Falko Saust,
Tobias Müller,
Jörn Marten Wille,
Markus Maurer

Entwicklungsbegleitendes Simulations- und Testkonzept für autonome Fahrzeuge in städtischen Umgebungen

Braunschweig : Inst. Für Regelungstechnik, Lehrstuhl für elektronische Fahrzeugsysteme, 2009

Veröffentlicht: 25.09.2009

http://www.digibib.tu-bs.de/?docid=00030369

Entwicklungsbegleitendes Simulations- und Testkonzept für autonome Fahrzeuge in städtischen Umgebungen

Dipl.-Ing. Falko Saust, Dipl.-Ing. Tobias Müller, Dipl.-Ing. Jörn Marten Wille
Prof. Dr.-Ing. Markus Maurer
Technische Universität Braunschweig, Lehrstuhl f. elektr. Fahrzeugsysteme,
Hans-Sommer-Str. 66, 38106 Braunschweig
saust@ifr.ing.tu-bs.de, tc.mueller@tu-bs.de, wille@ifr.ing.tu-bs.de, maurer@ifr.ing.tu-bs.de

Kurzfassung

Das Projekt „Stadtpilot“ hat das Ziel, aufbauend auf den Erfahrungen der DARPA Urban Challenge den Braunschweiger Stadtring vollständig autonom zu befahren. Im Rahmen des Projektes wird zur Gewährleistung der Sicherheit im öffentlichen Straßenverkehr ein entwicklungsbegleitendes Simulations- und Testkonzept umgesetzt. Das Konzept beinhaltet die Definition von festen Grundsätzen, die sowohl für die einzelnen Komponenten und ihre Bestandteile als auch für das Gesamtsystem erfüllt sein müssen. Neben der manuellen Erzeugung von Testfällen wird die automatisierte Evaluierung und Testfallgenerierung vorgestellt, in die insbesondere definierte Gütemaße und explizit zugelassene Fehlertoleranzen einfließen. Dadurch wird schon in einer sehr frühen Phase die Entwicklung robuster Module gefördert. Das Testkonzept wird ergänzt durch eine Verkehrssimulation. Dabei entstehen aus dem simulierten Verkehrsfluss kontinuierlich neue Testszenarien, so dass die Testtiefe nochmals erhöht werden kann. Erst wenn das System durch ein solches Testkonzept ausgereift ist, kann eine Erprobung unter realen Bedingungen erfolgen.

1. Einleitung

Die TU Braunschweig hat erfolgreich mit dem Forschungsfahrzeug „Caroline“ an der DARPA Urban Challenge 2007 teilgenommen. Die dort gesammelten Erfahrungen werden zurzeit in einem Folgeprojekt mit dem Namen „Stadtpilot“ vertieft. Ein interdisziplinäres Team bestehend aus dem Institut für Betriebssysteme und Rechnerverbund, dem Institut für Flugführung und dem Institut für Regelungstechnik hat sich zum Ziel gesetzt, eine vollständig autonome Fahrt auf dem Braunschweiger Stadtring zu absolvieren.

Die Fahrt auf dieser streckenweise baulich getrennten zweispurigen Straße beinhaltet u.a. Einfädeln in den fließenden Verkehr, Abbiegevorgänge an Kreuzungen und Spurwechselmanöver. Eine detaillierte Beschreibung des Projektes wird in [1] gegeben.

Der Einsatz eines autonomen Straßenfahrzeugs im realen Verkehr erfordert ein hohes Maß an Sicherheit und Zuverlässigkeit des Gesamtsystems sowie der einzelnen Komponenten und Module. Fehler der Software, die während des Betriebes auf öffentlichen Straßen auftreten, können schwerwiegende Konsequenzen haben. Unabhängig von den juristischen Folgen eines Unfalls sind bereits im Geschwindigkeitsbereich auf dem Stadtring nicht unerhebliche Gefährdungen für Verkehrsteilnehmer durch Fehlfunktionen möglich. Daher ist der Betrieb des Testträgers im Straßenverkehr erst mit einem ausgereiften und sicheren Fahrzeug möglich und stellt damit auch besondere Anforderungen an ein Testkonzept. In Verbindung mit der hohen Komplexität werden neue Ansätze zum Erreichen dieses Reifegrades benötigt.

Ein sehr zentraler Punkt des Testkonzeptes ist die Frage, wann ausreichend getestet wurde. Zur Klärung dieser Frage kann versucht werden, eine Aussage über die erreichte Testabdeckung zu treffen. Ein übliches Verfahren in der Softwaretechnik ist die Bestimmung von Überdeckungsmaßen wie die Funktionsüberdeckung oder auch flussorientierte Metriken wie die Pfadabdeckung [2].
Gerade bei Fahrzeugsystemen, die abhängig von anderen Verkehrsteilnehmern agieren, also in begrenztem Maße Entscheidungen treffen, können Überdeckungsmaße unzureichend sein. Durch geringfügige Änderung einzelner Parameter oder äußerer Einflussgrößen kann die getroffene Entscheidung oder Reaktion solch eines Systems grundlegend anders ausfallen, so dass durch die Komplexität dieser Systeme die Bestimmung einer Überdeckung kaum möglich ist.

In der Literatur wurden bereits umfassende Testkonzepte vorgestellt. Neben modellbasierten Ansätzen [3], [4] als Grundlage einer Automatisierung der Testfallerzeugung werden zur Abdeckung einer großen Zahl von Testfällen auch zufallsbasierte Verfahren eingesetzt, deren Anwendungen in [5], [6] beschrieben sind.
Als kombinierte Test- und Simulationsmethoden sind zur Erprobung im Bereich der Fahrerassistenzsysteme [7], [8] und [9] zu nennen.

In dieser Arbeit soll ausgehend von der geschilderten Problematik ein Simulations- und Testkonzept beschrieben werden, das den gesamten Entwicklungsprozess mit einschließt. Ziel ist es, durch robuste und ausgereifte Systeme die Sicherheit der autonomen Fahrt im öffentlichen Straßenverkehr zu gewährleisten.

2. Das Testkonzept

Das in dieser Arbeit entwickelte Konzept kombiniert mehrere bekannte Testkonzepte. Dazu zählen sowohl manuelle Tests als auch automatisierte Zufallstests (random testing). Nach [1] sind beide Methoden komplementär, es können je nach Anwendungsfall jeweils nur geringe Überschneidungen der gefundenen Fehler auftreten. Die beschriebenen Studien haben gezeigt, dass durch Zufallstests Fehler aufgedeckt werden, die durch manuelle Tests meist nicht gefunden werden. Umgekehrt finden Tester allerdings Fehler, die beim zufallsbasierten Testen nicht auftreten.

