T U M

I N S T I T U T F Ü R I N F O R M A T I K

Beschreibung der Modellierung eingebetteter Systeme und deren Umwelt als Grundlage fuer die Fehlermodellierung

Markus Pister

TUM-I0719
September 07

T E C H N I S C H E U N I V E R S I T Ä T M Ü N C H E N

TUM-INFO-09-I0719-0/0.-FI

Druck: Institut für Informatik der
Technischen Universität München

Zusammenfassung

Dieser technische Bericht beschreibt die Modellierung eingebetteter Systeme und derer Umgebung als Grundlage für die Fehlermodellierung. Eingebettete Systeme sind Rechnersysteme, die in eine physikalische Umwelt eingebunden sind und dort Prozesse regeln. Dabei haben sie Schnittstellen zu anderen Rechnersystemen, zum Menschen, zur Elektrik und zur Mechanik. Entsprechend müssen bei einer Spezifikation dieser Systeme Modellierungstechniken verwendet werden, welche kontinuierliche Modelle der Physik, die Datenfluss-Sicht und die algorithmische Sicht berücksichtigen.

Auf Basis der in dieser Arbeit aufgeführten Referenzmodelle und Modellierungstechniken können spezifische potentielle Fehler identifiziert, wie auch Fehlerverhalten spezifisch beschrieben werden. Es wird eine Grundlage für weitere Arbeiten der Fehlermodellierung und Funktionssicherheitsanalyse gelegt.

Die Arbeit ist in vier Kernteile gegliedert. Der erste Teil beschreibt Referenzmodelle eingebetteter Systeme und Referenzmodelle zu den Abstraktionsebenen, sowie Eigenschaften der Modellierung der Systeme. Wesentlich sind die Schnittstellen eingebetteter Systeme (Sensoren, Aktoren, MMI, Kommunikationsinterface), die Modellierung verschiedener Domänen (Mechanik, Elektrik, Elektronik) und die Abstraktion von Implementierungsdetails. Der zweite Teil zeigt, wie mit Modellen das Verhalten beschrieben werden kann und wie mit Modellen die Artefakte beschrieben werden können, auf denen das Verhalten implementiert ist. Der dritte Teil erklärt Modellierungstechniken, mit denen Verhaltensmodelle spezifiziert werden können. Hierzu gehören Prädikate und Automaten (, so wie deren Sonderformen 'aufgelöste' Gleichungen und Betriebsmodi-Automaten). Der vierte Teil beschreibt zwei Arten von Abhängigkeiten zwischen Verhaltensmodellen, die bei der Modellierung regelmäßig auftreten. Eine Art der Abhängigkeit ist die Komposition, mit der verschiedene Modelle zu einem Ganzen zusammengefügt werden. Die andere Art der Abhängigkeit ist die Verfeinerung, in der ein Modell detaillierter beschrieben wird. Anhand dieser beiden Arten von Abhängigkeiten werden Ansatzpunkte gegeben, um potentielle Fehler zu identifizieren, die aus Abweichungen dieser Abhängigkeiten entstehen.

Inhaltsverzeichnis

1 Einleitung: Modelle und Modellierungstechniken

Dieser technische Bericht beschreibt Modelle und Modellierungstechniken zur Spezifikation eingebetteter Systeme[1] und derer Umgebung im Fahrzeug. Ziel ist es, anhand von Referenzmodellen und formalen Beschreibungen der Modellierungstechniken eine Basis zur systematischen Fehlermodellierung und damit zur Funktionssicherheitsanalyse (siehe [Lev95]) zu schaffen. Eine Funktionssicherheitsanalyse betrachtet nur die Fehler, die sie identifizieren und modellieren kann. Die Beschreibung der Fehler ist immer relativ zu der korrespondierenden Systemspezifikation und damit abhängig von den darin enthaltenen Modellen. Die Modelle sind somit ausschlaggebend für die Gestaltung einer Funktionssicherheitsanalyse. Kern ist hierbei der Begriff Modell, der in dieser Arbeit wie folgt definiert ist:

Definition 1.1 (Modell)
Ein *Modell* ist eine Abstraktion eines Sachverhalts, bei dem die für einen bestimmten Zweck wesentlichen Eigenschaften erhalten bleiben. (siehe auch [Min65]) □

Erläuterung Modell

In dieser Arbeit dienen sie der verständlichen Darstellung von Annahmen über Sachverhalte (als Referenzmodelle) und zur Ermöglichung eines systematischen theoretischen Umgangs mit Teilaspekten komplexer Sachverhalte (hinsichtlich der Spezifikation von Systemen). Bei Modellen findet immer eine Abstraktion der Sachverhalte statt, die nicht dem Zweck des Modells dienlich sind. Der Inhalt eines Modells ist ein so weit wie möglich vereinfachter Sachverhalt, ohne dass wesentliche Gegebenheiten verloren gehen. Die Inhalte, der Grad der Formalität und die Darstellung der Modelle beeinflussen deren Nutzen. So eignen sich formale bzw. mathematische Modelle besonders zur Verarbeitung mit Software-Werkzeugen, da sie präzise interpretierbar sind. Informelle Modelle können hingegen einem besseren Überblick bzw. Verständnis dienen. In dieser Arbeit sind die Inhalte der Modelle (angelehnt an [Lev95], S. 306):

- *Struktur*: die (statischen) Zwischenbeziehungen der Elemente entlang einiger Dimensionen (wie Raum, Zeit, relative Wichtigkeit oder logischen bzw. entscheidenden Eigenschaften).
- *Unterscheidende Qualitäten*: eine qualitative Beschreibung bestimmter Variablen, Parametern und Eigenschaften, die das System charakterisieren und ähnliche Strukturen unterscheiden (z.B. nichtfunktionale Anforderungen).
- *Verhalten / Dynamik*: eine Beschreibung der Parameter und Variablen in Verbindung mit den unterscheidenden Qualitäten bezüglich des Wertebereichs, des Zeitverlaufs und dem Grad der Gewissheit in relevanten Situationen, also entlang der Dimensionen Wertebereich, Wahrscheinlichkeit und Zeit.

Kapitelübersicht

In Abschnitt 2 werden mit informellen Referenzmodellen die in dieser Arbeit getroffenen Annahmen über den Aufbau (Struktur) und die Eigenschaften (unterscheidende Qualitäten) eingebetteter Systeme beschrieben. Ziel ist es, eine Systemvorstellung aufzubauen, in die die in Abschnitt 3 vorgestellten Modellarten eingeordnet werden können. Auch werden anhand dieser Referenzmodelle die Schwerpunkte der zu behandelnden Inhalte bei der System- und Fehlermodellierung abgeleitet.

Der darauf folgende Abschnitt 3 beschreibt, wie Modelle des Verhaltens eingebetteter Systemen und deren Umgebung aussehen können und wie Modelle der Artefakte aussehen können, auf denen das Verhalten implementiert ist.

Die Verhaltensmodelle können mit den Modellierungstechniken aus Abschnitt 4 spezifiziert werden. Im Wesentlichen werden verschiedene Darstellungen der Verhaltensmodelle aufgeführt, die spezifisch auf die Beschreibung bestimmter Aspekte der Modelle zugeschnitten sind. Hierzu gehören Prädikate und Automaten (, so wie deren Sonderformen 'aufgelöste' Gleichungen und Betriebsmodi-Automaten). Es wird skizziert, wie diese Modellierungstechniken als Grundlage zur Beschreibung von Fehlverhalten herangezogen werden können.

Die Modelle werden in einem Entwicklungsprozess systematisch und schrittweise erstellt. Bei diesem Vorgehen werden Modelle in präzisere Modelle überführt und Modelle von Teilen des Systems zusammengeführt. Die Abhängigkeiten zwischen diesen Modellen werden in Abschnitt 5 beschrieben.

[1] Eingebettete Systeme sind Rechnersysteme, die in mechatronische Gesamtsysteme eingebaut sind. [FGP04]

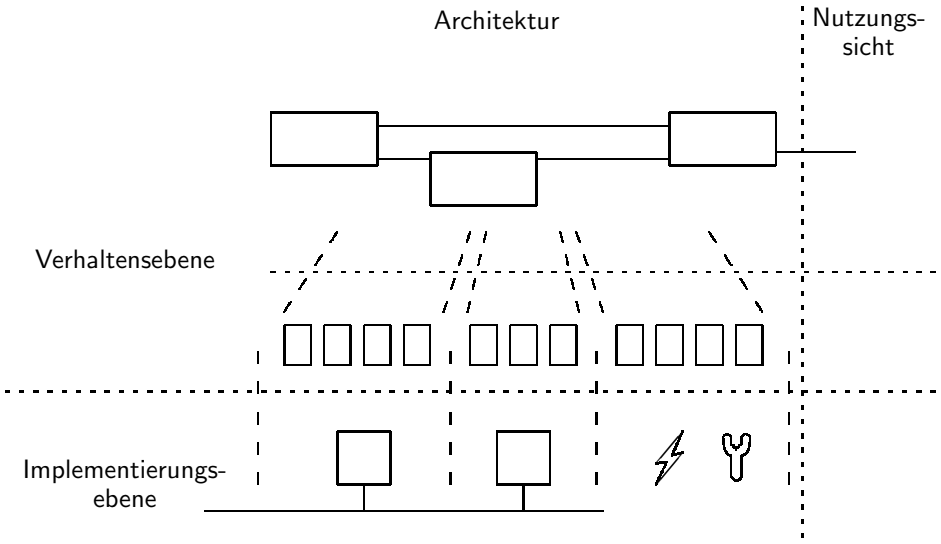


Abbildung 1: Abstraktionsschichten bei mechatronischen Systemen

Anhand dieser Abhängigkeiten werden Ansatzpunkte gegeben, um potentielle Fehler zu identifizieren, die aus Abweichungen dieser Abhängigkeiten entstehen.

Abschnitt 6 fasst schließlich die Ansatzpunkte zur Fehleridentifikation und -modellierung, die sich aus den vorhergehenden Abschnitten ergeben, zusammen.

2 Referenzmodelle

In diesem Abschnitt werden Systemvorstellungen präsentiert, die einen Überblick über die Modellierung eingebetteter Systeme geben. Sie stellen dar, welche Eigenschaften abstrahiert werden und welche Auswirkungen die Abstraktion auf die Modellierung hat.

Verhaltensebene

Ein *mechatronisches System* setzt sich aus mechanischen, elektrischen[2] und elektronischen[3] Anteilen zusammen. Die Anteile werden zu einem gemeinsamen Ganzen zusammengefügt und erfüllen gemeinsam eine Aufgabe. Der Schwerpunkt der Modellierung und der Funktionssicherheitsanalyse liegt auf den funktionalen Zusammenhängen zwischen den Anteilen des Systems. Unter funktional wird hierbei das Verhalten des Systems verstanden. Bei der Modellierung der Systeme gibt es entsprechend eine Abstraktionsebene, bei der eine Verhaltensextrahierung stattfindet und die restlichen Aspekte des Systems nicht betrachtet werden. Diese Ebene wird *Verhaltensebene* genannt. Abbildung 1 stellt diese funktionale Abstraktionsebene dar, in der die Verhaltensweisen der Bestandteile eines mechatronischen Systems modelliert werden können und für das gesamte System in Zusammenhang gebracht werden können. Mit dem steigenden Anteil an elektronischen Systemen in Fahrzeugen, die immer komplexere Steuerungsaufgaben bewältigen müssen, nimmt die Modellierung der Verhaltensebene einen immer wichtigeren Platz in der Entwicklung ein (siehe [AUT06]).

Domäneneigenschaften

Die Eigenschaften der Funktionen, die das Verhalten der Teile eines Systems beschreiben, sind, entsprechend der Domäne, der sie zugeordnet sind, verschieden. Die *Domäne 'Mechanik'* (dargestellt in Abbildung 1 als Zange) enthält Funktionen, die zeitkontinuierlich sind. Die mathematische Beschreibung der Funktionen ist hierbei durch die physikalischen Eigenschaften der Elemente vorgegeben. Viele Größen sind dort wertkontinuierlich. Einige Größen sind zusätzlich stetig. Zum Beispiel kann sich die Position eines trägen Teils nicht sprunghaft ändern. Auch die der *Domäne 'Elektrik'* (dargestellt als Blitz) zugeordneten Funktionen sind zeitkontinuierlich. Allerdings finden bedingt durch Relais oder andere Schaltelemente wesentlich öfter diskrete Wertsprünge statt. Bei den der *Domäne 'Elektronik'* zugeordneten Funktionen stehen Signale bzw. Informationen im Vordergrund. Diese sind

[2] Der Begriff 'Elektrik' bezieht sich auf die Versorgung mit Energie.
[3] Der Begriff 'Elektronik' bezieht sich hier auf den Signal-/Informationscharakter von elektrischem Strom.

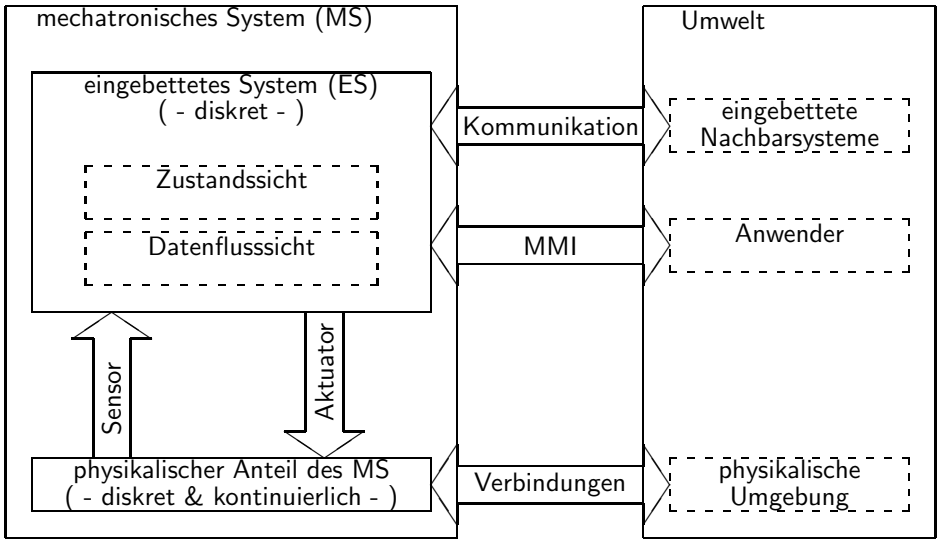


Abbildung 2: System und Umgebung (angelent an [FGP04] und [VDI04])

heutzutage nur noch in den seltensten Fällen wertkontinuierlich. Sie werden durch den Einsatz der Digitaltechnik meist mit diskreten Werten dargestellt. Die diskreten Werte der Signale sind innerhalb der Elektronik nur zu bestimmten Zeitpunkten relevant. Die Funktionen lassen sich so über Komponenten mit diskreten Berechnungszeitpunkten beschreiben (siehe [AD94]). Viele der Systeme sind zusätzlich getaktet, was innerhalb einer Taktung eine zeitdiskrete Beschreibung der Funktionalität mit einheitlichen Zeitabständen ermöglicht.

Verhaltensabstraktionsebenen

Die Elektronik eines mechatronischen Systems ist meist in einer Menge von Steuergeräten untergebracht, die über Bussysteme miteinander kommunizieren. Die Komplexität in den Steuergeräten ist meist so hoch, dass diese wie allgemeine Rechner organisiert sind. Sie haben ein Betriebssystem, eine Kommunikationsschnittstelle und die eigentlichen (Funktions-)Applikationen (siehe [AUT06]). Diese Rechnersysteme mit der Aufgabe der Steuerung physikalischer Prozesse nennt man *eingebettete Systeme* (siehe Abbildung 2). Aufgrund des Aufbaus der Steuergeräte bietet es sich an, verschiedene Abstraktionsebenen für die Funktionalität, also *Verhaltensabstraktionsebenen*, zu definieren. Diese Abstraktionsebenen sind, ähnlich wie in dem ISO/OSI-Schichtenmodell (siehe [ISO94]), durch Zwischenschichten realisiert. Auf der untersten Abstraktionsebene wird (ähnlich der OSI-Schicht-1) die tatsächliche Belegung der einzelnen Signale auf der Hardware betrachtet. Auf dieser Ebene wird zwar alles 'naturgetreu' nachgebildet, die Modelle werden aber schnell unübersichtlich und komplex. Je nach Bedarf lassen sich (analog zum OSI-Referenzmodell) eine Reihe von Zwischenebenen einziehen, bis hin zur Verhaltensebene der Applikation (analog OSI-Schicht-7). Auf dieser Ebene wird die gewünschte Funktionalität modelliert, die das beobachtbare Verhalten des Systems nach außen maßgeblich bestimmt. Das genaue Verhalten auf dieser obersten Ebene hängt von der Realisierung der darunter liegenden Ebenen ab. Es kann aber trotz der Realisierungsabhängigkeit eine angemessene Modellierung des Verhaltens des Systems stattfinden, wenn man davon ausgeht, dass die darunterliegenden Realisierungen der Ebenen das Verhalten nur gering oder nicht beeinflussen (siehe [WFH$^+$06]).

Zusammenhang zwischen Ebenen

Die jeweils höher liegenden Abstraktionsebenen beinhalten implizit die darunter liegenden Ebenen. Auch wenn das Verhalten bzw. die Funktionalität der darunter liegenden Ebene nicht direkt modelliert wird, können die Elemente / Funktionen der jeweils darunter liegenden Ebenen versagen, also ihren Dienst nicht erbringen und auf diese Weise Verhaltensfehler in die jeweils darüber liegende Ebene induzieren. Hat man sich in einem Modell für eine Abstraktionsebene entschieden, so ist es notwendig, die Schnittstelle bzw. Einbindung der darunter liegenden Ebenen, so wie die möglichen Fehler zu kennen, die an den Schnittstellen bzw. durch die abstrahierten Eigenschaften auftreten können. Die Einbindung der tiefer liegenden Abstraktionsebenen kann in einem Modell explizit modelliert werden. Durch die Angabe der Abstraktionsebene kann aber auch implizit eine Abbildungs-/Zugriffsvorschrift angenommen werden, von der auf potenzielle Fehler geschlossen werden kann. Abstrahiert man zum

Beispiel von der Verteilung der Funktionen auf Steuergeräte, so kommen bei der Übertragung der Signale zwischen den Systemen die Standard-CAN-Übertragungsfehler als potenzielle Fehler in Frage.

