Benjamin Zaiser
Matrikelnr.: 16129
Medieninformatik Bachelor
Hochschule der Medien Stuttgart

Bericht über das Integrierte Praktische Studiensemester (IPS) mit begleitendem Literaturstudium bei der DaimlerChrysler AG

1. März 2007, Böblingen

1. Kurzfassung

Dieser Bericht gliedert sich in zwei Teile. Zum Einen die Literaturarbeit, welche Themen behandelt, die ich während meines Integrierten Praktischen Studiensemesters bearbeitet habe. Zum Andern meinen Erfahrungsbericht über das IPS. Dieser beinhaltet detaillierte Aufgabenstellungen und den dazugehörigen Problemlösungen.

Inhaltsverzeichnis

2. Vorwort

Dieses Dokument enthält die Literaturarbeit und meinen Erfahrungsbericht bei der DaimlerChrysler AG in Böblingen/Hulb. Recht herzlich bedanken möchte ich mich bei Frau Dagmar Hermann, die mich bei dem ersten Projekt TMC und das ganze Semester über betreut hat. Außerdem möchte ich mich bei Andreas Hiller und Thomas Passegger bedanken, die mich während der Projekte Willwarn und GPS-Vergleich betreut haben und mir bei technischen Fragen zur Verfügung standen.

Dieser Bericht wurde mit LaTeXverfasst. Das hat den Grund, ein geeignetes System zu finden, um später meine Bachelor Thesis zu verfassen, da Microsoft Word manchmal ein eigenartiges Verhalten hat. Nach einer relativ langen Eingewöhnungszeit, erscheint mir LaTeXals eine sehr mächtiges und effektives Werkzeug, zum Verfassen großer Arbeiten.

Teil I.

Literaturstudium

Im nun folgenden Literaturstudium werden drei Themen behandelt. Diese sind TMC, PReVENT und GPS, wobei der Schwerpunkt auf dem zuerst genannten Thema liegt. Dies dient unter anderem zur Vermittlung eines Basiswissens, das für den eigentlichen Bericht des IPS erforderlich ist.

3. TMC

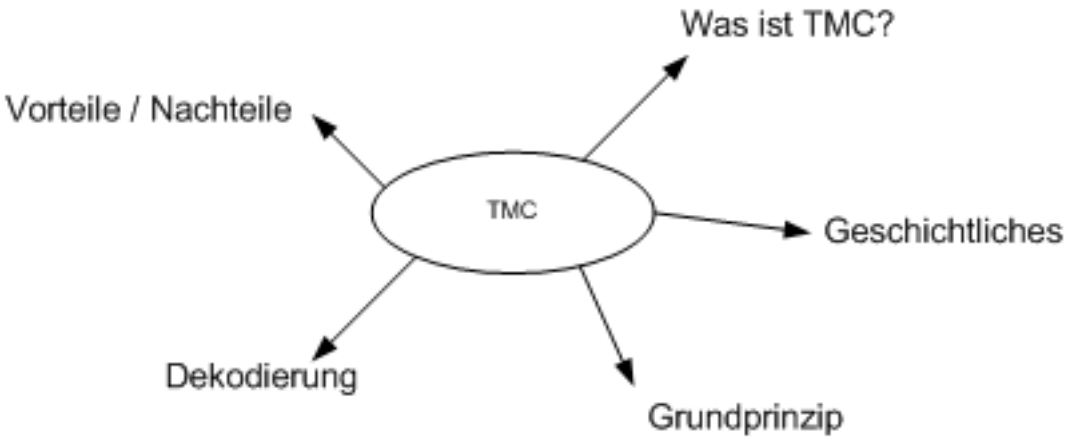


Abbildung 3.1.: MindMap zu dem nun folgenden Kapitel

3.1. Was ist RDS-TMC?

TMC bedeutet **T**raffic **M**essage **C**hannel und ist ein digitaler Radiodienst. Er wird dazu verwendet, Verkehrsstörungen an entsprechendes Empfangsgerät zu übertragen.
Als Empfangsgerät kann ein Autoradio das TMC fähig ist oder die heutzutage stark aufkommenden tragbaren Navigationssysteme fungieren. Die Meldungen werden über einen Hilfsträger mit einer Frequenz von 57kHz von den lokalen Radiosendern ausgestrahlt.

3.2. Geschichtliches

In den 60er Jahren nahm das Verkehrsaufkommen stark zu. Aufgrund der technischen Entwicklung der Radiosender stieg die Bandbreite eines Senders und somit waren die Weichen für einen Verkehrsfunk, der über den Radio-Kanal ausgestrahlt wird, gestellt. An Autobahnraststätten war damals ein Tonbandgerät aufgestellt, das sich, sobald ein sog. “Hinz-Triller” empfangen wurde, automatisch eingeschaltet hat, um den folgenden

Verkehrsbericht aufzunehmen. Der "Hinz-Triller" ist eine Tonfolge, die nicht in der E- oder U-Musik vorkommt und dient deshalb als eindeutiges Erkennungsmerkmal für den Start der Verkehrsfunkübertragung.
Mitte der 70er Jahre wurde die "Autofahrer-Rundfunk-Information (ARI)" eingeführt. Dieses System konnte unter anderem das Radio automatisch für die Durchsage lautstellen und Verkehrsfunksender durch einen automatischen Suchlauf erkennen. Am 1.März 2005 wurde das ARI-Signal abgeschaltet. [8]
Heute übernimmt das *Radio Data System* die Ausstrahlung des Verkehrsgeschehens. Dieses wurde 1983 entwickelt und dient zur Übertragung digitaler non-auditiver Signale über die Radiofrequenzen. Unter anderem hat es die Aufgabe Sendernamen, Sendeinhalte, Sparte, Zeitsynchronisation etc. zu übertragen oder einen Frequenzwechsel auszulösen. [7]

3.3. Grundprinzip

Die TMC Nachricht an sich kann den Autofahrer nicht über Stauereignisse informieren. Es gehören mehrere Akteure dazu, die nun nähers erläutert werden.

Erstellung der Daten

Zunächst muss das Verkehrsgeschehen beobachtet werden. Dies geschieht z.B. durch die Polizei, die bei einem Unfall oder ähnliches eine Nachricht verfassen und weiterleiten kann. Aber auch Verkehrsclubs wie z.B. der ADAC können Störungen melden oder die vielen Messstationen, die per Induktionsschleife in der Fahrbahn das Verkehrsaufkommen messen. Es ist auch möglich, selbst als Autofahrer Störungen zu melden. Dies geschieht durch einen Anruf der Hotlines vieler privater Radiosender.
Außerdem gibt es noch vollautomatische Verkehrsüberwachungsstationen. Die Gesellschaft für Verkehrsdaten mbH [3] bietet diesen Service an. Sie haben rund 4000 Sensoren an den Autobahnen aufgestellt, die die mittlere Geschwindigkeit und die jeweiligen Fahrzeugklassen (PKW / LKW) messen und per GSM an ein Rechenzentrum weiterleiten. Dieser Dienst stellt ein besseres und genaueres Abbild des Verkehrsgeschehens zur Verfügung und generiert somit genauere (Zeit und Ort) TMC Meldungen. Diese Meldungen sind auch als TMCPro bekannt und können für eine Jahrespauschale abonniert

werden. TMCPro Nachrichten werden über den Mobilfunk ausgestrahlt und über ein Handy, das mit dem Navigationssystem verbunden ist, empfangen. Seit kurzem werden die Nachrichten auch verschlüsselt über das RDS ausgestrahlt.

Verfassen der TMC Nachrichten

Die Daten müssen nun aufbereitet werden, um daraus entsprechende TMC Meldung zu generieren. Dabei muss unter anderem die Störung in einen sog. Event eingestuft, die Position genau bestimmt und mögliche Zusatzinformationen hinzugefügt werden. (Mehr dazu im Kapitel Dekodierung). Die Aufgabe der Erstellung der Nachrichten und deren Distribution übernehmen die Verkerhsinformationszentralen der Länder.

Ausstrahlen der Nachrichten

Die TMC Meldungen werden, wie bereits erwähnt, über die lokalen Radiosendestationen kontinuierlich ausgestrahlt. Sobald eine Störung vorliegt wird die dazugehörige TMC Meldung mindestens alle 15 Minuten wiederholt gesendet. Sollte das Empfangsgerät für eine Zeitspanne von 15 Minuten keine Wiederholung einer Nachricht empfangen haben, so wird die Störung automatisch als nicht mehr gültig eingestuft und gelöscht. Diese Gültigkeitsdauer kann auch durch einen Parameter in der Nachricht verändert werden. Das wiederholte Aussenden der Nachricht ist wichtig, damit auch Empfangsgeräte, die z.B. kurz nach dem ersten Ausstrahlen eine Nachricht eingeschaltet wurden, trotzdem noch über die Störung informiert werden. Eine Meldung kann auch explizit annuliert werden. Jede Eventklasse hat dabei einen eigenen Annulierungscode. Sobald das Empfangsgerät diesen empfängt, wird die Störung umgehend aus dem Speicher gelöscht.

Organisation für die Standardisierung

Das Aussehen einer TMC Meldung muss definiert werden. Dies übernimmt das CEN (Comité Européen de Normalisation). Diese Einrichtung hat das Alert-C coding protocol entwickelt, das in dem Alert-C Coding Handbook dokumentiert ist. [2] Wie im Kapitel Dekodierung nähers erklärt wird, benötigt man für die Dekodierung eine Datenbank. Diese wird ebenfalls von CEN gepflegt und bereitgestellt.
Diese Datenbank enthält eine Relation EventClasses mit den wichtigen Attributen:

- EventCode
- TextGerman

Ebenfalls enthält sie eine Relation LocationTable mit den wichtigen Attributen:

- LocationCode: (Bsp: 12532) Dieser wird in der TMC Nachricht übertragen.
- FirstName: (Bsp: Kirchheim(Teck)-West) Der Ortsname.
- SecondName: Dieser wird nur bei einer LinearReference verwendet. Siehe Direction, Extent, LocationCode
- RoadNumber: (Bsp: A8) Der Straßenname.
- LinearReference: (Bsp: 7207) Gibt einen LocationCode der nächst größeren Stadt an.
- PosOffset: (Bsp: 12533) Der nächste LocationCode an dieser Straße.
- NegOffset: (Bsp: 12531) Der vorhergehende LocationCode an dieser Straße.

Die Verwendung dieser Datenbank bei der Dekodierung wird später noch ausführlich erklärt.

3.4. Dekodierung

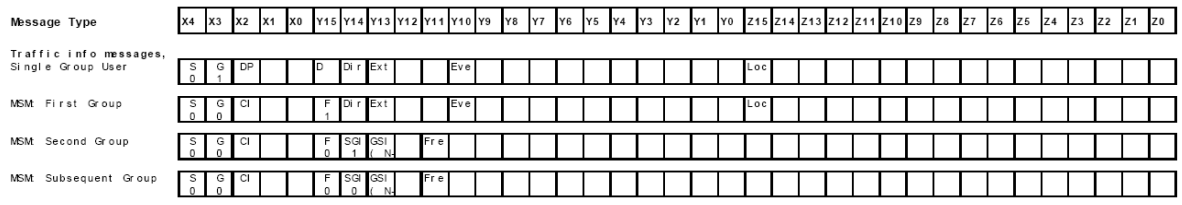


Abbildung 3.2.: Codierungstabelle der TMC Nachrichten

Bevor die Dekodierung von TMC Nachrichten erklärt wird, ist es wichtig zu wissen, dass es zwei verschiedene Übertragungsmöglichkeiten gibt. Falls für eine Verkehrsstörung zusätzliche Informationen übertragen werden sollen und eine einzige TMC Nachricht

(SingleGroupMessage) nicht ausreicht, gibt es die Möglichkeit, mehrere einzelne TMC Nachrichten zu einer Gruppe (MultiGroupMessage) zusammenzufassen. Bis zu fünf SingleGroupMessages können so zusammengefasst werden. Bei einer MultiGroupMessage ist das Group-Bit (G in Abbildung 3.2) auf 1 gesetzt. Die erste SingleGroupMessage der Gruppe ist wie eine normale Nachricht aufgebaut. Es unterscheidet sich nur das Diversion Bit. Dieses Bit wird bei einer MultiGroupMessage nur bei der ersten Nachricht gesetzt. Bei der zweiten Nachricht wird das Direction-Bit als SecondGroupIdentifier verwendet. Dieses Bit ist nur bei der zweiten Nachricht gesetzt. Die folgenden Nachrichten werden über einen laufenden Index identifiziert. Das Setzen dieser Bits ist notwendig, da der Übertragungskanal nicht gewährleistet, dass die Nachrichten nacheinander bzw. in der richtigen Reihenfolge am Empfänger ankommen.

Die Bits, die bei einer normalen Nachricht für Extent, EventCode und LocationCode verwendet werden, sind bei den zusätzlichen Nachrichten einer MultiGroupMessage zu einem Container zusammengefasst. In diesem Container werden Codepaare, die durch einen Platzhalter getrennt sind, übertragen. Dabei ist zu beachten, dass beim Empfänger die Container der zusätzlichen Nachrichten konkateniert werden müssen, um die "Liste" der CodePaare korrekt zu verarbeiten. Ein Codepaar besteht aus einem drei Bit langem Label gefolgt von weiteren Bits, deren Anzahl von dem verwendeten Label abhängt. Es sind folgende unterschiedliche Labels möglich:

Duration Gibt die Gültigkeitsdauer der Nachricht an. Dies muss hier angegeben werden, da das eigentliche Duration-Feld der ersten MultiGroupMessage anderweitig verwendet wird.

Control codes z.B. Dringlichkeit der Nachricht erhöhen / verringern, den Extent um 8 oder 16 erhöhen, ...

Length of route affected Hier kann die Staulänge, bzw. die Länge des betroffenen Streckenabschnitts angegeben werden.

Speed limit advice Es gibt die Möglichkeit eine empfohlene Höchstgeschwindigkeit für den betroffenen Abschnitt festzulegen.

Start stop times Angabe von Uhrzeiten, zu welchen das Ereignis startet oder endet. Wird vor allem bei längeren Baustellen verwendet.

Multi-event message Zusätzliche EventCodes für das Ereignis, z.B. Stau *und* Baustelle und Personen auf der Fahrbahn.

Eine Verkehrsmeldung wie z.B.

> A8: Stuttgart Richtung München, zwischen Esslingen und Wendlingen: 5km stockender Verkehr

wird nicht als eine Zeichenkette sondern in enkodierter Form übertragen. Bei einer Übertragungsgeschwindigkeit von 60bits/s (10 Meldungen pro Minute) wäre die Übertragung als Zeichenkette eine reine Resourcenverschwendung. Durch die Enkodierung müssen die Nachrichten im Empfänger zunächst dekodiert werden, um die Informationen im Navigationssystem anzuzeigen und auch entsprechend darauf zu reagieren zu können, wie z.B. dynamische Anpassung der Route.
Eine einfache TMC Meldung sieht z.B. so aus:

> 087065276C

Diese hexadezimale Zahl muss zunächst in die binäre Darstellung umgewandelt werden:

> (000)0 1000 0111 0000 0110 0101 0010 0111 0110 1100

Anhand des Alert-C Coding Handbooks [2] können die Zahlen der jeweiligen Bedeutung zugeordnet werden:

0	Systemnachricht
1	Group-Bit
000	Duration
0	Diversion
1	Direction
110	Extent
00001100101	EventCode
0010011101101100	LocationCode

Systemnachricht

Es gibt zwei verschiedene Arten von TMC-Nachrichten:

- Die “traffic info message” enthält Angaben über eine Verkehrsstörung.

