T U M

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

Dienstbasierte Spezifikation des ACC, Fallstudie

Alexander Harhurin und Judith Hartmann

TUM-I0834
Oktober 08

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-10-I0834-0/0.-FI

Druck: Institut für Informatik der
Technischen Universität München

Inhaltsverzeichnis

1 Einführung

Das Dokument ist im Rahmen des Verbundprojekts „Verteilte Entwicklung und Integration von Automotive-Produktlinien“ (VEIA), BMBF Förderungsnummer 01ISF15A, entstanden. Das Ziel des Projekts VEIA war es, auf der Grundlage von Konzepten der Produktlinientechnik eine Methode für die verteilte Entwicklung und Integration von Automotive-Systemen zu erarbeiten, die sich an den konkreten Anforderungen industrieller Entwicklungsprozesse orientiert und praktisch anwendbar ist.

Die Verantwortlichkeit der TU München war es, eine variierbare Spezifikation der Produktlinie bzw. eines Produkts zu erzeugen, die vollständig, konsistent und konform zu den Kundenwünschen ist. Dabei wurden Konzepte und Techniken erarbeitet, die die Sicherstellung der genannten Eigenschaften unterstützen. Der Schwerpunkt lag dabei auf der Betrachtung der funktionalen Anforderungen aus Nutzersicht.

Ziel dieses Dokumentes ist es, die erarbeiteten Konzepte zur dienstbasierten Modellierung der Funktionalität anhand einer Fallstudie aus dem Automobilbereich, der Adaptiven Cruise Control (ACC), darzustellen. Die entsprechenden theoretischen Grundlagen sind ausführlich in [5, 6, 3] beschrieben. Dennoch werden in den anschließenden Kapiteln die erarbeiteten Konzepte zur dienstbasierten Modellierung zunächst kurz rekapituliert (Kapitel 2.1) sowie zwei unterschiedliche Notationstechniken für dienstbasierte Spezifikation eingeführt: In Kapitel 2.2 wird eine automatenbaiserte Notationstechnik vorgestellt, in Kapitel 2.3 eine tabellarische Notationstechnik. Der Hauptteil des Dokuments widmet sich dann der Modellierung des Fallbeispiels. Die Funktionalität des ACC wird in Teilfunktionalitäten unterteilt, die zunächst unabhängig voneinander modelliert werden (Kapitel 4.1, 4.3 und 4.4). Dabei kommen beide Notationstechniken, automatenbasiert sowie tabellarisch, zum Einsatz. In Kapitel 4.5 werden die Teilfunktionalitäten schließlich unter Einführung von Querbeziehungen zur Gesamtfunktionalität ACC integriert.

2 Dienstbasierte Spezifikation

2.1 Allgemein

In einer dienstbasierten Spezifikation wird das Verhalten des Gesamtsystems in Form eines Dienstmodells strukturiert erfasst. Dabei wird für das Gesamtsystem die Systemgrenze festgelegt und das Verhalten des Gesamtsystems aus *Blackbox-Sicht* spezifiziert, d.h. es wird der Nachrichtenaustausch an der identifizierten Systemgrenze, also zwischen dem Gesamtsystem und seiner Umwelt festgelegt.

Das Dienstmodell setzt sich zusammen aus einer Menge von *Diensten* und *Querbeziehungen* zwischen diesen. Dienste liefern *formale Spezifikationen* von Teilfunktionalitäten des Systems (Kundenfunktionen) und können mittels geeigneter Notationstechniken, etwa durch I/O Automaten (siehe Kapitel 2.2), Tabellen (siehe Kapitel 2.3) oder Assumption/Guarantee-Spezifikationen (siehe [6]) repräsentiert werden. Querbeziehungen beschreiben, wie die modular

definierten Dienste zusammenspielen bzw. sich gegenseitig beeinflussen, um das gewünschte Gesamtverhalten zu erbringen. Querbeziehungen machen folglich die Interaktion zwischen Dienste explizit.

Die Zerlegung in Teildienste stellt keine Systemdekomposition in kommunizierende Einheiten dar, sondern lediglich eine hierarchische Strukturierung der Funktionalität, die das Gesamtsystem anbietet. Ein Teildienst stellt somit eine Projektion aus dem Gesamtverhalten dar. Die Systemgrenze ändert sich bei der Zerlegung in Teildienste nicht: auch Teildienste spezifizieren nur das Verhalten, wie es an der Grenze des Gesamtsystems beobachtbar ist. Insbesondere stellt die syntaktische Schnittstelle eines Teildienstes eine Projektion aus der Gesamtschnittstelle dar: Die Schnittstelle jedes Teildienstes entspricht einer Teilschnittstelle des Gesamtsystems. Von strukturellen Aspekten (vor allem Kommunikation zwischen Teilkomponenten) wird in der dienstbasierten Spezifikation abstrahiert.

Eine dienstbasierte Spezifikation beschreibt das Gesamtverhalten nicht notwendigerweise vollständig; das System kann nur *partiell* definiert sein, d.h. das Verhalten muss nicht für alle möglichen Eingabefolgen festgelegt sein.

Ausgangssituation Den Ausgangspunkt für die dienstbasierte Spezifikation bildet ein Satz von Anforderungen an das Systemverhalten. Diese Anforderungen können in unterschiedlicher Form vorliegen, beispielsweise als textuelle Dokumentation (z.B. Doors-Dokument oder rein textuelle Beschreibung in einem Lastenheft) oder als Sammlung von Use Cases.

Das Dienstmodell bildet das erste formale Modell für die funktionalen Anforderungen. Beim Übergang von den informellen Anforderungen zu einem Dienstmodell werden die funktionalen Anforderungen durch Dienste und deren Querbeziehungen umgesetzt. Die Umsetzung erfolgt jedoch nicht durch eine Abbildung einer Anforderung auf genau einen Dienst, vielmehr stehen die ausgearbeiteten Anforderungen und die Dienste in einer $n : m$ Beziehung. Eine Anforderung kann durch einen oder mehrere Dienste umgesetzt werden und ein Dienst kann eine oder mehrere Anforderungen realisieren.

Ziele Zusammengefasst ergeben sich auf Ebene des Dienstmodells folgende Ziele:

- Konsolidierung der funktionalen Anforderungen durch eine formale Präzisierung der Anforderungen, die das Systemverhalten aus Nutzersicht beschreiben;
- Erkennen und Auflösen von Widersprüchen in den funktionalen Anforderungen;
- Komplexitätsreduktion durch eine hierarchische Strukturierung der Funktionalität aus Nutzersicht;
- Besseres Verständnis der funktionalen Zusammenhänge durch formalisierte Erfassung und Analyse der Wechselwirkungen zwischen verschiedenen Funktionalitäten („Teildienste“);
- Separation of Concerns, d.h. eine geeignete Aufteilung der Systemfunktionalität in Teilfunktionalitäten, die von unterschiedlichen Entwicklern umgesetzt werden können.

Produktlinien Wie in [6] ausführlich beschrieben, ist unser Ansatz gut geeignet, funktionale Anforderungen an Produktlinien zu spezifizieren. Dabei entsteht eine dienstbasierte Spezifikation mit optionalen und alternativen Diensten, die nicht ein einziges System sondern eine Menge von ähnlichen Systemen beschreibt. Ein *optionaler* Dienst muss nicht von jeder gültigen Implementierung erfüllt sein, es handelt sich dabei um eine optionale Anforderung. Bezüglich *alternativer* Dienste gilt, dass jede gültige Implementierung aus einer Menge von *alternativen* genau einen Dienst erfüllen muss. Zusätzlich zu funktionalen Beziehungen zwischen Diensten kann die Spezifikation einer Produktlinie zwei weitere Beziehungen enthalten. Eine *requires*-Beziehung zwischen zwei Diensten fordert von einer Implementierung die Erfüllung des zweiten Dienstes, falls der erste Dienst erfüllt ist. Eine *excludes*-Beziehung besagt, dass wenn der erste Dienst erfüllt ist, muss der zweite Dienst nicht erfüllt sein. Die beiden Beziehungen machen nur dann Sinn, wenn die betroffenen Dienste optional oder alternativ sind.

2.2 Automaten

In diesem Kapitel wird eine automatenbasierte Notation für die dienstbasierte Spezifikation eingeführt. Das folgende Kapitel führt eine Technik zur Spezifikation einzelner Dienste ein, die jeweils eine bestimmte Teilmenge der funktionalen Anforderungen, nämlich genau einen Anwendungsfall, formalisieren. Danach wird gezeigt wie einzelne Dienste kombiniert werden können. Dabei unterscheidet man zwischen einer normalen und einer priorisierten Kombination. Als Ergebnis hat man eine Spezifikation, die aus formal definierten Diensten und Beziehungen zwischen ihnen besteht.

Das vorliegende Kapitel stellt eine sehr knappe Zusammenfassung des automatenbasierten Ansatzes dar. Alle formalen Definitionen und Erklärungen sind in [3] zu finden.

2.2.1 Dienst

Ein Dienst hat ein syntaktisches Interface, das aus einer Menge von Input- und einer Menge von Output-Ports besteht (siehe z.B. Abbildung 4 auf Seite 13). Dabei symbolisieren die leeren Kreise die Input-Ports und die schwarzen Kreise die Output-Ports.

Die Semantik eines Dienstes ist durch einen I/O Automaten definiert(vgl. Tabelle 1 auf Seite 14). Dabei werden Transitionen zwischen zwei Zuständen immer in der folgenden Form angegeben: $\{pre\}in/out\{post\}$. Die Vorbedingungen pre sind Bedingungen bzgl. Input-Ports und lokaler Variablen. Die Input- und Output-Mustern (in und out) definieren Werte an Input- und Output-Ports. Ein Input-Muster der Form $ip?iv$ verlangt den Wert iv am Input-Port ip. Ein Output-Muster der Form $op!ov$ verlangt den Wert ov am Output-Port op. Einzelne Muster werden zu einem größeres Muster zusammengefasst, getrennt durch ein Semikolon. Eine Transition kann gefeuert werden, falls der Boolesche Ausdruck $pre \wedge in$ mit den aktuellen Eingabewerten gleich $True$ ist. An den Output-Ports werden Werte ausgegeben, so dass der Boolesche Ausdruck $out \wedge post$ gleich $True$ ist. Einzelne Prädikate können weggelassen werden, falls sie leer sind, d.h. sie sind für alle möglichen Werte gleich $True$. Das Zeichen ε symbolisiert explizit, dass an einem Port keine Nachricht anliegt.

2.2.2 Komposition

Eine gültige Implementierung muß alle funktionalen Anforderungen an das System *gleichzeitig* erfüllen. Dies bedeutet, einzelne, unabhängig voneinander definierte Dienste müssen in größere Dienste kombiniert werden können. Damit werden einzelne funktionale Anforderungen in eine Spezifikation zusammengefasst. Unter einem kombinierten Dienst kann man sich einen Container von parallel laufenden Teil-Diensten vorstellen. Dabei können einzelne Dienste gemeinsame I/O Ports haben.

Der Automat der Komposition macht einen Schritt (eine seiner Transitionen wird gefeuert) in einem der folgenden Fällen:

- Die aktuelle Eingabe kann von allen seiner Teil-Dienste akzeptiert werden und ihre Outputs sind nicht widersprüchlich. In diesem Fall machen alle Teil-Dienste parallel einen Schritt und die Ausgabe muss die Zusicherung aller Dienste erfüllen.
- Die aktuelle Eingabe kann nur von einem der Teil-Dienste akzeptiert werden, d.h. die Eingabe liegt nicht in der Domäne der anderen Teil-Dienste. In diesem Fall macht nur der erste Dienst einen Schritt, die restlichen Teil-Dienste ändern ihren Zustand nicht. Die Ausgabe muss nur die Zusicherung des ersten Dienstes erfüllen.

Dies bedeutet, die Zusicherungen aller Dienste, welche die aktuellen Eingaben akzeptieren, müssen mit den aktuellen Ausgaben gleich *True* sein. Dienste, welche die aktuelle Eingabe nicht erfüllen, stellen keine Anforderungen an die aktuellen Ausgaben.

2.2.3 Priorisierte Komposition

In einer Spezifikation kommt es oft vor, dass bestimmte Ereignisse oder Abläufe höhere Priorität haben als die anderen. Zum Beispiel hat das Verhalten eines Systems in einem Notfall eine höhere Priorität als das Verhalten im Normalfall. Damit man in der Lage ist, solche Fälle spezifizieren zu können, wird in diesem Kapitel die Priorisierung zwischen verschiedenen Diensten eingeführt.

Die priorisierte Komposition priorisiert in jedem Schritt einen der beiden in der Komposition beteiligten Dienste. Dies bedeutet, der unterpriorisierte Dienst stellt keine Anforderungen an die Ausgabe: sein Automat macht keinen Schritt und stellt somit keine Anforderungen an die aktuellen Ausgaben. Dabei ist die priorisierte Komposition von einem speziellen Kontrolldienst S_P kontrolliert. Das Interface dieses Dienstes enthält die Input-Ports der beiden beteiligten Dienste $I_1 \cup I_2$ und keinen Output-Port. Die Menge der Transitionen dieses Dienstes ist in zwei Untermengen aufgeteilt $T = T_P \cup T'$. Wird eine Transition aus T_P gefeuert, hat der Dienst S_2 die höhere Priorität. Kann der Dienst die aktuelle Eingabe nicht akzeptieren (keine Transition wird gefeuert) oder eine Transition aus T' wird gefeuert, verhalten sich die beiden Dienste nach den Regeln der unpriorisierten Komposition. Mit anderen Worten, der Kontrolldienst S_P bestimmt Inputs, für die entweder ein Dienst oder die beiden Dienste Anforderungen an die Outputs stellen.

2.2.4 Konflikte zwischen Diensten

Der wesentliche Vorteil einer formalen Spezifikation ist die Möglichkeit, eine automatische Analyse ihrer Konsistenz durchzuführen. Eine dienstbasierte Spezifikation ist konsistent, falls es keine Konflikte zwischen einzelnen Diensten gibt. Die einfachste (und meist verbreitete) Art von Konflikten ist der Fall, wenn zwei Dienste am selben Output Port und im selben Zeitintervall unterschiedliche Nachrichten ausgeben. Das bedeutet, die Anforderungen an die Implementierung sind widersprüchlich. Formale Definitionen einzelner Konflikte sind in [3] zu finden.

2.3 Tabellen

Eine weitere Technik zur Spezifikation von Diensten sind Tabellen.

Die hier eingeführten Tabellen bestehen aus mehreren Spalten und Zeilen. Abbildung 1 stellt den Aufbau der verwendeten Tabellen schematisch dar.

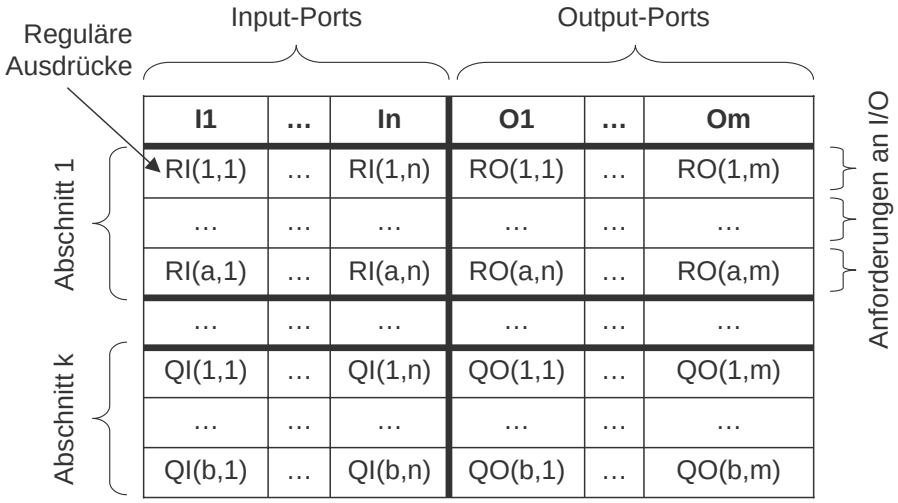

Abbildung 1: Aufbau der Tabellen zur Spezifikation von Diensten

Die Spalten der Tabelle (und ihre Beschriftung) ergeben sich durch das syntaktische Interface des Dienstes. Die Spalten unterteilen sich in einen Eingabeteil, der genau eine Spalte für jeden Eingabeport umfasst, und einen Ausgabeteil mit einer Spalte für jeden Ausgabeport. Jede Zeile enthält die Beschreibung einer Anforderung an das Systemverhalten durch die Angabe von Eingabeströmen an den Eingabeports und den als Reaktion geforderten Ausgabeströmen an den Ausgabeports. Dazu steht in jeder Zelle ein (erweiterter) regulärer Ausdruck, der die bis zu einem Zeitintervall anliegenden Ein- bzw. Ausgabenachrichten an dem zugehörigen Kanal beschreibt.

Erweiterte Reguläre Ausdrücke Die verwendete Notation für regulären Ausdrücke orientiert sich an dem IEEE POSIX-Standard 1003.1 zur Spezifikation von regulären Ausdrücken. So

wird z.B. das Zeichen „.“ für ein beliebiges Zeichen des entsprechenden Typs verwendet, d.h. jedes Element des entsprechenden Typs matched den regulären Ausdruck „.“. Zudem erlauben wir die Verwendung von Variablen innerhalb der regulären Ausdrücke. Diese können durch Bedingungen eingeschränkt werden. Beispielsweise matcht der regulärer Ausdruck „$a|a > 10$“ über dem Alphabet $\Sigma = \mathbb{N}$ alle Zahlen größer 10. Innerhalb der Tabellen, sind Variablen immer zeilenweise gebunden. Das heißt, in einem regulären Ausdruck können Variablen verwendet werden, die weiter vorne im regulären Ausdruck oder in anderen Zellen der jeweiligen Zeile gebunden sind. Beispielsweise beschreibt der Ausdruck „$aba.|b > 10$“ über dem Alphabet $\Sigma = \mathbb{N}$ vier aufeinander folgende natürliche Zahlen, von denen die erste und dritte einen beliebigen Wert annehmen dürfen aber identisch sein müssen, die zweite Zahl größer als 10 sein muss und die vierte Zahl beliebig sein kann. Korrekte Beispiele wären „3 11 3 4“ oder „27 100 27 1“.

Semantik der Tabelle Die regulären Ausdrücke in den Eingabespalten beschreiben die bis zum aktuellen Zeitintervall t beobachteten Nachrichtenströme an den Inputkanälen. Die regulären Ausdrücke in den Ausgabespalten beschreiben das als Reaktion geforderte Verhalten an den Ausgabekanälen ab dem nächsten Zeitintervall $t + 1$.