Nutzungsschnittstelle

Die Beschreibung eines mechatronischen bzw. eingebetteten Systems enthält als wesentlichen Bestandteil die Interaktion mit der Umwelt. Diese Interaktion wird durch das Verhalten des Systems an dessen Systemgrenze bestimmt. Die Grenze zwischen dem System und der Umwelt wird auch als *Nutzungsschnittstelle* bezeichnet. Im Gegensatz zu dem Verhalten des Systems nach außen steht die Architektursicht, die den Aufbau des Systems und das Zusammenwirken der Bestandteile des Systems beschreibt. Wie in Abbildung 1 (rechts senkrecht dargestellt) zu sehen, ist die Nutzungsschnittstelle unabhängig von den Abstraktionsebenen. Beschreiben die Modelle des Systems auf verschiedenen Abstraktionsschichten das gleiche Verhalten des Gesamtsystems, so ist auch die Nutzungsschnittstelle auf beiden Ebenen gleich. Im Verlauf der Entwicklung und mit der Modellierung bzw. Festlegung der Realisierung tiefer liegender Abstraktionsebenen kann eine Verfeinerung oder gar eine Änderung der Nutzungsschnittstelle einhergehen. Ziel ist es, im Laufe der Entwicklung Nutzungsschnittstellenbeschreibungen zu erhalten, die in einer Verfeinerungsbeziehung stehen, die in Richtung der detailliertesten Abstraktionsebene immer feiner wird. Auch die Fehler an der Nutzungsschnittstelle können nach den verschiedenen Abstraktionsebenen gegliedert sein. Jeder Fehler, der an der Nutzungsschnittstelle auftritt, ist Teil einer Funktionssicherheitsanalyse, da hier die Umwelt beeinflusst wird (siehe [Eri05]).

Regelungsversus Interaktionssicht

Betrachtet man die Elektronik als Kernstück der Funktionalität eines mechatronischen Systems, so wird mit dessen Modellierung der Großteil der Modellierung des Systems abgedeckt. Die Logik dieser eingebetteten Rechnersysteme hat die Aufgabe, zwei verschiedenartige Sichten zu verbinden. Aus Sicht der Regelungstechnik dienen die Systeme dazu, physikalische Prozesse zu steuern. Diese Steuerung findet über Funktionen statt, die abhängig von den Eingabewerten einen Ausgabewert berechnen. Sie sind im wesentlichen Diskretisierungen kontinuierlicher Funktionen. Die meisten dieser Funktionen haben, abgesehen von Integralwerten, keine große Menge an Zuständen, die das Verhalten über einen längeren Zeitraum beeinflussen. Die steigende Vernetzung mit anderen Systemen und die steigende Interaktion mit dem Menschen fordern auch die Sicht der ereignisbasierten Systeme, wie z.B. Automaten. Diese Systeme zeichnen sich dadurch aus, auf konkrete Ereignisse von außen zu reagieren und mit einer Menge verschiedener Zustände verschiedene Verhaltensweisen über längere Zeiträume zu zeigen, oder komplizierte algorithmische Aufgaben bewältigen zu können. Der Fokus bei diesen Systemen liegt in der Interaktion bzw. den Abläufen und nicht auf der Berechnung eines Funktionswertes.

Schnittstellen der Systeme

Abbildung 2 stellt die Arten von Systemen dar, die ein eingebettetes System umgeben. In der Regelungssicht werden die Systeme von Sensoren mit Werten beliefert, die physikalische Größen abgreifen. Die Sensoren sind die eingehende Schnittstelle von der kontinuierlichen Sicht zu der diskreten Rechnerwelt. Umgekehrt liefern die eingebetteten Systeme Werte an Aktuatoren, um physikalische Vorgänge zu beeinflussen. Sie sind die ausgehende Schnittstelle zur physikalischen Welt. Die Fehler, die an diesen Schnittstellen auftreten können sind über die Bauart und den physikalischen Wert bzw. dem mechanischen Verhalten des Restsystems bestimmt. An der Mensch-Maschinen-Schnittstelle können die Systeme Nachrichten vom Nutzer bekommen oder Nachrichten an den Nutzer senden. Von den Schnittstellen können systematisch potenzielle Fehler abgeleitet werden. So sind z.B. an der Mensch-Maschine-Schnittstelle (MMI) falsche Eingaben relevant. Bei der Interaktion mit anderen Systemen können zusätzlich spezifische Fehler zur Reihenfolge der Nachrichten identifiziert werden. Folgendes Beispiel skizziert die Schnittstellen eines Systems und daraus ableitbare Fehler:

Beispiel 2.1 (Schnittstellen als Indiz für potenzielle Fehler)
Gegeben sei ein *Lenksystem* eines Fahrzeugs mit einer *Überlagerungslenkung*. Eine Überlagerungslenkung berechnet abhängig von der Fahrsituation einen Überlagerungswinkel und addiert diesen zu dem Lenkradwinkel. Sie hat zwei Teilfunktionen: die *geschwindigkeitsabhängige Lenkübersetzung* und die *Stabilisierungsfunktion* (Gegenlenken bei instabilem Fahrzustand).

Über Sensoren bekommt die Überlagerungslenkung die aktuellen Werte der Gierrate[4], der Querbeschleunigung, des Lenkradwinkels und der Radgeschwindigkeiten. Mögliche ableitbare Fehler von dieser Schnittstelle sind *Sensorfehler* (z.B. Justierfehler, Sensorausfall).

Die Überlagerungslenkung (UL) interagiert mit der Raddrehzahlsteuerung (RDS).[5] Deren Stabilisierungsfunktion (Bremsen der Räder bei instabilem Fahrzustand) kann mit Hilfe der Überlagerungs-

[4] Die Gierrate ist die Drehung des Fahrzeugs um die Hochachse.
[5] Wird häufig auch als ESP bezeichnet.

lenkung teilweise später eingreifen. Der Fahrkomfort wird so erhöht, da ein Bremseingriff weniger komfortabel ist, als ein Lenkeingriff. Hierzu ist ein Austausch von Nachrichten notwendig, bei dem beide Systeme jeweils den Zustand des anderen Systems kennen müssen, um richtig zu handeln. Mögliche ableitbare Fehler von dieser Schnittstelle sind *Nachrichtenfehler* (z.B. falsche Nachricht, verspätete Nachricht).

Des Weiteren kann der Fahrer die Stabilisierungsfunktion der Überlagerungslenkung (und der Raddrehzahlsteuerung) mit einem Taster aktivieren, bzw. deaktivieren. Mögliche ableitbare Fehler sind *diskrete Sensorfehler* (z.B. Signal des Tasters ohne Berührung, keine Reaktion des Tasters bei Berührung). □

3 Modellierung der Systeme

Eigenschaften der Modelle

Eine Grundlage einer Funktionssicherheitsanalyse ist die Systemdefinition (siehe [Lev95]). Um eine automatische Verarbeitung dieser Systemdefinition zu ermöglichen, findet die Definition mit Modellen statt, die symbolisch und statisch sind. Es werden in dieser Arbeit Modelle eines Systems betrachtet, die für eine Funktionssicherheitsanalyse relevant sind. Die Modelle unterscheiden sich abhängig von der Art der Abstraktion, der Art des beschriebenen Systems und dem Zustand des beschriebenen Objekts. Besonders der Grad des Determinismus, wie auch die Kontinuität sind unterscheidende Eigenschaften (bzw. Qualitäten) der Modellelemente. Die Eigenschaften der verwendeten Modelle werden im Folgenden erläutert (nach [Lev95], S. 305):

- *symbolische Modelle*: Modelle, welche die strukturellen Eigenschaften eines Systems mit Eigenschaftsbeschreibungen und logischen Aussagen repräsentieren. Sie nutzen mathematische oder logische Operationen um das Verhalten des Originalsystems vorherzusagen.
- *statische Modelle*: Modelle, die ihre Merkmale zeitunabhängig beibehalten und nicht ändern.
- *deterministische Modelle*: Modelle, die auf gleiche Eingabefolgen oder Reize bei gleichen Randbedingungen immer das gleiche Ergebnis liefern.
- *unscharfe Modelle*: Modelle, die sich bei diskreten Funktionen deterministisch verhalten, bei kontinuierlichen Funktionen jedoch Abweichungen in einem Toleranzbereich zulassen.

Grad des Determinismus

Die meisten der Modelle eingebetteter Systeme sind deterministisch. Bei der Beschreibung physikalischer Vorgänge kann aufgrund der Realisierung (Bauteileungenauigkeit, nicht modellierte Umwelteinflüsse ...) eine Unschärfe angegeben werden, innerhalb derer der tatsächliche Funktionswert von dem ideal beschriebenen Wert abweichen kann. Diese Unschärfe kann zu Nichtdeterminismus an den Schnittstellen der diskreten Systeme zu den kontinuierlichen Systemen führen. Die Umwelt hingegen wird meist mit nichtdeterministischen Modellen dargestellt, da hier oft beliebige Verhaltensweisen angenommen werden müssen. Ebenso kann bei der Beschreibung der Fehler die Verwendung nichtdeterministischer Modelle notwendig sein, da z.B. ein lockerer Kontakt zu einer beliebigen Übertragung oder Nicht-Übertragung eines Signals führen kann.

Abschnitts-Übersicht

Dieser Abschnitt beschreibt, wie das Verhalten (bzw. die Funktionalität) eines Systems modelliert werden kann und wie die Artefakte modelliert werden können, auf denen das Verhalten abgelegt ist. Die Verhaltensmodelle sind hierbei mathematische Modelle, die aus Relationen zwischen Eingaben und Ausgaben bestehen. Die Implementierungsmodelle haben ihren Schwerpunkt auf der Struktur und werden als Instanzen von UML-Klassen-Diagrammen modelliert.

3.1 Modellierung der Verhaltensebene

Dieser Abschnitt beschreibt ein mathematisches Konzept zur Verhaltensmodellierung von Systemen. Es wird allgemein beschrieben, wie mit Relationen das Verhalten modelliert werden kann. Dieses Konzept ist die Grundlage für Modellierungstechniken, die in Abschnitt 4 zur Spezifikation von Systemen genutzt werden können.

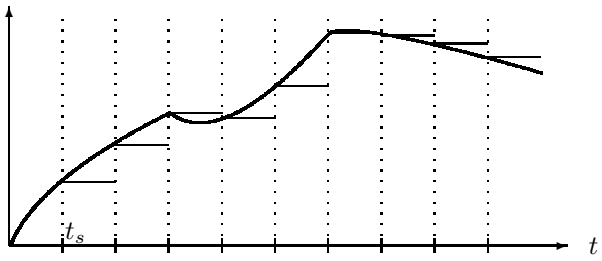


Abbildung 3: Diskretisierung einer kontinuierlichen Funktion

Systeme

Systeme werden auf funktionaler Ebene über ihre Schnittstelle und über ihr Verhalten definiert. Die *Schnittstelle*, über die ein System S mit seiner Umwelt kommuniziert, wird über die (Namen der) Kanäle, deren Datentypen und deren Charakterisierung als Eingabe- oder Ausgabekanal definiert.[6] Die *exemplarische Kommunikation* wird an jedem Kanal f über eine Funktion $f_i(t)$ dargestellt, welche den Wert, der an einem Kanal zu einer bestimmten Zeit t anliegt, angibt (Der Bezeichner f ist hierbei durch den Kanalnamen zu ersetzen und der Wert i identifiziert den exemplarischen Ablauf). Die Menge aller möglichen Funktionen an einem Kanal wird $\mathcal{F}_f = \bigcup f_i(t)$ genannt. Die Menge $\mathcal{F}_f$ stellt den *dynamischen Typ* eines Kanals dar (siehe [LSV95]). Die Definitionsmenge D_f der kanalwertbeschreibenden Funktionen ist bei kontinuierlichen Systemen die Menge der positiven reellen Zahlen einschließlich der Null $\mathbb{R}_0^+$. Die Zeit schreitet also konstant ab einem bestimmten Startzeitpunkt fort. Die Wertemenge aller möglichen exemplarischen Funktionen eines Kanals ist $W_{\mathcal{F}_f} = \bigcup W_{f_i(t)}$, also die Vereinigung der Wertemenge der einzelnen exemplarischen Funktionen. Diese Wertemenge wird auch als (statischer) *Datentyp* des Kanals bezeichnet, der die möglichen Belegungen des Kanals zeitunabhängig darstellt. Das *Verhalten* eines Systems wird über eine Relation $R_S \subseteq \mathcal{F}_f \times \mathcal{F}_g \times \ldots$ dargestellt, in der alle möglichen Kombinationen exemplarischer Eingabe- und Ausgabekommunikationen in Beziehung gebracht werden.

Kommunikation

Ein System kann entweder atomar sein oder aus mehreren Untersystemen (auch Komponenten genannt[7]) zusammengesetzt sein. Die Systeme werden über *Kommunikationskanäle* verbunden, das heißt, die Werte eines Ausgabekanals eines Systems werden als Eingaben eines anderen Systems genutzt. Kanäle, die nicht mit anderen Systemen verbunden sind, treten als Kanäle des übergeordneten Systems auf. Die *Kommunikation* zwischen Systemen findet ausschließlich über den Austausch von Funktionswerten über die Kanäle statt.

Zeit-Diskretisierung

Die Modellierung mechatronischer Systeme fordert die Berücksichtigung zweier verschiedener Aspekte. Zum einen wird die Mechanik und Elektrik modelliert. Hier stellen die Ein- und Ausgaben der Systeme Werte physikalischer Größen dar, die zu jedem Zeitpunkt einen bestimmten Wert haben. Zum anderen betrachtet man eingebettete Systeme, die auf diskreten Steuergeräten abgelegt sind. Die Werte an den Kanälen sind für die Logik dieser Systeme nur an diskreten Zeitpunkten relevant. Die physikalische Funktionalität der Steuergeräte wird hierbei abstrahiert, in dem für die kanalbeschreibenden Funktionen Konstanten verwendet werden, die sich zu den jeweiligen Taktzyklen ändern können (siehe [LSV01]). Auf der Verhaltensebene (siehe Abschnitt 2) ist für diese Systeme die Definitionsmenge D_f der Funktionen, die die Kanalwerte darstellen, diskret. Sie entspricht der Vielfachenmenge $\mathbb{V}_0^+(t_s)$, wobei t_s eine Taktlänge ist (siehe Abbildung 3). Die Folge der Kanalwerte zu den diskreten Zeitpunkten, also die Funktion f_i kann als ein Strom x von Werten dargestellt werden (siehe Definition unten aus [Bre01], [BS01]). Auf der Verhaltensebene haben in dieser Arbeit alle diskreten Systeme eine systemweit einheitliche (schnellste) Taktung mit einer systemweit einheitlichen Zeit. Eine langsamere Taktung muss für die Systeme explizit modelliert werden. Auf das Zusammenspiel zwischen kontinuierlichen Funktionen und zeitdiskreten Funktionen wird in Abschnitt 4.5 weiter eingegangen.

[6] In einigen Arbeiten wie z.B. [LSV01] werden diese Kanäle auch als externe (Kommunikations-) Variablen bezeichnet.

[7] Der Begriff 'System' wird in dieser Arbeit als Synonym für den Begriff 'Komponente' verwendet. Im Sprachgebrauch sind Komponenten meist Teile eines Gesamtsystems.

Definition 3.1 (Strom)
Sei M eine Menge von Werten. Ein *Strom* x über M ist eine endliche oder unendliche Sequenz von Elementen aus M.

Die Menge der Ströme wird im endlichen Fall mit M^*, im unendlichen Fall mit M^∞ bezeichnet. Die Menge aller Ströme über M ist $M^\omega = M^* \cup M^\infty$.

Der Ausdruck $\langle d_1, d_2, \dots d_n \rangle$ bezeichnet einen endlichen Strom der Länge n mit $d_1, d_2, \dots d_n \in M$, d_1 als ersten und d_n als letzten Element. $\langle\rangle$ ist der leere Strom. □

Stromverarbeitung

Die diskreten Zeit- und Werteigenschaften der Kanäle und die damit verbundene Interpretation der Belegungen als Nachrichten betonen besonders den Aspekt der Interaktion zwischen den Systemen und den Aspekt der Verarbeitung der Nachrichten. Es bieten sich zusätzlich zu den mathematischen Operatoren $+, -, *, /, ...$ eine Menge stromverarbeitender Operationen an, um mit den Nachrichten umzugehen (siehe [Bre01], [BS01]).

Definition 3.2 (Stromverarbeitende Operatoren)
$\#x$ bezeichnet die Länge des Stroms x. Für unendliche Ströme x gilt: $\#x = \infty$.

$ft.x$ bezeichnet das erste Element eines nicht leeren Stroms x. $rt.x$ bezeichnet den Strom x ohne sein erstes Element. $lt.x$ bezeichnet das letzte Element eines nicht leeren Stroms x.

$x.t$ bezeichnet den t-ten Wert eines Stroms x falls $\#x \geq t$.

$x \frown y$ bezeichnet die Konkatenation von Strömen. Für $\#x = \infty$ gilt: $x \frown y = x$.

Bezogen auf eine exemplarische Kanalfunktion kann f_i mit einem Strom x beschrieben werden, wobei der Wert $f_i(t)$ dem Wert $x.t$ entspricht. □

In [BDD+92] finden sich weitere ausführliche Beschreibungen von Operatoren und Definitionen. Da die Beschreibung der diskreten Systeme nicht im Fokus dieser Arbeit steht, wird hier auf die Quelle [BS01] verwiesen. Die Darstellung der hier genannten Operationen dient der Verdeutlichung des Charakters der diskreten Kommunikation der Systeme.

Formeln

Die Beschreibung der Relationen, die das Verhalten eines Systems anhand der zugehörigen Ein- und Ausgabefunktionen repräsentieren, soll mit Formeln realisiert werden. Hierzu ist es notwendig, die dort verwendeten Begriffe zu definieren:

Definition 3.3 (Formel & Term)
Die Menge aller Variablen wird mit VAR benannt. Jeder Variable $v \in VAR$ wird ein Typ zugewiesen, der den maximalen Wertebereich beschreibt.

Eine *Belegung* α weist jeder Variablen einen Wert $\alpha.v$ zu. VAL bezeichnet die Menge aller Belegungen.

Zwei Belegungen stimmen für eine Variablenmenge $V \subset VAR$ überein, wenn sie allen Variablen der Menge V den gleichen Wert geben:
$\alpha \overset{V}{=} \beta \overset{def}{=} \forall v \in V : \alpha.v = \beta.v$

Ein Term τ ist ein Ausdruck, der sich mit einer Belegung α zu einem Wert aus der Wertemenge W_τ des Terms auswerten lässt.

