der Entwicklung vorkommt. Es ist damit prinzipiell vergleichbar mit abstrakten Klassen in der Softwaretechnik, kann aber im Rahmen von Kompatibilitätsbetrachtungen (s. Abschnitt 2.3.2) auch wie eine Interface-Klasse behandelt werden. Als Ordnungsdaten werden hier die Komponentenklasse und das eindeutige Ordnungs- bzw. Identifikationsmerkmal des Modells innerhalb seiner Komponentenklasse angegeben, so dass dem Modell innerhalb der gesamten Modellbibliothek ein eindeutiger Platz zugeordnet werden kann. Eine Komponentenklasse ist dabei als eine Sammlung aller Modellvarianten einer funktionalen Simulationskomponente aufzufassen – zum Beispiel DC/AC-Wandler (s. Abschnitt 3).

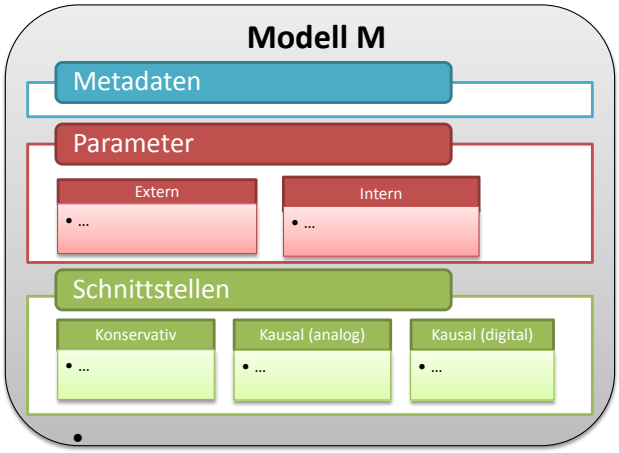

Bild 1 Standardisierter regulärer Modellaufbau

2.1.2 Parameterwerte

Bei den Parameterwerten sind zwei verschiedene Parametertypen zu unterscheiden, die externen und die internen Parameter. Erstere sind dabei von außen für den Anwender schreib- und lesbar zugänglich, während letztere hauptsächlich für modellinterne Funktionen gedacht sind und für den Anwender nur lesbar zugänglich sind. Definiert werden die Parameter beider Typen durch ihren lokal eindeutigen Namen, ihre Einheit – eine physikalische Einheit oder die dimensionslose Einheit „*[1]*“ – und einer Kurzbeschreibung des Parameters. Zusätzlich muss für jeden Parameter ein Default-Wert angegeben werden, da der Wert eines Parameters zum Zeitpunkt der Instanziierung des Modells bekannt sein muss. Bei externen Parametern ist der Default-Wert immer ein tatsächlicher Wert. Bei internen Parametern kann dies aber auch eine mathematische Formel unter Verwendung anderer Parameter sein. Parameterwerte lassen sich somit mit den privaten und öffentlichen Member-Konstanten aus der Softwaretechnik vergleichen.

[1] Unified Modeling Language [6]

2.1.3 Die Modellschnittstellen

Bei den Schnittstellen müssen ebenfalls zwei grundsätzliche Typen unterschieden werden, die konservative Schnittstelle und die Daten- oder kausale Schnittstelle. Grundsätzlich muss die Definition von Schnittstellen mit besonderer Sorgfalt und eindeutig erfolgen, da die Schnittstellen eines Modells hinsichtlich der Kompatibilität und Interoperabilität die höchste Relevanz besitzen. Die Festlegung des Namens, der Beschreibung und des Schnittstellentyps muss daher sehr sorgsam erfolgen.

2.1.3.1 Konservative Schnittstellen

Die konservativen Schnittstellen sind Schnittstellen über die die Modelle untereinander physikalische Größen austauschen, die an entsprechende Erhaltungssätze gekoppelt sind. Normalerweise existieren also bei physikalischen Schnittstellen immer eine Fluss- und eine Differenzgröße. Dementsprechend ist bei der Definition derartiger Schnittstellen immer die physikalische Bezugsdomäne anzugeben.