Jede Anforderung an das beobachtbare Systemverhalten wird also durch eine Zeile beschrieben, welche die beobachtbaren Eingabemuster an den Eingabekanälen mit geforderten Ausgabemustern an den Ausgabekanälen in Verbindung setzt. Jede derartige Kombination von passenden Ein- und Ausgabemustern schränkt die Menge der gültigen Kombinationen von Ein-/Ausgabeströmen über einem gegeben Interface weiter ein.

Die Gesamtspezifikation des Tabelle ergibt sich durch Kombination der Anforderungen, die in den verschiedenen Zeilen der Tabelle spezifiziert sind. Die Zeilen der Tabelle können in mehrere Abschnitte unterteilt sein. Die verschiedenen Abschnitte müssen alle gleichzeitig gelten, d.h. der Entwickler kann dadurch mehrere Anforderungen modellieren, die gleichzeitig erfüllt sein müssen. Die Zeilen innerhalb eines Abschnittes hingegen repräsentieren alternative Anforderungen (i.d.R. alternative Eingabe-Situationen), von denen in jedem Zeitintervall immer mindestens eine erfüllt sein muss. Sie sind gemäß ihrer Priorität angeordnet, d.h. sind die Eingabebedingungen verschiedener Zeilen eines Abschnitts erfüllt, muss nur die geforderte Reaktion der ersten passenden Zeile eingehalten werden.

2.3.1 Konflikte

Zwei Zeilen (und somit zwei Anforderungen) einer Tabelle stehen in Konflikt zueinander, wenn sie auf (mindestens) eine bestimmte Eingabesequenz unterschiedliche Ausgabereaktionen, d.h. unterschiedliche Nachrichten an einem gemeinsamen Ausgabeport, fordern.

Konflikte können nur zwischen Zeilen verschiedener Abschnitten der Tabelle auftreten, da die Zeilen innerhalb eines Abschnittes nach Prioritäten angeordnete sind. Innerhalb eines Abschnittes muss bei gegebener Eingabesituation nur die erste passende Zeile gelten. Somit kann innerhalb eines Abschnittes kein Konflikt auftreten. Da die Abschnitten hingegen gleichzeitig gelten müssen, ist es möglich, dass zwei Zeilen aus unterschiedlichen Abschnitten im Konflikt zueinander stehen.

Ziel ist es, am Ende eine Gesamtspezifikation zu haben, die in sich konsistent ist, d.h. bei der keine zwei Anforderungen im Konflikt zueinander stehen. Konflikte müssen folglich aufgelöst werden. Dies kann durch die Modifikation der modularen Spezifikationen erfolgen. Resultieren die Konflikte aus der Kombination von zwei Teildiensten (vgl. Kapitel 2.3.2), so bietet sich auch die Einführung von Querbeziehungen an. In den Querbeziehungen wird festgelegt unter welchen Eingabevoraussetzungen welcher Teildienst höhere Priorität hat. Durch diese Prioritätsregelung können Konflikte aufgelöst werden.

2.3.2 Kombination von Diensten

Theoretisch ist es möglich, von Beginn an die Funktionalität des Gesamtsystems in einer Tabelle zu spezifizieren. Aus verschiedenen Gesichtspunkten (Komplexität, Wiederverwendung, Variantenvielfalt,...) ist es jedoch erstrebenswert, die Funktionalität zunächst hierarchisch in Teilfunktionalitäten aufzuspalten. Die elementaren Teildienste werden dann modular in separaten Tabellen spezifiziert. Die modularen Tabellenspezifikation werden zu größeren Blöcken und schließlich zur Gesamtfunktionalität integriert.

In Folgenden wird die Kombination von verschiedenen Tabellenspezifikation erläutert. Dabei unterscheiden wir zwischen der Kombination ohne Querbeziehungen und der Kombination mit Querbeziehungen.

Kombination ohne Querbeziehungen Für die Kombination von zwei Teildiensten ohne Querbeziehungen wird neben den jeweiligen (Teil-)spezifikationen keine zusätzliche Information benötigt. Die Kombination erfolgt systematisch und ist daher automatisierbar. Sie umfasst die folgenden Schritte:

- Vereinigung der syntaktischen Interfaces der Teilspezifikationen.
- Erweiterung der Teilspezifikationen auf das gemeinsame Interface. Die Nachrichten an den zusätzlichen Eingabeports haben keine Auswirkungen auf das Verhalten und an den zusätzlichen Ausgabeports wird nichts gefordert. Dieses beliebige Verhalten wird durch das Zeichen „.“ in den Zellen der zusätzlichen Ports ausgedrückt.
- Parallelkomposition, d.h. Integration er Teilspezifikationen in verschiedene (parallel gültige) Abschnitte der gemeinsamen Tabelle.

Anschließend kann die resultierende Tabellenspezifikation auf Konflikte überprüft werden.

Kombination mit Querbeziehungen Wie bereits erwähnt, können Konflikte zwischen verschiedenen Teildiensten durch Querbeziehungen aufgelöst werden. Diese legen Prioritäten zwischen den Diensten fest, die bei der Kombination berücksichtigt werden müssen.

Formal werden die Querbeziehungen wieder in eigenen Tabellen definiert. Diese sind über das vereinigte syntaktische Interface beider Dienste (bzw. aller Dienste, die von dieser Querbeziehung betroffen sind) definiert. Querbeziehungen definieren keine neue Ausgabe, sondern legen nur fest, unter welcher Eingabebedingung welcher Dienst höhere Priorität hat. Wie normale Tabellen zur Spezifikation von Diensten, enthält der Eingabeteil wieder reguläre Ausdrücke,

die die entsprechenden Eingabesituationen charakterisieren. In den Ausgabezellen wird jedoch kein bestimmtes Verhalten durch die Angabe von regulären Ausdrücken festgelegt, sondern es wird nur auf die entsprechenden Dienste verwiesen, die das Verhalten in der gegebenen Eingabesituation festlegen.

Da diese Querbeziehungen bei der Kombination berücksichtigt werden müssen, ist eine einfache Parallelkomposition nicht mehr möglich. Vielmehr bleiben zwei Möglichkeiten:

1. Erweiterung der modularen Teilspezifikationen um das höherpriore Verhalten der anderen Dienste und Parallelkomposition der erweiterten Teilspezifikationen.
2. Integration der Teilspezifikationen in einen Abschnitt unter Berücksichtigung der Prioritäten.

In Kapitel 4.4.7 werden beide Methoden anhand des Fallbeispiels erläutert.

2.3.3 Produktlinienaspekte

Wie in Kapitel 2.1 beschrieben, eignet sich der verwendete dienstbasierte Ansatz gut um nicht nur einzelne Produkte sondern auch ganze Produktfamilien zu beschreiben. Hierfür muss zwischen optionalen und nicht optionalen Diensten, d.h. Diensten die in jeder Konfiguration verbaut werden müssen, unterschieden werden.

In dem Dienstediagramm werden mit den für Produktlinien üblichen Konzepten (Alternativen, requires- und excludes-Beziehungen, vgl. [7]) definiert, welche Kombinationen von optionalen Diensten zu gültigen Konfigurationen der Produktlinie führen.

In der Tabellennotation ermöglichen wir zudem eine Zuordnung des Verhaltens (der Anforderungen) zu unterschiedlichen Konfigurationen. Hierzu wird die Konfiguration als zusätzliche Eingabe in die Tabellenspezifikation mit aufgenommen. Es wird eine zusätzliche Spalte für einen *Konfigurationsvektor* eingeführt. Der Konfigurationsvektor ist vom Typ $2^{(\mathbb{B}^n)}$, wobei n die Anzahl aller optionalen Dienste ist. Jedes Bit steht für genau einen optionalen Dienst. Der Konfigurationsvektor ordnet einer Zeile genau die Menge von Konfigurationenen zu, die die Anforderung, die in der entsprechenden Zeile formalisiert ist, erfüllen müssen. Anderen Konfigurationen müssen diese Anforderung nicht erfüllen.

In der Tabelle nutzen wir wieder reguläre Ausdrücke um den entsprechenden Konfigurationsvektor zu beschreiben. Ein entsprechender regulärer Ausdruck hat hierbei immer die Länge n. Er bestimmt eine (Menge von) Konfiguration(en), indem er für jeden optionalen Dienst festlegt, ob er in den Konfigurationen ausgewählt sein muss (entsprechendes Zeichen auf 1 gesetzt), nicht ausgewählt sein muss (entsprechendes Zeichen auf 0 gesetzt) oder ob es egal ist, ob er ausgewählt ist oder nicht (entsprechendes Zeichen auf . gesetzt). Angenommen es gibt drei otpionale Dienste in einem System. Der reguläre Ausdruck „.1.“ beschreibt dann alle Konfigurationen, bei denen der zweite optionale Dienst gewählt ist. Der erste und dritte optionale Dienst dürfen in dieser Konfiguration gewählt sein oder nicht. Die Anforderung, die in der entsprechende Zeile der Tabelle formalisiert wird, muss dann für all diese Konfigurationen gelten. Sie muss jedoch nicht erfüllt sein in einer Konfiguration, in der der zweite optionale Dienst nicht gewählt wurde.

In der Tabelle wird nicht explizit spezifiziert, welche Kombinationen nicht möglich sind (auf Grund von Alternativen, requires- oder excludes Beziehung). Wie erwähnt erfolgt dies außerhalb der Tabelle im Dienstediagramm.

3 Anwendungsvoraussetzungen

In diesem Kapitel skizzieren wir kurz die Voraussetzungen, unter denen der eingeführte dienstbasierte Ansatz sinnvoll eingesetzt werden kann. Daneben zeigen wir auch auf, unter welchen Voraussetzungen der Ansatz weniger geeignet ist. Dabei konzentrieren wir uns auf die Einordnung des Ansatzes in bestimmten Phasen des Entwicklungsprozesses und seine Eignung für bestimmte Typen von Systemen. Abschließend zeigen wir noch, was unser Ansatz in Bezug auf eine Produktlinienentwicklung leisten kann.

3.1 Einordnung in den Entwicklungsprozess

Der vorgestellte dienstbasierte Ansatz adressiert die frühen Phasen des Entwicklungsprozesses, die Requirements Engineering Phase. In dieser Phase sind die Informationen über das System oft nur unvollständig bekannt. Diese anfänglichen Unvollständigkeit kann durch die Partialität der Dienste in unserem Ansatz gut abgebildet werden.

Unser Ansatz ist besonders für die späteren Phasen des Requirements Engineering Prozesses geeignet, wenn die relevanten Anforderungen bereits weitgehend erhoben wurden. Er dient nicht nur der Modellierung der vorliegenden Anforderungen an das System, sondern insbesondere der Analyse der Anforderungen: Die bereits erhobenen funktionalen Anforderungen können mittels Diensten formalisiert und analysiert werden. Durch die Analysen können zum einen fehlende Anforderungen und zum anderen Konflikte in den existierenden Anforderungen aufgezeigt werden. Daher unterstützt unser Ansatz neben der Analyse gleichzeitig auch die Erhebung von neuen Anforderungen.

Der dienstbasierte Ansatz führt letztenendes zu einer formalen Anforderungsspezifikation, die präziser, eindeutiger und konsolidierter ist als gängige, textuelle Anforderungsspezifikationen. Daher stellt eine dienstbasierte Spezifikation eine geeignete, überprüfbare Ausgangsbasis für die nachgelagerten Entwicklungstätigkeiten, insbesondere den Aufbau einer logischen Architektur dar. Auch weitere Phasen wie Test oder Wartung profitieren von der verbesserten Spezifikation. Testfälle können direkt aus der Spezifikation generiert werden oder in anderen Worten: die Implementierung kann direkt gegen die Spezifikaiton getestet werden.

Der dienstbasierte Ansatz unterstützt jedoch nicht den Entwurf der Systemarchitektur. Auch die hierarchische Strukturierung in Dienste entspricht nur einer Strukturierung nach Kundenfunktionalitäten und keiner Strukturierung nach fachlichen Zuständigkeiten, wie für eine logische Architektur gefordert. Für das Design einer guten Architektur sind nicht-funktionale Anforderungen ausschlaggebend. Diese bleiben in dem vorgestellten Ansatz bislang völlig unberücksichtigt.

Zusammengefasst beschäftigt sich der dienstbasierte Ansatz mit der Formalisierung von funktionalen Anforderungen an das Systemverhalten. Der Ansatz ist geeignet für die späten Phasen

des Requirements Engineerings (insbesondere Analyse von Anforderungen), unterstützt aber auch die Erhebung von Anforderungen und stellt eine geeignete Ausgangsbasis für die späteren Phasen des Entwicklungsprozesses dar.

3.2 Systemtypen

Der Schwerpunkt des dienstbasierten Ansatzes liegt auf der Strukturierung, Spezifikation und Analyse der Systemverhaltens. Er ist in erster Linie für die Spezifikation von *multi-funktionalen Systemen* geeignet. Unter einem multi-funktionalen System versteht man ein System, das mehrere einander beeinflußende Funktionen anbietet. Diese Funktionalitäten können mittels einzelner Dienste spezifiziert werden. Abhängigkeiten zwischen den Funktionalitäten können durch formale Analysen identifiziert werden und durch die Einführung von Querbeziehungen explizit dargestellt werden. Natürlich kann der dienstbasierte Ansatz auch zur Spezifikation einer einzelnen Funktion verwendet werden. Allerdings liefert er hier keinen Mehrwert, da sein großer Nutzen in der Analyse und Modellierung der Zusammenspiels von Funktionalitäten liegt.

Der dienstbasierte Ansatz wurde entwickelt zur Spezifikation von diskreten Kontrollsystemen aus dem Bereich der eingebetteten Systemen. Ein *diskretes System* ist eine technische oder organisatorische Einheit mit einer klar definierten Systemgrenze, über die es mit seiner Umgebung mittels diskreter Events interagiert. In dem hier vorgestellten Ansatz sind die diskreten Events die Nachrichten, die in einem Zeitintervall über die Ports empfangen bzw. gesendet werden. Für kontinuierliche Systeme, wie z.B. für den Entwurf von Regelungen, bzw. für hybride System ist unser Ansatz dagegen nicht geeignet. Wir gehen davon aus das kontinuierliche Anteile, wie Regler, bereits in diskretisierter Form zur Verfügung stehen.

Bei der dienstbasierten Spezifikation gehen wir, wie gerade erwähnt, von einer fest definierten Systemgrenze aus und spezifizieren, strukturieren und analysieren das Systemverhalten, wie es an der Systemgrenze beobachtbar ist. Der Ansatz konzentriert sich somit auf die Spezifikation der Interaktion zwischen dem eingebetteten System und seiner Umgebung. Die Umgebung kann hierbei ein Mensch (Benutzerinteraktion), aber auch ein anderes mechanisches System sein. Es wird keine Systemdekomposition im Sinne einer Architektur durchgeführt. Falls der Schwerpunkt darauf liegt, die Interaktion von Teilsystemen abzubilden, ist der dienstbasierte Ansatz nicht unmittelbar geeignet. Im Falls einer bereits vorgegebenen Aufteilung des Systems in Teilsystemen, die intensiv miteinander kommunizieren müssen, ist es sinnvoller zwei getrennte Spezifikationen für jedes der Teilsysteme zu erstellen, mit einem fest definierten Interface zwischen diesen.

Neben dem hier vorliegenden und weiteren Fallbeispielen aus der Automobildomäne, wurde der Ansatz auch für die Spezifikation weiterer Fallbeispiele z.B. aus der Automatisierungstechnik erfolgreich eingesetzt. Die Anwendung des Ansatzes ist somit nicht beschränkt auf die Automobildomäne, sondern geeignet beliebige Kontrollsysteme aus dem Bereich der eingebetteten Systeme zu spezifizieren.

Dank der Modularität und Kompositionalität ist der dienstbasierte Ansatz auch zur Spezifikation komplexer Systeme geeignet. Lediglich die Spezifikationen der einzelnen Teilfunktionalitäten sind manuell zu erstellen. Die Spezifikation des Gesamtsystems kann automatsisch

generiert werden. Die Entwicklung kann hierbei auch verteilt (sowohl räumlich als auch organisatorisch) stattfinden. Einzelne Funktionalitäten können zunächst unabhängig voneinander von mehreren Teams und aus verschiedenen Perspektiven spezifiziert werden. Anschließend können die getrennt modellierten Teil-Spezifikationen dank der Kompositionalität unserers Ansatzes problemlos integriert werden.

Im Bereich der eingebetteten Systeme sind insbesondere auch technische Randbedingungen von Bedeutung. Wie andere nicht-funktionale Eigenschaften werden auch technische Details sowie Implementierungsdetails in dem dienstbasierten Ansatz ausgeblendet. Des weiteren ist unser Ansatz nicht geeignet komplexe algorithmische Berechnungen abzubilden. Auch für betriebliche Informationssysteme ist unser Ansatz nicht geeignet, da die für diesen Systemtyp nötige Datenmodellierung bei unserem Ansatz nicht im Vordergrund steht.

3.3 Produktlinien

Die Entwicklung von Produktlinien, d.h. von Familien von ähnlichen Produkten anstelle von Einzelprodukten, wird durch den dienstbasierten Ansatz dank seiner Modularität und Kompositionalität wie folgt unterstützt: Variable Teilfunktionalitäten können als eigene optionale Teildienste spezifiziert werden, die bei Bedarf in das System integriert werden können oder auch nicht. Auch die explizite Modellierung von Abhängigkeiten zwischen Diensten als Querbeziehungen unterstützt die Entwicklung von Produktfamilien. Eine Anpassung der modularen Teilverhalten zur Auflösung von Konflikten zwischen optionalen Diensten wäre nicht sinnvoll, da die Dienste auch unabhängig voneinander verbaut werden können. Querbeziehungen hingegen müssen nur berücksichtigt werden, wenn die beteiligten optionalen Dienste ausgewählt wurden. Andernfalls müssen sie nicht beachtet werden.

Wie in [6] ausführlich beschrieben, entsteht eine dienstbasierte Spezifikation mit optionalen Diensten (und optinonalen Querbeziehungen), die nicht ein einziges System sondern eine Menge von ähnlichen Systemen beschreibt. Neben *optionalen* Diensten, die optionale Anforderungen formalisieren und somit nicht von jeder gültigen Konfiguration erfüllt sein müssen, führen wir weiter das Konzept von *alternativen* Diensten sowie *requires-* und *exclude*-Beziehungen zwischen optionalen Diensten ein. Diese legen näher fest, welche Dienste von gültigen Konfiguration erfüllt sein müssen. Es wurde weiter gezeigt, wie die Konsistenz der gesamten Produktlinie abgesichert werden kann. Das Verhalten eines Variationspunktes wurde hierfür als Vereinigung der Verhalten seiner variablen Teildienste unter Berücksichtigung evtl. requires-/excludes Beziehungen definiert. Diese Konzepte wurden bisher allerdings nur theoretisch eingeführt und an keinem größeren Fallbeispiel erprobt.