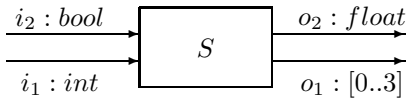
Eine Formel Φ ist ein Ausdruck (und Sonderfall eines Terms), der sich mit einer Belegung α zu einem boolschen Wert $\{true, false\}$ auswerten lässt.

Die Auswertung einer Belegung α einer Formel Φ zu *true* wird als $\alpha \vDash \Phi$ notiert.

$\Phi[w/v]$ beschreibt die Formel Φ, in der alle Vorkommen der freien Variable $v \in VAR$ durch die Variable w ersetzt werden.

$\Phi[\Phi_j/\Phi_i]$ beschreibt für eine Formel der Form $\Phi = (\boldsymbol{\Phi_1 \wedge \Phi_2 \wedge ...})$, die Formel, in der alle Vorkommen der Teilformel Φ_i durch die Teilformel Φ_j ersetzt werden.

$free(\Phi)$ bezeichnet die Menge der freien Variablen in Φ. □



Name	DatenTyp	Art
i_1	int	input
i_2	bool	input
o_1	[0 .. 3]	output
o_2	float	output

Abbildung 4: Darstellung der Schnittstelle eines Systems

Bindung

Mit Hilfe dieser Definitionen lässt sich eine Formel Φ an ein System S binden in dem in der Formel die freien Variablen v_i durch die Kanäle $f, g, \ldots$ ersetzt werden, also $\Phi[f/v_1][g/v_2]\ldots$ gilt. Als Belegungen der Variablen können dann entweder die Wertverläufe oder die aktuellen Werte der Kanäle des Systems verwendet werden. Eine genaue Beschreibung der Bindung der Formeln an ein System ist in Abschnitt 4.1 zu finden.

Definition 3.4 (Schnittstelle)
Eine Schnittstelle eines Systems S ist ein Tupel (I_S, O_S), wobei die endliche Menge $I_S \subseteq VAR$ die Eingabe- und die endliche Menge $O_S \subseteq VAR$ die Ausgabekanäle des Systems S sind. □

Die in dieser Arbeit betrachteten Schnittstellen eines Systems sind statisch, d.h. eine Schnittstelle ändert sich zur Laufzeit nicht. Systeme mit Schnittstellen, die sich zur Laufzeit ändern, werden u.a. in [GSB97] betrachtet. Eine Darstellung von Schnittstellen ist in in Abbildung 4 gegeben.

Definition 3.5 (Verhaltensrelation)
Eine Relation R_S eines Systems S welche n-Tupel zwischen den exemplarischen Eingabe- und Ausgabekommunikationen enthält. Es gilt:
$R_S \subseteq \mathcal{F}_{I_S} \times \mathcal{F}_{O_S}$

Hierbei entspricht $\mathcal{F}_{I_S}$ dem Tupel der dynamischen Datentypen der Eingabekanäle, also $\mathcal{F}_{I_S} = \mathcal{F}_f \times \mathcal{F}_g \times \ldots$ mit $f, g, \ldots \in I_S$ und $\mathcal{F}_{O_S}$ dem der Ausgabekanäle.

Die Menge aller Verhalten zu einem System S ist:
$REL_S = \mathcal{P}(\mathcal{F}_{I_S} \times \mathcal{F}_{O_S})$.

$dom.R_S$ ('Domain R') ist die Menge der Eingabe-Tupel, die in der Relation R_S enthalten sind. (Die Relation R_S kann eine Teilmenge von $\mathcal{F}_{I_S} \times \mathcal{F}_{O_S}$ sein).

$rng.R_S$ ('Range R') ist die Menge der Ausgabe-Tupel, die in der Relation R_S enthalten sind. □

Das Verhalten eines Systems S wird durch eine Verhaltensrelation R_S definiert. Die Relation R_S ist im unendlichen und im kontinuierlichen Fall nur durch Hilfsmittel wie Formeln beschreibbar. Unabhängig von der Beschreibbarkeit ist die Nutzung der Formeln üblich, da die Beschreibungen deutlich effizienter und lesbarer durchgeführt werden können. Die Beschreibung mit Formel der Relation R_S mit Formeln ist in Abschnitt 4 erklärt.

Realisierbarkeit

Prinzipiell lassen sich mit diesen Relationen auch Verhaltensweisen beschreiben, die von Systemen nicht realisierbar sind. So können z.B. Verhalten beschrieben werden, bei denen die Systeme auf Eingaben reagieren, die in der Zukunft stattfinden. Ein reales System kann aber nur auf Eingaben reagieren, die bereits stattgefunden haben. In [BS01] wird gezeigt, dass für ein System die Realisierbarkeit im Wesentlichen nur durch eine fehlende Kausalität und bzw. oder durch eine Verletzung der Linkstotalität nicht sichergestellt werden kann. Ein linkstotales System zeigt für jede Eingabe der Umgebung eine Reaktion. Bei einem kausal monotonen System kann ein ausgegebener Wert nicht mehr zurückgenommen werden. Bei diskreten Systemen haben zwei Eingabeströme mit dem selben Präfix auch zwei Ausgabeströme mit dem selben Präfix zur Folge.

Definition 3.6 (kausale Monotonie)
Ein System S ist *kausal monoton*, wenn eine Verlängerung der Eingabe i zu i' immer zur Folge hat, dass die Ausgabe entweder unverändert bleibt, oder sich auch verlängert. Es gilt:
$\forall i, i' \in \mathcal{F}_{I_S}, o, o' \in \mathcal{F}_{O_S} : ((i, o) \in R_S \wedge i \sqsubseteq i' \wedge (i', o') \in R_S) \Rightarrow o \sqsubseteq o'$ □

Definition 3.7 (Linkstotalität)
Ein System S ist *linkstotal*, wenn die verhaltensbeschreibende Relation R_S für jede mögliche Eingabe i eine Ausgabe o aufweisen kann. Es gilt:
$\forall i \in \mathcal{F}_{I_S} : \exists o \in \mathcal{F}_{O_S} : (i, o) \in R_S$ oder kurz: $dom.R_S = \mathcal{F}_{I_S}$ □

Diese Eigenschaften von Systemen sind in vielen Beschreibungstechniken implizit vorhanden oder werden von Werkzeugen explizit gefordert. So ist z. B. in dem Programm Simulink, wie auch im Programm Stateflow diese Eigenschaft implizit vorhanden, denn die Ausgaben können nur unabhängig von den Eingaben, oder in Abhängigkeit der bereits vorhandenen Eingaben stattfinden.

Laufzeitbedingungen

Die Beschreibung des Verhaltens eines Systems ist von der Umwelt unabhängig. Ein System wird in eine Systemlandschaft integriert und es treten in den meisten Fällen nicht alle Eingabekombinationen auf, die für das System über die Schnittstellen möglich wären. Ein Hilfsmittel bei der Spezifikation eines Systems für einen konkreten Einsatz in einer Systemumgebung ist die Angabe von Bedingungen, die im Einsatz erfüllt sein müssen. Mit dieser Angabe von Bedingungen wird die Menge der Elemente der verhaltensbeschreibenden Relation so reduziert, dass das System nur noch für die in dem konkreten Einsatz vorkommenden Pfade linkstotal ist. Diese Bedingungen werden *Laufzeitbedingungen* genannt. Dieses bedingte und somit tatsächlich genutzte Verhalten wird als R_S^{assert} bezeichnet, mit $R_S^{assert} \subseteq R_S$. Ein Beispiel zu Laufzeitbedingungen wird in Abschnitt 4.1 gezeigt.

3.2 Modellierung der Implementierungsstruktur

Zweck

Dieser Abschnitt beschreibt, wie Modelle der Struktur der Artefakte aussehen, auf denen das im vorherigen Abschnitt 3.1 spezifizierte Verhalten implementiert ist. Diese Modelle dienen nicht der Modellierung des Verhaltens. Sie dienen der Zuordnung spezifischer Eigenschaften der Implementierungselemente zu den Verhaltens-Spezifikationen. Schwerpunkte bei diesen Modellen sind die Struktur zwischen den Elementen und die Attribute der Elemente. Von diesen Strukturen und Attributen können in weiteren Arbeiten potenzielle Fehler und Fehlverhalten systematisch abgeleitet werden. So kann zum Beispiel ein CAN-Bus bis zu einem gewissen Grad eine Abschirmung gegen elektromagnetische Strahlungen haben. Ein davon abgeleiteter Fehler wäre die elektromagnetische Störung, die keine Übertragung zulässt.

Klassifizierung

Die Implementierungsmodelle sind entsprechend dem Design der Systeme verschieden, können aber anhand gemeinsamer Eigenschaften klassifiziert werden. Die gemeinsamen Eigenschaften der Klassen werden in dieser Arbeit mit Klassendiagrammen der Unified Modelling Language (UML) definiert (siehe [Fow03], [ISO05]). Ein Implementierungsmodell ist dann eine Instanz (UML-Objektmodell) des jeweiligen Klassendiagramms. Hierbei ist $\mathbb{C}$ die Menge der Klassen, die $\mathbb{I}$ die Menge der Objekte (bzw. Instanzen), $CI \subseteq \mathbb{C} \times \mathbb{I}$ die Relation zwischen den Klassen und den Instanzen und $II \subseteq \mathbb{I} \times \mathbb{I}$ die Relation zwischen den Instanzen miteinander verbundenen Objekten.

Faktoren für die Klassifizierung

Die Klassen der Implementierungsmodelle sind spezifisch auf die jeweiligen Verhaltensmodelle ausgerichtet. Sie hängen von der Abstraktionsebene des Verhaltensmodells ab, von der Domäne, die modelliert wird, von dem Grad der Zerlegung der Funktionsarchitektur, von dem Architekturkonzept und von den konkreten physikalischen Eigenschaften. Im Folgenden werden die beeinflussenden Faktoren anhand von Beispielen verdeutlicht:

- *Abstraktionsebene des Verhaltensmodells:* Ein Verhaltensmodell, welches die Datentypen abstrahiert, hat ein Implementierungsmodell, bei dem eine Verhaltensgetreue Umsetzung der Datentypen eine Eigenschaft ist. Ist hingegen in dem Verhaltensmodell der Datentyp bereits exakt vorgegeben, so ist diese Eigenschaft im Implementierungsmodell nicht gegeben.
- *modellierte Domäne:* Das Verhalten der Domäne Elektronik wird in elektronischen Bauteilen implementiert. Entsprechend wird das Verhalten anderer Domänen in Bauteilen anderer Domänen implementiert. Die Domäne beeinflusst so die Artefakte, die modelliert werden.

- *Granularität der Komponenten:* Entsprechen die kleinsten Systeme des Verhaltensmodells der Elektronik, so ist es nicht notwendig, im Implementierungsmodell die Struktur der Steuergeräte und Busse darzustellen. Sind hingegen die kleinsten Systeme einzelne Steuergeräte, so ist deren Verbindung über Busse relevant.

- *Architekturkonzept:* Das Verhalten der Systeme kann mit verschiedenen Architekturkonzepten implementiert werden. Ein Beispiel ist eine Implementierung der Elektronik nach dem AUTOSAR-Konzept [AUT06] oder nach dem in Abbildung 5 aus einem Projekt erhobenen Modell.

- *konkrete physikalische Eigenschaften:* Findet die Übertragung von Daten in einem Bus-System elektrisch statt, so spielt bei den Leitungen die Empfindlichkeit gegenüber elektromagnetischer Strahlung eine Rolle. Bei optischer Übertragung hingegen hat die elektromagnetische Strahlung keinen Einfluss auf die Leitungen.

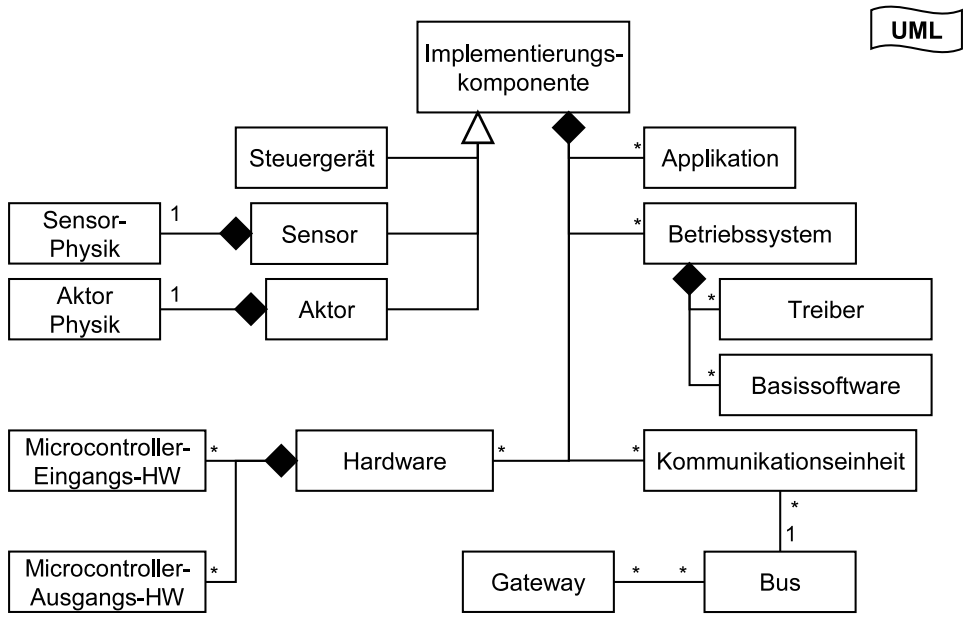


Abbildung 5: Beispiel eines Implementierungs-Klassendiagramms (vgl. [BSN07])

Abbildung Verhalten zur Implementierung

Die Elemente der Verhaltensmodelle werden denen der Implementierungselemente zugeordnet. Sei $\mathbb{S}$ die Menge der Systeme der Verhaltensmodelle und $\mathbb{I}$ die Menge der Implementierungselemente. Die Zuordnung der Systeme der Verhaltensmodelle zu den Implementierungselementen ist eine Relation $SI \subseteq \mathbb{S} \times \mathbb{I}$. Ein Implementierungsmodell ist für ein Verhaltensmodell geeignet, wenn jedes Blatt der Systemhierarchie einem Implementierungselement zugewiesen werden kann. Jedem System kann so eine Menge potenzieller Eigenschaften und somit potenzieller Fehler zugewiesen werden.

Eigenschaften der Elemente

Den Implementierungselementen werden Eigenschaften zugewiesen, anhand derer potenzielle Fehler abgeleitet werden können. Diese Eigenschaften sind Qualitätseigenschaften. Diese umfassen im Groben: Funktionalität, Zuverlässigkeit, Benutzbarkeit, Effizienz, Änderbarkeit und Übertragbarkeit. Eine detailliertere Referenzauflistung dieser Eigenschaften ist in der Norm ISO 9126 ([ISO01]) zu finden. Betrachtet man zum Beispiel ein Steuergerät, so gehört zur Zuverlässigkeit unter anderem eine gewisse Robustheit gegen thermische Belastung, elektromagnetische Strahlung und mechanische Erschütterungen.

Zusammenfassung

Zusammenfassend enthalten die Implementierungsmodelle die Struktur zwischen den Implementierungselementen und die qualitativen Eigenschaften dieser Elemente. Auf eine genauere Beschreibung der Ausprägungen der Implementierungsmodelle wird hier verzichtet, da der Schwerpunkt dieser Arbeit auf der Analyse des Verhaltens liegt. Grundlegend können von diesen Modellen potenzielle Fehler

anhand der qualitativen Eigenschaften abgeleitet werden, deren Wirkung entlang der Struktur ermittelt werden und schließlich deren Erscheinen im Verhaltensmodell über die Abbildung *SI* verfolgt werden.

4 Modellierungstechniken

Spezifikation des Verhaltens

Abschnitt 3.1 hat gezeigt, wie mit Relationen zwischen kanalbeschreibenden Funktionen das Verhalten von Systemen modelliert werden kann. Dieser Abschnitt widmet sich Modellierungstechniken, mit denen diese Relationen spezifiziert werden können. Die Techniken sind teilweise auf die zu modellierenden Aspekte zugeschnitten. Ausgehend von den allgemeinen Spezifikationen mit Formeln werden darauf aufbauend Sonderformen der Formeln betrachtet. Für die Regelungssicht werden zu den Ausgabekanälen aufgelöste Gleichungen vorgeschlagen. Für die Interaktionssicht werden Automaten vorgeschlagen. Diese beiden Sichten können mit Betriebsmodi-Automaten kombiniert werden. Letztlich wird diskutiert, ob und wie mit diesen Modellierungstechniken die komplexe physikalische Umgebung der Systeme geeignet abstrahiert werden kann.

4.1 Blackbox Spezifikationen

Prädikate

Eine *Black-Box Spezifikation* beschreibt das Verhalten eines Systems S mit einem Prädikat. Ein *Prädikat* ist eine Formel Φ, deren freien Variablen Eingabe- und Ausgabekanäle des Systems S sind und genau die Tupel exemplarischer Eingaben und Ausgaben von S in R_S sind, für die Φ wahr wird.

Bindung der Variablen

Die Prädikate nutzen Variablen zur Charakterisierung der Tupel. Hier gibt es zwei Möglichkeiten bei der Bindung der Variablen an die Kanäle. Die erste Möglichkeit ist, die Variablen der Formeln mit exemplarischen Kommunikationen an den jeweiligen Kanälen zu belegen. Diese Art der Blackbox-Spezifikation mit stromgebundenen Formeln ist in [Bre01] ausgeführt. Es gibt verschiedene Formeln, die das Verhalten R_S beschreiben. $PRED_S^{Stream}$ ist die Menge aller Formeln, die das Verhalten eines Systems S mit exemplarischen Kommunikationen beschreiben.

Definition 4.1 (stromgebundenes Prädikat)
Eine Formel $[\![S]\!] \in PRED_S^{Stream}$ aus der Menge:
$PRED_S^{Stream} \stackrel{def}{=} \{\Phi | (free(\Phi) \subseteq (I_S \cup O_S)) \wedge (\forall \alpha \in \mathcal{F}_{I_S} \times \mathcal{F}_{O_S} : (\alpha \vDash \Phi) \Leftrightarrow (\alpha \in R_S))\}$
Die Menge der Prädikatenmengen für die Verhaltensweisen R_S von S heißt $PRED_{REL_S}^{Stream}$. □

operative Bindung

Die zweite Möglichkeit der Variablenbindung ist, die freien Variablen der Formeln mit Werten der Kanäle in Bezug zu einem Zeitpunkt t zu belegen. In diesem Fall erfüllt eine Kombination von Strömen $x_1, x_2, ...$ an den jeweiligen Kanälen das Prädikat, wenn zu jedem Zeitpunkt t die Formel für die Werte $x_1.t, x_2.t, ...$ erfüllt ist.