2.1.3.2 Datenschnittstellen

Die Daten- oder kausalen Schnittstellen dienen dem Austausch von physikalischen Größen, die nicht an Erhaltungssätze gebunden sind, oder dem Austausch von klassischen I/O-Daten. Zu unterscheiden sind hier dementsprechend noch einmal die beiden Untertypen der analogen und digitalen Datenschnittstelle. Dazu passend muss dann auch die Angabe einer Einheit oder eines Datentyps erfolgen. Zu beachten ist bei Schnittstellen dieses Typs, dass für sie Richtungen definiert werden müssen (In, Out, In-Out) und dass sie eine Datenbreite größer eins besitzen können, sodass diese ebenfalls anzugeben ist.
Exemplarisch ist in **Tabelle 1** die Metadefinition für das Basismodell einer einfachen Batterie dargestellt.

2.1.4 Entwicklungsregeln

Neben dieser formalen Definition einer standardisierten Metadefinition der Modelle müssen zusätzlich noch Regularien definiert werden, die bei der Weiterentwicklung von Modellen eine weiterhin korrekte Funktion und die Kompatibilität zu den Vorgängermodellen sicherstellen. Vor allem müssen dabei Regularien zur korrekten Ableitung von Modellen berücksichtigt werden. Grundsätzlich sollten Modellableitungen nur erweiternd erlaubt sein, das heißt, es dürfen nur Schnittstellen oder Funktionalität hinzugefügt werden. Eine Ausnahme stellt hier die Umwandlung des Schnittstellentyps dar, wobei diese im Normalfall nur von der kausalen Schnittstelle hin zur konservativen Schnittstelle erfolgen sollte.
In diesem Zusammenhang ist es auch wichtig sich über Konzepte wie Mehrfachvererbung und Polymorphie Gedanken zu machen, die einerseits die Flexibilität und Adaptionsfähigkeit erhöhen, andererseits aber auch die Komplexität der Bibliotheksstruktur deutlich vergrößern.
Als wichtigste Grundregel muss aber gelten, dass die einzelnen Modelle ihre eigene korrekte Funktionalität selbst sicherstellen müssen. Das heißt, dass eine Modellvariante auch dann noch korrekt funktionieren sollte, wenn die neu hinzugekommenen Schnittstellen nur mit Dummy- oder Default-Werten gespeist werden. Diese Funktionsgarantie muss zumindest für eine gewisse, fest definierte Anzahl an Vorgängern gelten (Abwärtskompatibilitätstiefe).

2.2 Definition der Bibliotheksstruktur

Nach der Standardisierung der Metadefinition für die Einzelmodelle müssen auch die Strukturen definiert werden, die der Bibliothek eine feste Ordnung verleihen. Grundsätzlich müssen im Kontext einer Modellbibliothek dabei zwei Ordnungsebenen unterschieden werden. Zum einen existiert eine Ordnung innerhalb der einzelnen Komponentenklassen (s. Abschnitt 2.2.1), die „lokale Ordnung“, aus der die Ableitungen und Kompatibilitäten innerhalb dieser Klasse ersichtlich werden.
Zum anderen existiert eine „globale Ordnung“, die die Kompatibilitäten und Interoperabilität zwischen den einzelnen Komponentenklassen beschreibt.

Modellname	Simple_Battery_Lev_0				
Metadaten					
Typ	Modell				
Komponentenklasse	Energy_Storages				
Klassen-ID	0				
Beschreibung	Einfaches Batteriemodell mit konstanter Quellspannung und konstantem Innenwiderstand				
Parameter					
Typ	**Name**	**Einheit**	**Default-Wert**	**Beschreibung**	
extern	Q0_cell	As	1.0	Ladung pro Zelle	
extern	V0_cell	V	1.0	Spannung pro Zelle	
extern	n_cells_per_block	[1]	1	Anzahl serieller Zellen per Block	
extern	n_blocks	[1]	1	Anzahl paralleler Blöcke	
extern	SoC0	[1]	0.5	Initialer Ladezustand	
intern	R_int	Ohm	0.1	Innenwiderstand der Batterie	
Schnittstellen					
Physikalisch/ konservativ					
Name	**Einheit/Domäne**	**Beschreibung**			
P	Elektrisch	Positive Anschlussklemme			
N	Elektrisch	Negative Anschlussklemme			
Daten/ kausal					
Typ	**Name**	**Einheit/Datentyp**	**Breite**	**Richtung**	**Beschreibung**
analog	SoC	[1]	1	Out	Aktueller Ladezustand