In diesem Dokument wird zusätzlich für die tabellarische Notationstechnik gezeigt, wie die Zuordnung von gewissen Anforderungen zu optionalen Diensten explizit vorgenommen werden kann. Dies ermöglicht es, sehr einfach die Spezifikation für eine Konfiguration (ein konkretes Produkt) aus der Spezifikaiton der ganzen Produktlinie abzuleiten. Allerdings ist bei diesem Vorgehen nur noch die Konsistenzanalyse für einzelne Produkte nicht mehr für die gesamte Produktlinie möglich.

Im Gegensatz zu vielen Ansätzen, die sich mit Produktlinien befassen, werden mit dem dienstbasierten jedoch keine neuen Konzepte zur Modellierung von gültigen Konfigurationen, d.h.

Erweiterungen von klassischen Feature Trees (siehe [7]), eingeführt. Die um optionale Dienste, Alternativen und requires-/excludes-Beziehungen erweiterten Dienstediagramme, die die hierarchische Strukturierung der Funktionalität in Teildienste graphisch repräsentieren, entsprechen im Wesentlichen den klassischen Feature Trees. Diese können dann mit den gängigen Methoden formalisiert werden, z.B. mittels Grammatiken (siehe [4]).

Wie dargestellt, wird variables Verhalten lediglich in optionale Dienste ausgegliedert, die ausgewählt werden können oder nicht. Es wurden bisher keine tragfähigen Konzepte zur expliziten Spezifikation von Verhaltensvarianz erarbeitet.

4 Fallstudie

Das elektronische Fahrzeugsystem „Adaptive Cruise Control ACC“ von Bosch (siehe [1]) ermöglicht eine kombinierte Geschwindigkeits- und Abstandskontrolle. Wie ein normaler Fahrgeschwindigkeitsregeler (Tempomat) lässt das ACC das Fahrzeug mit einer gewählten Geschwindigkeit fahren. Holt das eigene Fahrzeug ein vorausfahrendes Fahrzeug ein, bremst das ACC automatisch ab und hält einen vom Fahrer festgelegten Abstand. Sobald sich im Messbereich kein vorausfahrendes Fahrzeug mehr befindet, beschleunigt das ACC das Auto wieder auf die gewählte Geschwindigkeit.

Die Anforderungen an das ACC-System sind im Wesentlichen dem Dokument [1] entnommen, allerdings an einigen Stellen leicht angepasst oder vereinfacht, um die Verständlichkeit der Fallstudie zu erhöhen.

In den folgenden Kapiteln werden die verschiedenen Teilfunktionalitäten des ACC-Systems näher beschrieben und als Dienste modelliert. Abbildung 2, das Dienstediagramm für das ACC, zeigt die hierarchische Dekomposition der Gesamtfunktionalität in Teildienste und die Abhängigkeiten zwischen diesen. Diese Teilfunktionalitäten werden in den anschließenden Ka-

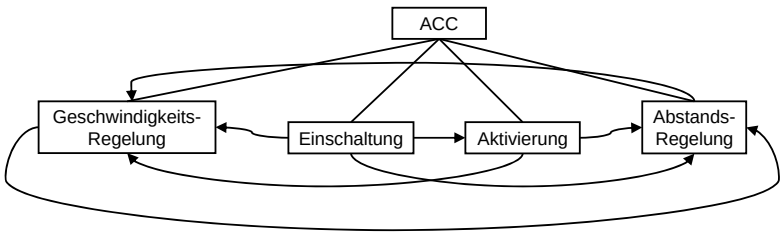

Abbildung 2: Dienstediagramm `ACC`

piteln zunächst getrennt voneinander spezifiziert. Für die `Einschaltung`, `Aktivierung` und `Geschwindigkeitsregelung` kommt dabei die automatenbasierte Notationstechnik aus Kapitel 2.2 zum Einsatz. Für die `Abstandsregelung` hingegen wird die tabellarische Notation aus Kapitel 2.3 verwendet. In Kapitel 4.5 werden die Teilfunktionalitäten zur Gesamtfunktionalität `ACC` integriert.

Alle Ports des syntaktischen Interfaces des Gesamtsystems sowie die zugeordneten Datentypen sind in Tabelle 28 auf Seite 43 zu finden.

4.1 Einschaltung

Das ACC ist vom Fahrer aktiv einzuschalten. Bei einigen Fahrzeugmodellen ist das ACC zunächst erst einmal durch einen Hauptschalter freizuschalten. Bei anderen Modellen befindet es sich gleich mit dem Drehen des Zündschlüssels in Position „Zündung an“ im passiven Wartezustand. Nach der Einschaltung befindet sich das ACC in einem „Stand by“ Zustand: der Fahrer kann die Wunschgeschwindigkeit und die Wunschzeitlücke eingeben, das ACC regelt die Geschwindigkeit jedoch nicht. Zusätzlich ist im ACC ein Mechanismus zur Spannungsüberprüfung eingebaut. Sinkt die gemessene Versorgungsspannung unter einen bestimmten Spannungswert ab, dann wird das ACC ausgeschaltet.

Abbildung 3 stellt das Dienstediagramm für die Teil-Funktionalität `Einschaltung` dar. Im

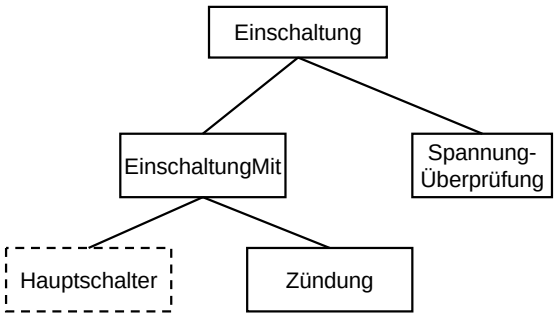

Abbildung 3: Dienstediagramm `Einschaltung`

Folgenden sind einzelne Teil-Dienste mittels I/O Automaten (vgl. Kapitel 2.2) spezifiziert.

4.1.1 Einschaltung mit Zündschlüssel

Befindet sich die Zündung in Position „Zündung aus“, darf das ACC keine Steuersignale an ihre Umgebung schicken. Der Dienst aus Abbildung 4 und Tabelle 1 formalisiert dieses Szenario. Sind in einer Transition bestimmte Ports nicht angegeben, sind sie bei diesem Zustandsüber-

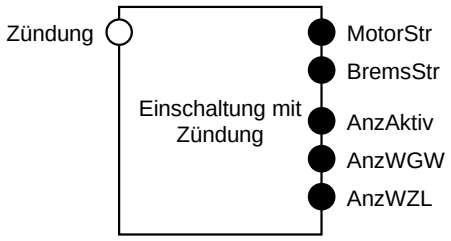

Abbildung 4: Einschaltung mit Zündschlüssel (Interface)

gang nicht spezifiziert – jeder beliebige Wert aus dem Porttyp ist zugelassen. Dieser Dienst ist

Nr.	Von	Hin	Transition
01	zAus	zAus	Zündung?ε/MotorStr!ε;BremsStr!ε;AnzWGW!ε;AnzWZL!ε;AnzAktiv!ε
02	zAus	zAn	Zündung?an/
03	zAn	zAus	Zündung?ε/MotorStr!ε;BremsStr!ε;AnzWGW!ε;AnzWZL!ε;AnzAktiv!ε

Tabelle 1: Einschaltung mit Zündschlüssel (I/O Automat)

nichtdeterministisch. Der Nichtdeterminismus wird jedoch in Kombination mit anderen Diensten (zumindest teilweise) aufgelöst. Der Automat hat zwei Zustände: `zAn` (die Zündung ist angeschaltet) und `zAus` (die Zündung ist ausgeschaltet). Befindet sich der Dienst im Zustand `zAn`, sind alle Output-Ports unterspezifiziert. Befindet sich der Dienst im Zustand `zAus`, dürfen an den angegebenen Output-Ports keine Nachrichten anliegen.

4.1.2 Einschaltung mit Hauptschalter

Der optionale Dienst `Hauptschalter` aus Abbildung 2 ist dem Dienst `Zündung` sehr ähnlich. Er hat einen Input-Port `Schalter` und dieselben Output-Ports wie der Dienst `Zündung`. Liegt an dem Input-Port keine Nachricht an(d.h. das ACC ist mit dem Schalter ausgeschaltet), dürfen an den Output-Ports keine Nachrichten anliegen. Liegt an dem Input-Port die Nachricht `an`, sind die Output-Ports unterspezifiziert.

Ein optionaler Dienst bedeutet, dass nicht jede gültige Implementierung diesen Dienst erfüllen muss. Die Anforderung, die dieser Dienst formalisiert ist optional. Mehr dazu in [6].

4.1.3 Spannungsüberprüfung

Sinkt die gemessene Versorgungsspannung unter einen bestimmten Spannungswert ab, unterbricht das ACC seine Arbeit – es darf keine Steuersignale an seine Umgebung schicken. Ist die Versorgungsspannung wieder im erlaubten Bereich, darf das ACC seine Regelung wieder aufnehmen. Der Dienst mit dem Interface aus Abbildung 5 und dem Verhalten aus Tabelle 2 formalisiert dieses Szenario. Solange der aktuelle Spannungswert den minimal erlaubten Wert

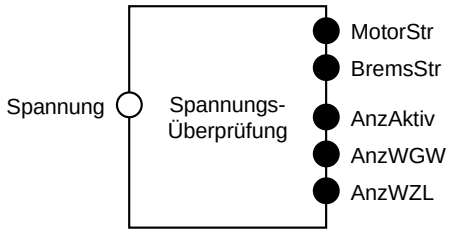

Abbildung 5: Spannungsüberprüfung (Interface)

nicht unterschreitet, ist der Dienst im Zustand `OK`, ansonsten im Zustand `Fehler`. Im Zustand `OK` sind die Ausgaben nicht spezifiziert – jeder Wert aus dem Porttyp ist gültig. Im Zustand `Fehler` dürfen keine Nachrichten an den Output-Ports anliegen.

Nr.	Von	Hin	Transition
01	Fehler	Fehler	{x<minSpannung}Spannung?x/MotorStr!ε;BremsStr!ε;AnzWGW!ε; AnzWZL!ε;AnzAktiv!ε
02	Fehler	OK	{x≥minSpannung}Spannung?x/
03	OK	Fehler	{x<minSpannung}Spannung?x/MotorStr!ε;BremsStr!ε;AnzWGW!ε; AnzWZL!ε;AnzAktiv!ε

Tabelle 2: Spannungsüberprüfung (I/O Automat)

4.1.4 Kombinierter Dienst `Einschaltung`

Die Kombination der beiden Dienste `EinschaltungMitZündung` und `Spannungsüberprüfung` ergibt den kombinierten Dienst `Einschaltung` aus Abbildung 6 und Tabelle 3. Das Interface des Dienstes ist die Vereinigung der beiden Interfaces aus Abbildungen 4 und 5. Der kombinierte Automat ist die Kombination der beiden Automaten aus Tabellen 1 und 2.

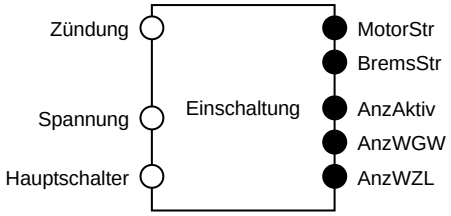

Abbildung 6: Kombinierter Dienst `Einschaltung` (Interface)

4.2 Aktivierung

Grundlegende Bedingung für die Aktivierung der Geschwindigkeitsregelung sind:

- Fahrgeschwindigkeit größer als minimale Wunschgeschwindigkeit,
- Bremspedal nicht getreten.

Sobald die Bedingungen für eine Aktivierung erfüllt sind und der Fahrer einen für die Aktivierung vorgesehenen Bedienknopf betätigt, beginnt das ACC die Geschwindigkeit zu regeln. Die für die Aktivierung vorgesehenen Knöpfe sind folgende: `Set`, `Set+`, `Set-`, und `Resume`. Ihre genauere Funktionalität ist in Kapitel 4.3.1 beschrieben. Die Deaktivierung erfolgt durch das Betätigen eines Ausschalters (Knopf `I/O`) oder des Bremspedals. Weitere Bedingung für eine Deaktivierung ist das Unterschreiten der minimalen Geschwindigkeit. Nach der Deaktivierung befindet sich das ACC im Zustand „stand by“.

Der Dienst mit dem Interface aus Abbildung 7 und dem Verhalten aus Tabelle 4 formalisiert diese Szenarios. Falls an einem der Input-Ports `Resume`, `Set`, `Set+` oder `Set-` eine Nachricht anliegt, die aktuelle Geschwindigkeit (`aktGW`) größer als die minimal erlaubte Geschwindigkeit ist, das Bremspedal nicht getreten ist (am Port `Bremse` liegt ein ε an) und das ACC nicht aktiv ausgeschaltet wird (am Port `I/O` ein ε), geht der Dienst in den Zustand `aktiv` über

Nr.	Von	Hin	Transition
01	zAus/OK	zAus/OK	{x≥minSpannung}Spannung?x;Zündung?ε/ MotorStr!ε;BremsStr!ε;AnzWGW!ε;AnzWZL!ε;AnzAktiv!ε
02	zAus/OK	zAus/Fehler	{x<minSpannung}Spannung?x;Zündung?ε/ MotorStr!ε;BremsStr!ε;AnzWGW!ε;AnzWZL!ε;AnzAktiv!ε
03	zAus/OK	zAn/OK	{x≥minSpannung}Spannung?x;Zündung?an/
04	zAus/OK	zAn/Fehler	{x<minSpannung}Spannung?x;Zündung?an/ MotorStr!ε;BremsStr!ε;AnzWGW!ε;AnzWZL!ε;AnzAktiv!ε
05	zAus/Fehler	zAus/OK	{x≥minSpannung}Spannung?x;Zündung?ε/ MotorStr!ε;BremsStr!ε;AnzWGW!ε;AnzWZL!ε;AnzAktiv!ε
06	zAus/Fehler	zAus/Fehler	{x<minSpannung}Spannung?x;Zündung?ε/ MotorStr!ε;BremsStr!ε;AnzWGW!ε;AnzWZL!ε;AnzAktiv!ε
07	zAus/Fehler	zAn/OK	{x≥minSpannung}Spannung?x;Zündung?an/
08	zAus/Fehler	zAn/Fehler	{x<minSpannung}Spannung?x;Zündung?an/ MotorStr!ε;BremsStr!ε;AnzWGW!ε;AnzWZL!ε;AnzAktiv!ε
09	zAn/OK	zAus/OK	{x≥minSpannung}Spannung?x;Zündung?aus/ MotorStr!ε;BremsStr!ε;AnzWGW!ε;AnzWZL!ε;AnzAktiv!ε
10	zAn/OK	zAus/Fehler	{x<minSpannung}Spannung?x;Zündung?ε/ MotorStr!ε;BremsStr!ε;AnzWGW!ε;AnzWZL!ε;AnzAktiv!ε
11	zAn/OK	zAn/Fehler	{x<minSpannung}Spannung?x;Zündung?ε/ MotorStr!ε;BremsStr!ε; AnzWZL!ε;AnzAktiv!ε
12	zAn/Fehler	zAus/OK	{x≥minSpannung}Spannung?x;Zündung?aus/ MotorStr!ε;BremsStr!ε;AnzWGW!ε;AnzWZL!ε;AnzAktiv!ε
13	zAn/Fehler	zAus/Fehler	{x<minSpannung}Spannung?x;Zündung?ε/ MotorStr!ε;BremsStr!ε;AnzWGW!ε;AnzWZL!ε;AnzAktiv!ε
14	zAn/Fehler	zAn/OK	{x≥minSpannung}Spannung?x;Zündung?ε/AnzAktiv!standBy
15	zAn/Fehler	zAn/Fehler	{x<minSpannung}Spannung?x;Zündung?ε/ MotorStr!ε;BremsStr!ε;AnzWGW!ε; AnzWZL!ε;AnzAktiv!ε

Tabelle 3: Kombinierter Dienst `Einschaltung` (I/O Automat)

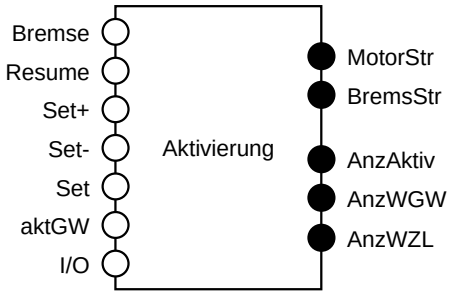

Abbildung 7: Aktivierung der Geschwindigkeitsregelung (Interface)

(Transition 01). In dem Zustand muss am Port `AnzAktiv` die Nachricht `aktiv` anliegen. Alle anderen Output-Ports sind unterspezifiziert. Der Dienst bleibt in dem Zustand, solange keine der oben genannten Bedingungen verletzt wird (Transition 04). Wurde mindestens eine der Bedingungen verletzt, geht der Dienst in den Zustand `passiv` über (Transition 03). In dem

Nr.	Von	Hin	Transition
01	passiv	aktiv	$\{g > minGW \wedge \neg(x = \varepsilon \wedge y = \varepsilon \wedge v = \varepsilon \wedge w = \varepsilon)\}$ Bremse?ε;aktGW?g;Resume?x;Set?y;Set+?v;Set-?w;I/O?ε / AnzAktiv!aktiv
02	passiv	passiv	$\{b \neq \varepsilon \vee g < minGW \vee (x = \varepsilon \wedge y = \varepsilon \wedge v = \varepsilon \wedge w = \varepsilon) \vee z = aus\}$ Bremse?b;aktGW?g;Resume?x;Set?y;Set+?v;Set-?w;I/O?z / AnzAktiv!standBy;MotorStr!ε;BremsStr!ε;AnzWGW!ε;AnzWZL!ε;
03	aktiv	passiv	$\{b \neq \varepsilon \vee g < minGW \vee z = aus\}$ Bremse?b;aktGW?g;I/O?z / AnzAktiv!standBy;MotorStr!ε;BremsStr!ε;AnzWGW!ε;AnzWZL!ε;
04	aktiv	aktiv	$\{g > minGW\}$ Bremse?ε;aktGW?g;I/O?ε/ AnzAktiv!aktiv

Tabelle 4: Aktivierung der Geschwindigkeitsregelung (I/O Automat)

Zustand muss am Port `AnzAktiv` die Nachricht `standBy` und an allen anderen Output-Ports keine Nachrichten anliegen. In diesem Zustand bleibt der Dienst, solange das ACC nicht wieder aktiviert wird (Transition 02).