Definition 4.2 (operationelles Prädikat)
Eine Formel $[\![S]\!]^{op} \in \mathsf{PRED}_S^{\mathsf{op}}$ aus der Menge:
$\mathsf{PRED}_S^{\mathsf{op}} \stackrel{def}{=} \{\Phi | (\forall \boldsymbol{t \geq 0 : \Phi[\Phi_{i_1}^t / \Phi_{i_1}] \ldots [\Phi_{i_n}^t / \Phi_{i_n}][\Phi_{o_1}^t / \Phi_{o_1}] \ldots [\Phi_{o_m}^t / \Phi_{o_m}]}) \in \mathsf{PRED}_S^{\mathsf{Stream}})\}$
mit $\{i_1, ...i_n\} = I_S$ und $\{o_1, ...o_m\} = O_S$
und $\forall 1 \leq a \leq n : \Phi_{i_a} = (\boldsymbol{i_a}) \wedge \Phi_{i_a}^t = (\boldsymbol{i_a.t})$ und $\forall 1 \leq a \leq m : \Phi_{o_a} = (\boldsymbol{o_a}) \wedge \Phi_{o_a}^t = (\boldsymbol{o_a.t})$ □

Definition 4.3 (Operationalisierung)
Eine Abbildung von einem operationellen Prädikat Φ auf ein strombasiertes Prädikat:
$\Phi[\uparrow \mathbb{T} \uparrow] :: \mathsf{PRED}_S^{\mathsf{op}} \times \mathcal{P}(\mathbb{N}) \rightarrow \mathsf{PRED}_S^{\mathsf{stream}}$
$\Phi[\uparrow \mathbb{T} \uparrow] \stackrel{def}{=} (\forall \boldsymbol{t \in \mathbb{T} : \Phi[\Phi_{i_1}^t / \Phi_{i_1}] \ldots [\Phi_{i_n}^t / \Phi_{i_n}][\Phi_{o_1}^t / \Phi_{o_1}] \ldots [\Phi_{o_m}^t / \Phi_{o_m}]})$
mit $\{i_1, ...i_n\} = I_S$ und $\{o_1, ...o_m\} = O_S$
und $\forall 1 \leq a \leq n : \Phi_{i_a} = (\boldsymbol{i_a}) \wedge \Phi_{i_a}^t = (\boldsymbol{i_a.t})$ und $\forall 1 \leq a \leq m : \Phi_{o_a} = (\boldsymbol{o_a}) \wedge \Phi_{o_a}^t = (\boldsymbol{o_a.t})$ □

Teilformeln

Ist ein System S mit einer Blackbox-Spezifikation definiert, so kann das gegebene Prädikat $[\![S]\!]$ die *konjunktive Form*
$[\![S]\!] = \bigwedge_{i \in [1..n]} (\Phi_i)$
haben. Die Menge aller Prädikate mit konjunktiver Form zu einem System S ist: $PRED(\bigwedge)^{Stream}_{REL_S}$. Die einzelnen Formeln Φ_i entsprechen formalen Anforderungen oder anderen Systembeschreibungen. Sie können als Teilprädikate verstanden werden, die das System aus Sicht der jeweiligen Formel beschreiben. Diese Zerlegung kann die Verständlichkeit der Spezifikation erhöhen, in dem verständliche Eigenschaften explizit hervorgehoben werden. Teilformeln sind unter anderem dazu geeignet, um sich nur auf eine Teilmenge der Eingabe- und Ausgabekanäle zu beziehen. Eine besondere Form sind Formeln, die sich nur auf gewisse Eingabebedingungen beziehen, also partiell gültig sind. Diese Prädikate werden auch *Assumption / Guarantee-Spezifikationen* genannt und haben die Form $\Phi^{Assumption} \Rightarrow \Phi^{Guarantee}$. Das folgende Beispiel zeigt, wie ein System mit Blackbox-Spezifiakationen definiert werden kann.

Beispiel 4.1 (Black-Box-Spezifikation eines Systems)
Bei abgeschalteter Überlagerungslenkung verhält sich das Lenksystem S wie eine herkömmliche Lenkung. Es gilt dann ein Prädikat $[\![S]\!]^{op} = (\boldsymbol{\Phi_1 \wedge \Phi_2 \wedge \Phi_3})$ mit folgenden Teilformeln:

Die Übersetzung vom Lenkradwinkel (Eingabe i_{lrw}) zum Vorderradwinkel (Ausgabe o_{vrw}) hat im Gültigkeitsbereich $[-12rad..12rad]$ den konstanten Wert 0.05:
$\Phi_3 = (\boldsymbol{(-12rad \leq i_{lrw} \leq 12rad) \Rightarrow (o_{vrw} = 0.05 * i_{lrw})})$

Der Betrag des Vorderradwinkels ist bei allen Lenkradwinkeln, deren Betrag größer als $12rad$ im Bogenmaß ist, kleiner gleich $0.6rad$ (, da das Rad sonst an den Radkasten stößt):
$\Phi_1 = (\boldsymbol{(i_{lrw} < -12rad) \Rightarrow (o_{vrw} \leq -0.6rad)})$
$\Phi_2 = (\boldsymbol{(i_{lrw} > 12rad) \Rightarrow (o_{vrw} \leq 0.6rad)})$ □

Laufzeit-bedingungen

Die Laufzeitbedingungen, die in einem System gelten sollen, werden ebenfalls mit Formeln angegeben. Da sie nicht das Verhalten des Systems beschreiben, sondern Bedingungen an dessen Einsatz, werden sie gesondert betrachtet. Die Formeln $[\![S]\!]_{Assertion}$ werden nur dann betrachtet, wenn für das System eine Konsistenzprüfung in seiner Systemlandschaft gemacht werden soll.

Beispiel 4.2 (Einsatz von Laufzeitbedingungen)
Das in vorherigem Beispiel spezifizierte Lenksystem S mit abgeschalteter Überlagerungslenkung kann wesentlich einfacher spezifiziert werden, wenn man annimmt, dass die Eingaben der Umwelt eingegrenzt sind. Folgende Aussagen sollen gelten.

Die Übersetzung vom Lenkradwinkel zum Vorderradwinkel hat den konstanten Wert 0.05:
$\Phi_1 = (\boldsymbol{o_{vrw} = 0.05 * i_{lrw}})$

Alle Ausführungen münden in einem Vorderradwinkel kleiner / gleich $0.6rad$:
$\Phi_2 = (\boldsymbol{|o_{vrw}| \leq 0.6rad})$

Für ein System S gilt dann $[\![S]\!]^{op} = \Phi_1$ und die Laufzeitbedingung $[\![S]\!]^{op}_{Assertion} = \Phi_2$. □

Bezug zur Zeit

Abschließend wird in diesem Abschnitt ein Operator eingeführt, um mit einem operationellen Prädikat algorithmische Aufgabendefinieren zu können. Ein System greift auf die aktuelle Belegung zu einem Zeitpunkt t, aber auch auf Belegungen der Vergangenheit zu Zeitpunkten $(t - a)$ zurück, wenn es sich diese gemerkt hat. Bei einer Beschreibung eines diskreten Systems S mit einem Prädikat $[\![S]\!]^{op}$ muss dementsprechend die Möglichkeit bestehen, sich auf vergangene Werte zu beziehen. Die Realisierung dieses Zugriffs geschieht üblicherweise über einen $delay()$-Operator. Dieser Operator verhält sich wie ein Puffer (FIFO-Stack), der die eingehenden Werte speichert und nach einer angegebenen Zeit wieder freigibt. Viele Sprachen bieten diesen Operator nur für die Zeit eines Berechnungsschrittes an, wobei durch Verschachtelung der Operatoren längere Verzögerungen formulierbar sind, und fordern die Angabe eines Initialwertes, der die Ausgabe des Operators beim ersten Rechenschritt bestimmt[8]. Formal ist der Operator wie folgend definiert:

[8] Die Sprache Lustre bietet einen 'pre'-Operator, der den Wert eines Ausdrucks im vorhergehenden Berechnungsschritt angibt und einen '→'-Operator zur Initialisierung im ersten Schritt.(siehe [HCRP91])

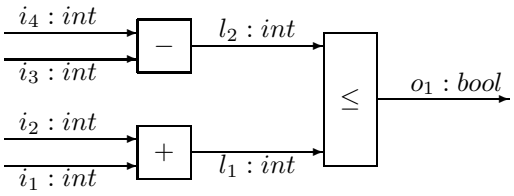


Abbildung 6: Graphische Darstellung einer Funktion

Definition 4.4 (Delay-Operator)
Der *Delay-Operator* zur Referenzierung vergangener Werte ist eine Konkatenation des Initialstroms $\langle a_1, ...a_n \rangle$ mit dem zu betrachtenden Strom x:
$delay_{\langle a_1,...a_n \rangle}(x.t) \stackrel{def}{=} (\langle a_1, ...a_n \rangle \frown x).t$ □

Möchte man verschiedene Prädikate zu bestimmten Systemzeiten gelten lassen, so lässt sich dies mit einem künstlich angelegten Zählerkanal realisieren, der in jedem Rechenschritt inkrementiert wird.

4.2 Spezifikation mit zu Ausgabekanälen aufgelösten Gleichungen

mathematische Formeln

Die Beschreibung mechanischer bzw. elektrischer Systeme, sowie die anfängliche Beschreibung der physikalischen Prozesse, die von eingebetteten Systemen gesteuert werden, finden typischerweise in einer zeit- und wertkontinuierlichen Logik statt. Um ein simulierbares System zu erhalten, ist es notwendig, dass die Beschreibungen berechenbar und für die eingebetteten Systeme deterministisch sind. Die Beschreibung eines simulierbaren Verhaltens dieser Systeme kann mit mathematischen Gleichungen realisiert werden, die zu den jeweiligen ausgabekanalbeschreibenden Variablen aufgelöst sind. Für jeden Ausgabekanal $o_j \in O_S$ entsteht so eine Formel der Form:

Definition 4.5 (aufgelöste Gleichung)
Ein simulierbares Verhalten für eine Ausgabe o_j in einem Prädikat $[\![S]\!]^{op}$ ist gegeben durch eine zu o_j aufgelöste Gleichung der Form:

$\Phi_{o_j} \stackrel{def}{=} (o_j = \tau_{o_j})$ mit $o_j \in O_S$ und $free(\tau_{o_j}) \cap \{o_j\} = \emptyset$.
(Siehe Definition 3.3 zu $free$ und τ.)

Die Menge aller zu einer Ausgabe o_j hin aufgelösten Gleichungen Φ_{o_j} in einem System S ist: $PRED^{op}_{REL\_o_j}$. □

Beispiel 4.3 (Spezifikation mit aufgelösten Gleichungen)
Das Verhalten der geschwindigkeitsabhängigen Lenkübersetzung des Lenksystem S aus Beispiel 2.1 wird folgendermaßen simulierbar festgelegt:

Die Ausgabe liefert es den Wert des Vorderradwinkels o_{vrw} un hängt von der Geschwindigkeit i_v und dem Lenkradwinkel i_{lrw} ab. Das Verhalten wird über ein Ausgabeprädikat definiert, also $[\![S]\!] = \Phi_{o_{vrw}}$. Nach Definition 4.5 hat die Formel damit die Form $\Phi_{o_{vrw}}(o_{vrw} = \tau_{o_{vrw}})$.

Der Term, der die Funktion zur Berechnung des Ausgabewertes beschreibt, ist:
$\tau_{o_{vrw}} = (\boldsymbol{i_{lrw}/(16 * \max(1, (|i_v|/100km/h))})$ □

Anwendbarkeit

Eine vollständige Beschreibung eines Systems anhand aufgelöster Gleichungen erhält man durch eine Konjunktion der jeweils dem Ausgabekanal zugeordneten Formeln, also $[\![S]\!]^{op} = \Phi_{o_1} \wedge \Phi_{o_2} \wedge ... \Phi_{o_n}$. Die Verwendung aufgelöster Gleichungen ist keine Garantie für ein ausführbares System. Die möglichen Fehlformulierungen können aber auf ein überschaubares Maß reduziert werden (z.B. Division durch Null). Die Darstellungen der Gleichungen kann in Textform, aber auch wie in Simulink (siehe Abbildung 6) graphisch realisiert werden. Ebenso kann eine Untergliederung in Teilfunktionen stattfinden, um Redundanzen bei der Beschreibung zu vermindern.

Präzision

Die mathematischen Gleichungen liefern präzise Ergebnisse mit einer unendlichen Genauigkeit. In der Realität finden jedoch immer Abweichungen statt, die in einem Toleranzbereich liegen. Teilweise möchte man gewisse Ungenauigkeiten in den Formeln zulassen, um weiteren Gestaltungsfreiraum bei der Implementierung zu geben. Oft wird eine gewisse Ungenauigkeit der Beschreibungen implizit angenommen. So findet z.B. bei Simulink die Simulation über eine Diskretisierung der Zeit statt, die entsprechende Abweichungen von der Beschreibung mit Formeln mit sich bringt. Mathematische Gleichungen wie in Simulink können also ein ideales Verhalten beschreiben, sind aber für eine Spezifikation zu präzise und müssen aufgeweicht werden. Eine System kann in diesen Fällen mit 'relaxierenden' Formeln beschrieben werden, in denen die Ausgaben mit einem Toleranzbereich versehen werden. Für eine Gleichung kann bei der Spezifikation eine obere und untere Schranke angegeben werden, welche die Toleranzgrenzen explizit beschreiben. Diese Angabe der Grenzen erlaubt es, den Toleranzbereich flexibel zu gestalten. Eine Formel Φ_x wird so zu einer Formel $\Phi_x^{upper} \wedge \Phi_x^{lower}$. Eine weitere Möglichkeit besteht in der Angabe eines Toleranzbereiches, in dem eine relative oder absolute Abweichung a akzeptabel ist. Aus Fallstudien haben sich folgende für diese Arbeit relevanten Toleranzangaben ergeben:

Definition 4.6 (relaxierte Beschreibungen)

Relaxierende Beschreibungen einer Ausgabe o_j in einem Prädikat $[\![S]\!]^{op}$ ist gegeben durch: $\Phi_{o_j}^{\pm a} \stackrel{def}{=}$
$(o_j \stackrel{<\pm a>}{=} \tau_{o_j}) \stackrel{def}{=} (\tau_{o_j} - a \leq o_j \leq \tau_{o_j} + a)$
$\Phi_{o_j}^{\div a} \stackrel{def}{=} (o_j \stackrel{<\div a>}{=} \tau_{o_j}) \stackrel{def}{=} (|\tau_{o_j} - o_j| \leq |\tau_{o_j} * a|)$
$\Phi_{o_j}^{\div a \pm b} \stackrel{def}{=} \Phi_{o_j}^{\div a} \vee \Phi_{o_j}^{\pm b}$

□

Ergänzung der Gleichungen

Die relaxierte Beschreibung des Systems kann um zusätzliche Prädikate ergänzt werden, die eine Anwendbarkeit der konkreten Funktion sicherstellen. Eine Forderung an eine Funktion ist z.B. die Stetigkeit, die für fast alle Steuerungen von Mechaniken notwendig ist. Ebenso kann hinsichtlich der Steigung der Funktion angegeben werden, ob diese differenzierbar sein soll. Ein weiterer Punkt ist die Angabe der Monotonie, ob eine Funktion monoton steigend oder fallend sein soll, bzw. ob die Steigung einer Funktion monoton steigend oder fallend sein soll.

Beispiel 4.4 (Spezifikation mit relaxierten aufgelösten Gleichungen)

Das Lenksystem S mit geschwindigkeitsabhängiger Lenkübersetzung soll nicht zu stark von einem herkömmlichen Lenksystem mit konstanter Lenkübersetzung abweichen. Für das Übersetzungsverhältnis zwischen dem Lenkradwinkel (i_{lrw}) und dem Vorderradwinkel (o_{vrw}) möchte man offen lassen, wie eine konkrete Implementierung der Lenkübersetzung aussieht (damit z.B. jeder Anbieter noch sein spezielles Wissen einbringen kann). Für das Lenksystem S gilt dann (siehe Definition 3.3 zu $free$ und τ und Defnition 4.4 zu $delay$):

$[\![S]\!]^{op}_{Assertion} = (\boldsymbol{|o_{vrw}| \leq 0.6})$ und $[\![S]\!]^{op} = (\boldsymbol{\Phi_{o_{vrw}}^{\pm 0.2} \wedge \Phi_1 \wedge \Phi_2})$

mit:

Verhalten: $\tau_{o_{vrw}} = (\boldsymbol{0.05 * i_{lrw}})$

Differenzbergrenzung: $\Phi_1 = (\boldsymbol{|o_{vrw} - delay_{\langle o_{vrw}.0 \rangle}(o_{vrw})| < 0.0002})$

Monotonie: $\Phi_2 = (\boldsymbol{sign(i_{lrw} - delay_{\langle i_{lrw}.0 \rangle}(i_{lrw})) = sign(o_{vrw} - delay_{\langle o_{vrw}.0 \rangle}(o_{vrw}))})$ □

4.3 Spezifikation mit Zustandsautomaten

schrittweises Verhalten

Das Verhalten einiger Komponenten ist algorithmischer Natur. Das heißt, die Komponenten führen Schritt für Schritt ihre Berechnungen durch. Zustandsautomaten bieten die Möglichkeit, ein Systemverhalten schrittweise zu beschreiben. Ein Schritt besteht aus einem Zustandsübergang (inklusive dem identischen Zustandsübergang), der durch bestimmte Werte oder Wertfolgen an den Eingabekanälen und einem Trigger-Signal ausgelöst wird. Ebenso können mit einem Zustandsübergang die Werte an den Ausgabekanälen geändert werden. Weitere Beschreibungen von Zustandsautomaten findet man in [Bre01] und [BS01]. Der unten definierte Automat ist speziell auf getaktete Rechnersysteme abgestimmt, bei der die Belegungen der Variablen die Werte an den Kanälen zu einem Zeitpunkt t sind.

Definition 4.7 (Zustandsautomat)
Ein *Zustandsautomat* A wird durch das 5-Tupel $A_S = (I, O, L, S_0, \delta, \Phi^{trigger})$ bestimmt.

Die Mengen I und O enthalten die Eingabe- und Ausgabekanäle (Variablen) des Systems. Die Menge L enthält die lokalen Variablen.

Der *Zustand* des Automaten A ist eine Belegung α aller Variablen der Menge $I \cup O \cup L$ mit den an den Kanälen anliegenden Werten (im Sinne der Bindung $[\![.]\!]^{op}$). Die nicht-leere Menge $S_0 \subseteq VAL$ enthält die Startzustände.