Tabelle 1 Metadefinition eines Batteriemodells

Aus der Verschmelzung der beiden Ordnungsebenen lassen sich dann die klassenübergreifenden Kompatibilitätsbeziehungen der einzelnen Modelle und Modellvarianten ermitteln.
Als geeignete Datenstruktur, die gleichzeitig Hierarchien abbilden und sich flexibel an modulare Weiterentwicklungen anpassen kann, bieten sich dabei Graphen an, bei denen die Einzelmodelle und Modellklassen auf Knotenmengen abgebildet werden.

2.2.1 Der Klassenentwicklungsgraph

Für die Beschreibung der lokalen Ordnung innerhalb einer Komponentenklasse existieren zwei Graphen, die sich allerdings eine Knotenmenge V_K und eine Wurzel R_K teilen. Die nichtleere Knotenmenge $V_K = \{v_{K,i} \,|\, i \in N_0^+\}$ wird durch die zu einer Komponentenklasse zugehörigen Einzelmodelle gebildet. Da eine Komponentenklasse nicht leer sein darf, existiert mit dem jeweilige Basismodell in jedem Klassentwicklungsgraph ein expliziter Wurzelknoten $R_K = v_{K,0}$.
Für den Klassenentwicklungsgraphen $G_K = (V_K, E_K, R_K)$ beschreiben die streng abwärts gerichteten Kanten $e_{i,j} = (v_{K,i}, v_{K,j}) | \, i < j$ der Kantenmenge E_K dann die Ableitungsbeziehungen der einzelnen Modellvarianten. Zyklen sind dabei nicht erlaubt – striktes Erweiterungsprinzip. Dabei ist der Ausgangsgrad $d^+_{G_K}(v_{K,i})$ der Knoten generell größer oder gleich 0. Für den sinnvollen Fall, dass Mehrfachvererbungen bei der Ableitung erlaubt sind, ist der Eingangsgrad $d^-_{G_K}(v_{K,i}) \geq 1$ – mit Ausnahme der Wurzel.
Mit Hilfe dieser Datenstruktur bietet sich auch eine Möglichkeit, den einzelnen Modellen ein eindeutiges Ordnungs- bzw. Identifikationsmerkmal zuzuweisen. Dabei werden die einzelnen Knoten einer Hierarchieebene von links nach rechts durchnummeriert und beim Übergang in eine andere Hierarchieebene für jede entlanggelaufene Kante ein Punkt eingefügt. Dadurch entstehen dann innerhalb einer jeden Komponentenklasse eineindeutige Identifikatoren $ID(v_{K,i}) = a.b.\cdots.z \mid a, b, \cdots, z \in N_0^+$, die zugleich die Position innerhalb der Komponentenklasse wiedergeben.

2.2.2 Der Kompatibilitätsgraph

Parallel zum Klassenentwicklungsgraphen existiert noch der Klassenkompatibilitätsgraph $G_{KK} = (V_K, E_{KK}, R_K)$, der sich lediglich in der Kantenmenge unterscheidet. Die gerichteten Kanten $e_{i,j} = (v_{K,i}, v_{K,j})$ der Kantenmenge E_{KK} sind dabei entweder aufwärts oder horizontal gerichtet und beschreiben somit die versionshistorische Abwärtskompatibilität der einzelnen Modellvarianten der Komponentenklasse. Abwärts gerichtete Kanten können hier nicht existieren, da bei der Modellentwicklung ein striktes Erweiterungsprinzip gefordert ist. Trivialerweise muss hier für den Eingangsgrad Knoten $d^-_{G_{KK}}(v_{K,i}) \geq 0$ gelten; und für den Ausgangsgrad entsprechend $d^+_{G_{KK}}(v_{K,i}) \geq 1$.