Grundlage der Implementierung und damit auch der Testfallerzeugung sind die zuvor definierten Anforderungen. Viele Testkonzepte bauen explizit oder implizit auf vollständigen Anforderungen auf [2], [3]. Gerade bei autonomen Straßenfahrzeugen ist bedingt durch die komplexe Umwelt eine absolut vollständige Definition der Anforderung in der nötigen Detailtiefe nicht mit vertretbarem Aufwand möglich. Je nach Realisierung des Systems müssen alle möglichen Situationen entweder direkt oder indirekt durch die Anforderungen abgedeckt sein. Wird dabei bedacht, dass eine Situation nicht nur durch ihren momentanen Zustand, sondern auch durch die vorherigen Zustände definiert ist, entsteht schnell ein sehr großer Situationsraum. Der Umstand einer unvollständigen Anforderungsdefinition muss folglich in einem Simulations- und Testkonzept berücksichtigt werden. In den betreffenden Abschnitten wird jeweils darauf Bezug genommen.

Testen auf drei Prüfebenen

Das Testen wird prinzipiell sowohl als notwendiger Schritt zur Steigerung der Qualität als auch zur Überprüfung der geforderten Qualität verstanden.
Aufgrund der hohen Komplexität des zu testenden Systems sind reine Black-Box-Tests nicht ausreichend, da ohne Kenntnis der inneren Struktur des Systems die notwendige Testtiefe nicht erreicht werden kann. Folglich sind Tests notwendig, die sich nicht nur aus der

Spezifikation heraus ableiten, sondern auch aus der Implementierung des Systems und dessen Bestandteilen (White-Box-Tests). Dabei ist ein modularer Aufbau des Gesamtsystems eine notwendige Voraussetzung, so dass die einzelnen Module unabhängig voneinander entwickelt und in Betrieb genommen werden können. Die Modularität ermöglicht es auch, die Bausteine einzeln und entkoppelt vom Gesamtsystem zu testen.

Daraus wird ein Testen auf drei verschiedenen Prüfebenen motiviert. Diese Ebenen leiten sich wie beschrieben aus der Struktur des Systems ab und gliedern sich in Unit-, Modul- und Systemebene [2].

Mit Hilfe eines Unittest-Frameworks werden gleich ab der Implementierung der ersten Methode Tests durchgeführt. Unittests[1] erfolgen auch für Klassen oder Teilstrukturen der Module.

In Tests auf Modulebene wird die Funktion des jeweiligen Moduls überprüft und gegen die vorgegebenen Spezifikationen getestet. Diese Tests erfolgen durch Anwendung verschiedener Testmethoden und der zugehörigen Werkzeuge.

Durch die Modularität können die einzelnen Komponenten getrennt entwickelt werden, müssen anschließend jedoch in einem gemeinsamen Verbund funktionieren. Daher ist eine Funktionsüberprüfung der Module auch in größeren Verbünden erforderlich. Durch gegenseitige Wechselwirkungen ist es möglich, dass Module zwar einzeln fehlerfrei arbeiten, gemeinsam jedoch in ihrer Funktion eingeschränkt sind. Sobald Module einzeln ausreichend getestet wurden, werden sie daher im Modulverbund getestet. Es entsteht ein fließender Übergang zum Systemtest, bei dem große Teile des Systems bzw. das System als Ganzes getestet werden, so dass abschließend die Funktionalität des Gesamtsystems sichergestellt wird.

Erst durch die Unterteilung in verschiedene Prüfebenen, die ein Testen von der kleinsten Einheit bis zum Gesamtverbund ermöglicht, wird die Komplexität des Gesamtsystems beherrschbar.

[1] Für eine weiterführende Beschreibung von Unittests wird auf [10] verweisen.

3. Grundsätze des Testkonzeptes

Um die Vorteile der erwähnten Verfahren zu kombinieren, werden sechs Grundsätze des Testkonzeptes definiert:

- Entwicklungsbegleitende Tests
- Manuelle Testfallerzeugung
- Automatisierte generische Testfallerzeugung
- Simulationsbasierte Testfallgenerierung
- Automatisierte Validierung und Evaluierung
- Förderung robuster Software

Diese Grundsätze müssen in allen drei beschriebenen Prüfebenen erfüllt sein und werden nachfolgend erläutert.

Entwicklungsbegleitende Tests

Die Unterteilung in mehrere Prüfebenen macht erst ein entwicklungsbegleitendes Testen möglich. So werden bereits ab der ersten implementierten Einheit mittels Unittests Testfälle durch den Entwickler definiert und durchgeführt. Der Vorteil hierbei ist z.B. die gezielte Abdeckung komplexer Softwarepfade, die zu späteren Zeitpunkten nicht mehr effizient möglich wäre. Fortgesetzt wird das Testen auf der nächsthöheren Ebene, indem erste Module zum einen durch den/die Entwickler selbst und zum anderen durch einen Testingenieur überprüft werden. Schließlich werden, wie beschrieben, mit immer mehr fertiggestellten Modulen Modulverbünde und das Gesamtsystem durch den Testingenieur getestet. Auf diese Weise wird ein entwicklungsbegleitendes Testen realisiert.

Manuelle Testfallerzeugung

Die manuelle Testfallerzeugung geschieht wie gefordert auf Unit-, Modul- und Systemebene. Auf Unitebene werden zunächst statische Softwaretests durchgeführt, durch die noch vor der Programmausführung die Software beispielsweise durch eine statische Code-Analyse auf formale Fehler überprüft wird [16]. Weiterhin erfolgen dynamische Tests[2], darunter Unittests, die zu einem großen Teil die Entwickler selbst während der Implementierung definieren. Die

[2] Auf statische und dynamische Tests wird detailierter in [16] eingegangen.

statischen Softwaretests sowie ein Großteil der definierten Unittests werden automatisch beim Erzeugen der Software vorgenommen. Entwicklungsbegleitend werden wie bereits beschrieben auch Unittests unmittelbar während der Implementierung durchgeführt, um z.B. gezielt Programmpfade zu überprüfen. Analog zur Unitebene werden auf Modulebene ebenfalls Unittests ausgeführt. Auf Systemebene steht vermehrt die Überprüfung der reinen Funktion im Vordergrund.
Die Definition von Tests durch die Modulentwickler selbst kann dazu führen, dass die Auswahl sehr einseitig erfolgt, da die Modulentwickler durch den Entwicklungsprozess voreingenommen sein können. Daher werden Tests nicht nur von den Modulentwicklern selbst definiert und durchgeführt. Stattdessen sind zu jedem Modul und Modulverbund Anforderungen und Spezifikationen definiert. Daraus werden sowohl durch den Entwickler als auch durch den Testingenieur entsprechende Testfälle abgeleitet und anhand von Expertenwissen erweitert.