Man beachte, dass der Dienst `Aktivierung` im Wesentlichen das Verhalten des *deaktivierten* ACC spezifiziert. Dabei erzwingt er, dass an den Ports `MotorStr`, `BremsStr` und `AnzWGW` keine Nachrichten anliegen. Das Verhalten des *aktivierten* ACC ist jedoch unterspezifiziert – beliebige Nachrichten können an den Output-Ports anliegen. Dieses Verhalten wird dann mit den weiteren Diensten verfeinert (siehe nächstes Kapitel).

4.3 Geschwindigkeitsregelung

Der Fahrer stellt an den Bedienelementen eine gewünschte Fahrzeuggeschwindigkeit ein. Daraufhin gleicht die Regelung die momentane Fahrzeuggeschwindigkeit an diese Wunschgeschwindigkeit an. Abbildung 8 stellt das Dienstediagramm der Teil-Funktionalität `Geschwindigkeitsregelung` dar. Die Funktionalität besteht aus drei Teil-Funktionalitäten: der Benutzer-Inter-

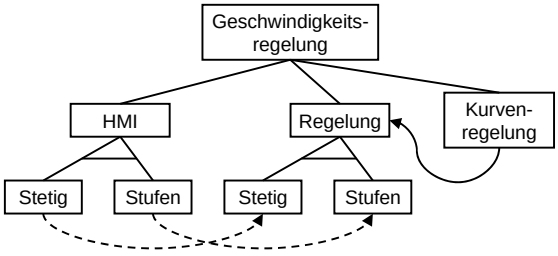

Abbildung 8: Dienstediagramm `Geschwindigkeitsregelung`

aktion (Dienst `HMI`) zum Setzen und Anzeigen der Wunschgeschwindigkeit, der Regelung der Geschwindigkeit (Dienst `Regelung`) und der Regelung der Geschwindigkeit in einer Kurve (Dienst `Kurvenregelung`). Zwischen den Diensten `Regelung` und `Kurvenregelung` besteht eine Priorisierungsbeziehung. Das ACC gibt es in zwei (*alternativen*) Varianten: Die Wunschgeschwindigkeit kann entweder in 1km/h- oder 10km/h-Einheiten eingegeben werden. Entsprechend sind Dienste `HMI` und `Regelung` in jeweils zwei alternative Dienste (`Stetig` und `Stufen`)

aufgeteilt. Zwischen ihnen besteht jeweils eine *requires*-Beziehung. Im Folgenden sind einzelne Dienste und Beziehungen mittels I/O Automaten (vgl. Kapitel 2.2) spezifiziert.

4.3.1 HMI: Setzen und Anzeigen der Wunschgeschwindigkeit (stufenweise)

Man beachte, dass dieses Szenario keine Geschwindigkeitsregelung spezifiziert, sondern die Benutzerinteraktion beim Setzen und Anzeigen der Wunschdaten. Die eigentliche Regelung wird in Kapitel 4.3.3 spezifiziert.

Für das Setzen der Wunschgeschwindigkeit gibt es vier Möglichkeiten:

1. Übernahme der Ist-Geschwindigkeit als Wunschgeschwindigkeit (Taste `Set`).
2. Übernahme der zur Ist-Geschwindigkeit nächst höheren Stufe (Taste `Set+`). Wahl der Wunschgeschwindigkeit in Schritten von 10 km/h nach oben.
3. Übernahme der zur Ist-Geschwindigkeit nächst niedrigeren Stufe (Taste `Set-`). Wahl der Wunschgeschwindigkeit in Schritten von 10 km/h nach unten.
4. Übernahme der gespeicherten Wunschgeschwindigkeit (Taste `Resume`).

Der Dienst mit dem Interface aus Abbildung 9 und dem Verhalten aus Tabelle 5 spezifiziert diese Szenarios. Die ersten vier Transitionen des Automaten formalisieren genau die vier oben beschriebenen Szenarios. Abhängig von der aktuellen Geschwindigkeit (dem Wert am Port `aktGW`) und davon welche der vier Tasten gedrückt ist, wird der entsprechende Wert in einer lokalen Variable `lGW` gespeichert und am Port `AnzWGW` ausgegeben. Ist keine der Tasten gedrückt, wird am Port `AnzWGW` der gespeicherte Wert ausgegeben. Da der I/O Automat aus Tabelle 5 nur einen Zustand hat, wird in der Tabelle die Angabe der Ausgangs- und Ziel-Zustände weggelassen.

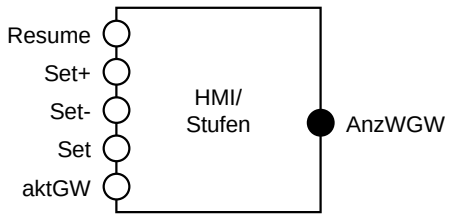

Abbildung 9: HMI/Stufen (Interface)

Nr.	Transition
01	$\{x \neq \varepsilon\}$aktGW?g;Resume?ε;Set?x;Set+?ε;Set-?ε/AnzWGW!g {lGW=g}
02	$\{x \neq \varepsilon\}$aktGW?g;Resume?ε;Set?ε;Set+?x;Set-?ε/AnzWGW!$(g+10)$ {lGW=$(g+10)$}
03	$\{x \neq \varepsilon\}$aktGW?g;Resume?ε;Set?ε;Set+?ε;Set-?x/AnzWGW!$(g-10)$ {lGW=$(g-10)$}
04	$\{x \neq \varepsilon\}$Resume?x;Set?ε;Set+?ε;Set-?ε/AnzWGW!lGW
05	Resume?ε;Set?ε;Set+?ε;Set-?ε/AnzWGW!lGW

Tabelle 5: HMI/Stufen (I/O Automat)

Man beachte, dass der hier beschriebene Dienst partiell ist – es gibt Eingaben, auf die er keine definierten Ausgaben produzieren kann. Bei der Spezifikation wurde angenommen, dass keine zwei Tasten gleichzeitig (in einem Takt) gedrückt werden können. D.h., falls in einem Takt zwei Nachrichten an zwei Input-Ports anliegen, hat der Dienst kein definiertes Verhalten – jede mögliche Ausgabe ist dann gültig.

4.3.2 HMI: Setzen und Anzeigen der Wunschgeschwindigkeit (stetig)

Genau wie in Kapitel 4.3.1. Die Stufen sind allerdings nicht 10 sondern 1 km/h.

4.3.3 Regelung der Wunschgeschwindigkeit (stufenweise)

Die Regelung gleicht die momentane Fahrzeuggeschwindigkeit an die vom Fahrer eingestellte Wunschgeschwindigkeit an.

Der Dienst mit dem Interface aus Abbildung 10 und dem Verhalten aus Tabelle 6 formalisiert dieses Szenario. Abhängig von der aktuellen Geschwindigkeit (dem Wert am Port `aktGW`) und dem welche der drei Tasten `Set`, `Set+` und `Set-` gedrückt ist, wird der entsprechende Wert in einer lokalen Variable `lGW` gespeichert und an den Ports `MotorStr` und `BremsStr` die Soll-Werte ausgegeben. Die Soll-Werte werden mit den folgenden zwei Funktionen berechnet.

$$g(x,y) = \begin{cases} maxVerz & \text{falls } (x-y) > 70 \\ starkVerz & \text{falls } 30 > (x-y) \leq 70 \\ verz & \text{falls } 10 > (x-y) \leq 30 \\ \varepsilon & \text{sonst} \end{cases}$$

$$f(x,y) = \begin{cases} verz & \text{falls } 0 < (x-y) \leq 10 \\ beschl & \text{falls } (x-y) < 0 \\ \varepsilon & \text{sonst} \end{cases}$$

Da der I/O Automat nur einen Zustand hat, wird in der Tabelle die Angabe der Ausgangs- und Ziel-Zustände weggelassen. Genau wie der Dienst aus Kapitel 4.3.1, ist der Dienst `Regelung`

Abbildung 10: Geschwindigkeitsregelung (Interface)

partiell – es wird angenommen, dass nur eine Taste gleichzeitig gedrückt werden kann.

Nr.	Transition
01	$\{x \neq \varepsilon\}$ aktGW?y;Set?x;Set+?ε;Set-?ε / MotorStr!$f(y, lGW)$; BremsStr!$g(y, lGW)$ $\{lGW = y\}$
02	$\{x \neq \varepsilon\}$ aktGW?y;Set?ε;Set+?x;Set-?ε / MotorStr!$f(y, lGW)$; BremsStr!$g(y, lGW)$ $\{\text{lGW}=(y + 10)\}$
03	$\{x \neq \varepsilon\}$ aktGW?y;Set?ε;Set+?ε;Set-?x / MotorStr!$f(y, lGW)$; BremsStr!$g(y, lGW)$ $\{\text{lGW}=(y - 10)\}$
04	aktGW?y;Set?ε;Set+?ε;Set-?ε / MotorStr!$f(y, lGW)$; BremsStr!$g(y, lGW)$

Tabelle 6: Geschwindigkeitsregelung (I/O Automat)

4.3.4 Regelung der Wunschgeschwindigkeit (stetig)

Genau wie in Kapitel 4.3.3. Die Stufen sind allerdings nicht 10 sondern 1 km/h.

4.3.5 Kurvenregelung

Befindet sich das Fahrzeug in einer Kurve, darf das ACC das Fahrzeug nicht beschleunigen. Das physikalische Messprinzip beruht auf der Winkelmessung an der Lenksäule. Ist der Winkel der Lenksäule größer als 7%, darf am Port `MotorStr` keine Nachricht anliegen. Der Dienst mit dem Interface aus Abbildung 11 und dem Verhalten aus Tabellen 7 spezifiziert dieses Szenario. Der Dienst ist partiell – für alle Eingaben mit $lRWinkel \leq 7$ hat er kein definiertes Verhalten, d.h., alle Ausgaben am Port `MotorStr` sind gültig.

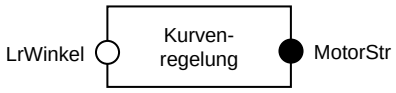

Abbildung 11: Kurvenregelung (Interface)

Nr.	Transition
01	$\{y > 7\}$ lRWinkel?y/MotorStr!ε;

Tabelle 7: Kurvenregelung (I/O Automat)

4.3.6 Priorisierung: Kurvenregelung vs. Regelung

Die Kurvenregelung hat eine höhere Priorität als die Geschwindigkeitsregelung immer dann, wenn der Winkel der Lenksäule größer ist als 7%. Die Priorisierung mit dem Interface aus Abbildung 12 und dem Verhalten aus Tabelle 8 formalisiert diese Anforderung. Immer wenn der Priorisierungsautomat einen Schritt machen kann (d.h. wenn der Wert am Port `lRWinkel` größer 7 ist), muss der Automat aus Tabelle 7 einen Schritt machen können. Der Automat aus Tablle 6 bleibt stehen und stellt somit keine Anforderungen an die Output-Ports aus Abbildung 10.

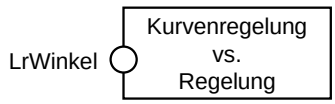

Abbildung 12: Priorisierung: Kurvenregelung vs. Regelung (Interface)

Nr.	Transition
01	$\{y > 7\}$ lRWinkel?y

Tabelle 8: Priorisierung: Kurvenregelung vs. Regelung (I/O Automat)

4.3.7 Kombination: `HMI` und `Regelung`

Die Kombination der Dienste `HMI/Stufen` und `Regelung/Stufen` ergibt den kombinierten Dienst aus Abbildung 13. Dabei handelt es sich um die einfache Kombination aus Kapitel 2.2.2. Da die beiden Dienste keine gemeinsamen Output Ports haben, ist der kombinierte Automat

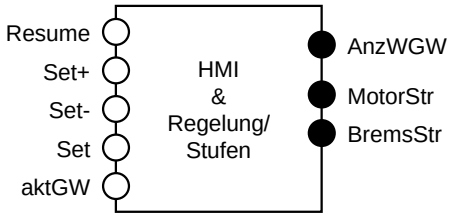

Abbildung 13: Kombinierter Dienst: `HMI/Stufen` & `Regelung/Stufen` (Interface)

ein Produktautomat der beiden Automaten aus Tabellen 5 auf Seite 18 und 6 auf Seite 20. Auf seine Darstellung wird an dieser Stelle verzichtet.

4.3.8 Kombination: `Regelung` und `Kurvenregelung`

Die Kombination der Dienste `Regelung/Stufen` und `Kurvenregelung` unter Berücksichtigung der Priorisierung aus Kapitel 4.3.6 ergibt den kombinierten Dienst aus Abbildung 14 und Tabelle 9. Dabei handelt es sich um eine priorisierte Komposition aus Kapitel 2.2.3. Das syntaktische Interface des kombinierten Dienstes ist die Vereinigung der Interfaces aus Abbildungen 10 auf Seite 19 und 11 auf Seite 20. Der kombinierte Automat ist die Kombination der Automaten aus Tabellen 6 auf Seite 20 und 7 auf Seite 20 unter Berücksichtigung der Priorisierung aus Kapitel 4.3.6. Entsprechend dieser Priorisierung hat der Dienst `Kurvenregelung` immer Vorrang, wenn am Port `LrWinkel` ein Wert größer 7 anliegt (in diesem Fall darf am Port `MotorStr` keine Nachricht anliegen). Im kombinierten Automaten sind das die ersten vier Transitionen. In allen anderen Fällen gilt die Regel der normalen Komposition (die letzten vier Transitionen).

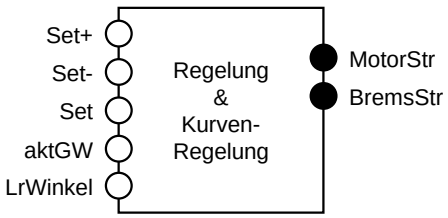

Abbildung 14: Kombinierter Dienst: `Regelung/Stufen` & `Kurvenregelung` (Interface)

Nr.	Transition
01	$\{x \neq \varepsilon \wedge y \leq 7\}$ aktGW?y;Set?x;Set+?ε;Set-?ε;LrWinke?y / MotorStr!$f(y, lGW)$; BremsStr!$g(y, lGW)$ $\{lGW = y\}$
02	$\{x \neq \varepsilon \wedge y \leq 7\}$ aktGW?y;Set?ε;Set+?x;Set-?ε;LrWinke?y / MotorStr!$f(y, lGW)$; BremsStr!$g(y, lGW)$ {lGW=$(y + 10)$}
03	$\{x \neq \varepsilon \wedge y \leq 7\}$ aktGW?y;Set?ε;Set+?ε;Set-?x;LrWinke?y / MotorStr!$f(y, lGW)$; BremsStr!$g(y, lGW)$ {lGW=$(y - 10)$}
04	$\{y \leq 7\}$ aktGW?y;Set?ε;Set+?ε;Set-?ε;LrWinke?y / MotorStr!$f(y, lGW)$; BremsStr!$g(y, lGW)$
05	$\{x \neq \varepsilon \wedge y > 7\}$ aktGW?y;Set?x;Set+?ε;Set-?ε;LrWinke?y / MotorStr!ε; BremsStr!$g(y, lGW)$ $\{lGW = y\}$
06	$\{x \neq \varepsilon \wedge y > 7\}$ aktGW?y;Set?ε;Set+?x;Set-?ε;LrWinke?y / MotorStr!ε; BremsStr!$g(y, lGW)$ {lGW=$(y + 10)$}
07	$\{x \neq \varepsilon \wedge y > 7\}$ aktGW?y;Set?ε;Set+?ε;Set-?x;LrWinke?y / MotorStr!ε; BremsStr!$g(y, lGW)$ {lGW=$(y - 10)$}
08	$\{y > 7\}$ aktGW?y;Set?ε;Set+?ε;Set-?ε;LrWinke?y / MotorStr!ε; BremsStr!$g(y, lGW)$

Tabelle 9: Kombinierter Dienst: `Regelung/Stufen` und `Kurvenregelung` (I/O Automat)

4.3.9 Kombinierter Dienst Geschwindigkeitsregelung

Die Kombination der Dienste `HMI`, `Regelung` und `Kurvenregelung` ergibt den kombinierten Dienst `Geschwindigleitsregelung` (vgl. Abbildung 8). Da die Kombination nach demselben Schema wie in den Kapiteln 4.3.7 und 4.3.8 erfolgt, wird an dieser Stelle die Darstellung des Gesamtautomaten weggelassen.

4.3.10 Produktlinie

Wie bereits erwähnt, sind die Dienste `HMI/Stetig` und `HMI/Stufen` sowie `Regelung/Stetig` und `Regelung/Stufen` aus Abbildung 2 zwei Paare von alternativen Diensten. Somit beschreibt ein Dienstediagramm mehrere mögliche Produkte einer Produktlinie. Eine gültige Implementierung muss jeweils nur einen der alternativen Dienste erfüllen. Zwischen den Diensten `HMI/Stetig` und `Regelung/Stetig` sowie `HMI/Stufen` und `Regelung/ Stufen` besteht jeweils eine *Excludes*-Beziehung. Das bedeutet, dass wenn eine gültige Implementierung den Dienst `HMI/Stetig` (`HMI/Stufen` entsprechend) erfüllt, muss sie auch den Dienst `Regelung/Stetig` (`Regelung/ Stufen`) erfüllen. Mehr dazu in [6].

4.4 Abstandsregelung

Wird ein vorausfahrendes Objekt erkannt, das langsamer fährt als die eingestellte Wunschgeschwindigkeit, so stellt die Abstandsregelung des ACC sicher, dass das ACC-Fahrzeug dem langsamer fahrenden Fahrzeug folgt. Fährt das vorausfahrende Fahrzeug mit konstanter Geschwindigkeit, so folgt auch das ACC-Fahrzeug mit der gleichen Geschwindigkeit in dem ausgewählten Wunschabstand. Der Abstand ist dabei proportional zur Geschwindigkeit. Dies entspricht einer konstanten Zeitlücke. Diese kann von dem Benutzer eingestellt werden, je nach Fahrzeugtyp kontinuierlich oder stufenweise. Der eingestellte Wunschabstand (genauer: die eingestellte Wunschzeitlücke) wird dem Fahrer angezeigt.