- Die “system message” enthält Informationen für das Empfangsgerät, wie z.B. Frequenzwechsel oder ähnliches.

Da die Systemnachrichten nur technisch bezogene Informationen für den Empfänger enthalten, wird an dieser Stelle nicht näher darauf eingegangen.

Group-Bit

Dieses Bit gibt an, ob es sich bei der aktuellen Nachricht um eine SingleGroupMessage oder um ein Teil einer MultiGroupMessage handelt.

Duration

Die Duration gibt die Gültigkeitsdauer der Nachricht an. Standardmäßig ist diese auf 15 Minuten gesetzt. Das bedeutet, die Nachricht wird automatisch gelöscht, falls keine Wiederholung empfangen wird. Es sind acht verschiedene Gültigkeitsdauern definiert: keine, 15min, 30min, 1h, 2, 3h, 4h, bis zum Ende des Tages.

Diversion

Durch Setzen dieses Bits, ist es möglich der Nachricht eine Umleitungsempfehlung zu geben. Diese wird dann aus der LocationTable ausgelesen. Mehr dazu später.

EventCode

Das genaue Ereignis einer TMC Meldung kann mithilfe einer Datenbank und des Event-Codes extrahiert werden. In der Datenbank sind 1590 solcher Events definiert, z.b. “Ausfahrt gesperrt”, “Stau”, “Unfall”, “stockender Verkehr”, “Gefahr durch defektes Fahrzeug”, ...

Die vielen unterschiedlichen Events sind zusätzlich noch in sog. Eventklassen zusammengefasst. Bei der Darstellung einer TMC Nachricht könnte jeder Eventklasse z.B. ein Symbol zugeordnet werden. Somit muss nicht für jeden einzelnen Event eine Zuordnung getroffen werden.

Direction, Extent, LocationCode

Diese drei Parameter einer TMC Nachricht geben den Ort des Ereignisses an. Der Ursprung der Störung kann mithilfe der in 3.3 beschriebenen Datenbank und des LocationCodes extrahiert werden. Diese Datenbank gibt dann Auskunft über den Ort, die Straße, Land, usw. Bei einem Punktereignis ist der Extent = 0. Die Richtung ist dabei irrelevant. Falls aber das Ereignis eine Streckenabschnitt betrifft, muss folgender Algorithmus angewendet werden, um alle betroffenen LocationCodes zu extrahieren und den Anfang (firstLocCode) der Störung zu lokalisieren:

```
1  int i = 0; int firstLocCode = currentLocCode = -1;
2  if(i == extent) firstLocCode = currentLocCode, Stop
3  else
4          i++
5          if(direction == 1) currentLocCode = posOffset
6          else currentLocCode = negOffset
7          GoTo 2.
```

Der Extent gibt dabei an, wie oft man in die, durch Direction angegebene, Richtung (posOffset / negOffset) "springen" muss, um zum Anfang (in Fahrersicht) der Störung zu kommen.

Nun kann die Nachricht (hier eine MultiGroupMessage),

02C8652A9D 0242080000

zunächst dekodiert

LocCode: 12531
Direction: 1
Extent: 1
Event: 104
...

und dann als lesbarer Text mithilfe der Datenbank dargestellt werden.

A8, Stuttgart Richtung München,
zwischen Wendlingen und Kirchheim/Teck,
Unfall, 4km Stau

Der LocationCode würde bei dieser Nachricht Kirchheim/Teck entsprechen. Unfall und 5km Stau sind in dem Container der MultiGroupMessage gespeichert. Wendlingen wird über den LocationCode, Direction und dem Extent ermittelt. Der LocationCode von Stuttgart steht im Attribut LinearReference. München ist der SecondName der Zeile des LocationCodes von Stuttgart.

Zum besseren Verständnis sind hier noch die passenden Auszüge aus der Datenbank:

LocCode	RoadNum	FirstName	SecondName	LinRef	NegOff	PosOff
7207	A8	Stuttgart	München	50167	7206	7210
12531	A8	Wendlingen		7207	12530	12532
12532	A8	Kirchheim T.		7207	12531	12533

3.5. Vorteile / Nachteile

Vorteile

TMC bietet dem Autofahrer einen Vorteil, da die Verkehrsnachrichten jederzeit empfangen bzw. abgerufen werden können. Man ist nicht mehr an die halbstündige Durchsage des Radiomoderators gebunden. Ein weiterer großer Vorteil, im Zusammenhang mit einem Navigationssystem, ist die dynamische Anpassung der Route. Sollte das System eine TMC Nachricht für einen auf der Route befindlichen Streckenabschnitt detektieren, kann automatisch eine alternative Route berechnet werden, ohne dass der Fahrer eingreifen muss. So ist ein wesentlich entspannteres, schnelleres und umweltschonenderes Fahren möglich.

Nachteile

Der Nachteil am Verkehrsfunk an sich, ist seine Aktualität bzw. Verlässlichkeit. Aus eigener Erfahrung kann ich sagen, dass eine Verkehrswarnung manchmal erst dann empfangen wird, wenn man bereits seit ca. 15 Minuten im Stau steht. Das Phänomen eines Staus oder stockenden Verkehrs ist zu dynamisch, um von TMC Nachrichten korrekt abgebildet zu werden.

4. PReVENT

Dieses Kapitel soll einen kurzen Überblick über das PReVENT Projekt schaffen, um das Unterprojekt WILLWARN besser einordnen zu können.

Allgemeines

Das PReVENT Consortium besteht aus 52 Partnern. Angefangen bei Automobilherstellern wie z.B. DaimlerChrysler AG, Audi, BMW, Volvo, ... über Forschungsinstitute (Hochschule für Technik und Wirtschaft des Saarlandes, Fraunhofer-Gesellschaft, Laboratoire Central des Ponts et Chaussees, ...) bis hinzu Europäischen Vereinigungen wie z.B. ERTICO oder CLEPA.
PReVENT hat sich unter anderem als Ziel gesetzt, bis 2010 die Zahl der Unfälle zu halbieren und die Zusammenarbeit der europäischen Automobilindustrie zu fördern. Aufgrund der Größe dieses Projekts und auch der hoch gesteckten Ziele ist es in 14 Unterprojekte aufgeteilt. In Abbildung 4.1 werden alle Unterprojekte in ihrer jeweiligen Funktion dargestellt.

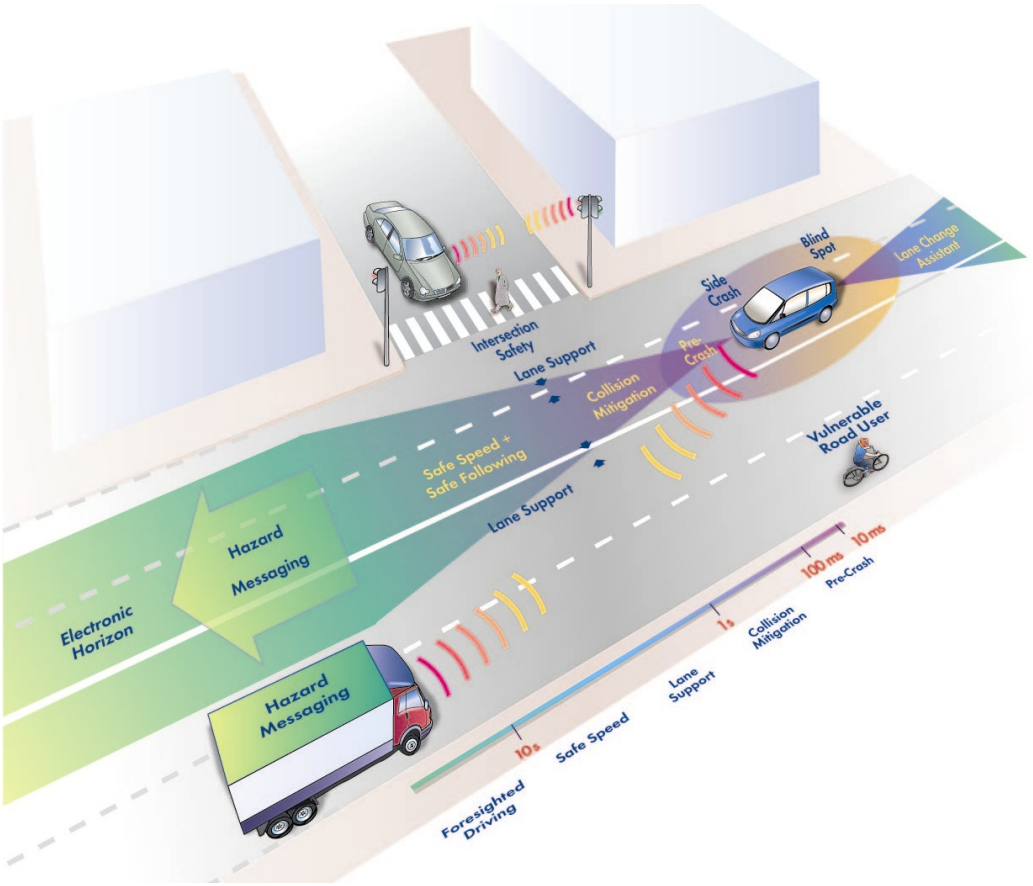


Abbildung 4.1.: Übersicht der PReVENT Unterprojekte

Für einen besseren Einblick werden nun ein paar ausgewählte Subprojekte erläutert.

4.1. SASPENCE

Das Projekt SASPENCE (Save speed and distance) gehört zu der Gruppe “Safe Speed and Safe Following”. Da viele Unfälle durch überhöhte Geschwindigkeiten (vor allem in Kurven) verursacht werden, soll SASPENCE den Fahrer vor möglichen Gefahrensituationen warnen bzw. helfen, diese unfallfrei wieder zu verlassen.
Damit das System das aktuelle Fahrverhalten analysieren kann, müssen Daten von Sensoren erfasst werden. Zur Verfügung stehen Radarsensoren, Car2Car Kommunikation, GPS Position, digitale Straßenkarten und die Überwachung der Fahrzeugumgebung durch Kameras. Dadurch kann erfasst werden, ob sich das Fahrzeug vielleicht zu schnell auf eine

Kurve zu bewegt, Hindernisse auf der Fahrbahn sind, oder der Sicherheitsabstand zum vorausfahrenden Fahrzeug zu gering wird. Sollte eine Gefahr bestehen, wird der Fahrer über eine Anzeige informiert. Sobald das System ein Hindernis detektiert wird eine Gefahrenwarnung über die Car2Car Kommunikation gesendet (mehr dazu im Willwarn Projekt). Es ist auch möglich, den Fahrer über ein in der jeweiligen Situation optimales Verhalten zu instruieren.

Abbildung 4.2.: Warnungsanzeige

4.2. WILLWARN

WILLWARN (Wireless local danger warning) warnt den Fahrer über Gefahren, die vor ihm liegen. Voraussichtiges Fahren und die frühe Erkennung von Gefahren ist die Grundlage von sicherem Fahren und der Unfallvermeidung. Sobald das Fahrzeug in eine Gefahrensituation gerät, erzeugt es automatisch eine Warnmeldung. Durch eine Fahrzeug-Fahrzeug Kommunikation (drahtloses Netzwerk aller Fahrzeuge in einer bestimmten Reichweite) wird diese Warnung auf andere Fahrzeuge übertragen, die wiederum die Nachricht weiterverteilen. Hat ein Fahrzeug eine solche Meldung empfangen und nähert sich deren Ursprung, wird der Fahrer über die bevorstehende Gefahr audiovisuell informiert, sodass dieser sich darauf vorbereiten und entsprechend reagieren kann.

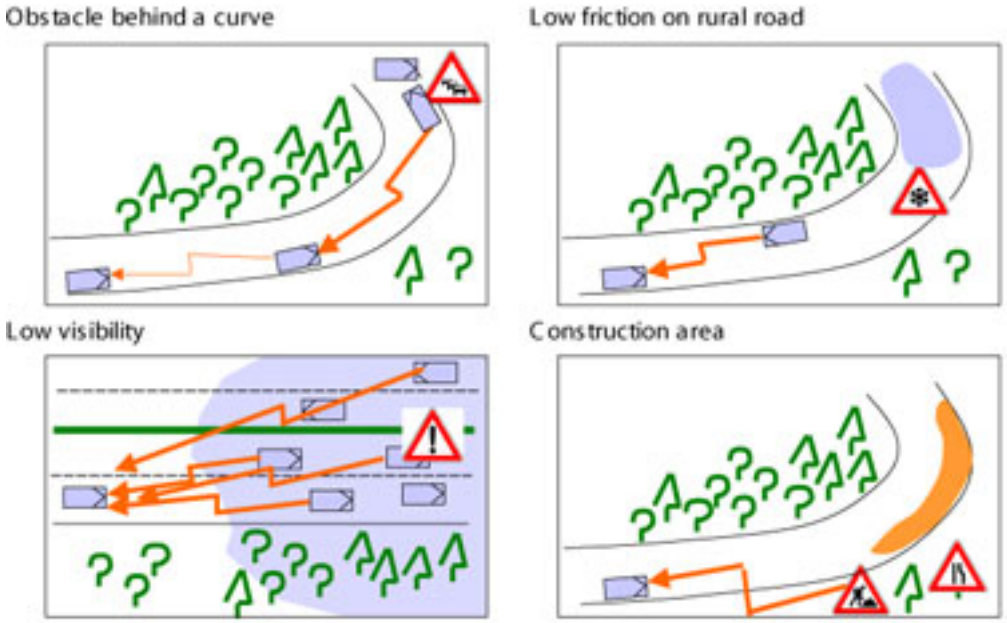


Abbildung 4.3.: Willwarn Anwendungszenario

4.3. LATERAL SAFE

Dieses Projekt bietet Unterstützung im Lane Support (siehe Abbildung 4.1). Mehr als 38% der Unfälle entstehen bei einem Fahrbahnwechsel. Um dieses Risiko zu senken überwacht ein Multi-Sensor System 270 Grad der Fahrzeugumgebung. Dieses System besteht aus Kameras, Long Range Radar und Short Range Radar. Das System erkennt Fahrzeuge, die sich nähern, Hindernisse, Fahrzeuge die sich im toten Winkel befinden und Objekte, die sich vor oder hinter dem Fahrzeug befinden. Die Applikation unterstützt den Fahrer bei einem Spurwechsel und warnt ihn bei Kollisionsgefahren.

Abbildung 4.4.: LateralSafe: die verschiedenen Sensoren am Fahrzeug

4.4. INTERSAFE

Die Gefahr eines Unfalls bei Kreuzungen ist relativ hoch, da neben den Ampelsignalen auch viele unterschiedliche Verkehsteilnehmer beachtet werden müssen. Ziel von INTERSAFE ist es, dem Fahrer zu assistieren, indem die Position des Fahrzeugs durch Erkennung von Straßenmarkierungen genau überwacht wird, sowie durch Kommunikation mit der Ampelanlage und einer Wegvorhersage aller anderen Teilnehmer.