Automatisierte generische Testfallerzeugung

Die manuellen Testfälle werden ergänzt durch ausgedehnte Zufallstests, um die Anzahl der abgedeckten Testfälle signifikant zu steigern. Bei diesen Zufallstests werden die Eingangsdaten zufällig erzeugt. Aufgrund der Komplexität der Systeme ist eine vollständige Abdeckung aller möglichen Eingangsdaten nicht möglich, so dass auch für eine automatisierte Erzeugung die Definition von Rahmenbedingungen erforderlich ist. Eine genaue Betrachtung und anschließend eine Vorgabe eines limitierenden Bereichs, von Schrittweiten oder weiterer Randbedingungen zur Erzeugung der Testfälle sind notwendig, um die Anzahl der durchzuführenden Tests auf ein technisch und zeitlich mögliches Maß zu beschränken.
Diese zufallsbasierten Tests benötigen zwangsläufig eine automatisierte Auswertung, um die hohe Zahl der generierten Testfälle auswerten zu können. Erst dadurch steigern Zufallstests auch deutlich die Effizienz.
Für die Automatisierung der Auswertung ist ein sogenanntes Testorakel notwendig, um für einen durchgeführten Testfall die Entscheidung treffen zu können, ob das Verhalten des jeweiligen Moduls oder Systems fehlerhaft ist [11]. Das Testorakel muss dementsprechend in Abhängigkeit der Eingangsdaten Kenntnis über das korrekte Ausgangsverhalten besitzen. Daher wird für den Test jedes Moduls, der Modulverbünde und des Gesamtsystems ein Testorakel geschaffen, mit dem eine automatische Auswertung möglich ist. Das Orakel ist als

Referenz für das Systemverhalten anzusehen und kann beispielsweise aus Systemmodellen, Spezifikationen oder anderen maßgeblichen Bewertungskriterien gewonnen werden.
Die Problematik in der Erstellung eines Orakels liegt im Umfang und in der Komplexität der auszuwertenden Daten, die eine Vorhersage des korrekten Verhaltens sehr aufwendig gestaltet, besonders für Systeme deren Zustände nicht beobachtbar sind.

Bei allen zufallsbasierten Tests muss sichergestellt sein, dass ein zufällig generierter Testfall reproduziert werden kann. Wird durch die automatisierte Auswertung der Testfälle ein Fehler oder eine Abweichung festgestellt, müssen die Eingangsdaten und Randbedingungen des ausgeführten Testfalls festgehalten werden, um eine exakte Reproduzierbarkeit zu gewährleisten.

Simulationsbasierte Testfallgenerierung

Ein wichtiger Bestandteil der zufallsbasierten Testfallgenerierung, insbesondere für Module die mit ihrer Umwelt interagieren, ist die Simulation. Die Eingangsdaten werden für das zu testende System mit Hilfe möglichst realitätsnaher System- bzw. Umfeldmodelle erzeugt. Dadurch werden implizit neue Testfälle generiert. Es ist evident, dass eine effiziente Überprüfung des Systemverhaltens nur automatisiert erfolgen kann. Eine solche automatisierte Auswertung bzw. die Implementierung eines geeigneten Testorakels stellt gerade für komplexere Module eine Herausforderung dar. Lösungsansätze hierfür werden in nachfolgenden Abschnitten vorgestellt.
Die Simulation stellt einen Lösungsansatz für das Problem der Testabdeckung dar. Aufgrund der Komplexität der zu testenden Systeme ist eine vollständige Testabdeckung nicht realisierbar. Folglich muss eine Auswahl von möglichst relevanten Testfällen getroffen werden. Als relevant werden dabei Testfälle angesehen, die zum einen besonders häufig auftretende Situationen oder aber zum anderen besonders kritische Situationen abdecken. Diese Definition ist an die Berechnung des Risikos *R* mit Gl. (1) angelehnt, wobei *P(E)* die Eintrittswahrscheinlichkeit eines Fehlerereignisses *E* und *I(E)* das zu erwartende Schadensausmaß (impact) des Fehlerereignisses darstellt.

$$\boldsymbol{R = P(E) \cdot I(E)} \tag{1}$$

Für den eingangs vorgestellten Anwendungsfall stellt die detaillierte Verkehrsimulation eine Lösung für die vorangehend erwähnten Probleme der Testfallerzeugung und -auswahl dar.

Eine Simulation erzeugt implizit Testfälle. Gleichzeitig ist die Erzeugung besonders häufiger Situationen ebenfalls implizit gegeben. Für die Generierung von relevanten Testfällen müssen jedoch noch, gemäß der vorangehend gegebenen Definition, kritische Situationen gezielt in die Simulation integriert werden.

Ein möglicher Ansatz zur Lösung dieser Problematik stellt die systematische Gliederung in einzelne kritische Verkehrsszenarios dar. Diese Gliederung erfolgt anhand des Unfalltypenkataloges der Unfallforschung der Versicherer (UDV) [20], der auf den Empfehlungen zur Auswertung von Straßenverkehrsunfällen der Forschungsgesellschaft für Straßen- und Verkehrswesen basiert [15]. Die Bestimmung des Unfalltyps ist ein wichtiger Bestandteil in den Unfalluntersuchungen der Unfallforschung.

Der Unfalltyp kennzeichnet die Situation, die auslösend für den Unfall war. Die Zuordnung des Unfalltyps basiert nur auf der auslösenden Verkehrssituation. Der weitere Verlauf eines Konfliktes fließt nicht in die Zuordnung ein. Durch eine dreistellige Ziffernfolge sind etwa 300 verschiedene Unfalltypen codiert und gliedern sich in folgende Kategorien:

1. Fahrunfall
2. Abbiege-Unfall
3. Einbiegen/Kreuzen-Unfall
4. Überschreiten-Unfall
5. Unfall durch ruhenden Verkehr
6. Unfall im Längsverkehr
7. Sonstiger Unfall

Diese Kategorien sind nochmals untergliedert, so dass der Typ einer unfallauslösenden Konfliktsituation sehr spezifisch angegeben werden kann (siehe Abbildung 1).

Da diese Gliederung der Unfalltypen empirisch aus der Unfallanalyse entstanden ist, können diese Unfalltypen als kritische Situationen im Straßenverkehr verstanden werden und stellen somit gemäß (1) aufgrund der „teuren“ Folgen (*I(E)* ist groß) relevante Testfälle für ein autonomes Straßenfahrzeug dar.

Es bleibt anzumerken, dass die Unfalltypen und Unfallstatistiken keinen alleinigen Maßstab für autonome Fahrzeuge im Straßenverkehr darstellen, sondern Ereignisse des realen Verkehrs widerspiegeln. Dessen ungeachtet bleiben sie gefährliche und kritische Situationen, die sich fortwährend im Straßenverkehr ereignen und somit auch für autonome Straßenfahrzeuge äußert relevant sind.