Die Relation $\delta \subseteq VAL \times VAL$ ist die Menge der *Transitionen* $(\alpha, \beta) \in \delta$, wobei α den Zustand vor und β den Zustand nach dem Zustandsübergang beschreibt.

Automaten sind *linkstotal*. Es gilt:
$\forall \alpha, \beta, \gamma \in VAL : ((\alpha, \beta) \in \delta \wedge \gamma \overset{L \cup O}{=} \beta) \Rightarrow (\alpha, \gamma) \in \delta$

Ein *Kontrollzustand* $k = (V_k, \beta^k)$ ist ein abstrakter Zustand, der sich auf die Belegung β^k einer Teilmenge $V_k \subset I \cup L \cup O$ bezieht und somit eine Menge von Zuständen umfasst. Es gilt: $k \overset{def}{=} \{\alpha | \alpha \overset{V_k}{=} \beta^k\}$.

Ein *Datenzustand* ist ein Kontrollzustand $d = (V_d, \beta^d)$, der sich auf die lokalen und die Ausgabevariablen bezieht, also $V_d = L \cup O$.

Ein *Datenkontrollzustand*(dkz) ist ein Kontrollzustand $l = (V_l, \beta^l)$, der sich auf einen Teil der lokalen und der Ausgabevariablen bezieht, also $V_l \subseteq L \cup O$.

Eine *linkstotale Transition* $(\alpha, d) \subseteq \delta$ mit der Belegung α und dem Datenzustand d ist eine Menge von Transitionen, die Transitionen von α zu jedem Zustand aus d umfasst, also $(\alpha, d) = \{(\beta, \beta') | \beta = \alpha \wedge \beta' \in d\}$

Eine *Datenkontrollzustand-Transition* $(k, l') \subseteq k \times l'$ ist eine Menge linkstotaler Transitionen mit dem Kontrollzustand $k = (V_k, \beta^k)$ und dem Datenkontrollzustand $l' = (V_{l'}, \beta^{l'})$, wobei gilt $V_{l'} \subseteq V_k$.

Ein Zustandsübergang findet statt, wenn es in δ mindestens ein Tupel (α, β) gibt, für das die Belegung α erfüllt ist. Die Belegungen der Variablen aus $L \cup O$ ändern sich dabei nicht, wenn nicht gleichzeitig die Formel $\Phi^{trigger}$ erfüllt ist. □

Formeln

Die Menge der Zustände, die mit einem Kontrollzustand k zusammengefasst wird, kann durch eine Formel Φ_k beschrieben werden, die bei einer Auswertung zu 'wahr' einen gültigen Zustand bezeichnet. Die Menge der Startzustände S_0 kann ebenfalls mit einer Formel Φ_{S_0} beschrieben werden.

Die Relation δ enthält oft viele Elemente und wird so meist nicht direkt als Menge von Tupeln angegeben, sondern wird durch eine Formel

$[\![\delta]\!] = (\boldsymbol{\Phi_1^\delta \vee \Phi_2^\delta \vee ...})$

beschrieben. Die Teilformeln beschreiben jeweils eine Datenkontrollzustand-Transition und zeichnen sich dadurch aus, dass sie als freie Bezeichner nur Variablen aus der Belegung vor dem Zustandsübergang und Variablen aus der dem Zustandsübergang folgenden Belegung enthalten[9]. Sie beschreiben des Weiteren nur Mengen linkstotaler Transitionen[10]. Die Relation δ ist dann definiert als [11]:

$$\begin{aligned} \delta \quad = \quad & \{(\alpha, \beta) | (\alpha, \beta \vDash [\![\delta]\!]) \wedge (\alpha \vDash \Phi^{trigger})\} \\ & \cup \{(\alpha, \beta) | (\alpha \overset{L \cup O}{=} \beta) \wedge ((\alpha \vDash \neg \Phi^{trigger}) \vee (\nexists (\alpha, \gamma) \vDash [\![\delta]\!]))\} \end{aligned}$$

Blackbox-Konvertierung

Die Beschreibung der Automaten A_S kann systematisch in eine Blackbox-Spezifikation $[\![S]\!]^{op}$ mit einer Laufzeitbedingung $[\![S]\!]^{op}_{Assertion}$ umgewandelt werden. Hierbei werden die Variablenreferenzen v und v' durch $delay(v)$ und v ersetzt. Es gilt $[\![S]\!]^{op} = [\![\delta]\!][delay_{S_0.0}(V)/V][V/V']$, wobei V der Menge der feien Variablen entspricht. $[\![S]\!]^{op}_{Assertion}$ beschreibt die Startzustände. Einige Begriffe werden im weiteren Verlauf der Arbeit wegen dieser Konvertierbarkeit nicht explizit für Automaten, sondern nur für Blackbox-Spezifikationen beschrieben.

[9] Üblicherweise werden die dem Zustandsübergang folgenden Variablenbelegungen mit dem Zeichen ' versehen (also x und x').

[10] Die Beschränkung auf linkstotale Transitionen ist gegeben, wenn Formeln keine Variablenbelegung x' für eine Eingabevariable $x \in I$ referenzieren.

[11] Es ist entweder ein Triggerzeitpunkt und der Zustand ist in der Transitionsmenge enthalten (dann schaltet die beschriebene Transition), oder der Zustand ist nicht enthalten, bzw. es ist kein Triggerzeitpunkt (dann ändern sich nur die Eingaben).

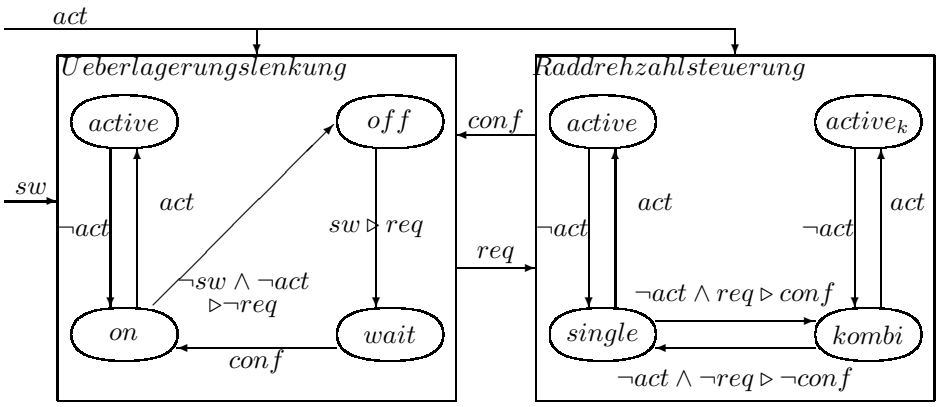


Abbildung 7: Beispieldarstellung zweier Automaten (siehe Beispiel 4.5)

Trigger Automaten dienen der Beschreibung von getakteten Systemen. Da alle Modelle einer einheitlichen schnellsten Taktung unterliegen, kann mit der Angabe der Formel $\Phi^{trigger}$ eine langsamere spezielle Taktung angegeben werden. Diese langsamere Taktrate wird durch Überprüfen eines von außen kommenden *Trigger*- oder Takt-Signals realisiert, welches dann entsprechend ausgewertet wird. Findet in einem Takt keine Transition statt, so bleiben die Werte der lokalen Variablen und der Ausgabevariablen gleich (und somit die Werte der Kanäle des Systems).

Konsistenz Mit Formeln beschriebene Automaten sind immer linkstotale und kausale Systeme. Die Linkstotalität ergibt sich daraus, dass die Formeln nur linkstotale Transitionen beschreiben und die fehlenden Transitionen, die ein Akzeptieren jeder Eingabe zulassen, automatisch ergänzt werden. Das System zeigt also für jede Eingabe ein Verhalten. Ebenso ist die Kausalität gegeben, da die einzelnen Transitionen immer nur den aktuellen Zustand und den Folgezustand betreffen. Eine weitere Eigenschaft bei Automaten ist die Schaltbereitschaft. Ein Automat A ist schaltbereit für eine Belegung α, wenn er von dieser Belegung zu einer anderen Belegung β eine Transition ausführen kann.

Darstellung Automaten eignen sich besonders zur Modellierung algorithmischer Abläufe. Die Beschreibung des Verhaltens mit Handlungsschritten erlaubt die anschauliche Modellierung (siehe Abbildung 7) komplexer Interaktionen zwischen den Systemen. Des Weiteren können die Zustände und Transitionen der Automaten formal in einer zu Black-Box-Konvertierbaren Darstellung gefasst werden, wie in folgendem Beispiel gezeigt wird.

Beispiel 4.5 (Spezifikation mit Automaten)
Die Stabilisierungsfunktion der Raddrehzahlsteuerung(RDS) und die Stabilisierungsfunktion der Überlagerungslenkung(UL) kommunizieren miteinander. Die Zustände 'active' bedeuten, dass die Systeme jeweils gerade eingreifen. Die Zustände 'single' und 'kombi' bedeuten, das die (eingeschaltete) RDS gerade die UL berücksichtigt ('kombi') oder nicht ('single'). Die UL kann entweder eingeschaltet sein ('on'), abgeschaltet sein ('off') oder auf die Bestätigung der RDS warten('wait'), dass diese im Zustand 'kombi' ist. Die Überlagerungslenkung soll nur dann eingeschaltet werden können, wenn die RDS im Kombi-Modus ist. Die Überlagerungslenkung kann mit dem Schalter 'sw' ein bzw. ausgeschaltet werden.

Dieses in Abblildung 7 dargestellte Verhalten der Systeme wird in Formeln mit zwei Automaten dargestellt:
$A_{RDS} = (I_{RDS}, O_{RDS}, L_{RDS}, S_{RDS_0}, \delta_{RDS}, \Phi^{trigger}_{RDS})$ und
$A_{UL} = (I_{UL}, O_{UL}, L_{UL}, S_{UL_0}, \delta_{UL}, \Phi^{trigger}_{UL})$
Die einzelnen Elemente sind wie folgt definiert:
$I_{RDS} = \{req : bool, act : bool\}$
$O_{RDS} = \{conf : bool\}$
$L_{RDS} = \{state_{RDS} : \{active, kact, single, kombi\}\}$
$S_{RDS_0} = \{\alpha_{RDS} | state_{RDS} = single \wedge conf = false\}$
$\Phi^{trigger}_{RDS} = (\boldsymbol{true})$
$[\![\delta_{RDS}]\!] =$

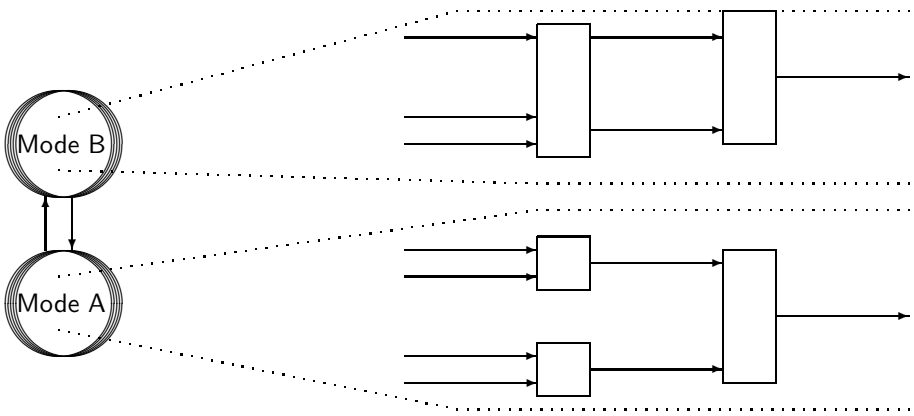


Abbildung 8: Beispieldarstellung zweier Betriebsmodi

$(\boldsymbol{act = true \wedge state_{RDS} = single \wedge state'_{RDS} = active \wedge conf' = conf})$
$\vee\ (\boldsymbol{act = false \wedge state_{RDS} = active \wedge state'_{RDS} = single \wedge conf' = conf})$
$\vee\ (\boldsymbol{act = false \wedge state_{RDS} = single \wedge req = true \wedge state'_{RDS} = kombi \wedge conf' = true})$
$\vee\ (\boldsymbol{act = false \wedge state_{RDS} = kombi \wedge req = false \wedge state'_{RDS} = single \wedge conf' = false})$
$\vee\ (\boldsymbol{act = true \wedge state_{RDS} = kombi \wedge state'_{RDS} = kact \wedge conf' = conf})$
$\vee\ (\boldsymbol{act = false \wedge state_{RDS} = kact \wedge state'_{RDS} = kombi \wedge conf' = conf})$

$I_{UL} = \{sw : bool, act : bool, conf : bool\}$
$O_{UL} = \{req : bool\}$
$L_{UL} = \{state_{UL} : \{active, on, off, wait\}\}$
$S_{UL_0} = \{\alpha_{UL} | state_{UL} = off \wedge req = false\}$
$\Phi_{UL}^{trigger} = (\boldsymbol{true})$
$[\![\delta_{UL}]\!] =$
$(\boldsymbol{act = true \wedge state_{UL} = on \wedge state'_{UL} = active \wedge req' = req})$
$\vee\ (\boldsymbol{act = false \wedge state_{UL} = active \wedge state'_{UL} = on \wedge req' = req})$
$\vee\ (\boldsymbol{act = false \wedge sw = false \wedge state_{UL} = on \wedge state'_{UL} = off \wedge req' = false})$
$\vee\ (\boldsymbol{sw = true \wedge state_{UL} = off \wedge state'_{UL} = wait \wedge req' = true})$
$\vee\ (\boldsymbol{conf = true \wedge state_{UL} = wait \wedge state'_{UL} = on \wedge req' = req})$ □

4.4 Spezifikation mit Betriebsmodi-Automaten

Betriebsmodus

Automaten bieten mit Kontrollzuständen die Möglichkeit, mehrere Zustände zusammenzufassen. Das Verhalten eines Systems kann mit Kontrollzuständen übersichtlicher beschrieben werden. Ein Sonderfall der Kontrollzustände sind die *Betriebsmodi*. Mit ihnen werden verschiedene Verhaltensmuster unterschieden, die auch mit verschiedenen Modellen dargestellt werden können. Abbildung 8 stellt den Begriff des Betriebsmodus graphisch dar (siehe auch [BBR+05]). Für ein System gilt in einem Betriebsmodus das Verhalten, das in einem Modellteil spezifiziert ist und in einem anderen Betriebsmodus das Verhalten, das in einem anderen Modellteil spezifiziert ist. Je nach dem, in welchem Betriebsmodus sich das System gerade befindet, werden die entsprechenden Ausgaben der Teilmodelle aus dem System weitergeleitet. Die Umschaltlogik zwischen den Betriebsmodi wird in einem expliziten Betriebsmodi-Automaten modelliert.

Kombination der Sichten

Der Einsatz dieser Betriebsmodi-Darstellung eignet sich zur Modellierung des Zusammenspiels der Regelungssicht und der Interaktionssicht. In den jeweiligen Teilmodellen der Betriebsmodi können regelungstechnische Funktionen untergebracht werden, die anhand der Zustände der Betriebsmodi-Automaten aktiviert werden. Das Zusammenspiel der Sichten wird explizit dargestellt und es können entsprechende Verifikations- und Modellierungsrichtlinien den jeweiligen repräsentativen Modellteilen zugeordnet werden. Die Betriebsmodi-Automaten selbst kommunizieren abgesehen von der Information über ihren aktuellen Zustand nicht nach außen. Die Interaktion mit anderen Komponenten wird in

getrennten Zustandsautomaten modelliert, wodurch der Charakter der Betriebsmodi-Automaten als Sicht-Schnittstelle hervorgehoben wird.

Beispiel 4.6 (Betriebsmodi eines Systems)
Der Fahrer kann der bei der geschwindigkeitsabhängigen Lenkübersetzung mit einem Schalter zwischen zwei Kennlinien für die Übersetzung wählen. Jedes mal, wenn das Lenkrad und die Räder gerade aus zeigen, wird entsprechend des Fahrerwunsches umgeschaltet. Die Modelle für die Regelung der Lenkübersetzung (welche den Kennlinien der Modi entsprechen) und das Modell für die Umschaltlogik (Automat) können bei einer Modellierung als Betriebsmodi-Automat getrennt entwickelt und analysiert werden. □

Aktivierung der Teilmodelle

Bei der Modellierung mit einem Automaten mit Kontrollzuständen umfasst der Zustand des Systems alle Variablenwerte der Komponenten. Wird eine Komponente mit Betriebsmodi modelliert, so gibt es verschiedene Möglichkeiten, die Teilmodelle miteinander zu kombinieren. Das Zusammenspiel zwischen den Teilmodellen wird zum einen bezüglich der Rechenleistung festgelegt und zum anderen bezüglich des Informationsaustausches. Bezüglich der Rechenleistung gilt:

- Die Teilmodelle werden parallel ausgeführt und gerechnet.
  Auch wenn ein Teilmodell gerade nicht seine Ausgaben aus der Komponente senden kann, führt es trotzdem Berechnungen aus. Diese Modellierung erfordert nicht die Berücksichtigung eines Abschaltkonzeptes, benötigt aber für jedes der Modelle Rechenleistung.

- Nur das aktive Teilmodell wird ausgeführt.
  Bei dieser Art der Koordination werden die Teilmodelle sequentiell ausgeführt, wobei die Sequenz durch den Betriebsmodi-Automaten bestimmt wird. Hier ist die benötigte Rechenleistung geringer, da schlimmstenfalls nur die Leistung des rechenintensivsten Teilmodells benötigt wird. Hierbei muss ein Abschaltkonzept berücksichtigt werden.

Übergang zwischen den Modi

Bei Informationssystemen werden die Teilmodelle meist aus Effizienzgründen sequenziell geschaltet, also immer nur das aktive Teilmodel ausgeführt. Bei der Sequenziellschaltung werden in [BRS05] drei Arten des Zusammenspiels der Komponenten bezüglich der einzelnen Variablen bei einem Modus-Wechsel dargestellt. Es wird entweder bei einem Modus-Wechsel die Variable des aktivierten Teilmodell immer neu initialisiert, oder sie behält den Zustand, den sie bei der Deaktivierung des Teilmodells zuletzt gehabt hat, oder sie übernimmt den Wert von dem zuletzt aktivierten Teilmodell. Aus der Modellierungsidee der Funktionsmodellierung mit Simulink ist jedoch die Parallelkomposition der Teilmodelle mit herkömmlichen Mitteln leicht zu realisieren. Werden die Modelle parallel ausgeführt, so ergeben sich in der Praxis zwei Möglichkeiten, die eine relativ unabhängige Modellierung der Teilsysteme zulassen:

- Es findet keine Kommunikation zwischen den Teilmodellen statt.
  Die verschiedenen Teilmodelle werden parallel voneinander gerechnet und es wird je nach Betriebsmodus entschieden, welche Ausgabe des jeweiligen Modells ausgegeben werden. Bei einer Umschaltung zu einem anderen Betriebsmodus ist es egal, welche Ausgaben das andere Teilmodell gehabt hat.