Die Basisfunktionalität `Abstandsregelung Normalbetrieb` wird bzw. kann optional um folgende Funktionalitäten erweitert werden:

- `Abstandsregelung nach Einscheren eines vorausfahrenden Fahrzeugs`: Das ACC erkennt Situationen in denen ein anderes Fahrzeug, z.B. nach einem Überholvorgang, kurz vor dem eigenen Fahrzeug einschert. Die Situation „Einscheren“ wird erkannt durch Unterschreiten eines Minimalabstands zum vorausfahrenden Fahrzeug. Fährt das vorausfahrende Fahrzeug deutlich schneller als das eigene Fahrzeug (positive Relativgeschwindigkeit) so ist keine starke Bremsreaktion erforderlich, sondern nur ein langsames Einregeln des Abstands. Bei negativer Relativgeschwindigkeit hingegen soll das Fahrzeug schnell abbremsen. Zusätzlich wird ein Warnsignal an den Fahrer ausgegeben. Diese Funktionalität ist nicht optional, sondern in jeder ACC-Variante verbaut.
- `Abstandsregelung bei angestrebtem Überholvorgang`: Wird ein vorausfahrendes Fahrzeug erkannt, aber der Blinker ist links gesetzt, so erkennt das ACC einen angestrebten Überholvorgang. Daher bremst das ACC nicht ab, auch wenn der eingestellte Wunschabstand unterschritten wird, sondern beschleunigt und leitet somit den Überholvorgang ein. Dem Fahrer wird diese Situation durch Ausgabe eines Warnsignals signalisiert. Der Beschleunigungsvorgang wird abgebrochen, wenn der Minimalabstand unterschritten wird. Bei der Abstandsregelung bei angestrebtem Überholvorgang handelt es sich um eine optionale Erweiterung der Basisfunktionalität.
- `Pre-Crash Regelung`: Diese optionale Teilfunktionalität `Pre-Crash Regelung` regelt das Verhalten bei einem potentiellem Crash, d.h. wenn der kritischer Minimalabstand unterschritten ist. Hierbei werden zwei Varianten unterschieden.
  - Die Variante `Pre-Crash Warning` gibt bei Erkennung eines potentiellen Crashs eine Warnung aus und deaktiviert das System vorübergehend.
  - Die Variante `Pre-Crash Bremsen` gibt bei Erkennung eines potentiellen Crashs ebenfalls eine Warnung aus, leitet aber zudem eine Vollbremsung ein.

Neben den Regelungsdiensten umfasst die Abstandsregelung weitere Teildienste für die Benutzerinteraktion (`HMI`). Die Benutzerinteraktion umfasst folgende Teilfunktionalitäten:

- `Einstellung des Wunschabstands:` Damit die Abstandsregelung aktiv sein kann, muss der Fahrer einen Wunschabstand (genauer eine Wunschzeitlücke) einstellen können. Der eingestellte Wert wird dem Fahrer angezeigt. Hierfür gibt es zwei Varianten: die Eingabe der Wunschzeitlücke kann kontinuierlich oder stufenweise erfolgen.

- `Anzeige Objekterkennung:` Des Weiteren wird dem Fahrer signalisiert, ob das System ein Objekt in Fahrtrichtung erkannt hat und somit die Abstandsregelung aktiv ist oder nicht.

Abbildung 15, das Dienstediagramm für die Abstandsregelung, zeigt die hierarchische Dekomposition der Abstandsregelung in ihre Teildienste und die funktionalen Abhängigkeiten (Querbeziehungen) zwischen den Teildiensten. Optionale Dienste sind hierbei gestrichelt dargestellt.

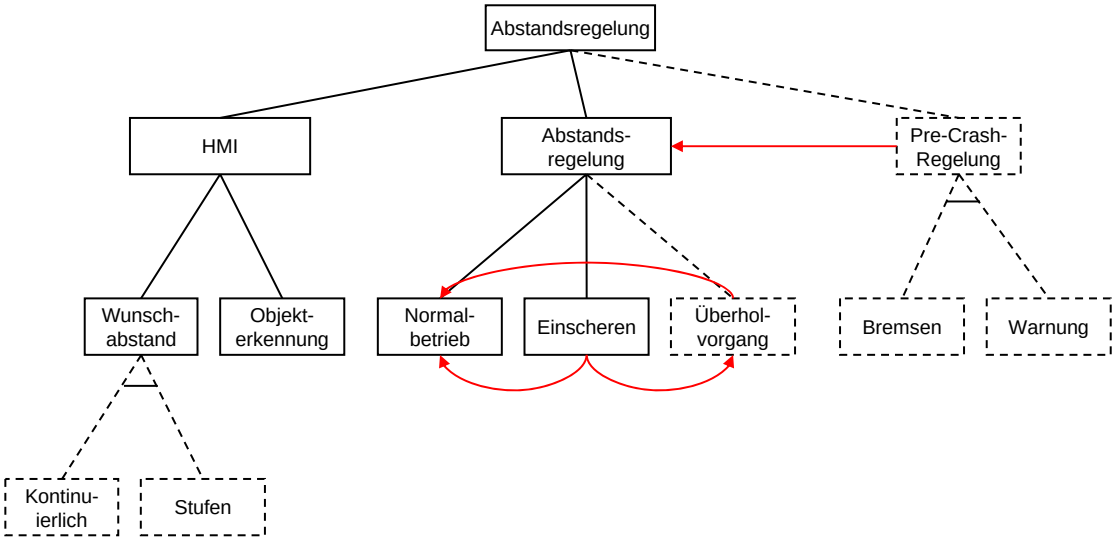

Abbildung 15: Dienstediagramm für die Abstandsregelung

Die Spezifikation der einzelnen Teildienste erfolgt in den folgenden Kapiteln. Im Gegensatz zur Geschwindigkeitsregelung, kommt hierbei die in Kapitel 2.3 eingeführte tabellarische Notation zum Einsatz.

4.4.1 HMI – Bedienung und Anzeige

Wie für die Geschwindigkeitsregelung (siehe Kapitel 4.3) sind auch für die Abstandsregelung einige Benutzerinteraktionen notwendig. Die notwendige Schnittstelle zu dem ACC wird wieder über verschiedene Bedien- und Anzeigeelemente realisiert.

Die Benutzerinteraktion (`HMI`) umfasst die `Kontinuierliche Einstellung des Wunschabstands`, die `Stufenweise Einstellung des Wunschabstands` sowie die `Anzeige Objekterkennung`. Die einzelnen Teilfunktionalitäten werden in den folgenden Abschnitten genauer beschrieben.

Kontinuierliche Einstellung des Wunschabstands Abbildung 16 zeigt das syntaktische Interface des Dienstes `Kontinuierliche Einstellung des Wunschabstands`. Die Typen der verwendeten Ports können der Tabelle 28 auf Seite 43 entnommen werden.

Abbildung 16: Kontinuierliche Einstellung des Wunschabstands (Interface)

Über den Eingabeport `WZL` kann eine beliebige Wunschzeitlücke im Bereich von 1-3 Sekunden eingestellt werden (z.B. über ein Drehrad). Es wird angenommen, dass an dem Port `WZL` in jedem Zeitintervall ein Signal anliegt. Es ist nicht definiert, wie sich das System verhalten soll, wenn kein Signal anliegt. Dies muss in einer spätere Implementierung ergänzend definiert werden oder es wird z.B. durch die mechanische Realisierung sichergestellt, dass in jedem Takt ein Signal anliegt. Der eingestellt Wert für die Wunschzeitlücke wird dem Fahrer angezeigt (Output-Port `AnzWZL`). Durch Multiplikation mit der aktuellen Geschwindigkeit des eigenen Fahrzeuges ergibt sich hieraus der geforderte Wunschabstand zu einem vorausfahrenden Fahrzeug.

Die entsprechende tabellarische Verhaltensspezifikation ist in Tabelle 10 dargestellt.

`WZL`	`AnzWZL`
x	x

Tabelle 10: Kontinuierliche Einstellung des Wunschabstands (Tabellarische Spezifikation)

Liegt in einem Zeitintervall t an dem am Port `WZL` ein beliebiger Wert x an, so wird genau dieser Wert im nächsten Zeitintervall $t+1$ auf den Port `AbzWZL` ausgegeben. x ist hierbei eine Variable des entsprechenden Typs der Ports `WZL` bzw. `AbzWZL`.

Stufenweise Einstellung des Wunschabstands Die Einstellung des Wunschabstands kann auch diskret erfolgen. Die Wunschzeitlücke kann über entsprechende Benutzereingaben stufenweise erhöht bzw. erniedrigt werden. Der Wunschabstand wird dem Benutzer angezeigt und bei Änderung abgespeichert. Der Dienst hat somit zwei Eingabeports,

- den Port `WZL` auf dem die gespeicherte Wunschzeitlücke abgenommen werden kann und
- den Port `Step` für die Benutzereingabe,

und zwei Ausgabeports,

- den Port `WZL` auf den eine geänderte Wunschzeitlücke zur Speicherung ausgegeben werden kann und
- den Port `AnzWZL` zur Anzeige der aktuellen Wunschzeitlücke.

Das syntaktische Interface ist in Abbildung 17 dargestellt. Die Typen der verwendeten Ports können wieder der Tabelle 28 auf Seite 43 entnommen werden.

Die formale Verhaltensspezifikation ist in Tabelle 11 dargestellt.

Abbildung 17: Stufenweise Einstellung des Wunschabstands (Interface)

Step	WZL	AnzWZL	WZL
.	ϵ	3	3
+	$x \vert x \in \{1,2\}$	$x+1$	$x+1$
−	$x \vert x \in \{2,3\}$	$x-1$	$x-1$
.	x	x	x

Tabelle 11: Stufenweise Einstellung des Wunschabstands (Tabellarische Spezifikation)

Liegt keine Wunschzeitlücke an (WZL $= \epsilon$) – z.B. nach Systemstart – wird zunächst die Wunschzeitlücke auf den Defaultwert 3 Sekunden gesetzt. Dies entspricht dem größt möglichen Abstandswert. Diese Wunschzeitlücke wird angezeigt und abgespeichert, d.h. an beiden Ausgabeports AnzWZL und WZL wird im nächsten Takt eine 3 ausgegeben. Der Einstellung des Defaultwert ist hierbei unabhängig von einer möglichen, gleichzeitigen Benutzereingabe am Port Step (Step = .). Dies ist in der ersten Zeile der Tabelle 11 formalisiert. Die folgenden beiden Zeilen der Tabelle 11 beschreiben das Erhöhen bzw. Erniedrigen des Wunschabstands über die Eingabe am Port Step. Hat die Wunschzeitlücke noch nicht den Maximalwert von 3 Sekunden erreicht, d.h. ist die WZL gleich dem Wert $x = 1$ oder $x = 2$, so kann sie über die Eingabe Step $= +$ inkrementiert werden. Der entsprechend erhöhte Wert $x + 1$ wird im nächsten Takt auf die Ports AnzWZL und WZL ausgegeben. Analog kann über die Eingabe Step $= -$ die Wunschzeitlücke dekrementiert werden, sofern sie nicht auf bereits den Minimalwert von 1 Sekunden eingestellt ist. Die letzte Zeile formalisiert alle restlichen Fälle. Dies umfasst die Möglichkeiten, dass keine Eingabe am Port Step anliegt oder bereits der entsprechende Maximal- bzw. Minimalwert erreicht ist. In diesen Fällen wird die eingehende Wunschzeitlücke x unverändert auf die Ausgabeports ausgegeben.

Anzeige Objekterkennung Neben der eingestellten Wunschzeitlücke wird dem Fahrer angezeigt, ob ein vorausfahrendes Objekt erkannt wurde. Über den Eingabeport Obj wird dem System ein Objekt im Fahrschlauch signalisiert. Sobald ein Objekt erkannt ist, wird dies dem Fahrer auf dem Ausgabeport AnzObj angezeigt. Das syntaktische Interface dieses Dienstes ist in Abbildung 18 dargestellt, die Typen der Ports finden sich wieder in Tabelle 28 auf Seite 43, die zugehörige Verhaltensspezifikation in Tabelle 12.

Der Ausdruck $ja\ \epsilon^*$ in der Spalte Obj in der ersten Zeile der Tabelle 12 beschreibt, dass bis zum Zeitintervall t am Port Obj als letztes Signal ja und danach beliebig oft ϵ, d.h. kein Signal, empfangen wurde. In diesen Fällen wird dem Fahrer die Objekterkennung angezeigt. Es wird ja auf dem Port AnzObj ausgegeben.

Ist das letzte Statuskanal, das bis zum Zeitintervall t am Port Obj empfangen wurde, *nein* (be-

Abbildung 18: Anzeige Objekterkennung (Interface)

`Obj`	`AnzObj`
$ja\ \epsilon^*$	ja
$nein\ \epsilon^*$	$nein$

Tabelle 12: Anzeige Objekterkennung (Tabellarische Spezifikaiton)

schrieben durch den regulären Ausdruck $nein\ \epsilon^*$), so wird dem Fahrer keine Objekterkennung angezeigt. Es wird $nein$ auf dem Port `AnzObj` ausgegeben.

Man beachte, dass am Eingabe-Port nicht in jedem Zeitintervall ein Signal anliegen muss (`Obj` $= \epsilon$ zum Zeitintervall t möglich), dem Fahrer aber in jedem Zeitintervall der letzte bekannte Zustand signalisiert wird. Es ist nicht definiert, welche Ausgabe anliegt, wenn auch in den vorhergehenden Zeitintervallen nie ein gültiger Objekterkennungszustand (nie $Obj = nein$ bzw. $Obj = ja$) signalisiert wurde. Bei der späteren Umsetzung muss dieses Verhalten ergänzend definiert werden.

4.4.2 Abstandsregelung Normalbetrieb

Die Abstandsregelung soll eine möglichst schnelle Ausregelung der Abweichungen vom Sollwert realisieren. Auf geringe Abstands- und Geschwindigkeitsänderungen soll die Abstandsregelung träge reagieren, um einen möglichst hohen Komfort für den Fahrer zu erreichen. Die Entscheidung erfolgt anhand der Relativgeschwindigkeit zum vorausfahrenden Fahrzeug.

Für die Abstandsregelung werden folgende Inputports benötigt:

- `Obj` auf dem signalisiert wird, ob ein Objekt erkannt ist,
- `aktGW` auf dem die aktuelle Geschwindigkeit des eigenen Fahrzeuges anliegt,
- `GWObj` auf dem die aktuelle Geschwindigkeit des vorausfahrenden Fahrzeugs gemeldet wird,
- `WZL` auf dem die eingestellte Wunschgeschwindigkeit anliegt und
- `Abstand` auf dem der tatsächliche Abstand zu dem vorausfahrenden Fahrzeug anliegt.

Es wird auf folgende Outputports geschrieben:

- `MotorStr` auf dem Befehle an die Motorsteuerung weitergeleitet werden und
- `BremsStr` auf dem Befehle an die elektronische Bremssteuerung weitergeleitet werden, da eine stärkere Verzögerung des Fahrzeugs nicht mehr über die Motorsteuerung realisiert werden kann.

Das syntaktisches Interface dieses Dienstes ist in Abbildung 19 dargestellt. Die konkreten Typen der Ports können Tabelle 28 auf Seite 43 entnommen werden.

Abbildung 19: Abstandsregelung Normalbetrieb (Interface)

Tabelle 13 beschreibt das Verhalten dieses Dienstes.

`Obj`	`aktGW`	`GWObj`	`WZL`	`Abstand`	`MotorStr`	`BremsStr`
$ja\ \epsilon^*$	v	$v_o \vert v_o > v$	x	$d \vert d < x * v$	ϵ	ϵ
$ja\ \epsilon^*$	v	$v_o \vert v_o = v$	x	$d \vert d < x * v$	$verz$	ϵ
$ja\ \epsilon^*$	v	$v_o \vert v_o < v$	x	$d \vert d < x * v$	ϵ	$verz$
.	.	.	.	.	.	.

Tabelle 13: Abstandsregelung Normalbetrieb (Tabellarische Spezifikation)

Wurde ein Objekt erkannt, das den Wunschabstand unterschreitet (`Abstand` $<$ `WZL` $*$ `aktGW`) hängt die Reaktion von der Realtivgeschwindigkeit ab. Bei positiver Relativgeschwindigkeit, d.h. wenn das vorausfahrende Fahrzeug schneller als das eigene Fahrzeug fährt, wird aus Komfortgründen auf ein Abbremsen verzichtet. Es werden keine Befehle an Motor- bzw. Bremssteuerung geschickt ($MotorStr = BremsStr = \epsilon$). Die Objekterkennung wird wieder nicht kontinuierlich, sondern in unregelmäßigen Abständen signalisiert. Die erste Zeile der Tabelle 13 formalisiert dieses Verhalten: Wenn in einem Zeitintervall t

- `Obj` $= ja\ \epsilon^*$: ein Objekt erkannt wurde,
- `aktGW` $= v$: die aktuelle Geschwindigkeit den Wert v hat,
- `GWObj` $= v_o \vert v_o > v$: das vorausfahrende Objekt mit Geschwindigkeit v_o fährt, wobei v_o größer als die aktuelle Geschwindigkeit des eigenen Fahrzeuges v ist (positive Relativgeschwindigkeit),
- `WZL` $= x$: die Wunschzeitlücke auf den Wert x gesetzt ist und
- `Abstand` $= d \vert d < x * v$: der aktuelle Abstand den Wert d beträgt, wobei d den Wunschabstand $x * v$ unterschreitet, der sich aus der eingestellten Wunschzeitlücke x und der aktuellen Geschwindigkeit v berechnet,

dann gilt für die Ausgabeports im nächsten Zeitintervall $t + 1$

- `MotorStr` $= \epsilon$: es wird kein Signal an die Motorsteuerung geschickt und

- `BremsStr` $= \epsilon$: es wird auch kein Signal an die Bremssteuerung geschickt.

Analog formalisieren die folgenden zwei Zeilen der Tabelle 13 das Verhalten bei neutraler bzw. negativer Relativgeschwindigkeit, d.h. wenn das vorausfahrende Fahrzeug genauso schnell bzw. langsamer als das eigene Fahrzeug fährt (`GWObj` $= v_o | v_o = v$ bzw. `GWObj` $= v_o | v_o < v$, ansonsten wie oben). Bei neutraler Relativgeschwindigkeit soll leicht abgebremst werden, um den Wunschabstand wieder herzustellen. Dies kann über die Motorsteuerung erreicht werden ($MotorStr = verz$, $BremsStr = \epsilon$). Die negativer Relativgeschwindigkeit, d.h. der Wunschabstand ist schon unterschritten und das eigene Fahrzeug fährt noch schneller als das vorausfahrende Fahrzeug, ist eine stärkere Bremsreaktion erforderlich. Diese kann nicht mehr über die Motorsteuerung erreicht werden, daher $MotorStr = \epsilon$ und $BremsStr = verz$.