Die einzelnen Unfalltypen werden nun als Ausgangssituationen für Testpatterns, also als Entwurfsmuster und Vorlagen zur Erzeugung von Testfällen, verwendet. Auf Basis dieser Testpatterns können Testfälle erzeugt werden, die eine hohe Zahl der im Straßenverkehr auftretenden kritischen Situationen abdecken. Für jeden einzelnen Unfalltyp existiert ein eigenes Pattern, das durch diverse Parameter konfiguriert werden kann. Mit Hilfe dieser Parameter ist eine generische Erzeugung von Testfällen möglich, so dass diese mit der Einbettung in die Verkehrssimulation automatisiert erzeugt und anschließend ausgewertet werden können.

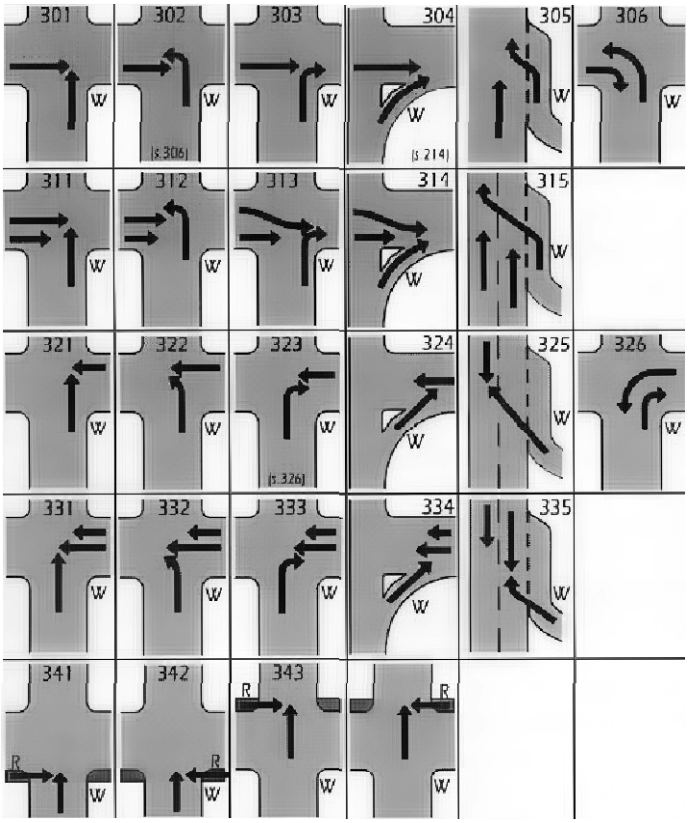

Abbildung 1: Auswahl der Unfalltypen eines Einbiegen/Kreuzen-Unfalls [15]

Mit einer detaillierten Verkehrssimulation wird neben der Problematik der Testabdeckung der Umstand einer unvollständigen Anforderungsdefinition berücksichtigt. Es werden die Situationen implizit durch eine mikroskopische Verhaltensbeschreibung sowie das modellierte Umfeld erzeugt. Mikroskopische Verhaltensbeschreibungen sowie deren Anforderungen sind mittlerweile gut erforscht. Für einen Überblick zu Verhaltensmodellen sei auf [13] und [21] verwiesen. Eine Herausforderung stellt hierbei allerdings die Anforderungsdefinition für die Modellierung des Umfelds dar. Diese bestimmt im Wesentlichen die Art der implizit erzeugten Testfälle. Die Anforderungen sind im Gegensatz

zu den Systemanforderungen jedoch besser beherrschbar, insbesondere wenn das zu betrachtende Szenario bekannt ist.

Automatisierte Validierung und Evaluierung

In den vorangehenden Abschnitten wurde auf die Problematik der Testfallerzeugung eingegangen. Nachfolgend soll auf die Auswertung von Testfällen näher eingegangen werden. Hier ist es für viele Testfälle nicht nur entscheidend, ob ein Testfall bestanden wurde oder fehlgeschlagen ist, sondern auch wie gut ein Testfall bestanden wurde. Als Beispiel sei hier die Bahnplanung erwähnt. Als K.O.-Kriterien sind u.a. eine Kollision oder das Verlassen eines erlaubten Bereiches anzusehen. Gleichzeitig sind darüber hinaus auch „weichere" Kriterien wie die Qualität der geplanten Bahn entscheidend. Damit kann insbesondere bei komplexen Modulen die Bewertung eines bestandenen Testfalls aufwendig sein.
Die Auswertung wird daher durch eine automatisierte Validierung und anschließende Evaluation der Ergebnisse durchgeführt. Die Validierung überprüft über das entsprechende Testorakel anhand von definierten Kriterien, ob ein Testfall bestanden wurde. Diese Kriterien gelten als K.O.-Kriterien und müssen folglich vollständig erfüllt sein, um den Testfall zu bestehen.
Die Evaluation hingegen überprüft nicht die Korrektheit der Funktion des Systems, sondern bewertet auf Basis von zuvor definierten Gütemaßen und explizit zugelassenen Fehlertoleranzen den Erfolg eines Testfalls.
Die Evaluation kann allgemein als Prozess der Untersuchung und der Bewertung verstanden werden [12]. Die Evaluation grenzt sich dabei von der Validierung ab, indem sie nicht funktionale sondern Qualitätskriterien betrachtet. Die reine Funktion des zu evaluierenden Systems gemäß den Anforderungen wird als gegeben vorausgesetzt. Das Ergebnis einer Evaluation ist demnach ein Qualitätsmaß [17].
Die Methoden der Evaluation lassen sich unabhängig von der Anwendungsdomäne beschreiben und klassifizieren. Eine Evaluation besteht im Allgemeinen aus mehreren (Qualitäts-)Kriterien, die zusammengenommen die Gesamtqualität des Systems oder des Moduls spezifizieren. In der Regel werden die Kriterien bestimmten Kategorien wie zum Beispiel Performance und Bahnqualität zugeordnet. Jeder Bereich kann sich dabei aus mehreren weiteren Unterkriterien zusammensetzen (vgl. Abbildung 2). Die hierarchische Anordnung erlaubt dabei eine Verfeinerung bzw. Zusammenfassung der Kriterien.