Abbildung 4.5.: Intersafe

5. GPS

Das global positioning system ist ein globales Ortungssystem betrieben vom amerikanischen Pentagon.

5.1. Funktionsweise

In einer Höhe von 20000km befinden sich 24 Navstar-Satelliten, die auf exakt vermessenen Bahnen um die Erde kreisen. Die Satelliten sind so angeordnet, dass mindestens sechs Satelliten auf jeder Position der Erde empfangen werden könnten. Da das ausgestrahlte Signal der Satelliten sehr schwach ist, muss für eine Ortung eine Sichtverbindung bestehen. Durch Berge oder Hochhausschluchten kann das Signal so stark beeinflusst werden, dass keine Ortung mehr möglich ist.

Jeder Satellit sendet kontinuierlich seine Position, seine Kennung und die aktuelle Uhrzeit. Außerdem sendet er noch die Daten über seine Umlaufbahn und die der anderen Satelliten, den sogenannten Almanach. Die Signale sind mithilfe einer Phasenmodulation auf eine Trägerfrequenz von 1575,42 MHz moduliert. [6]

Ein GPS Empfänger bestimmt die Position über die Signallaufzeit und entsprechender Triangulation. Er vergleicht seine aktuelle Uhrzeit mit der vom Satelliten empfangenen Sendezeit. Über diese Differenz weiß der Empfänger auf welcher "Kugel" um den Satelliten er sich befindet. Durch einen weiteren Satelliten schneiden sich die zwei Kugeln, und der Empfänger muss sich auf einem Kreis befinden. Ein dritter Satellit grenzt die Position weiter ein. Es verbleiben nur noch zwei Punkte. Mithilfe eines vierten Satelliten lässt sich die Position auf der Erde genau bestimmen. Für eine genaue Ortung müssen also vier unterschiedliche Satelliten sichtbar sein. [1]

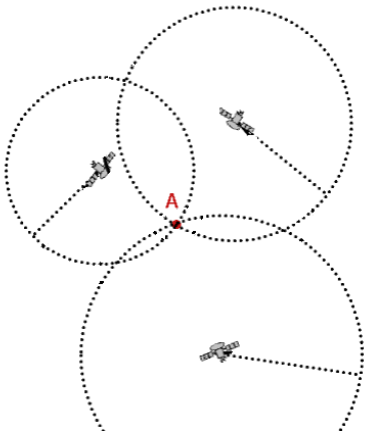


Abbildung 5.1.: Positionsbestimmung im zweidimensionalen Raum

Da die Uhrzeiten bei der Positionsbestimmung sehr wichtig sind, befindet sich in jedem Satellit eine Atomuhr. Der Empfänger passt seine Uhrzeit solange an, bis sich die vier Kugeln in einem Punkt schneiden.

Beim Einschalten eines Empfängers wird versucht, die Signale der einzelnen Satelliten zu empfangen. Aufgrund von verschiedener physikalischer Effekte (Doppler-Effekt) muss jede Frequenz und Phasenlage solange durchprobiert werden, bis der Empfänger synchron zum Signal ist und somit eine Korrelation stattfindet. Über den oben erwähnten Almanach weiß der Empfänger, wo sich die Satelliten zu einem bestimmten Zeitpunkt befinden. So kann im Voraus berechnet werden, welche Phasenlage das Signal hat und die Synchronisation / Positionsbestimmung ist bedeutend schneller.

Falls sich ein Empfänger für längere Zeit nicht bewegt, wird die Position nicht auf einem Punkt bleiben. Durch verschiedene Effekte in den Erdatmosphären werden die Signallaufzeiten beeinflusst und die berechnete Position kann bis zu ca. 20m von der eigentlichen Position abweichen.

Da die Positionsbestimmung bei einer schlechten Satellitenkonstellation (z.B. alle Satelliten sind in einer Linie) nur schlecht oder gar nicht möglich ist, wird ein Maß dafür eingeführt. Der sog. HDOP/VDOP (Horizontal/Vertical Dillution of Precision) Wert gibt an, wie sehr man sich auf die berechnete Position verlassen kann. Je kleiner der HDOP/VDOP WertHDOP Werte zwischen 1 und 8 sind normal. Bei größeren Werten, sind die Positionskoordinaten nicht verwendbar.

Teil II.

Bericht über das IPS

Für jedes von mir bearbeitete Projekt hatte ich einen Betreuer, den ich bezüglich technischer Fragen stets konsultieren konnte. Diese/Dieser wird zu Beginn der Projektbeschreibung genannt. Die Betreuung (Fragen zum Ablauf, ...) über das gesamte Praktikum hinweg hatte Frau Dagmar Hermann. Alle Projekte habe ich alleine bearbeitet.

6. Vorstellung der Firma

6.1. DaimlerChrysler AG

Gottlieb Daimler mit seinem Mitarbeiter Wilhelm Maybach und Carl Benz haben zeitgleich im Jahr 1886 das Automobil erfunden. In den 20er Jahren fusionierten die Benz&Cie. und die Daimler-Motoren-Gesellschaft zur Daimler-Benz AG mit Sitz in Berlin. Die DaimlerChrysler AG entstand dann 1998 aus der Fusion der Daimler-Benz AG und der Chrysler Corporation mit dem Hauptsitz in Stuttgart und Auburn Hills. DaimlerChrysler ist ein führender Anbieter von Pkw, Geländewagen, Sportwagen, Minivans und Pick-ups sowie der weltweit größte Hersteller von Nutzfahrzeugen. Seit 2005 ist Dr. Dieter Zetsche der Vorsitzender des Vorstandes und Leiter der Mercedes Car Group. In Abbildung 6.1 ist ein Ausschnitt der Firmenstruktur zu sehen, die den Ast zu der Abteilung zeigt, in der ich beschäftigt war.

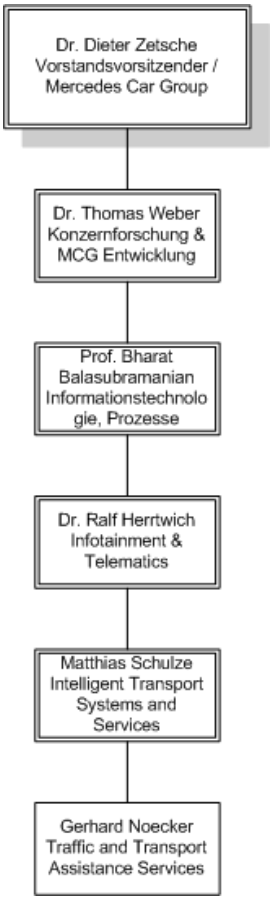


Abbildung 6.1.: Ausschnitt des Unternehmenorganigrammes

Das Unternehmen beschäftigte gegen Ende 2005 insgesamt 382 724 Mitarbeiter in 17 Ländern. Der Umsatz betrug 2005 149,8 Mrd Euro bei rund 4 Mio abgesetzten Personenwagen und rund 820 000 Nutzfahrzeugen.

Außer der Herstellung von Personenwagen und Nutzfahrzeugen stellt die DC AG auch Finanzdienstleistungen bereit. Die DaimlerChrysler AG ist aufgeteilt in 5 Gruppen:

- Mercedes Car Group (Mercedes, Smart und Maybach)
- Crysler Group (Chrysler, Jeep und Dodge)
- Truck Group (Mercedes-Benz, Freightliner, Mitsubishi Fuso, Sterling,...)
- Van, Bus und andere
- DaimlerChrysler Financial Services

6.2. Abteilung GR/ETI

Die GR/ETI Abteilung (Intelligent Transport Systems and Services) gehört zur Forschung und (Vor-) Entwicklungsabteilung mit Sitz in Böblingen/Hulb. Die Abteilung ist vor allem im Gebiet der Navigation tätig. Unter Anderem gehört dazu das Verbessern der aktuellen Navigationssysteme in den Fahrzeugen und auch die Entwicklung neuer Funktionen.

7. Projekt: TMC-Analyse

Das erste bearbeitete Projekt wurde von Dagmar Hermann betreut und begann am 1.September 2006 bis Ende Oktober 2006. Aufgrund des schon in der Literaturarbeit angesprochenen Nachteils der TMC Meldungen (Zuverlässlichkeit), soll ein Glaubwürdigkeitswert und eine Tendenz berechnet werden. Der Glaubwürdigkeitswert könnte später dem Autofahrer anzeigen, wie sehr er sich auf die gemeldete Verkehrsstörung verlassen kann. Aufgrund der Art und Anzahl der Wiederholungen einer TMC Nachricht (siehe Ausstrahlen der Nachrichten) könnte dieser Wert berechnet werden. Die Tendenz soll nur bei einer Staumeldung angezeigt werden.

7.1. Vorstellung des Projekts

Aufgabenstellung

Bewertung von RDS/TMC gemeldeten Störungen auf Basis ihrer Historie

- Implementierung eines Tools zur
  Decodierung und Analyse von archivierten RDS/TMC Meldungen (in Java)
- Statistische Analyse von archivierten Meldungen bezüglich ihres Aktualisierungsgrades
- Implementierung einer Methode zur Berechnung eines Glaubwürdigkeitswertes (in Java)
- Implementierung einer Methode zur Berechnung eines Tendenzwertes (in Java)
- Entwurf und Implementierung eines Demonstrators zur Veranschaulichung des Mehrwertes des beschriebenen Features (in Java)

Die Arbeitsumgebung bestand aus einem Standardarbeitsplatz (Rechner mit Pentium 4, 2,4GHz, 1GB RAM, zwei 17"TFT Bildschirmen) und einem Versuchsträger Mercedes E-Klasse mit eingebautem Windows XP Rechner und einem TFT-Display, das an der HeadUnit angebracht ist. Der Rechner wird mit einer Tastatur mit eingebautem Trackball bedient.

7.2. Ausführung

Aus den oben genannten Anforderungen wurde folgendes UseCase Diagram entwickelt:

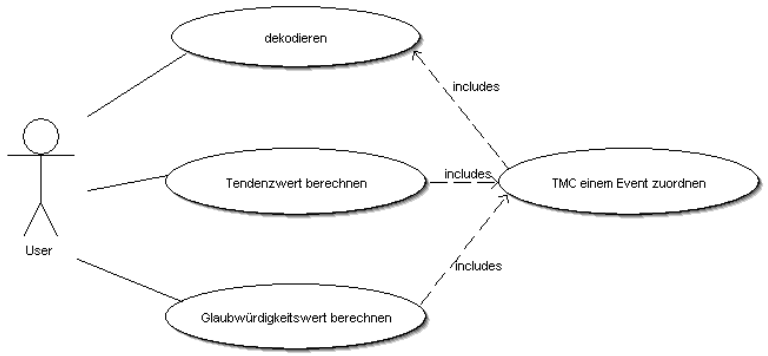


Abbildung 7.1.: UseCase Diagram

Für die Analyse der TMC Meldungen steht ein Archiv aller Meldungen aus dem Jahr 2005 vom WDR zur Verfügung. Das Archiv besteht aus mehreren Unterordnern, die den jeweiligen Monat und Tag angeben. Die Meldungen liegen in Form von Textdateien vor, deren Name den Zeitpunkt der Ausstrahlung angibt. Jede TMC Nachricht besteht aus zehn hexadezimalen Zahlen. Beispiel: Datei 011050.tmc im Ordner 2 im Ordner 5 enthält 087065276C. Dies bedeutet dass am 2.Mai 2005 um 1:10:50 Uhr eine TMC Meldung ausgestrahlt wurde. Um mit dem zur Verfügung stehenden Archiv effizient arbeiten zu können, habe ich zunächst ein Datenbankschema erstellt, in dem der Dateiname und die Inhalte durch ein Platzhalter getrennt, gespeichert ist. So kann mithilfe einer JDBC Anbindung einfach und schnell über die Nachrichten iteriert werden. Hierbei half mir das Skript aus der Datenbanken 1 Vorlesung.
Der Decoder wurde in Java implementiert unter der Verwendung der Eclipse 3.2 Entwicklungsumgebung. Dies stellte kein Problem dar, da ich durch die Belegungen der

Vorlesungen Softwareentwicklung 1-3 und Software Engineering in meinem bisherigen Studium gut darauf vorbereitet wurde.
Die Dekodierung einer TMC Meldung wurde bereits in der Literaturarbeit ausführlich erklärt.

Architektur

Für die Berechnung der oben genannten Ziele (siehe Aufgabenstellung) wurde eine Architektur entworfen.

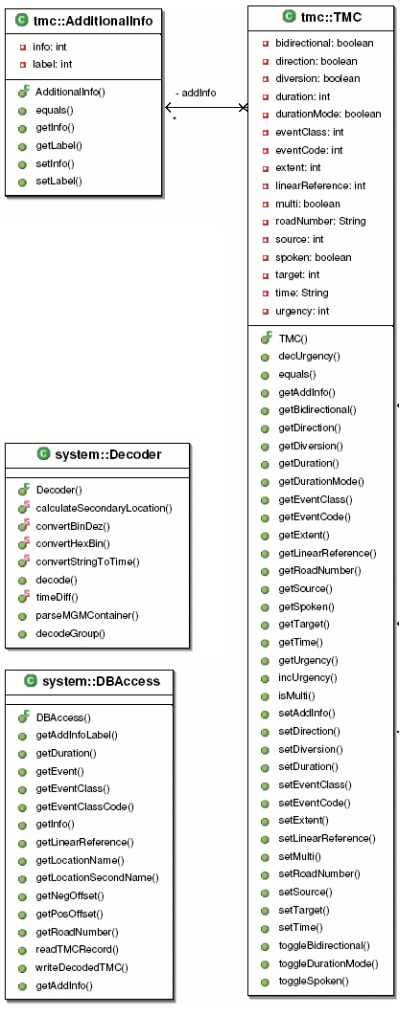


Abbildung 7.2.: Class Diagram

Die TMC Klasse dient als Datencontainer und enthält alle Attribute einer TMC Nach-

richt sowie deren Getter und Setter Methoden. Die Attribute sind nicht als Strings gespeichert, sondern entsprechen der dezimalen Darstellung der Werte. Die AdditionalInfo Klasse bildet eine Zusatzinformation, die in einer MultiGroupMessage (siehe Dekodierung) übertragen werden können, ab. Die Decoder Klasse enthält die Methode decode, die einen hexadezimalen String als Parameter bekommt und ein TMC Objekt zurückgibt. Um eine TMC Nachricht, die nur die dezimalen Werte einer Nachricht enthält, in Text umwandeln zu können, stellt die DBAccess Klasse die benötigten Funktionen bereit. Diese Klasse dient als Abstraktion der Datenbank.