Die letzte Zeile vervollständigt die Spezifikation. Sie legt fest, dass in allen anderen Fällen (beliebiger Wert „.“ an allen Eingabeports), die Ausgaben an den Ports `MotorStr` und `BremsStr` nicht näher definiert ist (beliebiger Wert „.“ gefordert).

4.4.3 Abstandsregelung nach Einscheren eines vorausfahrenden Fahrzeuges

Schert ein Fahrzeug knapp vor dem eigenen Fahrzeug ein, kann es passieren, dass ein minimaler Sicherheitsabstand (Konstante x_{min} multipliziert mit der aktuellen Geschwindigkeit) unterschritten wird. In diesem Fall ist eine stärkere Bremsreaktion als im Nomalfall erforderlich um möglichst schnell einen komfortablen Sicherheitsabstand wieder herzustellen. Zudem soll eine Warnung an den Fahrer ausgegeben werden.

Das syntaktische Interface, dargestellt in Abbildung 20, benötigt im Vergleich zu `Abstandsregelung Normalbetrieb` den Eingangsport `WZL` nicht, da das Unterschreiten des minimalen Abstands unabhängig von dem eingestellten Wunschabstand ist. Zusätzlich ist jedoch der Ausgabeport `Warnung` erforderlich (vgl. 19). Die konkreten Typen der Ports finden sich wieder in Tabelle 28 auf Seite 43.

Abbildung 20: Abstandsregelung nach Einscheren (Interface)

Die Verhaltensspezifikation dieses Dienstes findet sich in Tabelle 14.

Wie beim Normalbetrieb hängt die Reaktion von der Relativgeschwindigkeit ab. Bei positiver Relativgeschwindigkeit wird nur leicht verzögert. Hierzu wird das Signal $verz$ an die Motorsteuerung geschickt. Zusätzlich soll der Fahrer gewarnt werden (`Warnung` $= ja$). Dieses Verhalten wird in der ersten Zeile der Tabelle 14 formalisiert: Wenn in einem Zeitintervall t

- `Obj` $= ja\ \epsilon^*$: ein Objekt erkannt wurde,
- `aktGW` $= v$: die aktuelle Geschwindigkeit den Wert v hat,

Obj	aktGW	GWObj	Abstand	MotorStr	BremsStr	Warnung
$ja\ \epsilon^*$	v	$v_o \vert v_o > v$	$d \vert d < x_{min} * v$	$verz$	ϵ	ja
$ja\ \epsilon^*$	v	$v_o \vert v_o = v$	$d \vert d < x_{min} * v$	ϵ	$verz$	ja
$ja\ \epsilon^*$	v	$v_o \vert v_o < v$	$d \vert d < x_{min} * v$	ϵ	$starkVerz$	ja
.	.	.	.	.	.	$nein$

Tabelle 14: Abstandsregelung nach Einscheren (Tabellarische Spezifikation)

- `GWObj` $= v_o \vert v_o > v$: das vorausfahrende Objekt schneller als das eigene Fahrzeug fährt, d.h. mit einer Geschwindigkeit v_o fährt, die größer als die aktuelle Geschwindigkeit des eigenen Fahrzeuges v ist (positive Relativgeschwindigkeit) und
- `Abstand` $= d \vert d < x_{min} * v$: der aktuelle Abstand d einen Minimalabstand $x_{min} * v$ unterschreitet, der sich auf Basis der aktuellen Geschwindigkeit v berechnet,

dann gilt für die Ausgabeports im nächsten Zeitintervall $t + 1$

- `MotorStr` $= verz$: die Motorsteuerung soll eine Verzögerung verursachen,
- `BremsStr` $= \epsilon$: es wird kein Signal an die Bremssteuerung geschickt und
- `Warnung` $= ja$: es wird eine Warnung an den Fahrer ausgegeben.

Analog formalisieren die folgenden zwei Zeilen das Verhalten bei neutraler bzw. negativer Relativgeschwindigkeit. Bei neutraler Relativgeschwindigkeit soll stärker verzögert werden. Daher wird das Signal $verz$ an die Bremssteuerung (und ϵ an die Motorsteurung) geschickt Bei negativer Relativgeschwindigkeit muss aufgrund des geringen Abstands stark verzögert werden. Hierzu wird $starkVerz$ an die Bremssteuerung (und ϵ an die Motorsteurung) ausgegeben. In beiden Fällen wird ein Warnsignal ausgegeben, um den Fahrer auf den geringen Abstand aufmerksam zu machen.

Die letzte Zeile der Tabelle 14 formalisiert das Verhalten in den restlichen Fällen, die nicht durch die bereits beschriebenen Szenarien abgedeckt sind, d.h. Zeitintervalle in denen kein Objekt erkannt oder der minimale Abstand nicht unterschritten ist. In diesen Fällen soll keine Warnung ausgegeben werden (`Warnung` $= nein$). Das Verhalten an den Ports `MotorStr` oder `BremsStr` wird für diese Fälle nicht eingeschränkt (`MotorStr` = `BremsStr` = .).

4.4.4 Abstandsregelung bei angestrebtem Überholvorgang

Das ACC-System erkennt an einem gesetzten Links-Blinker die Überholabsicht des Fahrers. In diesem Fall soll auch bei Unterschreiten des Wunschabstands nicht abgebremst sondern beschleunigt werden. Der Fahrer muss über diesen Vorgang informiert werden: Es wird ein Warnsignal ausgegeben. Dieser Dienst ist optional.

Das syntaktische Interface ist in Abbildung 21 dargestellt. Im Vergleich zum syntaktischen Interface der `Abstandsregelung Normalbetrieb` enthält es den zusätzlichen Eingabeport `Blinker`. Der Eingabeport `GWObj` entfällt, da hier das Verhalten unabhängig von der Realtivgeschwindigkeit zum vorausfahrenden Fahrzeug ist. Als zusätzlicher Ausgabeport kommt wieder

der Port `Warnung` hinzu (vgl. Abbildung 19). Die konkreten Typen der Ports können Tabelle 28 auf Seite 43 entnommen werden.

Abbildung 21: Abstandsregelung Überholvorgang (Interface)

Die Verhaltensspezifikation ist in der nachfolgenden Tabelle 15 dargestellt. Die Formalisierung ist analog zu den vorausgehenden Tabellen 13 und 14. Die erste Zeile formalisiert das Verhalten, falls zwar ein Objekt erkannt und der eingestellte Wunschabstand unterschritten ist, aber der Linksblinker gesetzt (`Blinker` = $links$) ist. In diesem Fall soll beschleunigt werden (`MotorStr` = $beschl$) und eine Warnung (`Warnung`= ja) ausgegeben werden. In allen anderen Fällen soll keine Warnung ausgegeben werden (`Warnung` = $nein$). Das Verhalten an den Ports `MotorStr` und `BremsStr` ist nicht weiter beschränkt. Es darf ein beliebiges Signal anliegen (`MotorStr` = `BremsStr` = .)

Obj	aktGW	WZL	Abstand	Blinker	MotorStr	BremsStr	Warnung
$ja\ \epsilon^*$	v	x	$d \vert d \leq x * v$	$links$	$beschl$	ϵ	ja
.	.	.	.	.	.	.	$nein$

Tabelle 15: Abstandsregelung Überholvorgang (Tabellarische Spezifikation)

4.4.5 Pre Crash Warnung

Optional umfasst das ACC eine Pre Crash Regelung. Diese gibt es in zwei Varianten. In beiden Fällen wird bei unterschreiten eines kritischen Abstands, der auf einen potentiellen Crash hindeutet (Konstante $x_{critical}$ multipliziert mit der aktuellen Geschwindigkeit), eine Warnung an den Fahrer ausgegeben. In der ersten Ausbaustufe wird zudem die Abstandsregelung vorübergehend deaktiviert: es wird kein Signal an die Motor- bzw. Bremssteuerung ausgegeben. In der zweiten Ausbaustufe wird eine Vollbremsung eingeleitet.

Beide Varianten haben dasselbe syntaktische Interface, wie in Abbildung 22 dargestellt. Die konkreten Typen der Ports sind in Tabelle 28 auf Seite 43 aufgelistet.

Die Verhaltensspezifikation der beiden Varianten unterscheiden sich hingegen. Bei der Variante `Pre-Crash Warning` wird bei unterschreiten des kritischen Abstands eine Warnung ausgegeben und das System kurzfristig deaktiviert, d.h. kein Signal an die Brems- bzw. Motorsteuerung weitergegeben. Die formale Verhaltensspezifikation findet sich in Tabelle 16.

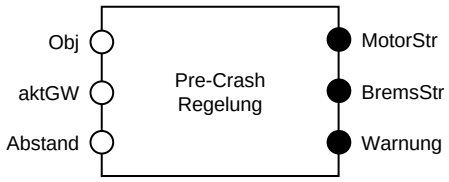

Abbildung 22: Pre-Crash Warning bzw. Pre-CrashBremsen (Interface)

Obj	$aktGW$	$Abstand$	$MotorStr$	$BremsStr$	$Warnung$
$ja\ \epsilon^*$	v	$d \vert d \leq x_{critical} * v$	ϵ	ϵ	ja
.	.	.	.	.	.

Tabelle 16: Pre-Crash Warning (Tabellarische Spezifikation)

Bei der Variante `Pre-Crash Bremsen` wird neben der Ausgabe der Warnung eine Vollbremsung eingeleitet. Hierzu wird das Signal $maxVerz$ auf den Ausgabeport `BremsStr` ausgegeben. Die entsprechende formale Verhaltensspezifikation findet sich in Tabelle 17.

Die Tabellen 16 und 17 sind analog zu den vorausgehenden Spezifikationen aufgebaut. Für eine nähere Erläuterung sei daher auf die Erklärungen zu den Tabellen 13, 14, 15 verwiesen.

4.4.6 Querbeziehungen zwischen den Teildiensten

Zwischen den in den vorausgegangenen Abschnitten dargestellten Teildiensten gibt es im Wesentlichen folgende Querbeziehungen:

- die `Pre-Crash Regelung` (beide Varianten) hat eine höhere Priorität als die anderen Abstandsregelungsdienste,
- die `Abstandsregelung nach Einscheren` hat die höchste Priorität zwischen den Abstansregelungsdiensten, die `Abstandsregelung bei Überholvorgang` die nächst höhere Priorität.

Formal werden die Querbeziehungen wieder in Tabellen erfasst. Sie legen fest unter welchen Bedingungen welcher Dienst eine höhere Priorität hat. Die Querbeziehungen sprechen über das vereinigte syntaktische Interface beider Dienste bzw. aller Dienste, die von dieser Querbeziehung betroffen sind. Aus Gründen der Lesbarkeit können Eingabekanäle, die für die Querbeziehung irrelevant sind, ausgeblendet werden.

Obj	$aktGW$	$Abstand$	$MotorStr$	$BremsStr$	$Warnung$
$ja\ \epsilon^*$	v	$d \vert d \leq x_{critical} * v$	ϵ	max_verz	ja
.	.	.	.	.	.

Tabelle 17: Pre-Crash Bremsen (Tabellarische Spezifikation)

Beziehung zwischen Pre Crash Regelung (PCR) und den (drei) Abstandsregelungsdiensten Wird der kritische Abstand unterschritten, so fordert der Pre-Crash Regelungsdienst (unabhängig davon welche Variante) anderes Verhalten als die Abstandsregelungsdienste. In der Variante ohne Vollbremsung fordert er eine Deaktivierung (Ausgabe ϵ an den Ports `MotorStr` und `BremsStr`), in der Variante mit Vollbremsung maximale Bremskraft (Ausgabe $maxVerz$ an dem Port `BremsStr` und ϵ an dem Port `MotorStr`). Dieses Verhalten ist bei keinem der Abstandsregelungsdienste vorgesehen. Die Dienste stehen also in einem Konflikt zueinander. Dieser Konflikt wird durch Einführung einer Querbeziehung aufgelöst: Die Pre-Crash Regelungsvarianten haben höhere Priorität als die Abstandsregelungsdienste. Wann immer der kritische Abstand unterschritten wird, muss das entsprechende Verhalten der gewählten Pre-Crash Variante ausgeführt werden. Dies gilt für alle Ausgabeports `MotorStr`, `BremsStr` und `Warnung`. Wie dieses Verhalten genau aussieht, ist nicht in der Querbeziehung spezifiziert, sondern in der modularen Spezifikation der Pre-Crahs Varianten. Die Tabelle legt nur die Verknüpfung der Teildienste fest. Die beschriebene Querbeziehung ist dargestellt in Tabelle 18.

Obj	aktGW	Abstand	MotorStr	BremsStr	Warnung
$ja\ \epsilon^*$	v	$d \vert d \leq x_{critical} * v$	PCR	PCR	PCR
.	.	.	.	.	.

Tabelle 18: Querbeziehung zwischen der Pre-Crash Regelung und der Abstandsregelung

Die erste Zeile der Tabelle definiert folgenden Zusammenhang: Wenn in einem Zeitintervall t

- `Obj` $= ja\ \epsilon^*$: ein Objekt erkannt wurde,
- `aktGW` $= v$: die aktuelle Geschwindigkeit den Wert v hat,
- `Abstand` $= d \vert d < x_{critical} * v$: der aktuelle Abstand den Wert d beträgt, wobei d den kritischen Abstand $x_{critical} * v$ unterschreitet, der sich basierend auf der aktuellen Geschwindigkeit v berechnet,

dann soll das Verhalten im nächsten Zeitintervall $t + 1$

- `MotorStr` = `PCR`: am Ausgabeport `MotorStr` der Spezifikation der Pre-Crash Regelung `PCR` folgen und
- `BremsStr` = `PCR`: am Ausgabeport `BremsStr` ebenfalls der Spezifikation der Pre-Crash Regelung `PCR` folgen.

In den Fällen, in denen die Querbeziehung keine Aussagen macht (hier alle anderen Fälle), müssen weiterhin *alle* kombinierten Querdienste gleichzeitig gelten. Die zweite Zeile formalisiert also, dass in allen anderen Fällen die Spezifikation der Regelungsdienste und der Pre-Crash Regelung erfüllt sein müssen. Da beide Pre-Crash Regelungen in diesen Fällen das Verhalten an den Ausgabeports nicht weiter einschränken, werden alle Konflikte zwischen Pre-Crash Regelung und Abstandsregelungsdienste durch diese Prioritätsbeziehung aufgelöst.

Beziehung zwischen den Abstandsregelungsdiensten Auch die Abstandsregelungsdienste fordern in gewissen Situationen unterschiedliches Verhalten. Diese Konflikte werden ebenfalls über die Einführung von Querbeziehungen aufgelöst.

Der Dienst `Abstandsregelung nach Einscheren` (`ARE`) behandelt Situationen in denen ein Fahrzeug kurz vor dem eigenen Fahrzeug einschert. Der kritische Abstand ist zwar noch nicht unterschritten (sonst wäre `Pre Crash Regelung` aktiv), aber ein minimaler Abstand. Dieser Dienst behandelt eine sicherheitskritische Situation und muss daher höhere Priorität als die anderen beiden Abstandsregelungsdienste.

Ein angestrebter `Überholvorgang` (`ARÜ`) spezifiziert eine weitere Spezialsituation, die das Normalverhalten überstimmt. Ist bei unterschrittenem Wunschabstand der Blinker links gesetzt, so wird das Verhalten folglich gemäß der Abstandsregelung für den Überholvorgang bestimmt.

Für das Zusammenspiel zwischen `Abstandsregelung nach Einscheren` und `Abstandsregelung bei angestrebtem Überholvorgang` ist ferner das Verhalten am Port `Warnung` zu klären. Der Dienst `Abstandsregelung nach Einscheren` fordert die Ausgabe *nein* am Port `Warnung`, wenn der minimale Abstand nicht unterschritten ist. Der Dienst `Abstandsregelung bei Überholvorgang` hingegen kann die Ausgabe *ja* am Port `Warnung` fordern, auch wenn der minimale Abstand nicht unterschritten ist. In diesem Fall soll der Dienst `Abstandsregelung nach Überholvorgang` höhere Priorität, d.h. es soll eine Warnung ausgegeben werden.

Die entsprechende Querbeziehung zwischen den drei Abstandsregelungsdiensten ist in Tabelle 19 abgebildet. Wird der minimale Abstand unterschritten, so bestimmt die `Abstandsregelung nach Einscheren` das Verhalten. Wird der minimale Abstand nicht unterschritten, aber ist bei unterschrittenem Wunschabstand der Blinker links gesetzt, so wird das Verhalten gemäß der `Abstandsregelung bei Überholvorgang` bestimmt. Ansonsten müssen alle Dienstespezifikationen gelten. Die Zeilen sind in der Tabelle wieder prioritätsgemäß angeordnet, d.h. die zweite Zeile gilt nur, wenn die erste nicht bereits gilt. Konkret heißt das, dass die `Abstandsregelung bei Überholvorgang` nur höhere Priorität als die beiden anderen Dienste hat, wenn der minimale Abstand nicht unterschritten ist (Priorität von `Abstandsregelung nach Einscheren` gemäß der ersten Zeile). Ansonsten ist die Formalisierung analog zu den bisherigen Tabellen. Auf eine genauere Erläuterung wird daher an dieser Stelle verzichtet.

Obj	aktGW	WZL	Abstand	Blinker	MotorStr	BremsStr	Warnung
$ja\ \epsilon^*$	v	.	$d \vert d < x_{min} * v$	.	ARE	ARE	ARE
$ja\ \epsilon^*$	v	x	$d \vert x_{min} * v < d \leq x * v$	$links$	ARÜ	ARÜ	ARÜ
.	.	.	.	.	.	.	.

Tabelle 19: Querbeziehung zwischen den Abstandsregelungsdiensten

Wieder regelt die Querbeziehung nur das Zusammenspiel der Dienste, nicht die konkrete Reaktion. Diese kann den modularen Spezifikationen entnommen werden.

4.4.7 Kombination der Teildienste zur Abstandsregelung

Nach der Spezifikation der einzelnen Teildienste und ihrer Querbeziehungen, können diese automatisch zu der Gesamtfunktionalität kombiniert werden.