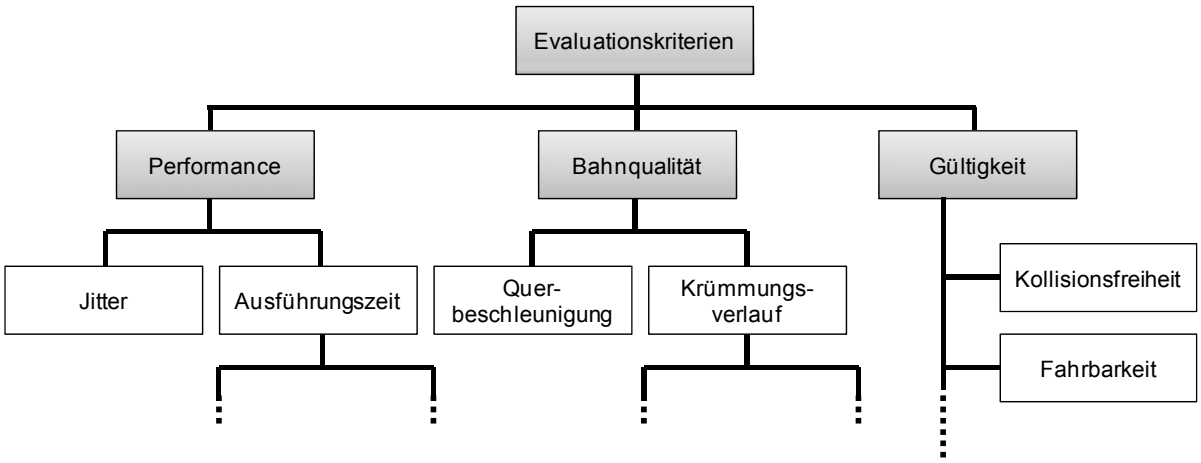

Abbildung 2: Evaluationskriterien für die Bahnplanung

Es gibt verschiedene Evaluationsmethoden, nach denen die vorhergehend beschriebenen Kriterien ermittelt werden können [17]. Für eine automatisierte Evaluierung, wie in diesem Anwendungsfall gefordert, werden experimentelle Methoden wie die bereits erwähnte Simulation und die Praxiserprobung verwendet.

Um Kriterien messbar zu machen, müssen ihnen Attribute zugeordnet werden. Der Attributwert selbst bzw. die Quantifizierung des Attributs kann durch verschiedene Evaluationstechniken bestimmt werden, die als Beschreibungen, wie verschiedene Bereiche eines Systems auf welche Weise zu bewerten sind, aufgefasst werden. Die Bewertung selbst kann sowohl qualitativ als auch quantitativ ausfallen [19]. Neben den Techniken selbst ist die Integration der Kriterien ein wesentlicher Bestandteil einer Evaluation. Dazu gehört sowohl die Gliederung der Kriterien in einer Struktur oder Hierarchie (vgl. Abbildung 2) als auch die Zusammenführung der Bewertungsergebnisse zu einem Zwischen- oder Gesamtergebnis.

Sollen Kriterien mit Ergebnissen unterschiedlichen Typs zusammengeführt werden, empfiehlt sich die Verwendung einer Qualitätsrate. Mit Hilfe einer Gewichtung der Unterkriterien ist neben einer Interpretation auch eine einfache Zusammenführung zu einem Gesamtergebnis möglich. Das Gewicht C_i beschreibt dabei die Relevanz eines Unterkriteriums für das übergeordnete Kriterium. Die Integration erfolgt letztlich nach Gl. (2) als gewichtete Summe der Qualitätsraten $Q_{Kriterium\ i}$, wobei die Summe aller Gewichtungen C_i 1 ergeben muss (vgl. Gl. (3)).

$$\mathbf{Q_{Gesamt} = \sum_i C_i \cdot Q_{Kriterium\ i}} \tag{2}$$

$$\mathbf{\sum_i C_i = 1} \tag{3}$$

$Q_{Kriterium\ i}$ = **Qualitätsrate für Kriterium i**

C_i = **Gewichtung der Qualitätsrate** $Q_{Kriterium\ i}$

Förderung robuster Software

Der Anspruch an die Qualität des zu entwickelnden Systems ist aufgrund des Einsatzes im Straßenverkehr sehr hoch. Daher wird nicht zuletzt durch intensives Testen versucht, ein möglichst fehlerfreies System zu schaffen. Ziel ist es außerdem, Software zu entwickeln, die robust gegenüber unvorhergesehenen Fehlern ist.
Nach [14] stellt Robustheit eine Qualitätsanforderung an ein Softwaresystem dar und ist wie folgt definiert: „Software ist robust, wenn fehlerhafte Eingaben zurückgewiesen werden, ohne dass sie in einen undefinierten Zustand gerät, sinnlose Ergebnisse berechnet oder gar unkontrolliert abstürzt. Ein robustes System stürzt niemals unkontrolliert ab, sondern reagiert auf Fehleingaben in einer kontrollierten Weise und befindet sich immer in einem wohldefinierten Zustand."

Daher müssen alle Module jederzeit auf unerwartete Fehler reagieren können. Im Idealfall kann das System trotz eines aufgetretenen Fehlers seine Funktion erfüllen. Sollte das System jedoch zu stark in der Funktion beeinträchtigt sein, muss es so robust sein, dass es noch in einen sicheren Fehlzustand überführt werden kann.

Zur Überprüfung und Förderung der Robustheit der einzelnen Systeme erfolgen bei den durchgeführten Tests auch gezielt fehlerhafte Eingaben. Weiterhin werden Testfälle erzeugt, die schärfere Anforderungen zum Bestehen dieses Testfalls an das System stellen als in der Validierung vorgesehen sind. Auch auf diese Testfälle muss das System angemessen reagieren können, ohne einen kritischen Zustand zu erreichen. Beispielsweise wird das Fahrzeug mit simulierten Verkehrssituationen konfrontiert, die schrittweise kritischer werden, um das Verhalten auch in Ausnahmesituationen bewerten zu können.

Praxiserprobung

Die vorgestellten Testmethoden sind notwendig, um einen gewissen Reifegrad der Systeme sicherzustellen. Eine eigentliche Fahrerprobung unter realen Bedingungen kann damit nicht ersetzt, aber zu einen ausgereifteren System hin verzögert werden. So muss nun als

notwendiger Schritt eine Praxiserprobung erfolgen, zunächst jedoch in einem abgeschlossenen Testgelände. Erst nachdem auch hier alle Testszenarien bestanden wurden, kann ein Praxiseinsatz auf öffentlichen Straßen erfolgen.

4. Umsetzung am Beispiel der Bahnplanung

Die Anwendung des vorgestellten Test- und Simulationskonzeptes soll im Folgenden am Beispiel des Bahnplanungsmoduls durchgeführt werden. Dabei wird Schritt für Schritt die Erfüllung der vorgestellten Grundsätze dargestellt.

Die Bahnplanung stellt ein wichtiges Modul innerhalb der autonomen Fahrzeugführung dar. Die Fahrzeugführung eines autonomen Straßenfahrzeugs setzt sich aus mehreren Teilsystemen zusammen (siehe Abbildung 3). Auf Basis der im Fahrzeug integrierten Umfelderkennung kann die Fahrzeugführung das Verhalten des Fahrzeugs und ein eventuell erforderliches Fahrmanöver festlegen und ausführen. Zunächst bereitet die Situationsanalyse die aktuellen Umfelddaten für eine Entscheidungsinstanz auf. Ein Fahrentscheider bestimmt einen passierbaren Korridor als Resultat des gewünschten Fahrmanövers. Es ist schließlich Aufgabe der Bahnplanung, innerhalb dieses sogenannten Fahrschlauchs eine geeignete Trajektorie zu berechnen, die der Fahrzeugregelung als Sollbahn dient.