- Die Teilmodelle können auf die Ausgabe der Gesamtkomponente zugreifen.
  Es findet eine unidirektionale Kommunikation von dem jeweils aktivierten Teilmodell zu den anderen Teilmodellen statt. Ein Modus entspricht so einem Kontrollzustand, dessen lokale Variablen in keinem anderen Zustand gelesen oder verändert werden.

formale Definition

Im Folgenden wird eine Definition eines Betriebsmodus formal beschrieben, die eine parallele Berechnung der Teilmodelle beschreibt. Die parallele Ausführung wurde gewählt, da die gängigen Datenfluss-Beschreibungssprachen für reaktive Systeme (z.B. Simulink oder Lustre) nur die parallele Zusammensetzung von Teilmodellen kennen (siehe [MR98b]) und so eine Umsetzung mir diesen Sprachmitteln möglich ist. Zur Verwendung anderer Mode-Interpretationen sei auf die Arbeiten [MR98b], [BRS05] und [HHK03] verwiesen.

Definition 4.8 (Betriebsmodi-Komponente)
Eine *Betriebsmodi-Komponente* S^{BM} mit der Schnittstelle (I^{BM}, O^{BM}) und dem Verhalten $[\![S^{BM}]\!]^{op}$ wird durch das Tupel
$S^{BM} = (A, T, O, \Phi^{mapping})$ bestimmt.

Gegeben sei eine Menge von Namen der Betriebsmodi der Komponente
$N_{BM} = \{bm1, bm2, ...\}$.

A ist ein Zustandsautomat mit der Einschränkung, dass dessen Ausgabe O_A aus einem Kanal $O_A = \{o_A\}$ besteht. Die Wertemenge des Kanals entspricht der Menge der Namen der Modi, also $W_{o_A} = N_{BM}$.

Die Menge T enthält die Komponenten, deren Ausgaben in den jeweiligen Betriebsmodi aus der Komponente ausgegeben werden, also
$T = \bigcup_{x \in N_{BM}} \{S_x\}$.

Die Eingabekanäle der Komponente S^{BM} sind die Vereinigung der Eingabekanäle der Komponenten in T und des Automaten A, also $I^{BM} = I_A \cup (\bigcup_{x \in N_{BM}} I_{S_x})$.

Die Menge $O^{BM} = \{o_{BM.1}, o_{BM.2}, ..., o_{BM.n}\}$ ist die Menge der Ausgabekanäle der Komponente S^{BM}. Sie kann o_A enthalten.

Jede Teilkomponente S_x mit $x \in N_{BM}$ hat eine (interne) Ausgabe
$O_x = \{o_{x.1}, o_{x.2}, ..., o_{x.n}\}$.

Das Prädikat $\Phi^{mapping} = \bigwedge_{x \in N_{BM}, 1 \leq z \leq n} \Phi^{mapping}_{x.z}$ ist eine Konjunktion von Teilprädikaten für jeden Betriebsmodi und jeden Ausgabekanal der Komponente. Die Teilprädikate haben die Form
$\Phi^{mapping}_{x.z} = ((o_A = x) \Rightarrow (o_{x.z} = o_{BM.z}))$.

Das Verhalten der Komponente ergibt sich aus dem Verhalten des Automaten, der Teilmodelle und des Mappings, also
$[\![S^{BM}]\!]^{op} = \Phi^{mapping} \wedge [\![A]\!] \wedge [\![S_{bm1}]\!]^{op} \wedge [\![S_{bm2}]\!]^{op} ...$ □

Implementierung

Die meisten Modellierungssprachen bieten nicht die Möglichkeit, Betriebsmodi explizit zu modellieren. Betriebsmodi lassen sich über ein Konstrukt mit herkömmlichen Mitteln relativ einfach erstellen. Die Umsetzung der Betriebsmodi in einer sequenziellen Programmiersprache geschieht durch Verwendung von if-Anweisungen und wird in [MR98a] erklärt. Bei einer graphischen Datenfluss-Modellierung werden die verschiedenen Teilmodelle in getrennten Teilkomponenten modelliert, wobei die Eingabekanäle der äußeren Komponente auch die Eingabekanäle der Teilkomponenten sind. Die Ausgaben der Teilkomponenten werden an einen so genannten *Schiedsrichter (engl. Arbiter)* gesendet. Dieser entscheidet, von welcher Teilkomponente die Ausgaben aus der äußeren Komponente herausgeleitet werden. Der Schiedsrichter besteht aus einem Automaten, der den aktuellen Zustand bestimmt und einem Schalter (engl. Switch) der entsprechend die Ausgaben weiterleitet. Abbildung 9 verdeutlicht diesen Aufbau. Je nach dem, welche der beiden oben gegebenen Interpretationen für Betriebsmodi gewählt sind, kann eine Rückkopplung der Ausgaben zu den Teilmodellen stattfinden (angedeutet mit gestrichelter Linie).

Beispiel 4.7 (Spezifikation mit Betriebsmodi-Komponenten)
Gegeben ist eine geschwindigkeitsabhängige Lenkübersetzung S_{BM}, deren Lenkübersetzung wahlweise zwei verschiedenen Kennlinien ($sport(s)$ und $komfort(k)$) entsprechen kann. Zur Vereinfachung findet die Umschaltung zwischen den Kennlinien sofort statt. Diese Komponente ist definiert als (siehe auch Abbildung 9)
$S^{BM} = (A, T, O^{BM}, \Phi^{mapping})$ mit:

$$
\begin{aligned}
A = (\ & \{taste : bool\}, \\
& \{modus : \{s, k\}\}, \\
& \{state : \{s, k, s_{pre}, k_{pre}\}\}, \\
& \{\alpha_{S_{BM}} | state = s \wedge modus = s\}, \\
& ((taste = true \wedge state = s \wedge state' = k_{pre} \wedge modus' = k) \\
& \wedge (taste = false \wedge state = k_{pre} \wedge state' = k \wedge modus' = k) \\
& \wedge (taste = true \wedge state = k \wedge state' = s_{pre} \wedge modus' = s) \\
& \wedge (taste = false \wedge state = s_{pre} \wedge state' = s \wedge modus' = s)), \\
& true\ )
\end{aligned}
$$

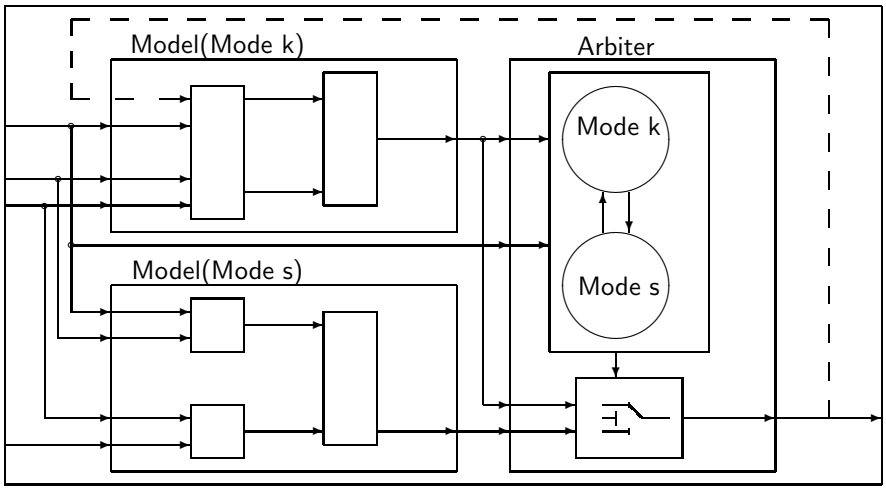


Abbildung 9: Beispielimplementierung zweier Betriebsmodi (siehe Bsp. 4.7)

$T = (S_s, S_k)$

$I_{S_s} = \{i_{lrw} : float, i_v : float\}$
$O_{S_s} = \{o_{vrw\_s} : float\}$
$[\![S_s]\!]^{op} = (\boldsymbol{o_{vrw\_s} = i_{lrw}/(16 * \mathsf{max}(1, (|i_v|/100km/h))})$

$I_{S_k} = \{i_{lrw} : float\}$
$O_{S_k} = \{o_{vrw\_k} : float\}$
$[\![S_k]\!]^{op} = (\boldsymbol{o_{vrw\_k} = i_{lrw}/(20 * \mathsf{max}(1, (|i_v|/100km/h))})$

$O_{BM} = \{o_{vrw}\}$
$\Phi^{mapping} = ((\boldsymbol{modus = s}) \Rightarrow (\boldsymbol{o_{vrw} = o_{vrw\_s}})) \wedge ((\boldsymbol{modus = k}) \Rightarrow (\boldsymbol{o_{vrw} = o_{vrw\_k}}))$ □

4.5 Spezifikation der kontinuierlichen Anteile

Eigenschaften der Schnittstelle

Die Besonderheit mechatronischer Systeme liegt in dem Zusammenspiel der auf der einen Seite stehenden Logik und Hardware der eingebetteten Rechnersysteme und des auf der anderen Seite stehenden physikalischen Teils, der aus dem physikalischen Teil des mechatronischen Systems und dem der Umwelt besteht (siehe Abschnitt 2). Die *eingebettete Systeme* sind getaktete Rechnersysteme, die mit diskreten Werten umgehen. Das Verhalten der physikalischen Systeme kann zusätzlich nichtdeterministisch, wertkontinuierlich und insbesondere zeitkontinuierlich sein. Die Schnittstellen des getakteten eingebetteten Systems zum physikalischen Teil werden als *Aktuatoren* (diskret zu kontinuierlich) und *Sensoren* (kontinuierlich zu diskret) bezeichnet. Dieser Abschnitt konzentriert sich auf die Modellierung des Verhaltens der Umgebung des eingebetteten Rechnersystems. Es werden keine neuen Modellierungstechniken eingeführt, sondern es wird darauf eingegangen, wie mit den bisher aufgeführten Modellierungstechniken die Physik hinsichtlich der Bewertung der Funktionssicherheit des eingebetteten Systems geeignet modelliert werden kann. Dazu werden die Eigenschaften der physikalischen Teile beschrieben und entsprechende Vereinfachungen bei der Beschreibung des Verhaltens genannt, um die Bewertung der Modelle zu ermöglichen.

Eigenschaften des physikalischen Teils

Die physikalische Umgebung des eingebetteten Systems hat zwei wesentliche Eigenschaften:

(1.) Das Verhalten des physikalischen Teils ist häufig von Faktoren beeinflusst, die nicht präzise erfasst werden können. So kann z.B. die Lenkung durch Unebenheiten der Straße beeinflusst werden. Besonders Systeme der Umwelt des mechatronischen Systems, in der auch der Faktor Mensch und andere natürliche Systeme Bestandteile sind, sind meist so komplex, dass sie nicht deterministisch modellierbar sind. Ein Modell des physikalischen Teils ist also meistens nichtdeterministisch.

(2.) Das Verhalten der Umgebung unterliegt nicht mehr der künstlich geschaffenen Logik, die programmiert wird, sondern ist durch die mechanische Bauweise oder physikalischen Umstände gegeben.

Die Komplexität des Systems ist also ein fester Faktor. Dies zeigt sich z.B. darin, dass die Mechanik meistens kontinuierlich ist. Sowohl die Zeit, wie auch die Variablen-Werte sind kontinuierlich.

freie Modellierung

Die Beschreibung des Verhaltens der physikalischen Umwelt kann theoretisch mit beliebigen Blackbox-Spezifikationen (siehe Abschnitt 4.1 und 4.2) stattfinden. Die Spezifikationen können zum Beispiel Integrale und trigonometrische Funktionen enthalten. Der Übergang zwischen dem kontinuierlichen und dem diskreten Teil des Systems an den Aktuatoren und Sensoren kann real modelliert werden. Nichtdeterminismus kann entweder mit relaxierten Gleichungen (siehe Abschnitt 4.2), Ungleichungen oder logischen Aussagen modelliert werden. Beliebige kontinuierliche nichtdeterministische Modelle sind mit heutigen Mitteln der Mathematik nicht verifizierbar und nur schwer analysierbar ([Sch03], S. 58). Dies ist die Ursache für die im Rest des Abschnitts genannten Beschreibungsmechanismen.

Sampling

Eine Vereinfachung bei der Beschreibung der Umgebung des eingebetteten Systems ist, die Beschreibung zu diskretisieren. Die Zeit wird zu diskreten Zeitpunkten abstrahiert. Dabei können entweder flexible Zeitspannen zwischen den Zeitpunkten zugelassen werden ([Sch03], [Hen96]), oder feste Zeitspannen verwendet werden. Zur Analyse eingebetteter Systeme eignet sich die Verwendung fester Zeitspannen, welche die Umgebung zu den Taktzeitpunkten des eingebetteten Systems betrachten ([Sch03], [HCRP91]). Des Weiteren werden die kontinuierlichen (eventuell sogar unendlichen) Wertebereiche der Variablen der Umgebung auf endliche Zahlen abgebildet (z.B. als Fließkomma- oder Integer-Zahlen einer bestimmten Bitlänge). Diese Diakretisierung der Zeit und der Werte wird Sampling genannt. Alle Modelle, die einer derartigen Diskretisierung zugrundeliegen, haben einen endlichen Zustandsraum. Zumindest theoretisch sind diese Modelle simulierbar und verifizierbar. In der Praxis wächst jedoch auch bei endlichen Wertemengen der Variablen der Zustandsraum sehr schnell, so dass eine automatische Verifikation schnell mit der Größe des Zustandsraums sehr viel Speicherplatz und Berechnungszeit braucht. Um eine Anwendung in der Praxis zu erlangen, ist eine systematische starke Einschränkung der Menge der diskreten Zustände nötig.

ND-Automaten

Ist ein abstraktes Systemmodell diskret, kann der Nichtdeterminismus der Umgebung mit ND-Automaten modelliert werden. Von einem Zustand gehen dann entsprechend mehrere Transitionen aus, die bei gleichen Eingaben schaltbereit sind. Bei einer Modellierung mit Automaten müssen geeignete abstrakte Zustände gefunden werden, mit denen die wichtigen Eigenschaften des Systems weiterhin dargestellt werden können. Eine genaue Beschreibung der Modellierung der Physik mit Automaten findet man in [OR04] und [ORS05].

Linearisierung

Eine Darstellung der Umwelt mit Automaten mit entsprechend abstrakten Zuständen ist nicht immer möglich und die Modelle sind auch nicht immer intuitiv verständlich. Eine Alternative besteht darin, nicht den Zustandsraum weiter einzuschränken, sondern die Funktionen, die das Verhalten beschreiben, zu approximieren. Hierzu werden die Funktionen in eine verifizierbare Form gebracht. Die am häufigsten verwendete Vereinfachung ist die lineare Approximation der Funktionen. Auf diese Weise erhält man Funktionen, die in (Un-)Gleichungssysteme umgeformt werden können, die systematisch mit Mitteln der Mathematik gelöst werden können (siehe [Hen96], [LSV95], [Sch03]).

5 Abhängigkeiten zwischen Modellen

schrittweises Vorgehen

Die bisher beschriebenen Modellierungstechniken ermöglichen es, ein System zu beschreiben. In einem Entwicklungsprozess wird ein Modell jedoch nicht sofort detailliert beschrieben, sondern es wird schrittweise erarbeitet. Ein Hilfsmittel bei der schrittweisen Modellierung eines Systems ist die Zerlegung in einfacher zu modellierende Teile, die Modellierung der Teile und dann das Zusammenfügen der Teile zu einem Ganzen. Zwei Modelle, die zusammen ein Ganzes ergeben, stehen in einer Kompositionsbeziehung. Ein weiteres Mittel der schrittweisen Modellierung ist, zuerst eine grobe Beschreibung zu erstellen, und diese dann immer detaillierter zu machen. Ein Modell, welches eine detailliertere Beschreibung eines anderen Modells ist, steht in einer Verfeinerungsbeziehung zu diesem.

Abschnittsübersicht

Dieser Abschnitt beschreibt die Abhängigkeiten Komposition und Verfeinerung für die in Abschnitt 4 genannten Modellierungstechniken. Von diesen beiden Abhängigkeiten können gezielt Fehler abgeleitet werden, die aus dem Umgang mit den Modellen entstehen.

5.1 Komposition

Systeme werden üblicherweise aus mehreren Teilkomponenten zusammengesetzt. Idealerweise können diese im Entwicklungsprozess voneinander unabhängig entwickelt werden. Allein die Strukturierung bringt oft bereits ein deutlich verständlicheres Bild des Systems. Dieser Abschnitt erläutert in Anlehnung an [Bre01] kurz die Voraussetzungen an eine *Komposition* von Systemen und deren Definition.