2.2.3 Globaler Bibliotheksgraph

Für die globale Ordnungsebene wird ein ungerichteter Graph ohne Wurzel ($G_B = (V_B, E_B)$) verwendet, da die einzelnen Komponentenklassen untereinander prinzipiell gleichberechtigt sind. Die in der Bibliothek zusammengefassten Komponentenklassen bilden die nichtleere Knotenmenge $V_B = \{v_{B,i} \,|\, i \in N^+\}$ des Graphen. Die ungerichteten Kanten $e_{i,j} = \{v_{B,i}, v_{B,j}\} = e_{j,i}$ der Kantenmenge E_B beschreiben dann die Kompatibilitätsbeziehungen der Komponentenklassen, welche immer in beide Richtungen gelten müssen.
Bild 2 zeigt exemplarisch den reduzierten Graphen für die entwickelte Bibliothek zur Analyse von Antriebssträngen von Elektrofahrzeugen.

3 Modellbibliothek für den Antriebsstrang von Elektrofahrzeugen

Die vorgestellte Methodik zur Entwicklung einer strukturierten Modellbibliothek wird hier für Komponenten eines Antriebsstrangs von Elektrofahrzeugen angewendet. Einen kleinen Ausschnitt aus den existierenden Komponentenklassen zeigt **Bild 2**. Exemplarisch soll hier kurz die Weiterentwicklung eines dreiphasigen Umrichter-Modells betrachtet werden.

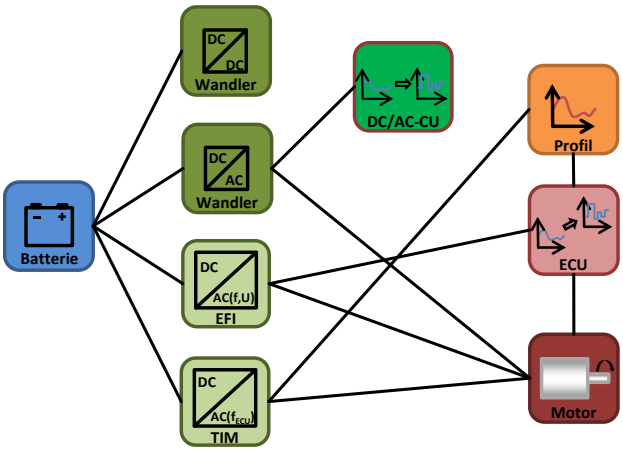

Bild 2 Globaler Bibliotheksgraph G_B

3.1 Definition der Komponentenklasse

Da eine Komponentenklasse nicht leer sein darf, findet ihre Definition zeitgleich mit der Definition ihres Basismodells – dem Wurzelknoten – statt. Für den vorliegenden Fall der DC/AC-Wandler-Klasse ist dies ein idealisiertes, verhaltensorientiertes Funktionsmodell. Dem DC/AC-Wandler werden über kausale Schnittstellen die Sollgrößen für die Spannungsamplitude, die Frequenz und der initiale Phasenwinkel der zu erzeugenden Wechselspannung vorgegeben. Dazu lässt sich noch ein prozentualer Wirkungsgrad einstellen. Da der Umrichter für den Einsatz in Fahrzeugantrieben gedacht ist, lässt sich auch noch die Richtung des erzeugten Drehfelds umschalten.

Bei dem Basismodell werden die Ausgangsgrößen dann rein mathematisch verhaltensbasiert erzeugt. Die Kopplung zwischen DC- und AC-Seite wird über die elektrischen Leistungen und die Verlustleistung realisiert.
Die elektrischen Größen werden über entsprechende konservative Anschlussklemmen übertragen.