Problem der Zuordnung: TMC-Event

Für die Berechnung eines Glaubwürdigkeitswertes (oder auch Verlässlichkeitswertes) und einer Tendenz eines Verkehrsereignisses dürfen nicht die einzelnen TMC-Nachrichten sondern so genannte “Events” betrachtet werden. Das heißt, die stets wiederholten Nachrichten (siehe Ausstrahlen der Nachrichten) müssen zu einer “Nachricht” (Event) zusammengefasst werden. Dies stellt ein Problem dar, da ein Verkehrsereignis keinen eindeutigen Identifizierer hat. Es muss nun ein Algorithmus entwickelt werden, der aufgrund der verschiedenen Attribute einer TMC Nachricht einen Event erstellt und weitere TMC Meldungen dem richtigen Event zuordnet. Dabei müssen folgende Fälle betrachtet werden:

Fall 1:

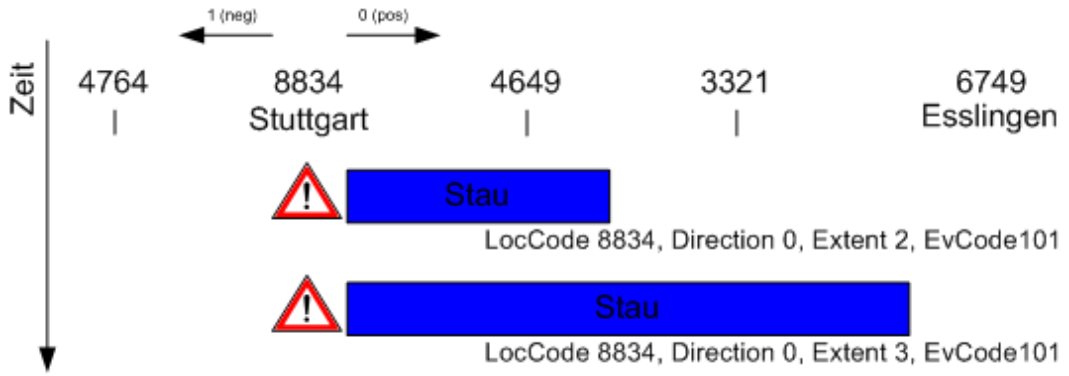


Abbildung 7.3.: Fall 1

Wie man in Abbildung 7.3 sieht, kann es vorkommen, dass sich ein Stau vergrößert. Dabei erhöht sich der Extent, während alle anderen Attribute gleich bleiben.
Es können also alle TMC Meldungen, bei denen LocationCode, Direction, EventCode gleich sind und der Extent in einem bestimmten Intervall (+/- 10) liegt, zu einem Event zusammengefasst werden.

Fall 2:

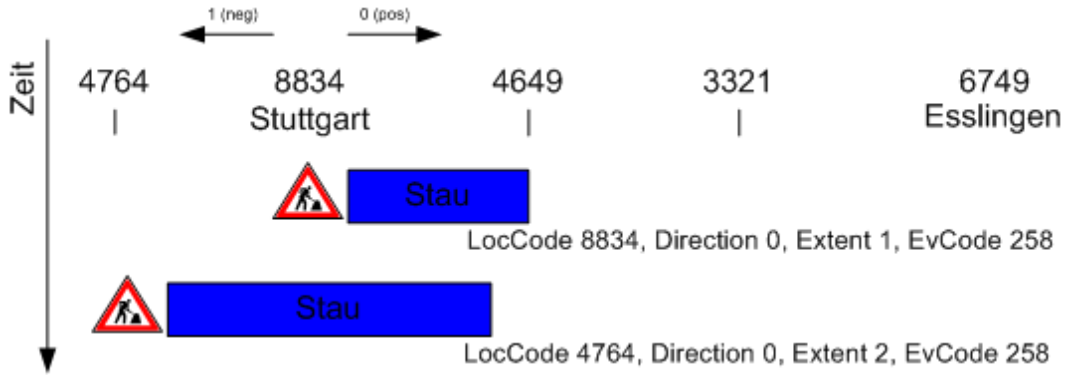


Abbildung 7.4.: Fall 2

In Abbildung 7.4 sieht man, wie sich z.B. ein Schwertransport oder eine Wanderbaustelle verschiebt. Das heißt es ändert sich LocationCode und Extent während alle anderen Attribute gleich bleiben. Zusätzlich muss überprüft werden, ob die neue TMC Meldung mindestens einen gemeinsamen LocationCode mit dem bereits bestehenden Event hat. Ist dies der Fall, so kann die TMC diesem Event zugeordnet werden.

Fall 3:

Eine TMC Meldung wird einfach wiederholt. Das heißt, die gesamte Nachricht entspricht einer einem Event bereits zugeordneten Nachricht, so kann diese dem Event zugeordnet werden.

Fall 4:

Eine TMC Meldung hat den gleichen LocationCode, Direction, Extent, EventCode aber unterschiedliche Zusatzinformationen (bei einer MultiGroupMessage), so kann die Nach-

richt auch einem entsprechend bestehenden Event zugeordnet werden.

Fall 5:

Wenn die Verkehrsstörung explizit aufgehoben wird (LocationCode, Direction, Extent sind gleich aber ein spezieller EventCode wird verwendet), so kann der Event gelöscht werden.

Fall 6:

Ist keiner der vorhergehenden Fälle eingetreten, so kann ein neuer Event erstellt werden. Die Attribute des Events werden entsprechend der TMC Meldung gefüllt.

Für die spätere Analyse enthält der Event alle zugehörigen TMC Nachrichten - auch die wiederholt gesendeten.

Diesem Algorithmus entsprechend wurde die Architektur angepasst.

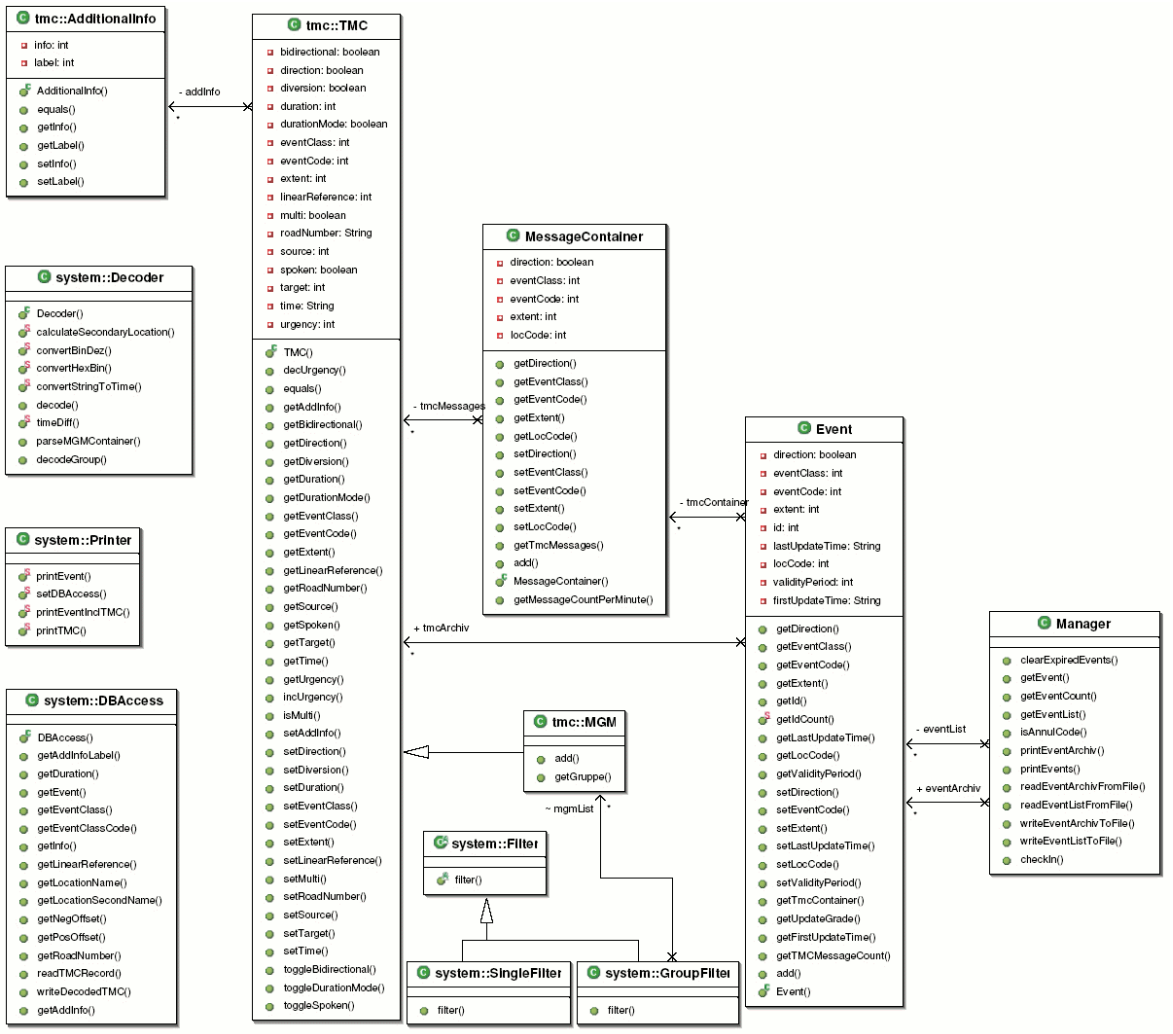


Abbildung 7.5.: Class Diagram

Ein TMC Objekt ist nun in einem MessageContainer gespeichert. Dies vereinfacht die spätere Verwendung bei der statistischen Analyse. Alle Nachrichten in einem MessageContainer sind genau identisch. Sobald eine TMC Nachricht empfangen und einem Event zugeordnet wird, muss intern eine weitere Zuordnung zu dem korrekten Messagecontainer stattfinden. Die Klasse Event bildet nun einen in Abschnitt 7.2 beschriebenes Verkehrsereignis ab. Die Klasse Manager verwaltet die einzelnen Events. Die Printer, MGM, Filter, ... Klassen helfen bei der Decodierung und Darstellung einer Nachricht.

Sobald eine veränderte TMC Meldung einem Event zugeordnet wird, müssen die Eventattribute entsprechend aktualisiert werden. Dies ist jedoch abhängig davon, wie sich

die TMC Meldung geändert hat. Es muss nun ein Algorithmus entworfen werden, der die Eventposition (LocationCode, Extent) und auch unter anderem die Staulänge entsprechend aktualisiert. Der Algorithmus wurde aufgrund folgender Fallunterscheidung implementiert:

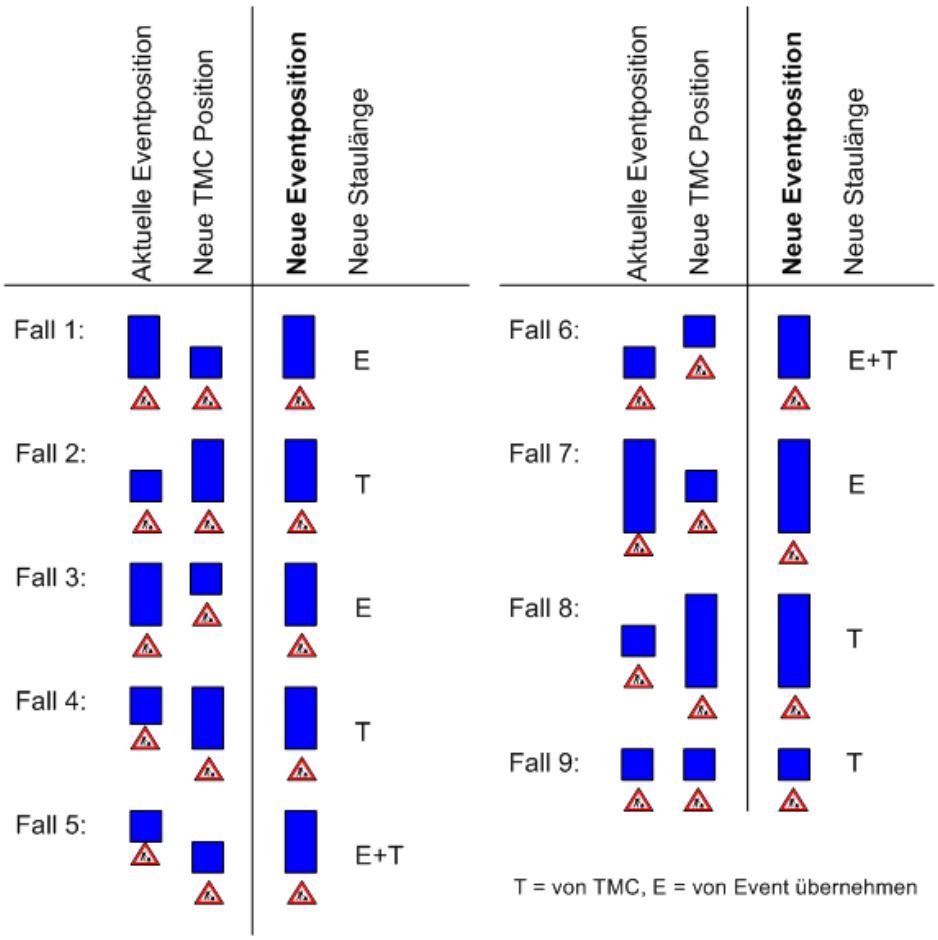


Abbildung 7.6.: Fallunterscheidung

Statistische Analyse

Aufgrund der nun implementierten Software kann eine statistische Analyse angefertigt werden. Dabei wurde eine repräsentative Auswahl an Tagen aus dem Archiv gewählt, da eine Analyse mit allen Nachrichten zu viel Zeit in Anspruch genommen hätte.

Für die Analyse der Events wurde ein spezieller Diagrammtyp entwickelt.

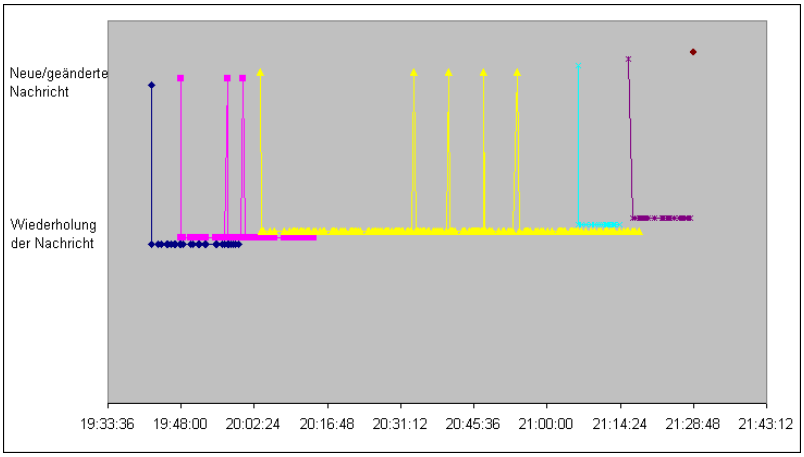


Abbildung 7.7.: Diagrammtyp für Events

Eine TMC Meldung ist in dem Diagramm als Punkt dargestellt. Jede Kurve entspricht einem MessageContainer. Eine neue oder geänderte TMC Meldung entspricht einem Punkt in der oberen Hälfte des Diagramms. Wird eine Meldung lediglich wiederholt, so wird der Punkt in der unteren Hälfte eingetragen.
Mithilfe dieses Diagramms kann später auch schön der Verlässlichkeitswert und die Tendenz eines Events veranschaulicht werden.