Diese Trajektorie wird durch die Bahnplanung sowohl auf Fahrkomfort als auch hinsichtlich der Fahrdynamik optimiert. Zum einen geschieht dies offline vor der Fahrt in Form einer a-priori-Planung, zum anderen online durch eine dynamische Anpassung an die veränderliche Umgebung während der Fahrt.

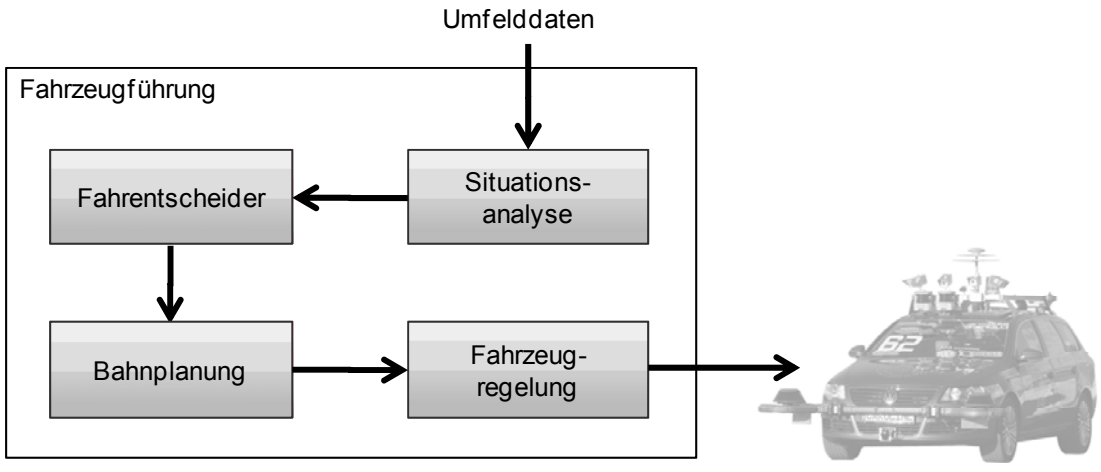

Abbildung 3: Schematischer Aufbau der autonomen Fahrzeugführung

Ein fehlerfreies Bahnplanungsmodul ist äußerst wichtig, da die erzeugte Trajektorie den Weg des Fahrzeugs darstellt. Fehler in diesem Modul könnten zu einer nicht mehr kollisionsfreien Bahn des Fahrzeugs führen, die unbedingt vermieden werden muss. Ausgiebige Tests können hier einen großen Beitrag leisten.

Entwicklungsbegleitende Tests

Wie vorab beschrieben, erfolgt das Testen des Moduls auf mehreren Ebenen. Auf Unit- und Modulebene sowie im Verbund mit anderen Komponenten. Durch einen frühzeitigen Beginn des Testens können bereits Fehler während der Implementierung erkannt und behoben werden. Auf Modulebene muss weiterhin die Funktion des ganzen Moduls überprüft werden. Durch eine Kombination aus manueller und generischer Testfallerzeugung soll die notwendige Reife des Moduls der Bahnplanung gewährleistet werden. Die Durchführung entwicklungsbegleitender Tests hat sich bei der Entwicklung des Bahnplanungsmoduls bereits bewährt.

Manuelle Testfallerzeugung

Die Problematik der Testfallerzeugung für das Bahnplanungsmodul besteht darin, dass die Zahl der möglichen Fahrschläuche theoretisch unbegrenzt ist, so dass die Abdeckung aller möglichen Testfälle nicht zu erreichen ist. Daher werden die manuell erstellten Testfälle auf gezielt ausgesuchte und besonders relevante Fälle beschränkt.
Zusätzlich zu den bereits erläuterten statischen Analysen und Unittests, in denen die elementaren Methoden und Komponenten des Moduls überprüft werden, basieren die manuellen Tests zu einem großen Teil auf aufgezeichneten Realdaten des Braunschweiger Stadtrings. Diese Testfälle bestehen aus einzelnen Teilstücken, angepassten Verkehrssituationen und der Gesamtstrecke.

Generische Testfallerzeugung

Die Erzeugung generischer Testfälle dient der Überprüfung der Funktion des Moduls bei dynamischen Änderungen der Strecke und unvorhersehbaren Situationen. Die dazu erforderlichen Fahrschläuche werden aus einzelnen zufällig ausgewählten und geformten Segmenten erzeugt. Diese Segmente leiten sich aus realen Straßensegmenten ab wie

verschieden lange Geraden, Kurven oder verengte Fahrbahnen. Dabei werden neben einspurigen auch mehrspurige Fahrbahnen erzeugt, um auch Spurwechsel zu ermöglichen. Die Auswahl und die Bestimmung der Eigenschaften wie Länge, Fahrbahnbreite oder Kurvenradien dieser Segmente erfolgen zunächst völlig zufällig. Abbildung 4 zeigt einen generisch erzeugten zweispurigen Fahrschlauch mit einer Gesamtlänge von 9,58km mit minimalen Kurvenradien von 19m.

Weiterhin kann die Erzeugung auch entsprechend parametriert werden, so dass beispielsweise eher kurvige oder eher gradlinige Fahrschläuche mit wahlweise kleineren oder größeren Kurvenradien entstehen, um die zu erzeugenden Testfälle an spezielle Umgebungen wie Autobahnen oder verwinkelte Wohngebiete anzupassen.