3.2 Modellvarianten

3.2.1 Modell mit idealisierten Schaltbrücken

Als zweite Modellvariante entstand ein erstes strukturorientiertes Modell, welches logisch gesehen auf der gleichen Hierarchieebene liegt wie das verhaltensorientierte Basismodell. Bei den nach außen sichtbaren Parametern und Schnittstellen hat sich dabei nichts verändert. Intern allerdings werden die AC-Spannungen nicht mehr mathematisch sondern über drei Halbbrücken mit idealen Schaltern erzeugt. Dadurch erlangen die AC-Größen ihr deutlich realistischeres und für Schaltwandler charakteristisches gepulstes Aussehen.

3.2.2 Schaltbrückenmodell mit Ausgangsfiltern

Die nächste Modellvariante des Frequenzumrichters wurde direkt von der vorherigen Modellvariante abgeleitet und erweitert diese intern um LC-Filterstrukturen. Diese dienen der Glättung der pulsförmigen Wechselspannungen, hin zu sinusförmigen Spannungen mit deutlich verringerten Oberwellen. Als zusätzliche externe Parameter sind dabei die Induktivitäts- und die Kapazitätswerte der Ausgangsfilter hinzugekommen.

3.2.3 EMV-Modell mit EMV-Transistormodellen

Als vierte Modellvariante ist ein Modell entstanden, welches auf der gleichen Hierarchieebene anzusetzen ist, wie die zweite Modellvariante, da hier lediglich die idealen Schalter durch für EMV-Untersuchungen realistischere Transistormodelle ersetzt wurden. Da die beiden Modellvarianten also nach außen formal identisch erscheinen, lässt sich in den Klassenentwicklungsgraphen nun zusätzlich eine abstrakte Modellvariante einfügen, aus der die zweite und vierte Modellvariante abgeleitet scheinen.
Für genauere EMV-Untersuchungen müssen in dem entsprechenden Teilbaum sowohl in der Breite, wie auch in der Tiefe noch viele weitere Modelle eingefügt werden, um Effekte wie die parasitären Eigenschaften von Filtern, die HF-Eigenschaften der funktionalen Komponenten bis hin zum Einfluss des Gehäuses auf die Störaussendungen des Umrichters zu betrachten.
Den Klassenentwicklungsgraph für den bisherigen Entwicklungsstand zeigt **Bild 3**. Die einzelnen Entwicklungsstufen und ihre Ableitungszusammenhänge sind dabei, wie gefordert, einfach und schnell erkennbar. Ebenfalls wird hier deutlich, dass die Methodik zur Bildung der Identifkatoren zu eindeutigen Merkmalen führt, die zugleich die Position der einzelnen Modellvarianten wiederspiegeln. **Bild 4** zeigt den zugehörigen Klassenkompatibilitätsgraphen.

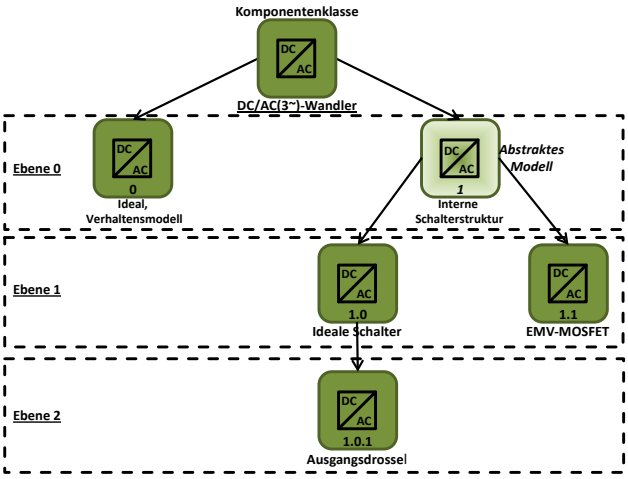

Bild 3 Klassenentwicklungsgraph für DC/AC-Wandler

4 Anwendungsbeispiel

Als Demonstrationsbeispiel für die einfache Austauschbarkeit von Modellvarianten ohne negative Beeinträchtigung der Gesamtfunktionalität soll hier der stark vereinfachte Antriebsstrang eines Elektrofahrzeuges (**Bild 5**) dienen, an dem zwei verschiedene Fragestellungen untersucht werden sollen. Verbunden sind die Komponenten dabei analog zu den in einem realen E-Fahrzeug vorhandenen physikalischen Schnittstellen.