Auf die Analyse werde ich nun nicht näher eingehen. Diese ist ausführlich dokumentiert in Kapitel 6 der Dokumentation des Projekts [9].

Verlässlichkeitswert

Der Verlässlichkeitswert wurde als Ganzzahl auf einer Skala von 100 - 0 definiert. Für die spätere Darstellung könnte eine Unterteilung in: hoch (100 - 80), mittel (79 - 60) und gering (59 - 0) stattfinden. Der Verlässlichkeitswert soll zum einen für archivierte TMC Ereignisse in Form einer Kurve dargestellt werden können und zum andern für aktuelle Ereignisse berechnet und im Auto dargestellt werden. Für eine aktuelle Darstellung muss dieser Wert alle 30 sec neu berechnet werden.

Der Verlässlichkeitswert kann nicht für alle Ereignisse gleich berechnet werden. Bei einer Baustelle fällt der Wert wesentlich langsamer als bei einem Stau. Da dieser mit ziemlicher Wahrscheinlichkeit schneller wieder verschwindet als die Baustelle. Bei einer

Gefahrenwarnung (z.B. Personen auf der Fahrbahn) muss der Wert wesentlich schneller abfallen als bei den anderen Ereignissen, da sich Menschen sicherlich nicht sehr lange auf der Fahrbahn aufhalten werden. Es wurden deshalb vier Kategorien gebildet: Stau, Gefahr, Baustelle und Standard (restliche). Bei der Berechnung wird eine Methode namens "getAverageUpdateCount()" verwendet. Dabei wird die Gesamtdauer des Events in 10 Minuten Intervalle unterteilt. Pro Intervall wird die Anzahl der empfangenen Meldungen über MessageContainer hinweg summiert. Über diese Intervalle wird dann ein Mittelwert gebildet und zurückgegeben.

Stau

Der Startwert der Verlässlichkeit beträgt 100. Bei einer Wiederholung der Nachricht wird der Verlässlichkeitswert wie folgt berechnet:

$$reliability' = reliability - 4 \cdot \frac{1}{getAverageUpdateCount()} \tag{7.1}$$

Sobald eine Nachricht geändert wird, beträgt die Verlässlichkeit wieder 100. Sollte für länger als 3 Minuten keine Meldung für den Event empfangen werden, kann davon ausgegangen werden, dass der Event nicht mehr existiert. Die Verlässlichkeit wird dann wie folgt berechnet:

$$reliability' = reliability - e^{\frac{duration[min]}{10}+1} \tag{7.2}$$

Baustelle

$$reliability' = reliability - 2 \cdot \frac{1}{getAverageUpdateCount()} \tag{7.3}$$

Da eine Baustelle nie einfach verschwinden wird, darf der Verlässlichkeitswert nicht unter 70 fallen. Fünf Minuten bevor der Event gelöscht werden würde (da keine Wiederholung empfangen wird), fällt der Wert nach der Formel 7.2 ab.

Gefahr

$$reliability' = reliability - 5 \cdot \frac{1}{getAverageUpdateCount()} \tag{7.4}$$

Standard

Bei den restlichen Ereignissen lohnt sich eine genauere Betrachtung nicht. Sie haben ständig 100%ige Verlässlichkeit.

Beispiel:

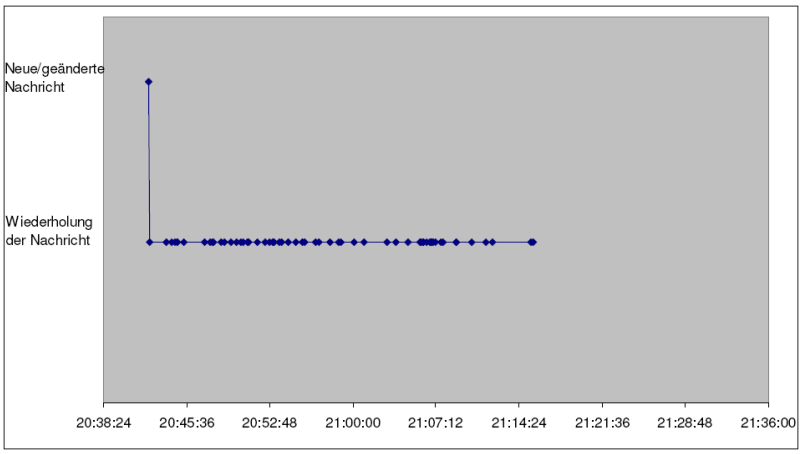


Abbildung 7.8.: Darstellung eines Events

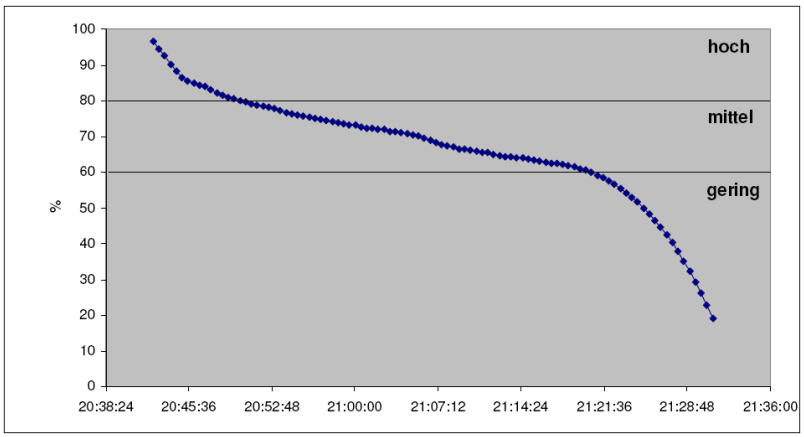


Abbildung 7.9.: Darstellung des Verlässlichkeitswertes

Die Architektur muss nun entsprechend angepasst werden. Die Klasse Event ist nun abstrakt. Die Klasse Stau und Gefahr erben von dieser Klasse. Baustelle erbt von Stau, da bei diesen beiden Klassen ein Stau entstehen kann und somit ein Tendenzwert berechnet werden muss. Die Methoden getReliabilty() und getTendency() werden von den Subklassen entsprechend implementiert.

Tendenzwert

Der Tendenzwert wird für Events mit einer Stauangabe berechnet. Aufgrund der Attributanpassung (siehe Abbildung 7.6) ist die Berechnung der Tendenz relativ einfach zu berechnen. Der Tendenzwert ist wie folgt definiert: gleich bleibend (orange) = 0, steigend (rot) = 1, absteigend (grün) = -1. Bei der Erzeugung eines Events mit einer Staulängenangabe beträgt die Tendenz 0. Sobald die Staulänge verändert wird, verändert sich der Tendenzwert entsprechend des Ergebnisses des Vergleichs zwischen der bisherigen und der neuen Staulänge.

Beispiel:

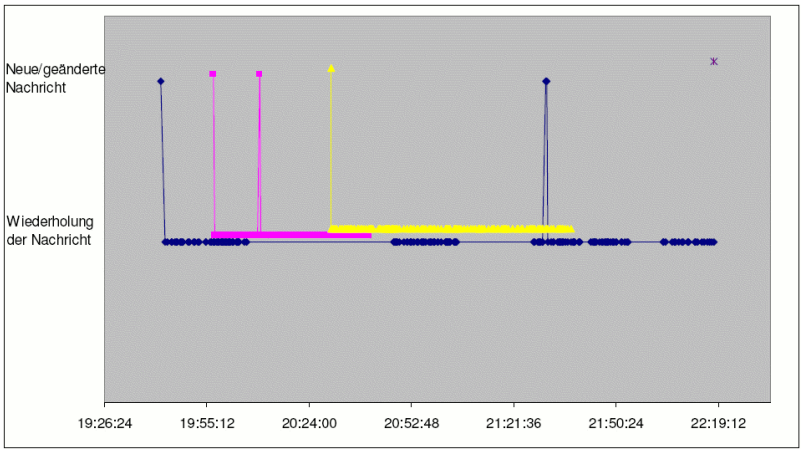


Abbildung 7.10.: Darstellung eines Events

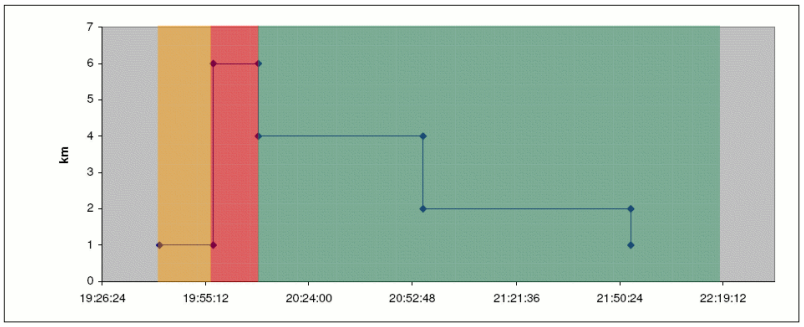


Abbildung 7.11.: Darstellung der Tendenz

Demonstrator

Es soll nun ein Demonstrator entworfen und implementiert werden, der den Mehrwert der zuvor beschriebenen Funktionen veranschaulicht. Die alte Architektur (Decoder, TMC, Event-Klassen) wurde hierbei verwendet und entsprechend erweitert. Dabei ist zu beachten, dass die Architektur ein anderweitig entwickeltes Softwaremodul einbindet, um dessen Funktion zu verwenden.

Architektur

Die neue Architektur sieht wie folgt aus:

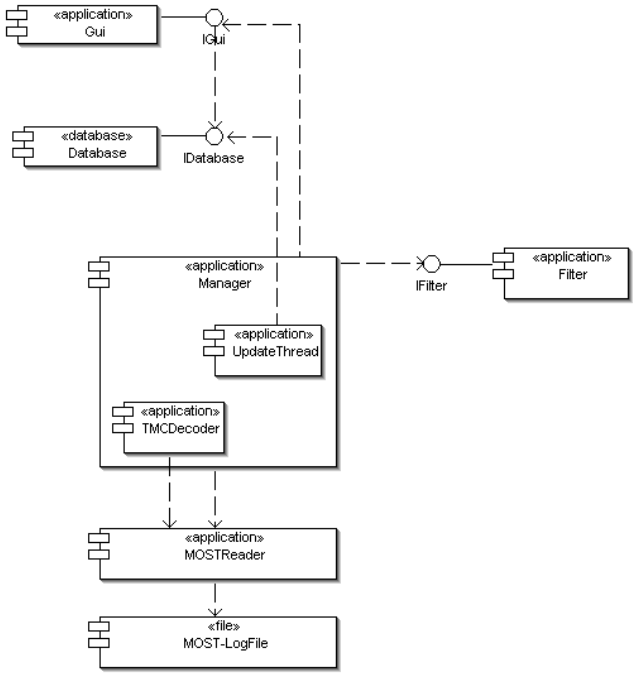


Abbildung 7.12.: Component Diagram

Aufgrund der Definition der Interfaces ist die Applikation sehr modular gehalten. Es kann z.B. relativ einfach eine alternative GUI entwickelt werden. Die Datenbank muss nach wie vor erhalten bleiben, da diese von einem externen Softwaremodul benötigt wird. Der Zugriff auf die empfangenen TMC Meldungen funktioniert über ein MOST-LogFile. Diese Datei wird von einem Programm geschrieben, das auf den MOST-Bus des

Versuchsträgers zugreift. Der MOST-Bus (Media Oriented Systems Transport) ist ein optisches Token-Ring Bussystem, das alle Telematikkomponenten (Bsp: Verstärker, Radio, Navigation, Command, Multi-Seat-Entertainment-Monitore, ...) eines Fahrzeugs miteinander vernetzt. Entsprechend dieser Ausgabe muss der MOSTReader angepasst werden, der dann die TMC Meldungen als String in den Decoder übergibt. Der Manager schreibt die erstellten Events in ein Datenbankschema. Da durch die Vorlesung Datenbanken und Anwendungen bereits einiges an Vorwissen über Hibernate vorhanden war wurde dieses Tool, zum Speichern der Events verwendet. (Hibernate ist ein Tool, das einfache Java Objekte in einer relationalen Datenbank persistiert.) Sobald sich an der Datenbank etwas geändert hat, wird die GUI benachrichtigt und entsprechend aktualisiert. Der UpdateThread berechnet alle 30 sec die Verlässlichkeitswerte der Events neu und aktualisiert entsprechend die Datenbank.
Ein komplett überarbeitetes Klassendiagramm der neuen Architektur inklusive der neuen GUI befindet sich im Anhang unter Abbildung Anhang Zusätzlich befindet sich im Anhang unter Abbildung Anhang ein ERM Diagramm, welches alle Datenbankrelationen enthält.

GUI

Die GUI soll so aussehen wie das aktuelle Navigationssystem der Mercedes S-Klasse. Da bei diesem Navigationssystem der Fahrer manchmal nicht weiß, wo er sich gerade befindet, wurde eine “Brot-Krumen-Navigation” hinzugefügt, die in jeder Ansicht angibt, wo man sich gerade im Navigationsbaum befindet. Auf Fenster, die übereinander dargestellt werden, wurde bewusst verzichtet, da dies nur unnötig verwirrend ist.
Der Navigationsbaum ist wie folgt aufgebaut:
Nach dem Start und der entsprechenden Menüauswahl wird eine alphabetisch sortierte Liste der Straßennamen aller empfangenen Verkehrsstörungen angezeigt. Diese Liste ist scrollbar und wird dynamisch erweitert. Das heißt, dass stets das aktuelle Element markiert bleibt und vor oder nach diesem die Straßennamen neue empfangener Meldungen eingefügt werden. Nach der Auswahl einer Straße sieht man alle Meldungen auf dieser Straße mit Angabe des Ortes und des Ereignisses. Die Detailansicht enthält zur Demonstration alle möglichen Angaben über die Verkehrsstörung. Der Verlässlichkeitswert wird dabei mittels eines JSliders dargestellt. Um die Verlässlichkeit auf einen Blick ablesen

zu können, wurde als Hintergrund ein Farbverlauf gewählt, der den aktuellen Wert für das Auge schnell erfassbar macht. Sofern eine Staulängenangabe vorhanden ist, wird die Tendenz als farbiges Wort nach der Staulänge angegeben. Die GUI wird ausschließlich mit den Pfeiltasten der Tastatur bedient. Im Anhang unter Abbildung 11.9 befindet sich ein State-Machine-Diagram für die GUI.
Bei der Implementierung der Liste gab es große Probleme. Da auf eine vorgefertigte JList nicht zurückgegriffen werden konnte (das Design konnte nicht angepasst werden), musste die Liste "von Hand" erstellt werden. Jeder Eintrag entspricht einem JPanel. Das Problem bestand bei der Implementierung des Scrollings der Liste. Es war die Funktion gefordert, dass das aktuelle Element stets markiert bleibt und neue Listenelemente entsprechend der Sortierung einfach eingefügt werden. Durch das geschickte Unterteilen ein kleinere Probleme gelang es dann doch noch, das Scrolling fehlerfrei zu programmieren.

Einige Screenshots der Applikation sind im Anhang gelistet (siehe **??**).