Kompositionsbegriff

Systeme agieren nur über ihre Ein- und Ausgabekanäle mit ihrer Umgebung. Ein komponiertes System entsteht so über die Verschachtelung der Kanäle, in dem angegeben wird, welche Ausgaben eines Systems als Eingaben eines anderen Systems dienen. Fügt man zwei Systeme zusammen, so dürfen ihre Ausgabekanäle keine gleichen Namen haben, um einen Identitätskonflikt zu vermeiden. Es gilt (angelehnt an [Bre01]):

Definition 5.1 (Komposition)
Gegeben sind zwei Systeme S und T mit den Schnittstellen (I_S, O_S) und (I_T, O_T)

und dem Verhalten

R_S und R_T

für die gilt:

$O_S \cap O_T = \emptyset$

Ein System U ist genau dann ein aus S und T zusammengesetztes System, wenn gilt: $U = S \otimes T$

Der Kompositionsoperator $\otimes$ für die Systeme S, T und U ist folgend definiert:

$\otimes :: \mathbb{S} \times \mathbb{S} \to \mathbb{S}$
$U = S \otimes T \overset{def}{\Leftrightarrow} ((I_U, O_U) = (I_S, O_S) \otimes (I_T, O_T)) \wedge (R_U = R_S \otimes R_T)$

Der Kompositionsoperator $\otimes$ für die Systeme (I_S, O_S), (I_S, O_S) und (I_U, O_U) ist folgend definiert:
$(I_U, O_U) = (I_S, O_S) \otimes (I_T, O_T) \overset{def}{\Leftrightarrow} (I_U = (I_S \cup I_T) \setminus (O_S \cup O_T)) \wedge (O_U = (O_S \cup O_T))$

Die Menge der Rückkopplungskanäle ist:
$L_U \overset{def}{=} (I_S \cup I_T) \cap (O_S \cup O_T)$ □

Black-Box

Im Falle der Black-Box-Spezifikationen ist die Komposition von zwei Systemen wesentlich einfacher auszudrücken. Für die Schnittstelle gilt die obige Kompositionsregel. Da die Kanäle über ihre Namen identifiziert werden und somit in den Gleichungen durch eindeutige Variablennamen repräsentiert werden, gilt für das Verhalten (siehe [Bre01]):

Definition 5.2 (Komposition (mit Prädikaten))
Gegeben seien die Verhaltensweisen $[\![S]\!]$ und $[\![T]\!]$ zweier Systeme S und T. Das Verhalten der Komposition $[\![S \otimes T]\!] = [\![S]\!] \otimes [\![T]\!]$ ist gegeben durch
$[\![S]\!] \wedge [\![T]\!]$ □

mehrere Systeme

Die Definition der Komposition gilt nur für zwei Systeme. Es kann durch eine iterative Anwendung der Komposition eine beliebig große Menge an Systemen zusammengefasst werden. Für den Kompositionsoperator gilt hierbei sowohl das Assoziativ- wie auch das Kommutativgesetz, d.h. die Reihenfolge des Zusammensetzens spielt keine Rolle. Wichtig ist jedoch, dass die zu verbindenden Kanäle in ihren Namen übereinstimmen. Hier kann mittels der Umbenennung, die einer Bijektion auf Kanalnamen entspricht, Abhilfe geschafft werden (siehe [Bre01]).

Serienschaltung

Ein Sonderfall der Komposition ist die *Serienschaltung*, bei der alle Ausgaben der jeweils vorhergehenden Komponente als Eingabe der folgenden Komponenten dienen. Bei dieser Art der Komposition ist somit eine rückkopplungsfreie Komponente beschrieben, die als Eingabe die Kanäle der ersten Teilkomponente und als Ausgabe die Kanäle der zweiten Teilkomponente hat. Beschrieben wird diese Art der Komposition für zwei Komponenten S und T mit $S \succ T$.

5.2 Verfeinerung

unterschiedliche Spezifikationen

Die Entwicklung eines Systems findet üblicherweise nicht immer in vollem Detaillierungsgrad statt. Es ist oft hilfreich, von dem tatsächlichen Verhalten zu abstrahieren. So wird z.B. bei der Entwicklung eines Systems zu Beginn eine eher grobe Beschreibung des Systems erstellt, die dann anhand von Entwurfsentscheidungen weiter und genauer beschrieben wird. Diese Entwurfsentscheidungen umfassen unter anderem die Aufteilung eines Systems in Untersysteme und die Wahl einer Systemarchitektur, aber auch die präziseren Anforderungen an das Soll-Verhalten, das mit einem fortgeschrittenen Reifegrad immer besser verstanden wird. Es ist auch zur besseren Handhabbarkeit und Analyse oft notwendig, bestimmte Eigenschaften eines Systems in einer für den jeweiligen Betroffenen gut lesbaren Beschreibung darzustellen, und ihn nicht mit Details zu überschütten.

Zusammenhang

Die in einem Entwicklungsprozess bestehenden Spezifikationen stellen immer das gleiche System dar und sind dementsprechend voneinander abhängig. Idealerweise stehen die Spezifikationen zu der jeweils detailliertesten Spezifikation in einer Verfeinerungsbeziehung. Eine Spezifikation ist eine *Verfeinerung* einer anderen Spezifikation, wenn sie im wesentlichen das gleiche System aus einer gleichen Sicht beschreibt, aber zusätzliche Eigenschaften und Konkretisierungen enthält. Es gibt je nach konkretem Verwendungszweck eine Menge verschiedener Verfeinerungsarten (siehe [BS01], [Bre01], [Bro02]). In dieser Arbeit hat die Verfeinerung folgende konkrete Aufgaben (die auch in Kombination auftreten können) zu unterstützen:

- Das Auflösen von Nichtdeterminismus (Verhaltensverfeinerung)
- Das Einschränken der Menge der Eingabeströme (Bedingte Verfeinerung)
- Das Sicherstellen einer Mindestfunktionalität nach der Verfeinerung (Bedingte sicherstellende Verfeinerung)
- Das Abbilden der Kanäle auf andere Kanäle mit anderen Datentypen (Schnittstellenverfeinerung)

Verhaltensverfeinerung

Im folgenden wird nun schrittweise der formale Begriff der Verfeinerung von einem einfachen Verfeinerungsbegriff hin zu einem allgemeinen Verfeinerungsbegriff aufgebaut. Die erste Variante der Verfeinerung kommt nur der Aufgabe des Auflösens des Nichtdeterminismus entgegen, ohne dabei die Menge der akzeptierten Eingabeströme zu reduzieren und ohne dabei auf die Menge der Ausgabeströme zu achten. Diese Verfeinerung wird Verhaltensverfeinerung genannt und ist wie folgt definiert (siehe [Bre01], [BS01]):

Definition 5.3 (Verhaltensverfeinerung)
Gegeben seien zwei Systeme mit der gleichen Schnittstelle aber mit verschiedenen verhaltensbeschreibenden Relationen S und T.
Das System T ist eine *Verhaltensverfeinerung* des Systems S
$S \rightsquigarrow T$
wenn jedes Verhalten von T auch ein Verhalten von S ist. Es gilt also:
$R_T \subseteq R_S$ und $dom.R_S = dom.R_T$ (Die Definition *dom* und *rng* ist in Abschnitt 3.1). □

bedingte Verfeinerung

Mit der Verhaltensverfeinerung lässt sich zwar die Menge der Ausgabeströme reduzieren, es ist aber nicht möglich, die Menge der Eingabeströme zu reduzieren, da bei der Komponente nur Nichtdeterminismus aufgelöst wurde. Die Einschränkung der Eingabe, wenn man z.B. die Eingabe von einem unendlichen Datentypen auf einen endlichen Datentypen reduzieren möchte, geschieht über eine bedingte Verhaltensverfeinerung, bei der über eine Kondition angegeben wird, welche Werte nicht berücksichtigt werden müssen.

Definition 5.4 (bedingte Verhaltensverfeinerung)
Gegeben seien zwei Systeme mit der gleichen Schnittstelle aber mit verschiedenen verhaltensbeschreibenden Relationen S und T, sowie eine Bedingung C
Das System T ist eine *bedingte Verhaltensverfeinerung* des Systems S
$S \rightsquigarrow_C T$
wenn jedes Verhalten von T auch ein Verhalten von S ist und die Eingabemenge reduziert werden kann. Es gilt also:
$R_T \subseteq R_S$ und $rng.C = dom.R_T$ und $rng.C \subseteq dom.R_S$ □

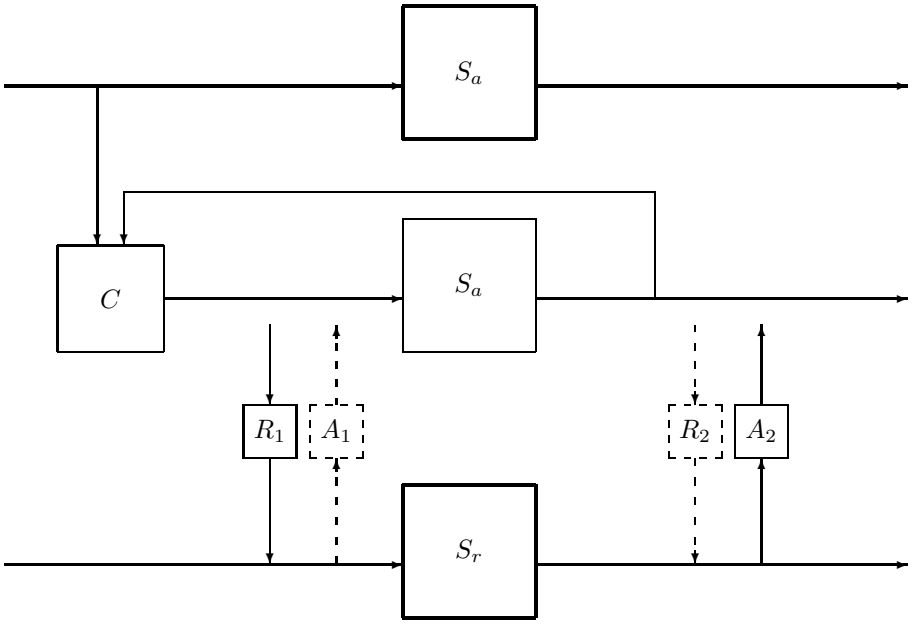


Abbildung 10: allgemeine Verfeinerung einer Komponente

Die Verwendung der Bedingung C bei der Beschreibung der bedingten Verhaltensverfeinerung kann theoretisch weggelassen werden, da eine Einschränkung des Eingabebereichs auch durch Auslassen der Bedingung $dom.R_S = dom.R_T$ bei der Verhaltensverfeinerung möglich wäre. Es ist jedoch für den Entwickler besser nachvollziehbar, wenn die Einschränkung explizit sichtbar ist. Der verkürzte Begriff der Verhaltensverfeinerung ist:

Bei Relationsspezifikation: $R_T \subseteq R_S$
Bei Blackboxspezifikation: $[\![T]\!] \Rightarrow [\![S]\!]$

Mindestfunktionalität

Bei der Spezifikation eines Systems liegen häufig auch Anforderungen vor, die eine gewünschte Mindestfunktionalität fordern, also eine minimale Menge an Ausgabeströmen. Der bisherige Begriff der Verfeinerung betrachtet nur, dass das Verhalten einer Komponente nicht erweitert wird. Über die Bedingung G kann bei der bedingten sicherstellenden Verfeinerung eine Mindestforderung an die verfeinerte Komponente weitergegeben werden.

Definition 5.5 (bedingte sicherstellende Verhaltensverfeinerung)
Gegeben seien zwei Systeme mit der gleichen Schnittstelle aber mit verschiedenen verhaltensbeschreibenden Relationen S und T, sowie die Bedingungen C und G.
Das System T ist eine *bedingte sicherstellende Verhaltensverfeinerung* des Systems S
$S \rightsquigarrow_{(C,G)} T$
wenn jedes Verhalten von T auch ein Verhalten von S ist und G berücksichtigt wird. Es gilt also:
$S \rightsquigarrow_C T$ und $G \vDash R_T$ □

Nachrichtenverfeinerung

Neben der Verfeinerung des Verhaltens kann auch eine Verfeinerung der Schnittstelle mit einhergehen. Hier können sich Datentypen, die Anzahl der Nachrichtenkanäle sowie die Repräsentation der Nachrichten verändern. Dies kann z.B. notwendig sein, wenn eine Funktion an ein spezifisches Steuergerät angepasst wird. Hier ist also ein allgemeinerer Begriff der Verfeinerung notwendig, der eine abstraktere Spezifikation mit einer konkreteren Spezifikation in Beziehung setzt (siehe Abbildung 10). Die Nachrichten des abstrakten Systems werden über eine Repräsentationsrelation in Beziehung mit dem konkreten System gesetzt. Eine Abstraktions- / Repräsentationsrelation zwischen zwei Systemen ist wie folgend definiert (siehe auch [Bro02]):

Definition 5.6 (Abstraktionsrelation & Repräsentationsrelation)
Eine Verfeinerungsbeziehung $g \overset{(A,R)}{\rightsquigarrow} f$ zwischen zwei Nachrichtenkanälen f und g ist durch eine *Abstraktionsrelation* $A \subseteq \mathcal{F}_f \times \mathcal{F}_g$ und eine *Repräsentationsrelation* $R \subseteq \mathcal{F}_g \times \mathcal{F}_f$ gegeben. Dabei muss gelten, dass beide Relationen surjektiv und linkstotal sind und eine konkrete Nachricht immer wieder auf die ausgehende abstrakte Nachricht schließen lässt, also gilt: $R \succ A = ID$ □

Schnittstellenverfeinerung

Auf Basis der Verfeinerungsbeziehung zwischen Nachrichtenkanälen ist es möglich, zwei Komponenten mit verschiedenen Schnittstellen in Beziehung zu setzen. Die jeweils verfeinerte Komponente hat das Verhalten der jeweils generelleren Komponente zu erfüllen. Die Definition einer entsprechenden korrekten Abstraktionsrelation und Repräsentationsrelation ist der kreative Anteil, der bei dieser Art der Verfeinerung als korrekt angenommen wird. Soll der dynamische Datentyp der abstrakteren Beschreibung bei der Verfeinerung eingeschränkt werden, so lässt sich dies über die Angabe einer zusätzlichen Bedingung regeln, die angibt, für welche Eingabeströme die Bedingung gelten soll (siehe Komponente C in Abbildung 10). Die Verfeinerung der Schnittstelle ist wie folgt definiert (siehe [Bre01], [BS01]):

Definition 5.7 (Schnittstellenverfeinerung)
Gegeben seien zwei Systeme mit den Schnittstellen (I_S, O_S) und (I_T, O_T) und den verhaltensbeschreibenden Relationen S und T. Gegeben seien auch eine Repräsentationsrelation R_1 mit der Schnittstelle (I_S, I_T), eine Abstraktionsrelation A_2 mit der Schnittstelle (O_T, O_S) und eine Bedingung C mit der Schnittstelle (I_C, I_S). Das System T ist eine *Schnittstellenverfeinerung* des Systems S
$S \overset{(R_1,A_2)}{\rightsquigarrow_{(C,G)}} T$
wenn gilt:
$S \rightsquigarrow_{(C,G)} (R_1 \succ T \succ A_2)$ □

Für diesen allgemeineren Verfeinerungsbegriff ist die Verhaltensverfeinerung der Sonderfall, in dem die Repräsentationsrelation und die Abstraktionsrelation der Identität entsprechen. Dieser erweiterte Verfeinerungsbegriff lässt auch Verfeinerungen zu, die nicht immer sinnvoll sind. Abgeänderte Definitionen und Kriterien zur Verfeinerung finden sich in [BS01] und [Bro98]. Der Begriff der Verfeinerung wird mit dem folgenden Beispiel verdeutlicht:

Beispiel 5.1 (Verfeinerung eines Systems)
Gegeben sei eine Spezifikation S einer geschwindigkeitsabhängigen Lenkübersetzung. Die Eingaben sind die Geschwindigkeit v ($W_{\mathcal{F}_v} = \mathbb{Z}$) und der Lenkradwinkel lrw ($W_{\mathcal{F}_{lrw}} = \mathbb{Z}$). Die Ausgabe ist der Vorderradwinkel vrw ($W_{\mathcal{F}_{vrw}} = \mathbb{Z}$). Es gelten folgende Aussagen:

Die Lenkung soll mit steigender Geschwindigkeit indirekter übersetzen:
$\Phi_1 = ((|\boldsymbol{v}| > |\mathbf{delay}_{\langle 0 \rangle}(\boldsymbol{v})|) \Rightarrow (|(\mathbf{delay}_{\langle 0 \rangle}(\boldsymbol{lrw})/\mathbf{delay}_{\langle 1 \rangle}(\boldsymbol{vrw}))| \leq |(\boldsymbol{lrw}/\boldsymbol{vrw})|))$

und der Vorderradwinkel soll auf Fahrerwunsch immer auf mindestens 15° eingeschlagen werden können:
$\Phi_2 = (\forall \boldsymbol{v} : \exists \boldsymbol{lrw} : (\boldsymbol{vrw} > \mathbf{15°}))$

Gelten für ein System S beide Aussagen, also $[\![S]\!]^{op} = (\boldsymbol{\Phi_1} \wedge \boldsymbol{\Phi_2})$, so erfüllen alle Kennlinien diese Spezifikation, die sich an diese Bedingungen halten.

Verhaltensverfeinerung:

Diese Spezifikation $[\![S]\!]^{op}$ wird nun in einer Spezifikation $[\![S_1]\!]^{op}$ präzisiert, in dem der Nichtdeterminismus beseitigt wird, also eine Kennlinie als Lösung ausgesucht wird. Es gilt:

$[\![S_1]\!]^{op} = (\boldsymbol{vrw} = \mathbf{round}(\boldsymbol{lrw}/(\mathbf{16} * \mathbf{max}(\mathbf{1}, |\boldsymbol{v}|/\mathbf{100}\boldsymbol{km}/\boldsymbol{h}))))$

Diese System erfüllt auch die Spezifikation $[\![S]\!]^{op}$. Es gilt also: $S \rightsquigarrow S_1$

bedingte Verhaltensverfeinerung:

Zusätzlich möchte man für die Implementierung vorsehen, dass der maximale Eingabewinkel kleiner 720° ist. Größere Werte werden nicht betrachtet. Die Bedingung C lautet:
$[\![C]\!]^{op} = (|\boldsymbol{lrw}| < \mathbf{720°})$

Damit kann eine Spezifikation S_2 definiert werden, die das gleiche Verhalten wie S_1 aufweist, der Datentyp des Eingabekanals lrw allerdings auf $[-719..719]$ reduziert ist. Es gilt:
$S \rightsquigarrow_C S_2$ und für S_2 gilt zusätzlich: $W_{\mathcal{F}_{lrw}} = [-719..719]$

bedingte sicherstellende Verhaltensverfeinerung:

Um die existentielle Eigenschaft des Mindestlenkwinkels sicherzustellen, wird die Bedingung $\Phi_G = (\exists \boldsymbol{lrw} : (\boldsymbol{vrw} > \mathbf{15°}))$ angegeben. Diese ist nicht bei jeder Verfeinerung $S \rightsquigarrow_C S_2$ erfüllt, da bei Geschwindigkeiten ab 300 km/h der Lenkradwinkel $< 720°$ nicht mehr ausreicht.

Eine Möglichkeit, diese Bedingung zu erfüllen ist die Angabe der verschärften Bedingung $[\![C_3]\!]^{op}$ für ein System S_3, bei dem zusätzlich die Eingabe v auf die üblichen 250 km/h eingeschränkt wird.
$[\![C_3]\!] = (\boldsymbol{|lrw| < 720° \wedge |v| < 250km/h})$

Damit gilt bei angepasstem Datentyp v auf $[-249km/h..249km/h]$:
$S \rightsquigarrow_{C_3} S_3$ und $S \rightsquigarrow_{(C_3,G)} S_3$.