Bei der Kombination von Diensten unterscheiden wir zwei Fälle:

- die Kombination von unabhängigen Diensten und
- die Kombination von abhängigen Diensten, d.h. mit Querbeziehung.

Wir werden die beide Arten der Kombination in den nächsten Abschnitten anhand von Beispielen erklären. Im letzten Abschnitt dieses Kapitels widmen wir uns dem Umgang mit optionalen Diensten.

Kombination von unabhängigen Diensten Als Beispiel für die Kombination unabhängiger Dienste betrachten wir die in Kapitel 4.4.1 beschriebenen Dienste zur Einstellung des Wunschabstands und zur Anzeige einer Objekterkennung. Da hier die Kombination unabhängiger Dienste erklärt werden soll, betrachten wir nur eine Variante der Einstellung des Wunschabstands und zwar den Dienst `Stufenweise Einstellung des Wunschabstands`. Das entsprechende syntaktische Interface dieses Dienstes ist in Abbildung 17 auf Seite 26 dargestellt, die Verhaltensspezifikation in Tabelle 11 auf Seite 26. Die syntaktische Interface des Dienstes `Anzeige Objekterkennung` kann Abbildung 18 auf Seite 27 entnommen werden, die Verhaltensspezifikation Tabelle 12 auf Seite 27.

Das syntaktische Interface des kombinierten Dienstes ergibt sich als Vereinigung der Interfaces der Teildienste. Es umfasst die Eingabeports `WZL`, `Step` und die Ausgabeports `WZL`, `AnzWZL` des Dienstes `Stufenweisen Einstellung des Wunschabstands` sowie den Eingabeport `Obj` und Ausgabeport `AnzObj` des Dienstes `Anzeige Objekterkennung`. Das resultierende Interface ist in Abbildung 23 dargestellt.

Abbildung 23: Kombination Stufenweisen Einstellung des Wunschabstands und Anzeige Objekterkennung (Interface)

Das Verhalten des kombinierten Dienstes ist definiert über dem gemeinsamen Interface. Der kombinierte Dienst muss die Anforderungen beider Teildienste erfüllen. Dies ist problemlos möglich, da die beiden Dienste nicht im Widerspruch zueinander stehen. In dem betrachteten Beispiel sind sie sogar über unabhängigen Teilinterfaces definiert, d.h. die beiden Teildienste haben keine gemeinsamen Kanäle, an denen ein Konflikt auftreten könnte.

Die kombinierte Verhaltensspezifikation ist in Tabelle 20 dargestellt. Gemäß der Semantik der verwendeten Tabellennotation, müssen verschiedene Abschnitte gleichzeitig gelten. Somit können die ursprünglichen Verhaltensspezifikationen einfach als zwei verschiedene Abschnitte in eine Tabelle integriert werden. An den Ports des jeweils anderen Dienstes wird dabei nichts gefordert, es darf eine beliebige Nachricht anliegen. Dies wird durch den regulären Ausdruck „.“ in den entsprechenden Zellen ausgedrückt (Eingabespalte `Obj` und Ausgabespalte `AnzObj` im ersten Abschnitt, Eingabespalten `Step` und `WZL` sowie Ausgabespalten `AnzWZL` und `WZL` im zweiten Abschnitt)

Analog können alle unabhängigen Dienste als eigene Abschnitte in eine Tabelle integriert werden. Die Möglichkeit verschiedene, parallel gültige Abschnitte innerhalb einer Tabelle zu haben,

Step	WZL	Obj	AnzWZL	WZL	AnzObj
ϵ	.	.	3	3	.
$+$	$x \vert x \in \{1,2\}$	.	$x+1$	$x+1$	.
$-$	$x \vert x \in \{2,3\}$	.	$x-1$	$x-1$	.
.	x	.	x	x	.
.	.	$ja\ \epsilon^*$	.	.	ja
.	.	$nein\ \epsilon^*$	.	.	$nein$

Tabelle 20: Kombination Stufenweisen Einstellung des Wunschabstands und Anzeige Objekterkennung (Tabellarische Spezifikation)

vereinfacht die Spezifikation und reduziert die Komplexität, da nicht alle möglichen Kombinationen explizit aufgelistet werden müssen. In diesem einfachen Beispiel werden anstelle von 8 Zeilen zur Auflistung aller möglichen Kombinationen nur 6 Zeilen benötigt.

Kombination von abhängigen Diensten (mit Querbeziehung) Für die Kombination müssen zunächst die Querbeziehungen ausgewertet werden, die einzelnen Teildienste entsprechend modifiziert und anschließend in eine Tabelle kombiniert werden. Die Modifikation der Tabellen der einzelnen Teildienste und deren Kombination wird hier an einem Beispiel exemplarisch dargestellt. Da alle benötigten Informationen in den Teilspezifikationen bzw. der Querbeziehung vorhanden sind, kann die Kombination im Allgemeinen automatisch erfolgen und erfordert keinen manuellen Designaufwand mehr.

Bei der Kombination wird zunächst die Tabelle jedes einzelnen Dienstes um die entsprechenden höherprioren Eingabebedingungen aus der Kombinationstabelle erweitert. Für die Fälle, in denen sie von einem anderen Dienst überstimmt werden, fordern die modifizierten Dienste anstelle des bisherigen Verhaltens ein beliebiges Verhalten (Ausgabe „.“). Die so modifizierten Teildienste können dann einfach parallel kombiniert werden, wie bei der Kombination ohne Querbeziehungen eingeführt.

Zur Illustration dieses Vorgehens führen wir die Kombination für die drei Abstandsregelungsdienste (`Abstandsregelung Normalbetrieb`, `Abstandsregelung nach Einscheren`, `Abstandsregelung bei angestrebtem Überholvorgang`) explizit durch. Das syntaktische Interface des Dienstes `Abstandsregelung Normalbetrieb` ist in Abbildung 19 dargestellt, die Verhaltensspezifikation in Tabelle 13 auf Seite 28. Das syntaktische Interface des Dienstes `Abstandsregelung nach Einscheren` findet sich in Abbildung 20 auf Seite 29, die zugehörige Verhaltensspezifikation in Tabelle 14 auf Seite 30. Für das syntaktisches Interface und Verhaltensspezifikation des Dienstes `Abstandsregelung bei angestrebtem Überholvorgang` siehe Abbildung 21 bzw. Tabelle 15 auf Seite 31.

Wie bei der Kombination ohne Querbeziehungen ergibt sich das syntaktische Interface als Vereinigung der Ports der Teildienste. Das resultierende Interface ist in Abbildung 24 dargestellt.

Das Verhalten kann jedoch nicht durch eine einfache Parallelkombination der Verhaltensspezifikation der modularen Abstandsregelungsdienste erfolgen, da die Dienste nicht unabhängig

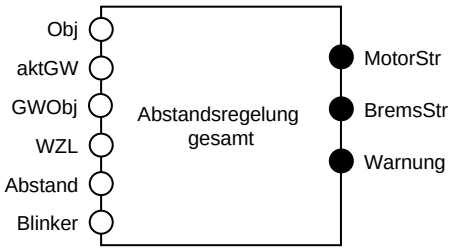

Abbildung 24: Kombinierte Abstandsregelung (Interface)

von einander sind. Im vorangehenden Kapitel haben wir im Wesentlichen folgende Querbeziehungen identifiziert:

- Die `Abstandsregelung nach Einscheren` hat höhere Priorität als die beiden verbleibenden Abstansregelungsdienste
- Ansonsten hat die `Abstandsregelung bei angestrebtem Überholvorgang` höhere Priorität als die `Abstandsregelung Normalbetrieb`.

Diese Abhängigkeiten wurden in den Tabellen 19 auf Seite 34 formal spezifiziert. Sie müssen bei der Kombination berücksichtigt werden.

Zunächst wird jeder Dienst gemäß der angegebenen Querbeziehung systematisch erweitert. Dabei wird für jede Zeile der Querbeziehung, in denen ein anderer Dienst eine höhere Priorität hat, eine entsprechende Zeile (mit derselben Eingabebedingung) in die Tabelle des modularen Dienstes eingefügt. Die Ausgaben dürfen von dem nieder-prioren Dienst nicht eingeschränkt werden. Daher wird das Zeichen „.“ in die entsprechenden Ausgabezellen eingefügt. Die erweiterten Dienste können dann wie bei der Kombination ohne Querbeziehung in eigene Abschnitte einer gemeinsamen Tabelle kombiniert werden. Das Ergebnis der schematischen Kombination ist in Tabelle 21 dargestellt.

Abschnitt 1 ist das Ergebnis der Erweiterung des Dienstes `Abstandsregelung nach Einscheren`. Hier ist die Priorität des Dienstes `Abstandsregelung bei angestrebtem Überholvorgang` zu berücksichtigen. Die entsprechende Zeile wurde eingefügt. Die durch Querbeziehung gegebene Priorität sind beim Einfügen zu berücksichtigen. Konkret heißt dies, dass die zusätzliche Zeile nicht beliebig sondern an der richtigen Stelle, hier als vierte Zeile, in die Tabelle einzufügen ist. Die ersten drei Zeilen der ursprünglichen Tabelle 14 auf Seite 30 erfüllen die Bedingung der ersten Zeile der Querbeziehung (Tabelle 19 auf Seite 34). Hier hat die `Abstandsregelung nach Einscheren` höhere Priorität. Die neue Zeile muss folglich danach eingefügt werden. Erst die letzte Zeile der ursprünglichen Spezifikation (Tabelle 14 auf Seite 30) ist durch die Priorisierung der `Abstandsregelung bei angestrebtem Überholvorgang` in der zweiten Zeile der Tabelle 19 auf Seite 34 betroffen. Daher muss die neue Zeile zuvor, also als vierte Zeile, eingefügt werden. Sie legt fest, dass unter den gegebenen Bedingungen (Blinker gesetzt) nicht wie ursprünglich definiert das Signal *nein* auf dem Port `Warnung` ausgegeben werden soll, sondern ein beliebiges Signal .. In den anderen Abschnitten, hier im zweiten Abschnitt zur `Abstandsregelung bei angestrebtem Überholvorgang`, wird näher spezifiziert, welches Signal ausgegeben wird.

Obj	aktGW	GWObj	WZL	Abstand	Blinker	MotorStr	BremsStr	Warnung
$ja\ \epsilon^*$	v	$v_o \vert v_o > v$	.	$d \vert d < x_{min} * v$	.	$verz$	ϵ	ja
$ja\ \epsilon^*$	v	$v_o \vert v_o = v$	.	$d \vert d < x_{min} * v$	.	ϵ	$verz$	ja
$ja\ \epsilon^*$	v	$v_o \vert v_o < v$	.	$d \vert d < x_{min} * v$	.	ϵ	$starkVerz$	ja
$ja\ \epsilon^*$	v	.	x	$d \vert d \leq x * v$	$links$	.	.	.
.	.	.	.	.	.	.	.	$nein$
$ja\ \epsilon^*$	v	$v_o \vert v_o > v$	.	$d \vert d < x_{min} * v$	.	.	.	.
$ja\ \epsilon^*$	v	.	x	$d \vert d \leq x * v$	$links$	$beschl$	ϵ	ja
.	.	.	.	.	.	.	.	$nein$
$ja\ \epsilon^*$	v	$v_o \vert v_o > v$	.	$d \vert d < x_{min} * v$	.	.	.	.
$ja\ \epsilon^*$	v	.	x	$d \vert d \leq x * v$	$links$	.	.	.
$ja\ \epsilon^*$	v	$v_o \vert v_o > v$	x	$d \vert d < x * v$	.	ϵ	ϵ	.
$ja\ \epsilon^*$	v	$v_o \vert v_o = v$	x	$d \vert d < x * v$	.	$verz$	ϵ	.
$ja\ \epsilon^*$	v	$v_o \vert v_o < v$	x	$d \vert d < x * v$	.	ϵ	$verz$	.
.	.	.	.	.	.	.	.	.

Tabelle 21: Kombinierte Abstandsregelung (Tabellarische Spezifikation)

Abschnitt 2 zeigt die Erweiterung des Dienstes `Abstandsregelung bei angestrebtem Überholvorgang` um die entsprechende Zeile des höher-prioren Dienstes `Abstandsregelung nach Einscheren`. Da die `Abstandsregelung nach Einscheren` höchste Priorität hat, wird die entsprechende Zeile gleich zu Beginn des Abschnitts eingefügt. Unter diesen Bedingungen beschränkt der Dienst `Abstandsregelung bei angestrebtem Überholvorgang` das Verhalten nicht. Der Dienst `Abstandsregelung nach Einscheren` setzt sich in der Kombination durch.

Abschnitt 3 zeigt die Erweiterung der `Abstandsregelung Normalbetrieb`. Sowohl die `Abstandsregelung nach Einscheren` als auch die `Abstandsregelung bei angestrebtem Überholvorgang` haben höhere Priorität. Die entsprechenden Zeilen werden zu Beginn des Abschnitts eingefügt. Nur wenn die Bedingungen aus diesen Zeilen nicht erfüllt sind, beschränkt die `Abstandsregelung Normalbetrieb` das Verhalten.

Zur Verbesserung der Lesbarkeit, kann die Tabelle durch Integration der Abschnitte vereinfacht werden. Hierbei wird die interne Priorität der Zeilen innerhalb einer Tabelle ausgenützt um die in den Beziehungstabellen explizit definierten Abhängigkeiten umzusetzen. Formal ist eine Integration der Abschnitte nicht nötig, aber sie kann einer vereinfachten Lesbarkeit dienen. Die Dekomposition in Teildienste ist allerdings aus der integrierten Tabelle nicht mehr ersichtlich. Das Ergebnis ist in Tabelle 27 dargestellt.

Die Kombination von unabhängigen sowie abhängigen Teildiensten wurde in den vorausgehenden Abschnitten nur exemplarisch anhand kleiner Teilausschnitte erklärt. Der explizite Aufbau der gesamten Tabelle (d.h. die Kombination aller Dienste) ist jedoch dank der formal fundierten Semantik nicht nötig, sondern kann nach dem vorgestellten Schema automatisch erfolgen.

Kombination unter Berücksichtigung von Produktlinienapekten Einige Teildienste der Abstandsregelung sind optional, z.B. die `Pre-Crash Regelung` oder die `Abstandsregelung bei angestrebtem Überholvorgang` (vgl. Abbildung 15 und Erläuterungen auf Seite 24). Dies muss bei der Kombination entsprechend berücksichtigt werden.

Obj	aktGW	GWObj	WZL	Abstand	Blinker	MotorStr	BremsStr	Warnung
$ja\ \epsilon^*$	v	$v_o \vert v_o > v$	.	$d \vert d < x_{min} * v$	.	$verz$	ϵ	ja
$ja\ \epsilon^*$	v	$v_o \vert v_o = v$	.	$d \vert d < x_{min} * v$	.	ϵ	$verz$	ja
$ja\ \epsilon^*$	v	$v_o \vert v_o < v$	.	$d \vert d < x_{min} * v$	.	ϵ	$starkVerz$	ja
$ja\ \epsilon^*$	v	.	x	$d \vert d \leq x * v$	$links$	$beschl$	ϵ	ja
$ja\ \epsilon^*$	v	$v_o \vert v_o > v$	x	$d \vert d < x * v$	.	ϵ	ϵ	$nein$
$ja\ \epsilon^*$	v	$v_o \vert v_o = v$	x	$d \vert d < x * v$	.	$verz$	ϵ	$nein$
$ja\ \epsilon^*$	v	$v_o \vert v_o < v$	x	$d \vert d < x * v$	.	ϵ	$verz$	$nein$
.	.	.	.	.	.	.	.	$nein$

Tabelle 22: Kombinierte Abstandsregelung (Vereinfachte Spezifikation)

Wie in Kapitel 2.3.3 eingeführt, ermöglicht die Tabellennotation eine Zuordnung des Verhaltens (der Anforderungen) zu unterschiedlichen Konfigurationen. Hierzu wird eine zusätzliche Spalte für einen *Konfigurationsvektor* eingeführt. Wir nutzen in dieser Spalte wieder reguläre Ausdrücke, um die Konfigurationen zu beschreiben, bei denen die entsprechende Zeile berücksichtigt werden muss. Die entsprechenden regulären Ausdrücke haben hierbei die Länge n, wobei n der Anzahl der optionalen Dienste im System entspricht. Für jede optionale Anforderung (Zeile) wird somit festlegt, in welchen Konfigurationen sie beachtet werden müssen.

In unserem Beispiel sind die relevanten, optionalen Blattdienste die `Kontinuierliche Einstellung des Wunschabstands`, die `Stufenweise Einstellung des Wunschabstands`, die `Abstandsregelung bei angestrebtem Überholvorgang` sowie die Dienste `Pre-Crash Bremsen` und `Pre-Crash Warnung`. Sie werden in dieser Reihenfolge in dem Konfigurationsvektor angeordnet, d.h. „10000“ beschreibt die Konfiguration, die die `Kontinuierliche Einstellung des Wunschabstands` umfasst, die anderen Dienste alle nicht.

In der Tabelle wird nicht explizit spezifiziert, welche Kombinationen nicht möglich sind (auf Grund von Alternativen, requires- oder excludes Beziehung). Dies erfolgt außerhalb der Tabelle im Dienstediagram. In unserem Fallbeispiel gibt es zwar keine excludes- oder requires-Beziehung. `Pre-Crash Bremsen` und `Pre-Crash Warnung` sowie `Kontinuierliche` und `Stufenweise Einstellung des Wunschabstands` sind jedoch Alternativen, d.h. diese Paar dürfen nie gemeinsam in ein System verbaut werden. Ansonsten sind alle Kombinationen an optionalen Diensten möglich.

Beschränken wir uns auf die Abstandsregelungsdienste. Hier gibt es nur einen optionalen Dienst und zwar der Dienst `Abstandsregelung bei angestrebtem Überholvorgang`. Der Konfigurationsvektor besteht folglich nur aus einem Bit, der reguläre Ausdruck hat die Länge 1. Die entsprechenden Anforderungen `Abstandsregelung bei angestrebtem Überholvorgang` müssen nur gelten, wenn die Abstandsregelung gewählt wurde, d.h. wenn das entsprechende Bit im Konfigurationsvektor gesetzt ist ($Konfig = 1$). Daselbe gilt auch für die die zugehörigen Querbeziehungen. Es ergibt sich daher die folgende modifizierte Tabelle 23 für die drei Abstandsregelungsdienste:

Um die entsprechende Spezifikation einer konkreten Konfiguration (oder einer bestimmten Teilmenge an Konfigurationen) zu erhalten, müssen einfach nur die passenden Zeilen ausgewählt werden. Die anderen Zeilen werden ausgeblendet. Die Spezifikation für eine Konfiguration mit `Abstandsregelung bei angestrebtem Überholvorgang` enthält alle Zeilen aus Tabelle 23.