Vergleiche

Zum Abschluss des Projekts sollten die Anzeigen von Verkehrsstörungen von den Navigationssytemen bei Audi und BMW verglichen werden. Da es relativ schwierig ist, im Internet solche Darstellungen zu bekommen ging ich zu den jeweiligen Autohäusern und fotografierte die TMC Meldungsanzeige ab. Auffallend ist, dass BMW noch eine Angabe in Minuten darstellt, die anzeigt, wie viel Zeit die Störung zusätzlich zur normalen Fahrzeit beanspruchen würde. Diese relativ einfache Berechnung wurde in die Applikation noch mit aufgenommen.

7.3. Ergebnis

Das Ergebnis der Arbeit ist zum einen eine Dokumentation der statistischen Analyse, sowie die Applikation, die den Verlässlichkeits- und Tendenzwert zu aktuellen Verkehrsstörungen anzeigt. Dieses Programm wurde im Versuchsträger installiert und getestet. Sollte nun eine längere Autobahnfahrt bevorstehen, kann die Software gestartet und somit die Zusatzfunktionen verwendet werden.

7.4. Fazit

Die Aufgabe dieses Projekts bestand darin, zu der normalen Darstellung der Verkehrsstörungen zwei Zusatzinformationen zu berechnen.
Die Anzeige einer Tendenz, ist eine Funktion, die einen Mehrwert bereitstellt. Indirekt wird dem Fahrer vermittelt, ob das Ereignis eher ernst zu nehmend ist oder vernachlässigt werden kann. Bei den statistischen Untersuchungen habe ich jedoch festgestellt, dass die Staulänge des öfteren nicht verändert wird. Außerdem kann es auch vorkommen, dass die Staulänge "hoch und runter springt". Dabei liegt die Tendenz natürlich oft daneben.
Der Verlässlichkeitswert stellt, meiner Meinung nach, keinen Mehrwert dar. Die Verlässlichkeit wird nicht aufgrund realer Ereignisse berechnet, sondern basiert auf den TMC Meldungen. Diese versuchen annähernd das reale Verkehrsgeschehen abzubilden, was ihnen nicht allzu häufig gelingt. Somit steht bereits die Berechnung auf "wackeligen Beinen".
Das Projekt an sich hat mir viel Spaß gemacht. Da ich das Programm von Grund auf neu implementieren durfte, musste ich mich an nichts binden und konnte völlig frei eine eigene Architektur entwickeln. Durch die Vorlesungen Softwareentwicklung, Software Engineering, Datenbanken und Anwendungen und Datenbanken 1 konnte ich mein bisher erlerntes Wissen in der Praxis einsetzen.

8. Projekt: WILLWARN

Die Bearbeitung des Willwarn Projekts begann Anfang November 2006 bis Anfang Februar 2007. Es wurde von Andreas Hiller betreut.
Mit der Entwicklung des Willwarn Projekts wurde bereits im Februar 2004 begonnen und Ende Januar 2007 fertiggestellt. Die beteiligten Gruppen sind: DaimlerChrysler, BMW Forschung und Technik, CNRS (Frankreich), TNO (Niederlande) und die Hochschule für Technik und Wirtschaft Saarbrücken. Für die Tests stand ein Mercedes E-Klasse und ein Smart fortwo mit eingebautem CarPC und NoW Router zur Verfügung. (Bilder im Anhang unter Anhang und Anhang)

8.1. Vorstellung des Projekts

Die genaue Aufgabe und die Einordnung dieses Projekts wurde bereits im Literaturstudium unter 4 bzw. 4.2 behandelt. Ich werde nun genauer auf den Aufbau der einzelnen Module dieses Softwareprojekts eingehen.

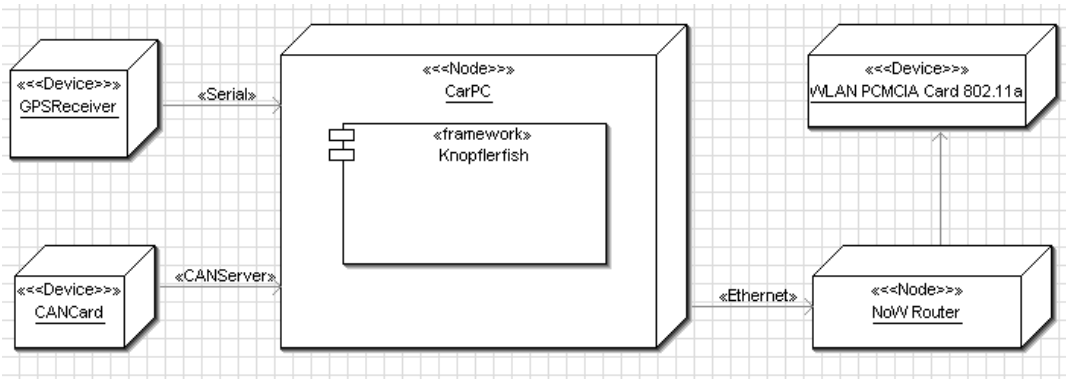


Abbildung 8.1.: Deployment Diagram

GPS

Der GPS-Empfänger bestimmt die aktuelle Position des Fahrzeugs und übermittelt sie über die serielle Schnittstelle an den CarPC.

CAN-Bus

Der CAN-Bus (Controller Area Network) ist ein asynchrones serielles Bussystem. Dieses wurde 1983 von Bosch entwickelt und dient zur Vernetzung von Steuergeräten in Kraftfahrzeugen. Dadurch kann der Kabelbaum beträchtlich reduziert werden und somit Gewicht und Kosten gespart werden. Über den CAN Bus kann die Software die Fahrzeugsensoren wie z.B. Querbeschleungigung, Licht, Motordrehzahl, Raddrehzahl, ABS-, ESP-, Blinker- und Scheibenwischeraktivitäten, ausgelesen werden.

CarPC

Jedes Fahrzeug hat einen fest eingebauten Computer mit dem Betriebssystem Windows 2000 oder Windows XP.

NoW-Router

Der NoW Router ist ein Notebook mit Fedora Core 4, ausgestattet mit einer Ubiquiti PCMCIA 802.11a/b/g Karte. Diese Karte hat eine hohe Sendeleistung (300mW) und es können zwei externe Antennen angeschlossen werden. Die Antennen sind zur besseren Kommunikation auf dem Dach des Fahrzeugs montiert. Der NoW Router ist mit einer gewöhnlichen Ethernet Verbindung an den CarPC angeschlossen.

NoW bedeutet Network on wheels und ist ein Projekt, das 2004 von den Firmen DaimlerChrysler, BMW, VW, Fraunhofer Institut, NEC und Siemens ins Leben gerufen wurde. Zum einen wurde ein verbesserter Treiber für die WLAN-Karte entwickelt, der schneller AdHoc-Verbindungen aufbauen kann, zum anderen wurde ein komplett neues Netzwerkprotokoll entworfen. Dieses Protokoll erlaubt die Kommunikation zwischen sich bewegenden Fahrzeugen. Dabei wurden die Probleme der extrem hohen Dynamik dieses Netzwerks wie z.B. Routing in der Stadt, bei sehr vielen Fahrzeugen (Überlastungsvermeidung des Kanals), Kommunikation auf dem Lande mit sehr wenigen Fahrzeugen, bedacht und gelöst.

Knopflerfish framework

Die gesamte Willwarn Software wurde in verschiedene Module aufgeteilt, die dann jeweils ein Partner zu implementieren hat. Die Kommunikation der Module untereinander kann wahlweise mit einem OSGi Framework oder über UDP Sockets stattfinden.

- GPS-Modul dient zur Auslesung der NMEA Strings der GPS Maus und Umwandlung in eine interne Darstellung der Position (AbsolutPosition). (Jeder Partner)
- HazardDetectionModule (HDM) dient zur Überwachung der Fahrzeugsensoren und implementiert Algorithmen zur Erkennung von Gefahren. (Jeder Partner)
- WarningMessageManagement (WMM) dient zur Verwaltung der Nachrichten und bildet das Hauptmodul. (Entwickelt von DaimlerChrysler)
- PositionRelevanceCheckModule (PRCM) dient zur Erkennung, ob eine Warnung für die aktuelle Position (Fahrtrichtung) relevant ist. (Entwickelt von TNO)
- Vehicle-to-VehicleCommunication (VVC) dient als Schnittstelle zwischen WMM und dem NoW Router. (Entwickelt von BMW)
- HazardWarningModule (HWM) dient zur Darstellung möglicher Gefahren für den Fahrer. (Jeder Partner)
- DemoUI dient zur Demonstration der gesamten Applikation. Sie zeigt sozusagen das Gedächtnis der Anwendung mit der aktuellen Fahrzeugposition auf einer Karte. Ebenfalls die sichtbaren Nachbarfahrzeuge, sowie alle Warnungen werden darauf angezeigt. Zur Demonstration können auch Warnungen durch betätigen eines Buttons manuelle erzeugt werden.

Über die UDP Schnittstellen ist die Software sehr modular gehalten. Es ist sogar möglich, einzelne Module auf andere Computer, mit anderen Betriebssytemen, auszulagern. Außerdem ist so eine Implementierung mit unterschiedlichen Programmiersprachen möglich.

Die meisten Module sind in JAVA implementiert. Wobei das HDM als C++ Applikation über die UDP Schnittstelle angesprochen wird.

Weg einer Warnung

Für die Verdeutlichung der Zusammenarbeit der Module wird nun die Entstehung, Übertragung und Anzeige einer Warnung erklärt.

Das GPS Modul liest sekündlich die Position aus und sendet diese an alle Module. Das PRCM speichert die GPS Positionen für die Erstellung eines Traces. Ein Trace ist eine Streckenmarkierung auf der die Warnung als relevant erkannt wird.

Durch (z.B.) eine Vollbremsung erkennt das HDM eine starke Verögerung. Dadurch wird ein Treshold überschritten und das HDM fordert einen Trace vom PRCM. Die Nachricht wird generiert und an das WMM gesendet.

Das WMM speichert die empfangene Nachricht in einer Datenbank und sendet sie an die DemoUI zur Darstellung. Das WMM durchläuft in einem bestimmten Intervall die Datenbank und sendet relevante Warnungen an das VVC. Dieses konvertiert die Warnung in eine spezielle NoW Nachricht (TSBMessage) und schickt diese an den Router.

Ein anderes Fahrzeug empfängt diese Nachricht und der Router leitet diese an das VVC weiter. Das VVC Modul entpackt die Nachricht, erstellt eine entsprechende Warnung und leitet diese an das WMM weiter.

Das WMM nimmt die empfangene Nachricht in die Datenbank auf und leitet alle relevanten Meldungen, nach einem positiven Positionscheck mit dem PRCM, an das HWM weiter.

Das HWM zeigt die Nachricht entsprechend in einem Fenster an.

Die detaillierte Funktion und Zusammenarbeit der Module sind in einem ActivityDiagram im Anhang dargestellt, siehe Abbildung 11.10.

8.2. Arbeitsaufgaben

Meine Aufgaben in diesem Projekt bestanden darin, die Applikation durch Funktionalität zu erweitern, sowie das Reimplementieren versch. (Teil-) Module.

Vorarbeit

Zunächst war eine Einarbeitung in den bestehenden Quellcode erforderlich. Die Software war dafür ausreichend dokumentiert ([4], [5]). Auch die Dokumentation im Quellcode war sehr hilfreich. Jedoch empfand ich es als relativ schwierig mich in fremden Quellcode

einzulesen, da die Struktur und das Zusammenspiel der einzelnen Komponenten doch recht komplex ist.

Für das bessere Verständnis des verwendeten OSGi Framework, war die Einarbeitung mithilfe eines Tutorials erforderlich. Dabei wurden zwei Bundles erstellt, die sich beim Framework registrieren und gegenseitig Funktionalitäten austauschen. Ein Bundle ist eine mit z.B. ANT erstellte JAR Datei, die ein erweitertes MANIFEST File besitzt. Darin wird angegeben, welche Packages das Bundle benötigt und welche Funktionen durch das Bundle bereitgestellt werden. Die Funktionen werden in Interfaces definiert, die als Java Package in der Eclipse Umgebung sichtbar sind und im Framework als Library-Bundle eingebunden werden. Jedes Bundle besitzt eine sogenannte Activator-Klasse. Diese beinhaltet die Aktionen, die bei dem Starten eines anderen Bundles, das eine benötigte Funktionalität besitzt, ausgeführt werden. Die Abhängigkeiten der Bundles untereinander stellt das Framework schön in einem Fenster dar.

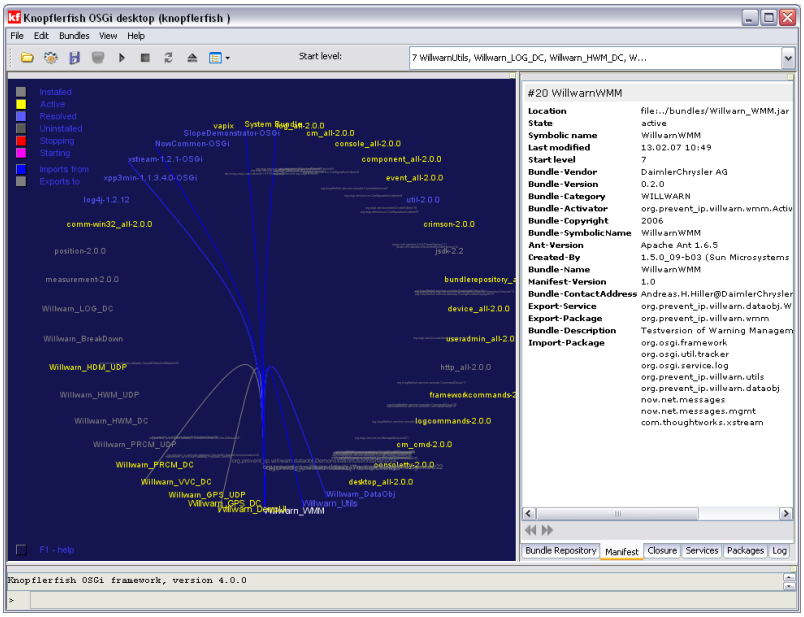


Abbildung 8.2.: Spin View Knopflerfish

Das Framework kann wahlweise über ein Fenster oder über die Konsole bedient werden. Die Bundles können dynamisch zur Laufzeit eingebunden, gestartet und gestoppt werden.

Die erste Aufgabe an diesem Projekt war, die OSGi Funktionalität zu überarbeiten.

Die Bundles wurden meist nicht richtig entfernt und es traten NullPointerExceptions auf. Durch ein Überarbeiten der Aktivator Klassen konnte dieses Problem behoben werden.

HWM

Abbildung 8.3.: Hazard Warning Module

Problem

Aufgrund eines sekündlichen Zeitintervalls wird der Pfeil, der in Abbildung 8.3 die Richtung der Gefahr anzeigt, nur sekündlich aktualisiert. Da dies einen sehr ruckhaften Eindruck auf den Fahrer macht, soll die Darstellung des Pfeils, sowie der Entfernung in einem kleineren Zeitintervall stattfinden. Dazu muss die Entfernung, welche immer sekündlich vom WMM aus aktualisiert wird, extrapoliert werden.