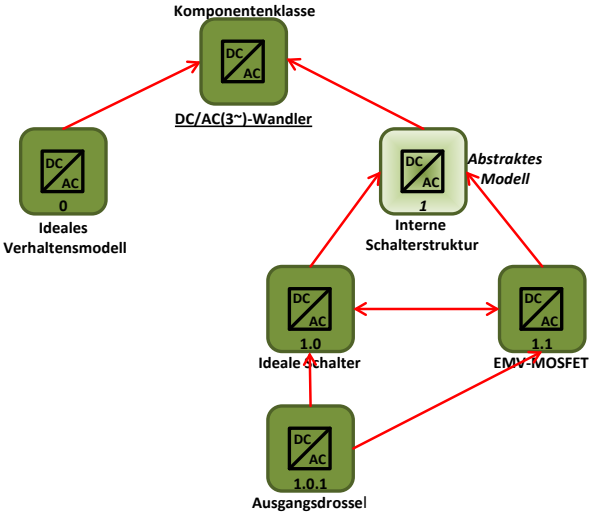

Bild 4 Kompatibilitätsgraph für DC/AC-Wandler

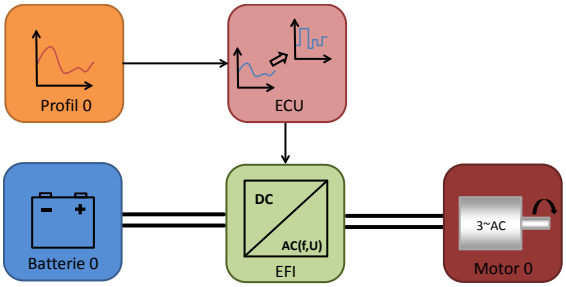

Bild 5 Vereinfachtes Antriebsstrangmodell

Im ersten Fall soll die Reichweite des Fahrzeugs untersucht werden. Im zweiten Untersuchungsfall sollen für EMV-Untersuchungen Störungen auf Leitungen betrachtet werden. Als auszutauschende Komponente dient dabei ein gesteuerter Wechselrichter (EFI – Electrical Frequency Inverter).

4.1 Reichweitenuntersuchung

Für die Reichweitenuntersuchung soll ein einfaches verhaltensbasiertes Basismodell des Frequenzumrichters ausreichen (vgl. Abschnitt 3.1). Durch die Vorgabe eines Lastprofils (*Profil 0*) wird der Frequenzumrichter (*EFI*) über die Motoransteuerung (*ECU*) gesteuert und so das Verhalten der elektrischen Maschine (*Motor 0*) kontrolliert. Beobachtet wird der Ladezustand (State of Charge - SoC) der Batterie.

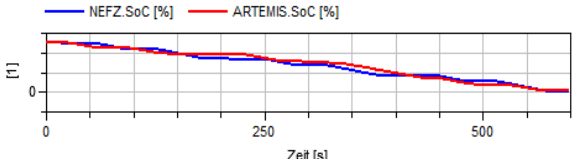

Bild 6 Ladezustand (SoC) der Batterie

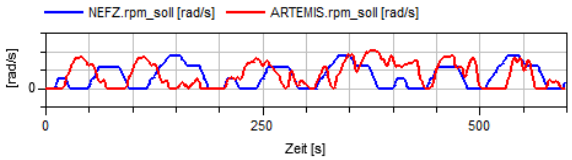

Bild 7 Solldrehzahlvorgaben der Fahrzyklen

Bild 7 zeigt die Solldrehzahl-Profile zweier Fahrzyklen.
In **Bild 6** ist der für die Reichweitenuntersuchung relevante Ladezustand der Batterie über den Verlauf den Zyklen abgebildet. Zu erkennen ist, dass für Reichweitenuntersuchungen einfache Verhaltensmodelle durchaus ausreichend sind.