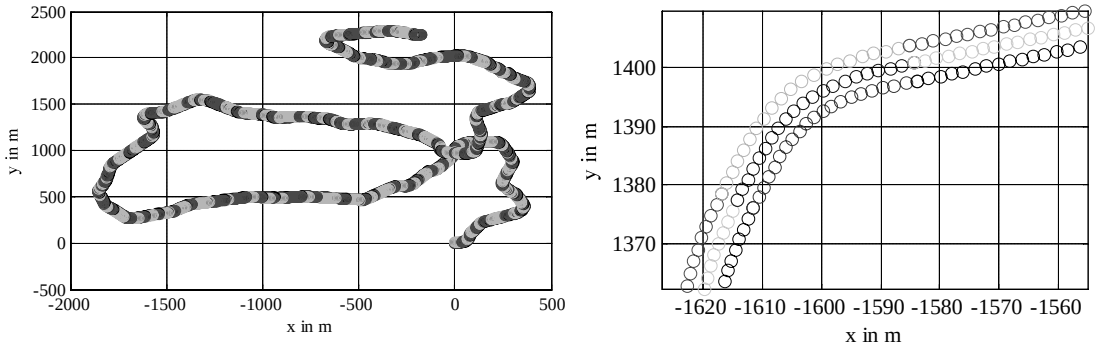

Abbildung 4: Generisch erzeugter zweispuriger Fahrschlauch

Evaluierung und Validierung mittels Gütemaßen und K.O.-Kriterien

Die durch die Bahnplanung berechneten Trajektorien werden mit Hilfe einer automatisierten Validierung und Evaluation beurteilt. Durch die Validierung wird überprüft, ob die Trajektorie einen gültigen Verlauf besitzt. Die Gültigkeit wird anhand mehrerer Kriterien beurteilt, wie z.B. der Anforderung, dass die Trajektorie einen kollisionsfreien Verlauf beschreibt. Die Trajektorie muss außerdem fahrdynamisch sinnvoll realisierbar sein, d.h. sie darf u.a. keine Lenkwinkelsprünge enthalten und einen minimalen Kurvenradius nicht unterschreiten.

Sind diese Kriterien vollständig erfüllt, erfolgt anhand von definierten Gütemaßen (siehe Abbildung 2) eine Evaluation des Ergebnisses. Die Gütemaße ermöglichen eine Bewertung, die beispielsweise verschiedene Parametrierungen des Moduls vergleichbar macht. Für die Bahnplanung sind nicht nur Performanzkriterien wie Ausführungszeiten oder die Echtzeitfähigkeit der Algorithmen entscheidend, sondern auch die Qualität der geplanten

Trajektorie und der Fahrkomfort. Die Evaluation erfolgt demnach u.a. durch eine gewichtete Bewertung anhand der auftretenden Querbeschleunigung, der Krümmungs- bzw. Lenkwinkeländerung und weiteren fahrdynamischen Einflussgrößen.

Förderung robuster Software

Anhand von gezielt fehlerhaften Eingangsdaten wird die Robustheit des Moduls überprüft. Es werden gezielt Fahrschläuche generiert, für die die Validierung des übergeordneten Fahrentscheiders fehlgeschlagen wäre. So weisen diese Fahrschläuche beispielsweise einen zu geringen Kurvenradius auf, so dass es der Bahnplanung nicht möglich ist, eine gültige Trajektorie zu planen. Diese nicht fahrbaren Korridore dürfen nicht zu einem Ausfall des Moduls führen. Das Modul muss in einen sicheren Zustand überführt werden, aus dem das Fahrzeug beispielsweise kontrolliert zum Halten gebracht werden kann.

Zufallsbasierte Testfallgenerierung auf Basis einer Simulation

Ein Integrationstest für das Modul der Bahnplanung ist der Verbund zusammen mit dem übergeordneten Modul des Fahrentscheiders und der nachfolgenden Fahrzeugregelung (siehe Abbildung 5). Anhand dieses Modulverbundes soll der Einsatz einer Verkehrssimulation zur Testfallerzeugung erläutert werden.
Die Verkehrssimulation simuliert das Umfeld des Fahrzeugs bestehend aus anderen Verkehrsteilnehmern und der urbanen Umgebung und stellt dem Fahrentscheider diese Daten als Objekte bzw. als gitterbasierte Daten zur Verfügung.
Die Ausgangsdaten der Fahrzeugregelung stellen die Eingangsdaten der Simulation dar, die daraus die simulierte Bewegung des Fahrzeugs und die Reaktionen der Umgebung ermittelt.
In dem Verbund erfolgt während der Simulation für jedes Modul die jeweilige Validierung und Evaluation, um die korrekte Funktion der einzelnen Module zu überprüfen und zu bewerten.

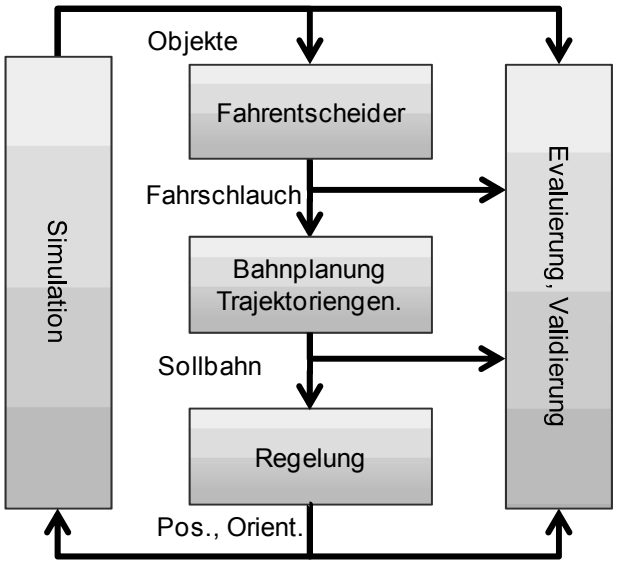

Abbildung 5: Simulation im Modulverbund

Um das Umfeld eines autonomen Straßenfahrzeugs simulieren zu können, ist eine möglichst detaillierte mikroskopische Verkehrssimulation erforderlich, die durch eine hohe Anzahl von simulierten Verkehrsteilnehmern einen realen Verkehrsfluss erzeugen kann. Dies setzt eine Modellierung der Längs- und Querdynamik der Fahrzeuge voraus, um eine realistische Fahrzeugbewegung abbilden zu können. Zur realitätsnahen Simulation des Braunschweiger Stadtrings ist neben der hohen Verkehrsdichte auch ein realistisches Verhalten der Verkehrsteilnehmer darunter auch Fußgänger notwendig, um eine Interaktion miteinander zu erreichen. Dieses Fahrerverhalten ist durch verschiedene Eigenschaften wie Beschleunigungsvermögen, Querverhalten, Wunschgeschwindigkeit und Wunschabstand charakterisiert.
Durch diese detaillierte Simulation ergeben sich kontinuierlich neue Verkehrssituationen. Der Schwerpunkt der Testszenarien kann durch verschiedene Parameter wie Verkehrsdichte, Geschwindigkeitsverteilung und andere Einflussgrößen bestimmt werden.

Darüberhinaus werden zur gezielten Überprüfung relevanter Situationen die bereits beschriebenen Testpatterns auf Basis der Unfalltypen genutzt.
Als Beispiel stellt Typ 301 (siehe Abbildung 6) die Konfliktsituation zwischen einem wartepflichtigen und einem von links kreuzenden bevorrechtigten Fahrzeug dar. Diese Situation kann für die Simulation anhand diverser Parameter darunter u.a. fahrzeugbezogene Parameter wie die Geschwindigkeit der Fahrzeuge, Abstand zur Kreuzung oder auch

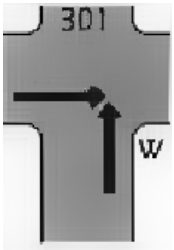

Abbildung 6: Unfalltyp 301

fahrbahnspezifische Ausprägungen wie Spurbreite oder Randbebauungen beliebig variiert werden, um möglichst viele Testfälle zu erzeugen.