Lösung

Durch einen zusätzlichen Thread wird die zukünftige Entfernung mithilfe der aktuellen Entfernung / Geschwindigkeit nach der Formel 8.1 extrapoliert.

$$distance' = distance - \frac{v[km/h] \cdot 1000}{3600} \cdot \frac{200[ms]}{1000} \tag{8.1}$$

Danach wird die Methode notifyWarning(), die normalerweise vom WMM aufgerufen wird, durch den Thread mit der zuvor extrapolierten Entfernung aufgerufen. Dies aktualisiert automatisch die Pfeilänge / -richtung und die Entfernungsanzeige. Alle 200ms

berechnet der Thread die Entfernung neu und ruft die notifyWarning() Methode auf. Wenn neue, “reale” Daten vom WMM empfangen werden, so werden diese Werte zur Extrapolation verwendet.

Problem

Normalerweise wird eine Warnung im HWM explizit vom WMM entfernt. Sollte dies aber aus unerwartenden Gründen nicht passieren, würde die Warnung im HWM ständig angezeigt bleiben, obwohl diese vielleicht gar nicht mehr gültig ist. Falls die Warnung aus dem HWM nicht mehr entfernt wird, würde die Extrapolation der Entfernung negative Wert erzeugen, die gegen $-\infty$ laufen.

Lösung

Nach einem bestimmten Zeitintervall muss die Warnung auf ihre Gültigkeit geprüft werden. Wird eine Warnung vom WMM empfangen, so wird ein Zeitstempel hinzugefügt oder aktualisiert. Bei dem Zeichnen der Nachricht wird dieser Zeitstempel mit der aktuellen Systemzeit verglichen. Sollte die Differenz einen bestimmten Treshold überschreiten, wird die Nachricht automatisch gelöscht.

Log-Window

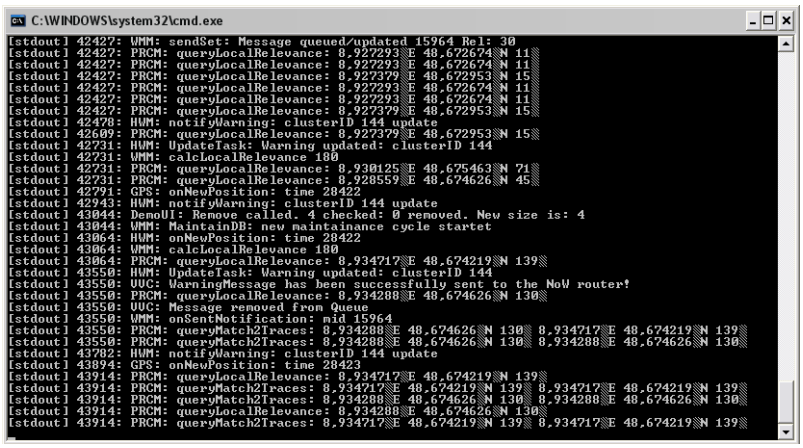


Abbildung 8.4.: Log-Ausgabe in der Console

Die Software besitzt zum Debuggen und zur allgemeinen Überwachung verschiedene LogLevels. Diese reichen von der Ausgabe von Fehlermeldungen (Level 1), die das weitere Funktionieren der Applikation stark beeinträchtigen, über die Ausgabe von Warnungen (Level 2), die zwar einen Fehler andeuten, der aber nicht weiters schlimm ist, bis hin zur Info (Level 3) oder Debug (Level 4) Ausgabe. Je nach gewähltem Level, wird sehr viel Text auf der Konsole ausgegeben und ein Mitlesen oder gar Debuggen gestaltet sich als sehr schwierig.

Lösung

Es soll ein neues Bundle erstellt werden, das die Log-Ausgabe der verschiedenen Module in sperate Textfelder umleitet. Das Bundle soll sich zur Laufzeit dynamisch aktivieren lassen und den Log-Stream automatisch von der Console in die Textfelder umleiten. Nach dem Stoppen des Bundles, soll der Log-Stream wieder zurück in die Console geleitet werden.

Sobald das Bundle geladen wird, ersetzt es die Klasse, die normalerweise die Funktion System.out.println() verwendet, durch eine JFrame abgeleitete Klasse, die den Log-Stream entsprechend parst. Die Buchstaben zwischen dem Anfang einer Logzeile und einem ":" Zeichen gibt das Modul an, aus dem die Zeile stammt. So kann jede Zeile eindeutig einem Modul und dementsprechend einem Textfeld zugeordnet werden. Durch die Verwendung der Ausgabe in einem Textfeld, bietet sich die Möglichkeit an, die einzelnen LogLevels grafisch hervorzuheben. Jeder LogLevel hat eine eigene Textfarbe und -stil. Es gibt außerdem die Möglichkeit den Text, ein spezielles durch Anklicken ausgewähltes Logfeld, in einem großen Textfeld anzeigen zu lassen. Siehe auch Screenshot in Abbildung 8.5.

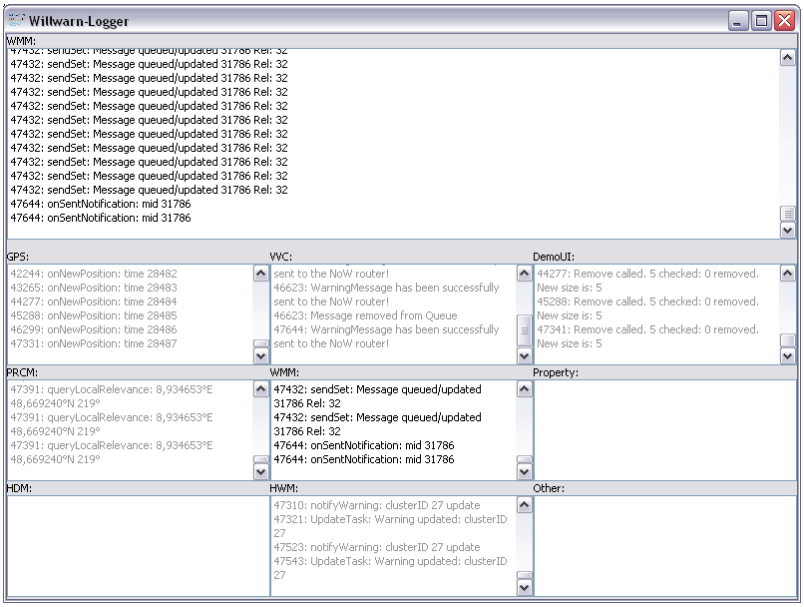


Abbildung 8.5.: Log-Ausgabe im Fenster

DemoUI

Für eine bessere Unterscheidung der einzelnen Fenster (Log-Window, HWM, DemoUI) sollten versch. Icons entworfen werden. So kann beim Fensterwechsel mit Alt+Tab schneller das gewünschte Fenster in den Vordergrund geholt werden. Die Icons wurden grafisch an das Willwarn Logo angepasst.

Abbildung 8.6.: Icons der verschiedenen Willwarn Fenster

Problem

Es wird die Funktion gewünscht, die aktuellen Warnungen, die sich in der Datenbank des WMMs befinden in eine Datei zu schreiben, um diese später einfach wieder einzulesen. Dabei soll die Datei in Textform geschrieben werden, so dass unter Umständen per Hand kleine Änderungen an den Warnungen vorgenommen werden können.

Lösung

Zunächst war die Idee, die Object Writer Funktion, welche standardmäßig im JDK enthalten ist, zu verwenden. Dabei werden die Java Objekte aber binär in die Datei geschrieben und ist somit nicht per Hand veränderbar. Eine andere Möglichkeit wäre, toString() Funktionen für die Warning Objekte zu implementieren und diese in eine Datei zu schreiben. Es müsste dann noch zusätzlich ein Parser entwickelt werden, der aus der Zeichenkette wieder ein Objekt zusammenbaut. Dies hat sich als sehr schwierig herausgstellt, da sich das Warning Objekt aus vielen weiteren Objekten zusammensetzt. Dabei muss für jedes Objekt eine toString() Methode sowie der dazugehörige Parser implementiert werden. Eine weitere Möglichkeit wäre die Verwendung der XML Reader/Writer Klassen, die standardmäßig im JRE integriert sind und in der Vorlesung Softwareentwicklung 2 vorgestellt wruden.. Dabei hat sich jedoch das Problem herausgestellt, dass alle Objekte JavaBEAN kompatibel sein müssen und es dürfen auch keine inneren Klassen verwendet werden. Die Anpassungen des Codes wären sehr tiefgreifend und zeitaufwändig gewesen. Durch einen glücklichen Zufall wurden die XStream Bibliotheken im Internet gefunden. Diese können auch sehr komplexe und ineinander verschachtelte Java Klassen in eine XML Datei schreiben. Dies ist die schnellste und eleganteste Lösung.

In der DemoUI wurden dementsprechend ein Menü-Eintrag angelegt, der das Lesen und Schreiben der Warnungen in ein XML File zur Verfügung stellt. Bei dem Einlesen der Warnungen muss noch die ExpirationTime einer Nachricht der aktuellen Systemzeit entsprechend angepasst werden. Sonst wären die Nachrichten sofort ungültig, da Meldungen die vielleicht Tage oder Wochen (Zeitpunkt des Speicherns) zurückliegen, als nicht mehr relevant markiert werden.

Die XStream Bibliotheken mussten noch entsprechend angepasst werden, um im OGSi Framework zu funktionieren. Dabei musste das MANIFEST entsprechend erweitert werden.

Problem

Die Darstellung der Warnungen in der DemoUI sieht wie folgt aus: Es werden ein Trace, ein Symbol, das die entsprechende Warnung angibt, und 3 Zahlen angezeigt. Die weiße

Zahl gibt die Reliability der Warnung an. Die grüne Zahl gibt die Anzahl der Meldungen pro Event an und die rote Zahl gibt die Anzahl der aufgehobenen Meldungen pro Event an. Sollte ein Fahrzeug zum Beispiel im Bereich einer Nebelwarnung liegen, aber nicht die Nebelscheinwerfer aktivieren, so wird eine "negative" Warnung erzeugt. Dies hat Einfluss auf die Reliability und Gültigkeit einer Warnung.

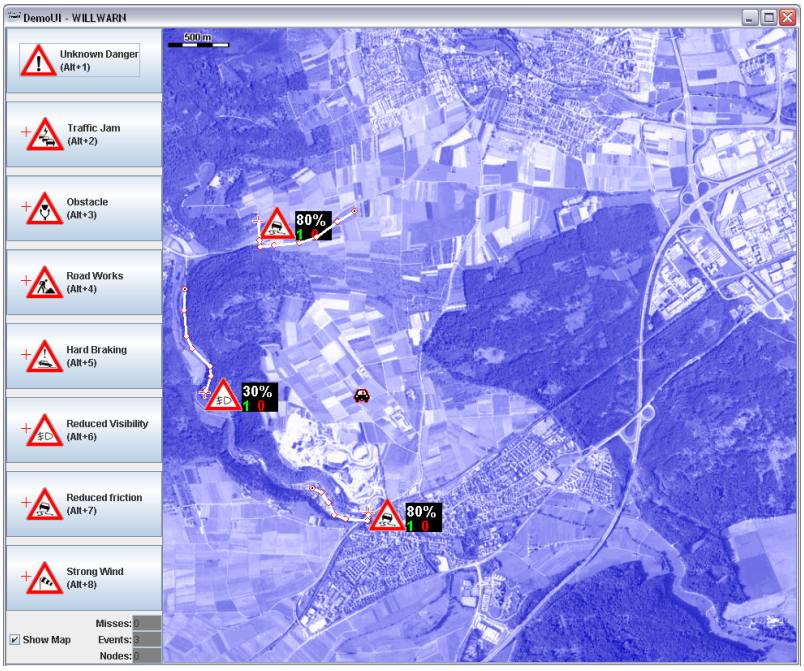


Abbildung 8.7.: DemoUI, seither

Diese Anzeige soll kompakter gemacht werden, da das Symbol und die Zahlen doch relativ viel Platz einnehmen und somit andere Warnungen überdecken können.

Lösung

Durch das Entfernen des roten Dreiecks des Symbols wird diese Anzeige wesentlich kleiner. Das Warnrot des Dreiecks wurde in ein ausgefülltes Dreieck umgewandelt, das auf den Ursprung der Warnung zeigt. Rechts daneben wird in kompakter Form die drei Zahlen sowie der Ursprung der Warnung anzeigt (siehe Screenshot 8.8).

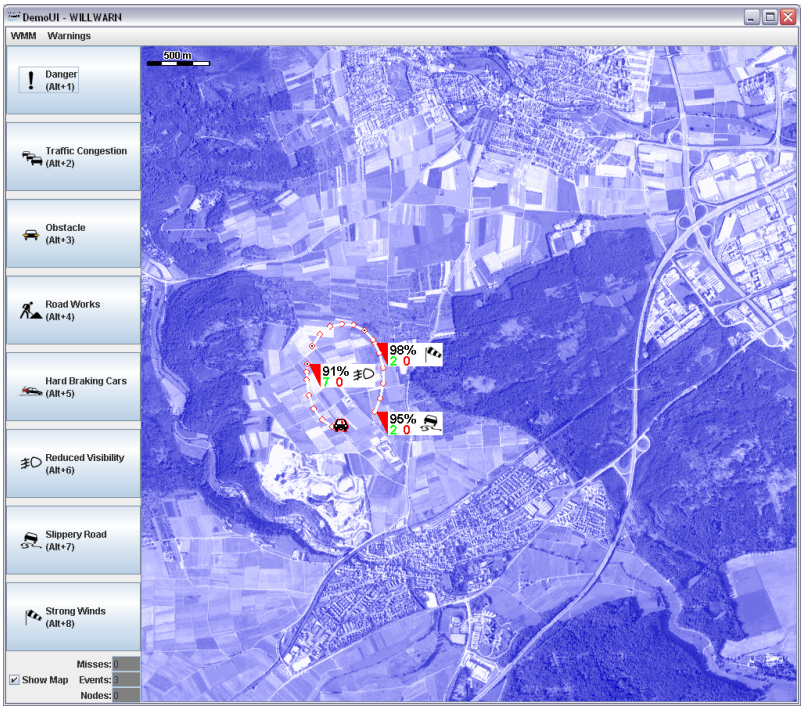


Abbildung 8.8.: DemoUI, in kompakterer und erweiterter Version

Problem

Die DemoUI bietet die Möglichkeit durch eine Checkbox die Anzeige der Karte zu deaktivieren. Dann wird einfach der Hintergrund ausgeblendet und das eigene Fahrzeug entsprechend der unteren linken und oberen rechten Ecke gezeichnet. (Bevor man die DemoUI startet, muss man diesen Ecken entsprechend Läüngen- und Breitenangaben zuordnen.)