Schnittstellenverfeinerung:

Letztlich soll der Datentyp der Kanäle v und lrw auf einen Kanal b mit einer 18-Bit-Integer-Darstellung zusammengefasst werden. Hierzu sind die jeweiligen Abstraktions- bzw. Repräsentationsrelationen $R_v, R_{lrw} und A_{vrw}$. Es gilt:

$$[\![R_v]\!]^{op} = (\quad \textbf{if } (-249 \leq v < 249) \textbf{ then } (b \textbf{ div } 1024 == v) \textbf{ else } (b \textbf{ div } 1024 \in [-256.. -250, 249..255]))$$

$$[\![R_{lrw}]\!]^{op} = (\quad \textbf{if } (-719 <= lrw < 719) \textbf{ then } (b \textbf{ mod } 1024 == lrw) \textbf{ else } (b \textbf{ mod } 1024 \in [-1024.. -720, 719..1023]))$$

$A_{vrw} = ID$

Ein System S_4 ist nur dann eine Verfeinerung, wenn es sich wie S_3 verhält und alle Abbildungen von 249 bzw. 719 gleich interpretiert. Es gilt also:
$[\![S_5]\!]^{op} = (\boldsymbol{vrw = sig(b \bmod 1024) * min(|b \bmod 1024|, 719)/(min(|(b \operatorname{div} 1024)|, 249)/100 + 1)})$
und $S \overset{(R_v,R_{lrw},A_{vrw})}{\rightsquigarrow_{(C_4,G)}} S_5$ □

Abweichungen von diesen Verfeinerungsbeziehungen sind potenzielle Fehlerquellen, die bei einer Funktionssicherheitsanalyse berücksichtigt werden können. So kann es zum Beispiel passieren, dass Datentypen gewechselt werden, ohne dass dabei auf Rundungsfehler geachtet wird. Diese sind somit potenzielle Fehler.

6 Ansatzpunkte für Fehlerbeschreibungen

Sichten

Aus den Sichten auf die Systeme und aus den jeweiligen Modellierungstechniken lassen sich Ansatzpunkte für die Identifikation und Modellierung von Fehlern ableiten (siehe Abbildung 11). Die Sichten sind nicht disjunkt, sondern betrachten die gleichen Artefakte aus verschiedenen Blickwinkeln. Grundlegend gibt es die Verhaltenssicht und Implementierungssicht. Die Implementierungssicht beschreibt die Artefakte, auf denen das Verhalten abgelegt ist (siehe Abschnitt 2). Die Implementierungssicht liefert im Wesentlichen Ansatzpunkte zur Identifikation der potenziellen Fehler eines Systems. In der Verhaltenssicht werden das Verhalten der potenziellen Fehler und das der Folgefehler in einem System betrachtet. Je nach dem, wie die Grenze zur Verhaltensebene gezogen wird, kommen in der Implementierungsstruktur entsprechende Fehler vor. Abstrahiert man zum Beispiel von der Verteilung der Funktionen auf Steuergeräte, so kommen bei der Übertragung der Signale zwischen den Systemen die Standard-CAN-Übertragungsfehler als potenzielle Fehler in Frage.

Implementierungssichten

Die Implementierungssicht (siehe Abschnitt 3.2) kann den Blickwinkel auf die Schnittstellen (siehe Beispiel in Abschnitt 2) und den auf die Artefakte selbst haben. Ansatzpunkte zur Identifikation potenzieller Fehler sind die Arten der Artefakte (Betriebssystem, SW-Modul, ...) und diedie Qualitätseigenschaften aus [ISO01] (Zuverlässigkeit, Robustheit, ...). Bei elektronischen Bauteilen sind dies z.B. unter anderen EM-Störungen oder thermische Störungen. Auch die Spezifika der Schnittstellen der Systeme liefern Ansatzpunkte zur Fehleridentifikation. So lassen sich aus der Art der Wertübermittlung, anhand Art der übermittelten Größe und dem Verhalten des Restsystems weitere potenzielle Fehler identifizieren. Wird z.B. die Beschleunigung des Fahrzeugs von einem Sensor gemessen, so

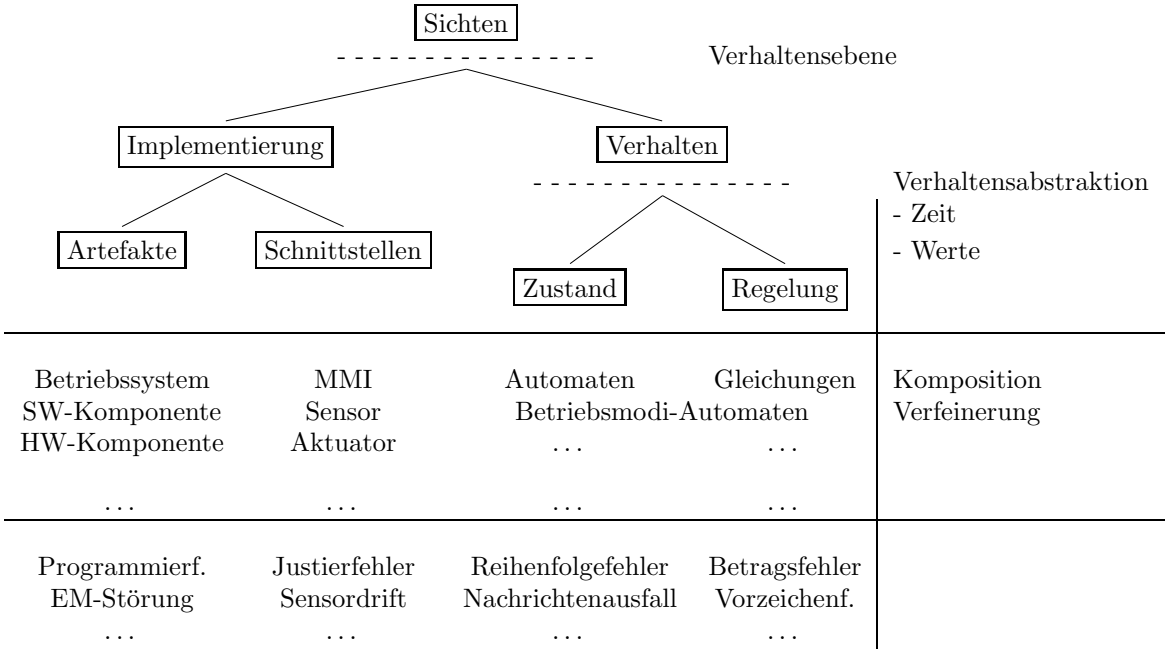


Abbildung 11: Sichten auf ein System als Grundlage zur Fehleridentifikation

kann es bei dem Sensor Justierfehler oder Sensordrift-Fehler geben. Bei der Kommunikation mit anderen Rechnersystemen können hingegen Nachrichtenfehler als potenzielle Fehler betrachtet werden. Den potenziellen Fehlern werden in der Implementierungssicht Wahrscheinlichkeiten zugewiesen. Diese können anhand von Statistiken, Erfahrungen und Schätzverfahren ermittelt werden.

Verhaltenssicht

Während in der Implementierungssicht den Fehlern ein Label und eine Wahrscheinlichkeit zugewiesen wird, ist in der Verhaltenssicht (siehe Abschnitt 3.1) der Einfluss der Fehler auf das Verhalten im Mittelpunkt. Der potenzielle Einfluss eines Fehlers auf das Verhalten hängt von der Modellierung bzw. der Sicht auf das Verhalten ab. Die Fehlverhalten können spezifisch für die Interaktionssicht und die Regelungssicht sein. In der Regelungssicht, welche z.B. mit Gleichungen (siehe Abschnitt 4.2) modelliert wird, sind Wertabweichungen relevant (z.B. Wert zu groß für 3 Sekunden). Zeit spielt hier eine untergeordnete Rolle. In der Interaktionssicht, welche z.B. mit Automaten (siehe Abschnitt 4.3) modelliert wird, sind Nachrichtenfehler relevant (z.B. Nachricht zu spät, Reihenfolgefehler, ...). Zeit und Zeitpunkte spielen hier eine wichtige Rolle. Schließlich sollten insbesondere auch die Zusammenhänge zu Betriebsmodi in einem Fehlermodell berücksichtigt werden können.

Abstraktion

Die Modelle können auf verschiedenen Verhaltensabstraktionsebenen stehen. So kann z.B. ein zeitkontinuierliches Verhalten eines Systems auf ein zeitdiskretes Verhalten abstrahiert werden. Verschiedene Modelle können so in Beziehung zueinander stehen (siehe Abschnitt 5). Aus dem Umgang mit Modellen lassen sich potenzielle Fehler identifizieren, welche durch Verletzungen der Verfeinerungs- und Kompositions-Beziehungen entstehen.

Analyse

Die Art der Modellierung und die Sichten auf ein System haben also direkten Einfluss auf die identifizierbaren potenziellen Fehler. Umgekehrt betrachtet ist die Beschreibung der Fehler immer relativ zu der korrespondierenden Sicht der Systemspezifikation beschrieben. Eine Funktionssicherheitsanalyse betrachtet entsprechend nur die Fehler, die sie identifizieren und modellieren kann. Ansatzpunkte für weitere Arbeiten sind zum einen Zuordnungen bestehender formaler Ansätze der Funktionssicherheitsanalyse zu den hier gegebenen Modellsichten, so wie die Betrachtung und der Entwurf von spezifischen Fehlermodellen und Funktionssicherheitsanalysemethoden.

7 Zusammenfassung

In dieser Arbeit wird mit Referenzmodellen eingebetteter Systeme und Modellierungstechniken die Grundlagen der Spezifikation dieser Systeme skizziert. Dabei werden Ansatzpunkte herausgestellt, an denen spezifische potenzielle Fehler identifiziert, wie auch Fehlerverhalten spezifisch beschrieben werden können. Diese Arbeit dient so als Grundlage für weitere Arbeiten der Fehlermodellierung und Funktionssicherheitsanalyse.

Die Arbeit ist in vier Kernteile gegliedert. Der erste Teil beschreibt Referenzmodelle eingebetteter Systeme und Referenzmodelle zu den Abstraktionsebenen und Eigenschaften bei der Modellierung der Systeme. Wesentlich sind die Schnittstellen eingebetteter Systeme (Sensoren, Aktuatoren, MMI, Kommunikationsinterface), die Modellierung verschiedener Domänen (Mechanik, Elektrik, Elektronik) und die Abstraktion von Implementierungsdetails.

Der zweite Teil erklärt, wie mit Modellen das Verhalten beschrieben werden kann und wie mit Modellen die Artefakte definiert werden können, auf denen das Verhalten implementiert ist. Das Verhalten wird als eine Relation zwischen kanalbeschreibenden Funktionen modelliert. Komplementär stehen die Implementierungmodelle, welche die Struktur und qualitativen Eigenschaften der Implementierung enthalten. Diese qualitativen Eigenschaften sind ein Ansatzpunkt zur Identifizierung potenzieller Fehler, deren Wirkung als Fehlverhalten im Verhaltensmodell dargestellt und in einer Funktionssicherheitsanalyse weiter analysiert werden kann.

Der dritte Teil zeigt Modellierungstechniken, mit denen die Verhaltensmodelle spezifiziert werden können. Hierzu gehören Prädikate und Automaten (‚so wie deren Sonderformen 'aufgelöste' Gleichungen und Betriebsmodi-Automaten). Die Modellierungtechniken sind auf spezifische Sichten hin ausgerichtet (Regelungssicht, Interaktionssicht, usw.). Diese spezifischen Modellierungstechniken bieten so eine Grundlage zu spezifischen Fehlerbeschreibungen.

Der vierte Teil zeigt zwei Abhängigkeiten zwischen Verhaltensmodellen, die bei der Modellierung regelmäßig auftreten. Eine Abhängigkeit ist die Komposition, mit der verschiedene Modelle zu einem Ganzen zusammengefügt werden. Die andere Abhängigkeit ist die Verfeinerung, in der ein Modell detaillierter wird. Anhand der Beschreibung dieser Abhängigkeiten können potenzielle Fehler identifiziert werden, welche durch Mißachtung der Abhängigkeiten entstehen.

Die sich aus den Kernteilen ergebenden Ansatzpunkte zur Fehleridentifikation und -modellierung werden schließlich zusammengefasst.

Literatur

[AD94] Alur, Rajeev ; Dill, David L.: *A theory of timed automata.* In: *Theoretical Computer Science* 126 (1994), Nr. 2, S. 183–235

[AUT06] AUTOSAR GbR. *Technical Overview V2.0.1.* 2006

[BBR+05] Bauer, Andreas ; Broy, Manfred ; Romberg, Jan ; Schätz, Bernhard ; Braun, Peter ; Freund, Ulrich ; Mata, Núria ; Sandner, Robert ; Ziegenbein, Dirk: *AutoMoDe — Notations, Methods and Tools for Model-Based Development of Automotive Software.* In: *Proceedings of the SAE 2005 World Congress.* Detroit, MI : Society of Automotive Engineers, April 2005

[BDD+92] Broy, Manfred ; Dederich, Frank ; Dendorfer, Claus ; Fuchs, Max ; Gritzner, Thomas ; Weber, Rainer: *The Design of Distributed Systems - An Introduction to FOCUS* / Institut für Informatik, Technische Universität München. 1992 (I-9202). – Forschungsbericht. www.tum.de

[Bre01] Breitling, Max D.: *Formale Fehlermodellierung für verteilte reaktive Systeme*, Lehrstuhl für Software und Systems Engineering, Institut für Informatik, Technische Universität München, Diss., 2001

[Bro98] Broy, Manfred: *Compositional Refinement of Interactive Systems Modelled by Relations.* In: *International Symposium Compositionality*, Springer, 1998, S. 130 – 149

[Bro02] Broy, M.: *Unifying Models and Engineering Theories of Composed Software Systems.* In: Broy, M. (Hrsg.) ; Pizka, M. (Hrsg.): *Models, Algebra and Logic of Engineering Software. Marktoberdorf Summer School 2002* Bd. 191, Springer, 2002

[BRS05] Bauer, Andreas ; Romberg, Jan ; Schätz, Bernhard. *The AUTOMODE Design Notation: Concepts, Consistency, and Timing.* 2005

[BS01] Broy, M. ; Stoelen, K.: *Specification and Development of Interactive Systems: Focus on Streams, Interfaces, and Refinement.* 1. Springer, 2001

[BSN07] Balzer, Heinrich ; Stein, Benno ; Niggemann, Oliver: *Diagnose in verteilten automotiven Systemen.* In: Gausemeier, Jürgen (Hrsg.): *5. Paderborner Workshop Entwurf mechatronischer Systeme* Bd. 210, Heinz Nixdorf Institut, March 2007. – ISBN 978–3–939350–29–3, S. 243–254

[Eri05] Ericson, Clifton A.: *Hazard Analysis Techniques for System Safety.* 1. John Wiley and Sons, Inc., Hoboken,New Jersey, 2005

[FGP04] Fleischmann, Andreas ; Geisberger, Eva ; Pister, Markus: *Herausforderungen für das Requirements Engineering eingebetteter Systeme* / Institut für Informatik, Technische Universität München. 2004 (I-0414). – Forschungsbericht. www.tum.de

[Fow03] Fowler, Martin: *UML Distilled: A Brief Guide to the Standard Object Modeling Language.* 1. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 2003. – ISBN 0321193687

[GSB97] Grosu, R. ; Stoelen, K. ; Broy, M.: *A Denotational Model for Mobile Point-to-Point Data-flow Networks with Channel Sharing* / Institut für Informatik, Technische Universität München. 1997 (I-9724). – Forschungsbericht. www.tum.de

[HCRP91] Halbwachs, N. ; Caspi, P. ; Raymond, P. ; Pilaud, D.: *The synchronous data-flow programming language LUSTRE.* In: *Proceedings of the IEEE* 79 (1991), September, Nr. 9, S. 1305–1320

[Hen96] Henzinger, Thomas: *The Theory of Hybrid Automata.* In: *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS '96).* New Brunswick, New Jersey : IEEE, 1996, S. 278–292

[HHK03] Henzinger, T.A. ; Horowitz, B. ; Kirsch, C.M.: *Giotto: a time-triggered language for embedded programming.* In: *Proceedings of the IEEE* 91 (2003), Nr. 1, S. 84–99

[ISO94] ISO. *ISO 7498 - Information processing systems - open systems interconnection - basic reference model*. 1994

[ISO01] ISO. *ISO 9126 - Software engineering – Product quality – Part 1: Quality model*. 2001

[ISO05] ISO. *ISO/IEC 19501 - Information technology - Open Distributed Processing - Unified Modeling Language (UML) Version 1.4.2*. 2005

[Lev95] Leveson, N.: *Safeware - System, Safety and Computers*. 1. Addison-Wesley Publishing Company, 1995

[LSV95] Lynch, Nancy A. ; Segala, Roberto ; Vaandrager, Frits W.: *Hybrid I/O automata*. In: *82*. ISSN 0169-118X : Centrum voor Wiskunde en Informatica (CWI), 31 1995, S. 16

[LSV01] Lynch, Nancy ; Segala, Roberto ; Vaandrager, Frits: *Hybrid I/O Automata Revisited*. In: *Lecture Notes in Computer Science* 2034 (2001), S. 403+

[Min65] Minsky, M.: *Matter, mind and models*. In: Kalenichl, A. (Hrsg.): *Proceedings International Federation of Information Processing Congress*, Spartan Books, 1965

[MR98a] Maraninchi, F. ; Remond, Y. *Running modes in a data-flow language: Modeautomata and their implementation*. 1998

[MR98b] Maraninchi, Florence ; Rémond, Yann: *Mode-Automata: About Modes and States for Reactive Systems*. In: *Lecture Notes in Computer Science* 1381 (1998), S. 185 ff.

[OR04] Ortmeier, F. ; Reif, W.: *Failure-Sensitive Specification: A formal method for finding failure modes* / Universität Augsburg, Institut für Informatik. 2004 (2004-3). – Forschungsbericht. www.uni-augsburg.de

[ORS05] Ortmeier, F. ; Reif, W. ; Schellhorn, G.: *Formal Safety analysis of a radio-based railroad crossing using Deductive Cause-Consequence Analysis*. 2005. – to be published

[Sch03] Schneider, Dr.-Ing. R. *Grundlagen der Simulationstechnik*. 2003

[VDI04] VDI. *Entwicklungsmethodik für mechatronische Systeme*. Beuth Verlag. 2004

[WFH+06] Wild, Doris ; Fleischmann, Andreas ; Hartmann, Judith ; Pfaller, Christian ; Rappl, Martin ; Rittmann, Sabine: *An Architecture-Centric Approach towards the Construction of Dependable Automotive Software*. In: *Proceedings of the SAE 2006 World Congress*, 2006

Abbildungsverzeichnis