Aus einer generisch erzeugten Instanz dieses Patterns werden die Anfangsbedingungen für die Simulation erzeugt. Bei Initialisierung des Patterns werden die Geometrie der Kreuzung festgelegt und anschließend die beiden beteiligten Fahrzeuge mit ihren jeweiligen Anfangszuständen platziert. Nach Abschluss der Simulation erfolgt eine automatisierte Auswertung des Situationsverlaufs, so dass direkt im Anschluss eine neu parametrierte Instanz erzeugt und simuliert werden kann.

Die Häufigkeit der angewendeten Patterns ist dabei abhängig von den jeweiligen Unfallzahlen des Patterns[3] um auch hier die Forderung nach Relevanz zu erfüllen.

Eine realistische Verkehrssimulation gepaart mit der gezielten Einstreuung kritischer Verkehrssituationen nach dem Unfalltypenkatalog stellt damit eine hervorragend geeignete Methode für die Erzeugung relevanter Testfälle dar.

5. Zusammenfassung

Das vorgestellte Simulations- und Testkonzept dient aufbauend auf den beschriebenen Grundsätzen der Sicherstellung des zum Einsatz eines autonomen Straßenfahrzeugs im öffentlichen Straßenverkehr notwendigen Reifegrades und der Robustheit der verwendeten Softwaresysteme.

Künstlich erzeugte und idealisierte Testdaten stellen zwar keinen Ersatz, jedoch eine notwendige Vorbedingung für die Erprobung unter realen Bedingungen dar. Besonders im Bereich der Simulation des Umfeldes und der Sensorik ist die Erzeugung realitätsnaher Daten äußerst komplex und aufwendig. Real erzeugte Daten unterscheiden sich daher von simulierten. Dennoch kann die Funktion der Module und des Systems in gewissen Grenzen in künstlicher Umgebung getestet werden, um den nötigen Reifegrad zur Erprobung in realer Umgebung sicherzustellen.

Das vorgestellte Konzept bereitet damit die autonome Fahrt im öffentlichen Straßenverkehr sinnvoll vor.

[3] Die Unfallzahlen der Unfalltypen werden u.a. durch das Statistische Bundesamt bekannt gegeben.

6. Literaturverzeichnis

[1] Wille, J.M.; Homeier, K.; Matthaei, R.; Nothdurft, T.; Ohl, S.; Sasse, A.; Saust, F.; Hecker, P.; Maurer, M.; Schumacher, W.; Wolf, L.: Der Stadtpilot. Autonomes Fahren auf dem Braunschweiger Stadtring. In: *AAET 2009. Automatisierungs-, Assistenzsysteme und eingebettete Systeme für Transportmittel* (2009)

[2] Hoffmann, D.: *Software-Qualität*. Springer, 2008

[3] Pretschner, A.: *Zum modellbasierten funktionalen Test reaktiver Systeme*. München, Technische Universität, Diss., 2003

[4] Horstmann, M: *Verflechtung von Test und Entwurf für eine verlässliche Entwicklung eingebetteter Systeme im Automobilbereich.* Braunschweig, Technische Universität, Diss., 2005

[5] Barltrop, K.J.; Friberg, K.H.; Horvath, G.A.: Automated Generation and Assessment of Autonomous Systems Test Cases. In: *IEEE Aerospace Conference* (2008), S.1-10

[6] Ciupa, I.; Meyer, B.; Oriol, M.; Pretschner, A: Finding Faults: Manual Testing vs. Random+ Testing vs. User Reports. In: *19th International Symposium on Software Reliability Engineering* (2008), S.157-166

[7] Breuer, K.: *Verkehrsflusssimulation zur Entwicklung von Fahrerassistenzsystemen*. Aachen, Rheinisch-Westfälische Technische Hochschule, Diss., 2004

[8] Gietelink, O.J.; Verburg, D.J.; Labibes, K.; Oostendorp, A.F.: Pre-Crash System Validation with PRESCAN and VEHIL. In: *IEEE Intelligent Vehicles Symposium* (2004), S. 913 – 918

[9] Bock, T.; Maurer, M.; Farber, G.: Validation of the Vehicle in the Loop (VIL) – A milestone for the simulation of driver assistance systems. In: *Proceedings of the IEEE Intelligent Vehicles Symposium* (2007), S. 612 - 617

[10] Thiruvathukal, G.K.; Läufer, K.; González, B.: Unit Testing Considered Useful. In: *IEEE Computational Science & Engineering* (2006), Nr. 8, S. 76 – 87

[11] Lehmann, E: *Time Partition Testing - Systematischer Test des kontinuierlichen Verhaltens von eingebetteten Systemen*. Berlin, Technische Universität, Diss., 2004

[12] Helmut Balzert: *Lehrbuch der Software-Technik: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung*. Heidelberg: Spektrum Verlag, 1998

[13] Brackstone, M.; McDonald, M.: Car-following: a historical review. In: *Transportation Research Part F* (1999), Nr. 2, S. 181-196

[14] Jähnichen, Heisel, Weber: *Softwaretechnik*, Vorlesungsmanuskript, Technische Universität Berlin, 1996.

[15] Institut für Straßenverkehr, Gesamtverband der Deutschen Versicherungswirtschaft: Sicherung des Straßenverkehrs –SVS- Teil 1, 1998.

[16] Liggesmeyer, P.: *Software-Qualität: Testen, Analysieren und Verifizieren von Software*. Heidelberg: Spektrum Verlag, 2002

[17] Müller, T.; Krieger, O.; Form, T.: Evaluierung von Offboard-Diagnosesystemen am Beispiel eines erfahrungsbasierten Diagnoseverfahrens. In: *Simulation und Test in der Funktions- und Softwareentwicklung für die Automobilelektronik*, 2008

[18] Oppermann, R.; Murchner, B.; Reiterer, H.; Koch, M.: *Softwareergonomische Evaluation - Der Leitfaden EVADIS II* (2. Auflage) Berlin; New York: Walter de Gruyter, 1992, S. 10ff.

[19] Bahsoon, R.; Emmerich, W.: Evaluating Software Architectures: Development, Stability, and Evolution. In: *AICCSA*, 2003

[20] Unfallforschung der Versicherer im Gesamtverband der Deutschen Versicherungswirtschaft: Interaktiver Unfalltypenkatalog, www.unfallforschung-der-versicherer.de Aktualisierungsdatum: 08.01.2009

[21] Toldeo, T: Driving Behaviour: Models and Challenges. In: *Transport Reviews* (2007), Nr. 27, S. 65-84