Konf	Obj	aktGW	GWObj	WZL	Abstand	Blinker	MotorStr	BremsStr	Warnung
.	$ja\ \epsilon^*$	v	$v_o \vert v_o > v$	.	$d \vert d < x_{min} * v$	.	$verz$	ϵ	ja
.	$ja\ \epsilon^*$	v	$v_o \vert v_o = v$	.	$d \vert d < x_{min} * v$	.	ϵ	$verz$	ja
.	$ja\ \epsilon^*$	v	$v_o \vert v_o < v$	.	$d \vert d < x_{min} * v$	.	ϵ	$starkVerz$	ja
1	$ja\ \epsilon^*$	v	.	t	$d \vert d \leq x * v$	$links$	$beschl$	ϵ	ja
.	$ja\ \epsilon^*$	v	$v_o \vert v_o > v$	x	$d \vert d < x * v$	.	ϵ	ϵ	$nein$
.	$ja\ \epsilon^*$	v	$v_o \vert v_o = v$	x	$d \vert d < x * v$	.	$verz$	ϵ	$nein$
.	$ja\ \epsilon^*$	v	$v_o \vert v_o < v$	x	$d \vert d < x * v$	.	ϵ	$verz$	$nein$
.	.	.	.	.	.	.	.	.	$nein$

Tabelle 23: Kombinierte Abstandsregelung (Spezifikation mit Produktlinieninformationen)

Die entsprechende Konfiguration entspricht dem Konfigurationsvektor „1“. Dieser Ausdruck matcht alle Einträge in der Spalte `Konf` der Tabelle 23. Um die Spezifikation für eine Konfiguration ohne `Abstandsregelung bei angestrebtem Überholvorgang` zu erhalten, muss nur die entsprechende Zeile (Zeile 4) aus der Tabelle gestrichen werden. Die entsprechende Belegung des Konfigurationsvektors wäre „0“. Dies matcht dem Ausdruck „1“ in Zeile 4 nicht und für die Spezifikation der Konfiguration wird die entsprechende Zeile gestrichen.

Das Vorgehen wurde wieder nur exemplarisch erläutert, kann aber analog auf das gesamte Fallbeispiel angewandt werden.

4.5 ACC

In den vorherigen Abschnitten wurden einzelne Dienste der Funktionalität `ACC` aus Abbildung 2 auf Seite 12 formalisiert. Nun werden die Teilfunktionalitäten unter Einführung von Querbeziehungen zur Gesamtfunktionalität integriert.

4.5.1 Priorisierung: Einschaltung vs. Restfunktionalität

Der Dienst `Einschaltung` hat die höchste Priorität in der Gesamtfunktionalität `ACC`. Immer wenn die minimale Spannung unterschritten ist oder die Zündung ausgeschaltet ist, ist das gesamte ACC ausgeschaltet. Im Dienstediagramm aus Abbildung 2 sind drei einzelne (identische) Priorisierungsbeziehungen zwischen dem Dienst `Einschaltung` und den restlichen drei Diensten zu finden. Alle drei fordern, immer wenn am Port `Zündung` keine Nachricht oder am Port `Spannung` ein Wert kleiner *minSpannung* anliegt, hat der Dienst `Einschaltung` eine höhere Priorität. Ansonsten gelten die Regeln der normalen Komposition.

Nr.	Transition
01	$\{s < minSpannung \vee z = \varepsilon\}$Zündung?z;Spannung?s

Tabelle 24: Priorisierung: Einschaltung vs. Restfunktionalität (I/O Automat)

Die Priorisierung hat zwei Input-Ports `Zündung` und `Spannung` und ist durch den Automaten aus Tabelle 24 formalisiert. Immer wenn der Automat seine Transition ausführen kann, hat

der Dienst `Einschaltung` eine höhere Priorität – der andere Dienst stellt keine Anforderungen an die Aussagen.

4.5.2 Priorisierung: Aktivierung vs. Geschwindigkeitsregelung

Der Dienst `Aktivierung` hat die zweit höchste Priorität in der Gesamtfunktionalität `ACC`. Immer wenn das Bremspedal getreten ist oder der Knopf `I/O` gedrückt ist, befindet sich das gesamte ACC im Zustand „StandBy“. Die Priorisierung aus Tabelle 25 formalisiert diese Anforderung. Immer wenn der Automat Transition 2 oder 3 ausführt, ist der Dienst `Aktivierung` höher priorisiert als der Dienst `Regelung` – die `Aktivierung` macht einen Schritt, die Regelung bleibt stehen und stellt somit keine Anforderungen an die Output-Ports. In allen anderen Fällen gelten die Regeln der normalen Kombination – die beiden Dienste müssen gleichzeitig einen Schritt machen.

Nr.	Von	Hin	Transition
01	passiv	aktiv	$\{g > minGW \wedge \neg(x = \varepsilon \wedge y = \varepsilon \wedge v = \varepsilon \wedge w = \varepsilon)\}$ Bremse?ε;aktGW?g;Resume?x;Set?y;Set+?v;Set-?w;I/O?ε
02	passiv	passiv	$\{b \neq \varepsilon \vee g < minGW \vee (x = \varepsilon \wedge y = \varepsilon \wedge v = \varepsilon \wedge w = \varepsilon) \vee z = aus\}$ Bremse?b;aktGW?g;Resume?x;Set?y;Set+?v;Set-?w;I/O?z
03	aktiv	passiv	$\{b \neq \varepsilon \vee g < minGW \vee z = aus\}$ Bremse?b;aktGW?g;I/O?z

Tabelle 25: Priorisierung: Aktivierung vs. G-Regelung (I/O Automat)

4.5.3 Priorisierung: Aktivierung vs. Abstandsregelung

Die Priorisierung zwischen den Diensten `Aktivierung` und `Abstandsregelung` ist der Priorisierung aus dem letzten Abschnitt identisch.

4.5.4 Priorisierung: Geschwindigkeits- vs. Abstandsregelung

Fährt das vorausfahrende Objekt schneller als die eingestellte Wunschgeschwindigkeit, so stellt das ACC sicher, dass das ACC-Fahrzeug nicht schneller als die Wunschgeschwindigkeit fährt. D.h., der Dienst `Geschwidigkeitsregelung` hat eine höhere Priorität als der Dienst `Abstandsregelung` immer, wenn die aktuelle Geschwindigkeit die Wunschgeschwidigkeit erreicht hat. Der Automat aus Tabelle 26 formalisiert diese Priorisierung. Immer wenn eine der Transitionen von 04 bis 07 ausgeführt wird, hat der Dienst `Geschwidigkeitsregelung` eine höhere Priorität, der Dienst `Abstandsregelung` stellt in diesen Zeitintervallen keine Anforderungen an die Ausgaben. Ansonsten gilt die Regel der normalen Komposition.

4.5.5 Priorisierung: Abstands- vs. Geschwindigkeitsregelung

Wird ein vorausfahrendes Objekt innerhalb des eingestellten Wunschabstandes erkannt, so regelt die `Abstandsregelung` (`AR`) die Geschwindigkeit des eigenen Fahrzeugs. Die `Abstandsregelung` hat somit in diesem Fall eine höhere Priorität als die `Geschwindigkeitsregelung`

Nr.	Transition
01	$\{y \leq lGW \wedge x \neq \varepsilon\}$ aktGW?y;Set?x;Set+?ε;Set-?$\varepsilon\{lGW = y\}$
02	$\{y \leq lGW \wedge x \neq \varepsilon\}$ aktGW?y;Set?ε;Set+?x;Set-?$\varepsilon\{\text{lGW}=(y+10)\}$
03	$\{y \leq lGW \wedge x \neq \varepsilon\}$ aktGW?y;Set?ε;Set+?ε;Set-?x$\{\text{lGW}=(y-10)\}$
04	$\{y > lGW\}$ aktGW?y;Set?ε;Set+?ε;Set-?ε/
05	$\{y > lGW \wedge x \neq \varepsilon\}$ aktGW?y;Set?x;Set+?ε;Set-?$\varepsilon\{lGW = y\}$
06	$\{y > lGW \wedge x \neq \varepsilon\}$ aktGW?y;Set?ε;Set+?x;Set-?$\varepsilon\{\text{lGW}=(y+10)\}$
07	$\{y > lGW \wedge x \neq \varepsilon\}$ aktGW?y;Set?ε;Set+?ε;Set-?x$\{\text{lGW}=(y-10)\}$

Tabelle 26: Priorisierung: Geschwindigkeits- vs. Abstandsregelung (I/O Automat)

(`GWR`). Die gilt jedoch nur, solange die aktuelle Geschwindigkeit nicht die Wunschgeschwindigkeit (Spalte `lGW`) überschreitet. In diesem Fall setzt sich, wie im vorausgegangenen Kapitel 4.5.4 beschrieben die `Geschwindigkeitsregelung` durch. Zusammen ergibt sich die folgende Querbeziehung, die im Gegensatz zu den vorausgegangenen Querbeziehungen mittels einer Tabelle spezifiziert wird.

Obj	aktGW	lGW	WZL	Abstand	MotorStr	BremsStr
.	v	$v_l \vert v > v_l$	.	.	GWR	GWR
$ja\ \epsilon^*$	v	.	x	$d \vert d < x * v$	AR	AR
.	.	.	.	.	.	.

Tabelle 27: Beziehung zwischen `Abstandsregelung` und `Geschwindigkeitsregelung`

Die erste Zeile formalisiert, dass wann immer die aktuelle Geschwindigkeit v die aktuelle Wunschgeschwindigkeit v_l überschreitet, die `Geschwindigkeitsregelung` das Verhalten an den Ports `MotorStr` und `BremsStr` bestimmt. Die zweite Zeile formalisiert, dass wann immer ein Objekt innerhalb des Wunschabstands erkannt wurde, die `Abstandsregelung` das Ausgabeverhalten an diesen Ports festlegt.

Man beachte, dass sich die Querbeziehung nur auf den Regelungsanteil der in Kapitel 4.3 bzw. 4.4 beschriebenen Funktionalitäten beschreiben. Der `HMI`-Anteil bleibt davon unbetroffen.

4.5.6 Gesamtfunktionalität ACC

Mit den zuletzt eingeführten Querbeziehungen zwischen den Diensten `Einschaltung`, `Aktivierung`, `Geschwindigkeitsregelung` und `Abstandsregelung` ist das Verhalten des Gesamtsystems vollständig beschrieben. Die Gesamtspezifikation kann aus den Teildiensten und den Querbeziehungen automatisch generiert werden. Für kleine Teilausschnitte wurde die Kombination von unabhängigen und von abhängigen Diensten sowohl für die automatenbasierte Notationstechnik (Kapitel 4.3.7 und 4.3.8) als auch für die tabellarische Notationstechnik (Kapitel 4.4.7) bereits gezeigt. Die Kombination zur Gesamtfunktionalität erfolgt analog und wird daher nicht nochmal erläutert. Neu ist, dass z.B. mit den Diensten `Geschwindigkeitsregelung` und `Abstandsregelung` zwei Dienste integriert werden, die mit unterschiedlichen Notationstechniken beschrieben wurden. Dies stellt kein Problem da, da beiden Notationstechniken dieselbe formal Fundierung zugrunde liegen. Daher können automatenbasierte Spezifikationen in

tabellarische Spezifikationen transformiert werden und umgekehrt. Eine Kombination ist somit kein Problem. Ebenfalls spielt es keine Rolle, ob die Querbeziehungen als Tabellen oder Automaten spezifiziert sind. Die Gesamtspezifikation kann dann sowohl als Tabelle als auch als Automat dargestellt werden.

4.6 Syntaktisches Interface

Portname	Typ	Beschreibung
Bremse	{an}	Bremspedal
Zündung	{an}	Die Position des Zündschlüssels
Set	{an}	Taste „Set“
Set+	{an}	Taste „+“
Set–	{an}	Taste „–“
Resume	{an}	Taste “Resume“
IO	{an}	Taste “I/O“
WZL	$[1, 3]$	gespeicherte Wunschzeitlücke (kontinuierlich)
WZL	{1, 2, 3}	gespeicherte Wunschzeitlücke (diskret)
Step	{+,-}	Taste zum verstellen der Wunschzeitlücke (diskret)
Obj	{ja, nein}	Objekt erkannt
aktGW	Int	Geschwindigkeit des eigenen Fahrzeugs
GWObj	Int	Geschwindigkeit des vorausfahrenden Fahrzeugs
Abstand	Int	Abstand zu vorausfahrendem Fahrzeug
Blinker	{links, rechts}	Blinker
Spannung	Int	Versorgungsspannung
LrWinkel	Int	Lenkradwinkel
MotorStr	{verz, beschl}	Motorsteuerung: leicht verzögern oder beschleunigen
BremsStr	{verz, starkVerz, maxVerz}	Bremssteuerung: einfach, stark oder maximal verzögern
AnzAktiv	{aktiv, standBy}	ACC Betriebsbereitschaft „stand by“
AnzWGW	Int	Anzeige der Wunschgeschwindigkeit
AnzWZL	Int	Anzeige der gewählten Wunschzeitlücke
AnzWZL	$[1, 3]$	Anzeige der gewählten Wunschzeitlücke (kontinuierlich)
AnzWZL	{1, 2, 3}	Anzeige der gewählten Wunschzeitlücke (schrittweise)
WZL	$[1, 3]$	gespeicherte Wunschzeitlücke (kontinuierlich)
AnzObj	{ja, nein}	Anzeige der Objekterkennung
Warnung	{ja}	Warnsignal

Tabelle 28: Ports aus dem syntaktischen Interface des Gesamtsystems

5 Zusammenfassung

In diesem Dokument wurde ein Ansatz zur dienstbasierten Modellierung von Funktionalität anhand eines ausführlichen Beispiels aus dem Automobilbereich, der Adaptiven Cruise Control (ACC), vorgestellt. Die Gesamtfunktionalität wurde in hierarchische Teildienste zerlegt, die zunächst unabhängig voneinander spezifiziert wurden. Hierfür wurden zwei unterschiedliche Notationstechniken eingeführt: eine automatenbasierte Notationstechnik und eine tabellarische

Notationstechniken. Die beiden Techniken basieren auf denselben formalen Grundlagen aus [6] und können daher äquivalent verwendet werden. Außerdem wurden Querbeziehungen zwischen den Teildiensten identifiziert und es wurde gezeigt wie diese sowohl mittels der automatenbasierten als auch der tabellarischen Notationstechnik spezifiziert werden können. Zuletzt wurde vorgestellt, wie unabhängige sowie durch Querbeziehungen vernetzte Teildienste zu größeren Diensten integriert werden.

Das Fallbeispiel zeigt, dass der dienstbasierte Ansatz gut geeignet ist, auch umfangreiche multifunktionale Systeme zu spezifizieren. Lediglich die Spezifikationen der einzelnen Teilsysteme und deren Querbeziehungen sind manuell zu erstellen. Die Spezifikation des Gesamtsystems kann automatsisch generiert werden. Die Anwendung des Ansatzes ist dabei nicht beschränkt auf die Automobildomäne. Vielmehr ist der Ansatz geeignet beliebige Kontrollsysteme aus dem Bereich der eingebetteten Systeme zu spezifizieren.

Durch die Modularität unseres Ansatzes ist er insbesondere auch für die verteilte Entwicklung (sowohl räumlich als auch organisatorisch) von größeren Systemen geeignet. Die vorliegende Fallstudie zeigt, dass ein System von mehreren Teams und aus verschiedenen Perspektiven parallel spezifiziert werden kann. Anschließend werden getrennt modellierte Teil-Spezifikationen problemlos zusammengeführt.

Auch der Umgang mit Variabilität, wie er in Produktlinien Anwendung findet, stellt kein Problem dar. Variable Teile werden als eigene optionale Teildienste spezifiziert, die bei Bedarf in das System integriert werden können oder auch nicht.

Ein weiterer Vorteil des Ansatzes ist, dass aufgrund der formalen Fundierung Konflikte zwischen einzelnen Anforderungen bzw. Teildiensten automatisch erkannt werden können. Durch die Einführung von Querbeziehungen können diese explizit ohne Änderung an den modularen Spezifikationen aufgelöst werden. Dies ist insbesondere im Kontext von Produktlinien sinnvoll, wenn Konflikte zwischen optionalen Teildiensten auftreten. Eine Anpassung der modularen Teilverhalten wäre hier nicht sinnvoll, da die Dienste auch unabhängig voneinander verbaut werden können. Die Querbeziehung hingegen ist nur dann aktiv, wenn beide Dienste ausgewählt sind.

In diesem Dokument wurde die Fallstudienbearbeitung manuell durchgeführt. Da alle Konzepte formal fundiert sind, steht einer Umsetzung in ein Werkzeug jedoch nichts im Weg. Es existiert bereits eine Integration des automatenbasierten Ansatzes in das Werkzeug AutoFocus [2]. Dabei wird die im Tool für logische Komponenten bereits existierende Simmulationsumgebung für den dienstebasierten Ansatz wiederverwendet. Die Semantik eines einzelnen Dienstes sowie ihre Komposition mussten neu implementiert werden. Die AutoFocus Semantik für Dienste ist [3] zu entnehmen.

Literatur

[1] *Adaptive Fahrgeschwindigkeitsregelung ACC*. Robert Bosch GmbH, 2001.

[2] AutoFocus 3. `http://af3.in.tum.de/`.

[3] Jewgenij Botaschanjan, Alexander Harhurin, and Leonid Kof. Service-based Specification of Reactive Systems. Technical Report TUM-I0815, Technische Universität München, 2008.

[4] Krzysztof Czarnecki, Simon Helsen, and Ulrich W. Eisenecker. Formalizing cardinality-based feature models and their specialization. *Software Process: Improvement and Practice*, 10(1):7–29, 2005.

[5] A. Gruler, A. Harhurin, and J. Hartmann. Modeling the functionality of multi-functional software systems. In *Proceedings of 14th Annual IEEE International Conference on the Engineering of Computer Based Systems (ECBS)*, March 2007.

[6] Alexander Harhurin and Judith Hartmann. Towards consistent specifications of product families. In *FM: 15th International Symposium on Formal Methods*, volume 5014 of *LNCS*. Springer Verlag, 2008.

[7] K. Kang, S. Cohen, J. Hess, W. Novak, and A. Peterson. Feature-oriented domain analysis (FODA) feasibility study. Technical report, SEI, CMU, Pittsburgh, PA, 1990.