4.2 EMV-Untersuchung

Für EMV-Untersuchungen hingegen werden entsprechend komplexere Modelle benötigt, die das Schaltverhalten eines Frequenzumrichters nachbilden. Die Modelle müssen also konkrete Schaltelemente, wie Leistungstransistoren, enthalten – in diesem Fall ein für EMV-Untersuchungen erstelltes MOSFET-Modell.
Bei Betrachtung von Rotorstrom (**Bild 8**) und Drehzahl (**Bild 9**) sind deutlich die Unterschiede zwischen den beiden Modellvarianten erkennbar. Diese Abweichungen sind aber lediglich auf die unterschiedlichen Genauigkeiten der Modelle zurückzuführen. Die korrekte Funktion des Gesamtsystems wurde hingegen nicht beeinträchtigt.
Anzumerken ist daher, dass beim Modellaustauch generell immer Rückwirkungen auf das Gesamtsystem auftreten, die hinsichtlich ihrer Kausalität zu identifizieren und entsprechend kritisch auf Plausibilität zu überprüfen sind.

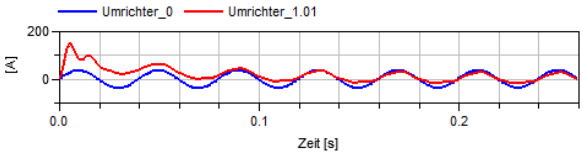

Bild 8 Rotorstrom der elektrischen Maschine

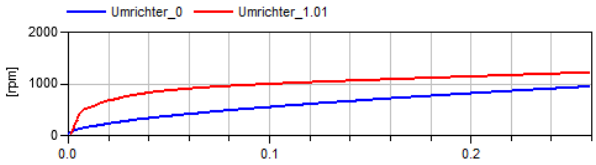

Bild 9 Mechanische Drehzahl der elektrischen Maschine

5 Zusammenfassung

Im vorliegenden Beitrag wurden zuerst grundsätzliche Überlegungen für die Strukturierung von Simulationsmodellen vorgestellt. Anschließend wurde eine Möglichkeit aufgezeigt, wie man die standardisierten Modelle effizient und strukturiert in eine Modellbibliothek einordnen und verwalten kann. Die praktische Anwendbarkeit der entwickelten Konzepte wurde dann anhand einer Modellbibliothek für simulationsbasierte Analysen an Antriebsträngen von Elektrofahrzeugen konkret gezeigt. Abschließend wurde an einem Beispiel dargestellt, dass mit Hilfe einer derart entwickelten Modellbibliothek Systemmodelle aufgebaut werden können, die sowohl eine hohe Flexibilität, als auch eine hohe funktionale Zuverlässigkeit bieten.
Die vorgestellten Konzepte treffen jedoch keine Aussagen hinsichtlich der Simulationszeit oder der Modellkomplexität, da diese Anforderungen oder Einschränkungen – zum Beispiel Echtzeitfähigkeit bei reduzierter Genauigkeit – sehr spezifisch vom Einsatzzweck der Modellbibliotheken abhängen. Ebenso werden keine Aussagen hinsichtlich der Simulationsumgebungen bzw. Simulationssprachen getroffen, da die vorgestellte Methodik, ebenso wie auch Algorithmen, auf einer Metaebene oberhalb konkreter Anwendungsfälle zu sehen ist.

6 Literatur

[1] Fritzson, Peter: Principles of object-oriented modeling and simulation with Modelica 2.1. IEEE Press, 2006
[2] Pelz, Georg: Mechatronic Systems. Wiley, 2003
[3] Otter, Martin: Objektorientierte Modellierung und Simulation von Antriebssystemen. Elektrische Antriebe, Kapitel 21, Springer, 2009
[4] Panreck, K.: Systembeschreibung zur Modellierung komplexer Systeme. at-Automatisierungstechnik, 4/99
[5] Zeigler, B.P.: Theory of Modeling and Simulation. John Wiley& Sons, 1976
[6] Object Management Group: UML-Specification 2.3. Mai 2010
[7] Hofmann, Peter: Hybridfahrzeuge. Springer, 2010