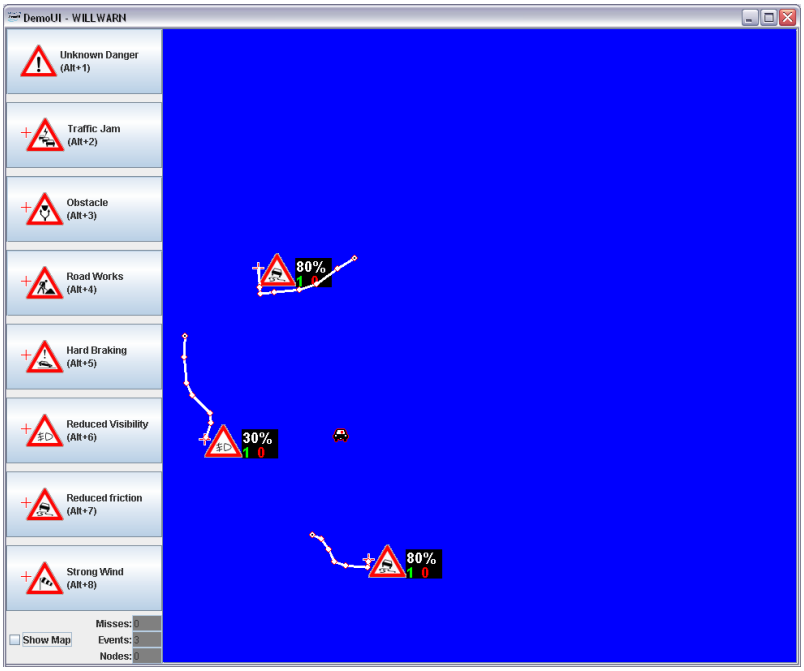


Abbildung 8.9.: DemoUI, seither, ohne Karte

Diese Anzeige soll überarbeitet werden. Das Fahrzeug soll sich immer in der Mitte des Fensters befinden. Die Warnungen bewegen sich dann dementsprechend um das Fahrzeug. Außerdem soll als Hintergrund ein Gitter anzeigt werden, dass eine Kästchenlänge von 100m hat. Der sichtbare Radius soll 4000m betragen.

Lösung

Sobald die Checkbox “Karte anzeigen” deaktiviert wird, muss das Auto in der Mitte des Frames gezeichnet werden und die Längen- und Breitenangaben der linken unteren Ecke (Origin) und der oberen rechten Ecke (Limit) müssen entsprechend der folgenden Formeln neu berechnet werden.

Die Formeln müssen für den Längengrad respektive Breitengrad angewendet werden:

$$x = 6371000.8 \cdot \frac{\Pi}{180} \cdot longitude \cdot \cos\left(latitude \cdot \frac{\Pi}{180}\right) \tag{8.2}$$

Abbildung 8.10.: Umwandlung in X/Y-Koordinaten

$$y = 6371000.8 \cdot \frac{\Pi}{180} \cdot latitude \tag{8.3}$$

Abbildung 8.11.: Umwandlung in X/Y-Koordinaten

$$x' = x - 4000 \cdot \cos\left(\tan^{-1}\left(\frac{height}{width}\right)\right) \tag{8.4}$$

Abbildung 8.12.: x für Origin mit Radius 4000m

$$y' = y - 4000 \cdot \sin\left(\tan^{-1}\left(\frac{height}{width}\right)\right) \tag{8.5}$$

Abbildung 8.13.: y für Origin mit Radius 4000m

$$latitude = \frac{y}{6371000.8 \cdot \frac{\Pi}{180}} \tag{8.6}$$

Abbildung 8.14.: Umwandlung in WGS84 Koordinaten

$$longitude = \frac{x}{6371000.8} \cdot \frac{\frac{180.0}{\Pi}}{\cos\left(latitude \cdot \frac{\Pi}{180}\right)} \tag{8.7}$$

Abbildung 8.15.: Umwandlung in WGS84 Koordinaten

Jedes weitere Neuzeichnen kann mithilfe der translate() Funktion des Graphic Objekts erledigt werden. Somit spart man sich den Rechenaufwand der Längen- und Breitenan-

gaben für Origin und Limit.

Das Gitter wird dem sichtbaren Radius entsprechend berechnet und als Bild im Speicher abgelegt. Das Bild ist nur um zwei Kästchen in der Höhe/Breite größer. So bleibt durch geschicktes Verschieben der Rand des Bildes nicht sichtbar.

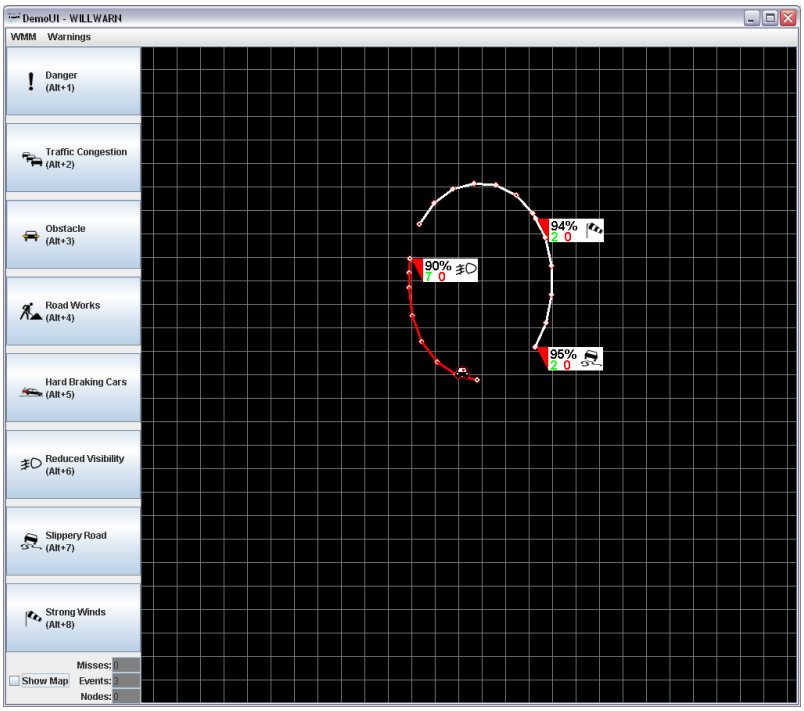


Abbildung 8.16.: DemoUI, jetzt, ohne Karte

VVC LAN

Für die finale Demonstration des gesamten PReVENT Projekts im Sommer 2007 in Paris, soll die Willwarn Software auch ohne den NoW Router mit anderen Fahrzeugen kommunizieren. Die Kommunikation zwischen den Fahrzeugen (in denen alle PReVENT Projekte installiert sind) läuft über eine WLAN Verbindung (bzw. Standard Ethernet), für deren Aufbau ein anderes Team zuständig ist. Die Aufgabe ist nun, ein neues VVC Modul zu implementieren, welches die Nachrichten nicht an einen NoW Router weiterleitet, sondern über ein LAN Interface mit anderen Rechnern kommuniziert.

Als Entwicklungsumgebung standen mir zwei Notebooks mit Suse 9.3 und Fedora Core 4 zur Verfügung, die über ein Crossover-Kabel verbunden werden konnten.

Lösung

Zunächst muss ein einfaches Protokoll entwickelt werden. Das VVC Modul in Rechner A empfängt sekündlich einen neue Position von dem GPS Empfänger. Diese Position wird immer an die Broadcast Adresse des Netzwerkes weitergeleitet. (Das WMM enthält einen sogenannten Neighbortable. Darin werden alle sichtbaren Nodes (andere Fahrzeuge) gespeichert) Sobald Rechner B das Broadcastpaket empfängt, wird der neue Node automatisch in die Neighbortable übernommenm, bzw. aktualisiert. Außerdem werden alle Warnings, die in der Datenbank des WMM enthalten sind, an den Sender des Broadcastpakets gesendet. Auf jeden Broadcast wird also mit dem Senden aller Warnings geantwortet. Der erhöhte Netzwerkverkehr ist tragbar, da bei der späteren Anwendung nicht mehr als 2-3 Fahrzeuge verwendet werden! Es müssen dadurch auch keine Acknowledge Pakete zur Bestätigung des korrekten Empfangs gesendet werden.

Sollte das Broadcastpaket von dem eigenen Rechner stammen, wird automatisch eine Methode aufgerufen, die den Neighbortable wieder löscht. Sobald das letzte Update eines Nodes älter als z.B. 3 sec ist, wird der Node automatisch gelöscht, da er sich offenbar nicht mehr in Reichweite befindet.

Sobald das VVC eine WarningMessage empfängt, wird diese automatisch an das WMM weitergeleitet.

Die WarningMessages und die GPS Positionen werden auf getrennten Ports empfangen

VVC

Das VVC Modul wurde ursprünglich von BMW implementiert. Da der Quellcode nicht zur Verfügung steht und man vielleicht ein paar Änderungen daran vornehmen möchte, soll dieses Modul neu implementiert werden. Hinzu kommt, dass das NoW Projekt eine neue Version der Router Software (Version 1.3) entwickelt hat und das alte VVC damit nicht kompatibel ist.
Das neue VVC soll mit dieser neuen Router Version kompatibel sein.

Lösung

Die Dokumentation der neuen Routersoftware offenbarte, dass sich manche Schnittstellen verändert haben und die Router untereinander nun komplett auf IPv6 umgestellt

wurden. Außerdem wurde das Handling der Neighbortable völlig neu konzipiert. Der Austausch dieser Neighbortable funktioniert nun über einen sogenannten Information-Connector. Dieser funktioniert nach dem "publish-subscribe-notify pattern". Das VVC Modul muss sich zunächst bei dem InformationConnector registrieren. Sobald der Router einen neuen Node empfängt, reagiert der Connector automatisch darauf und sendet einen Event an alle registrierten Objekte. Das erkannte Fahrzeug wird dann vom VVC an den Neighbortable des WMM weitergeleitet. Da dieser völlig neu implementiert werden musste, ist leider eine Kompatibilität mit dem alten VVC (für die NoW Router Version 1.2) nicht mehr gewährleistet.

Der NoW Router verwendet keine gewöhnlichen TCP/IP Pakete sondern spezielle Pakete, die die neuen Funktionen des dynamischen Netzwerks voll ausnutzen. Darunter gibt es z.B. die Mölichkeit, Nachrichten für ein bestimmtes geographisches Gebiet zu senden oder die Anzahl der Hops für eine Nachricht festzulegen. Für das VVC wurde ein entsprechender Pakettyp gewählt und implementiert.

Außerdem sah das ursprüngliche Design des VVCs vor, eine Queue zu verwenden. Nachrichten vom WMM werden in eine Queue eingereiht. Ein eigener Thread sendet stets das erste Elemente der Queue an den Router. Wurde das Paket korrekt empfangen wird, das Element aus der Queue entfernt. Dieses Konzept wurde beispielhaft implementiert. Das Problem ist, dass der Router in der jetzigen Version keine Bestätigung des korrekten Empfangs eines Pakets zurückschickt. Der Thread im VVC wartet also kurz und nimmt an, dass der Router das Paket korrekt empfangen hat. Sobald die Router Software die gewünschte Funktionalität unterstützt, ist das VVC durch das ändern einer Zeile schnell angepasst.

NoW Router einrichten

Für weitere Demonstrationen ist es nützlich einen zusätzlichen NoW Router zu haben. Diese sollte auf einem Notebook installiert werden.

Lösung

Diese Aufgabe hat sich als recht schwierig herausgestellt. Die Installation war zwar etwas dokumentiert, aber da ich noch ein Neuling in Sachen Linux war, wurde das Einbinden eines IPv6_queue Moduls in den Kernel sehr schwer! Nach dem Probieren mehrerer

Linux-Distributionen (Kubuntu 6.10, Suse Linux 10.2, Suse Linux 10.1, Fedora Core 3) und vielen Versuchen einer Kernelkompilation hat es letztendlich doch noch geklappt, die Router Software zum Laufen zu bringen.

8.3. Ergebnis

Das Endergebnis dieses Projekts ist eine funktionierende Version 1.2, die mit dem VVC von BMW und der NoW Router Version 1.2 zusammenarbeitet, eine Version 1.3 die mit der Router Version 1.3 zusammenarbeitet und einige Architekturänderungen, sowie die neue DemoUI beinhaltet. Zusätzlich das Notebook mit den getesteten Versionen des NoW Routers 1.2 und 1.3 und eine dazugehöriges Tutorial, wie man die Software installiert.

8.4. Fazit

Bei diesem Projekt habe ich sehr viel gelernt. Das OSGi Framework (SOA Architektur) war mir völlig neu. An das Arbeiten mit fertigem Quellcode musste ich mich erstmal gewöhnen, da manche Dinge einfach anders gelöst wurden, als man durch den eigenen Stil gewohnt ist.

Außerdem durfte ich an verschiedenen Testfahrten mithelfen, die stets eine willkommene Abwechslung waren. Auch die finale Demonstration des Projekts im November 2006 hat mich sehr beeindruckt. Durch das Anwesen aller beteiligten Partner aus ganz Europa wurde erst die wahre Größe dieses Projekts sichtbar.

Die Idee des WILLWARN Projekts ist, meiner Meinung nach, ein äußerst sinnvoller Einsatz der Car2Car Communication und ein lohnender Weg zum Ziel des unfallfreien Fahrens.

9. Vergleich GPS Empfänger

Dieses Projekt begann Mitte Februar 2007 bis Ende Februar 2007 und wurde von Thomas Passegger betreut.

9.1. Vorstellung des Projekts

Das Ziel war, den fest eingebauten GPS Empfänger der Mercedes S-Klasse mit einem SIRF-III Empfänger zu vergleichen. Die SIRF-III Empfänger werden heutzutage in den vielen tragbaren Navigationssytemen verbaut.

9.2. Arbeitsaufgaben

Folgende Eigenschaften der Empfänger soll verglichen werden:

- Position
- HDOP
- gemessene Fahrzeuggeschwindigkeit
- empfangene Satelliten
- Ausfallzeit
- Querabweichung

Die Ausfallzeit ist die Zeitdifferenz zwischen der letzten gemessenen Position vor einer Tunneleinfahrt und der ersten gemessenen Position nach einer Tunnelausfahrt. Die Querabweichung gibt die Abweichung der gemessenen Position zu einer Geraden an. Die Gerade entspricht dabei dem Straßenverlauf einer geraden Straße.

Für diese Eigenschaften mussten entsprechende Fahrtrouten gewählt werden. Viele Tunnels, Unterführungen und Häuserschluchten gibt es auf der B14 zwischen Heslach

Tunnel und BadCannstatt. Für die Berechnung der Querabweichung wurde die Olgastraße aufgrund ihres kerzengeraden Verlaufs gewählt.

Der SirfIII Empfänger wird über die RS232 Schnittstelle an ein Notebook angeschlossen. Über ein Simulink Modell, das bereits vorgefertigt ist, kann eine Matlab Matrize automatisch mit folgenden Werten erstellt werden:

- UTC Zeit
- Longitude
- Latitude
- Altitude
- Course
- Speed
- HDOP
- VDOP
- Satellites
- Fix Point

Da für die spätere Auswertung Matlab verwendet wird, kann so schnell damit weitergearbeitet werden. Der fest eingebauten Empfänger im Mercedes kann über ein Notebook, das mit der Headunit über eine Ethernetverbindung kommuniziert, ausgelesen werden. Die Software stellt eine Nachbarabteilung dafür bereit. Um mit der Logdatei, in der die GPS Position und die dazugehörigen Daten aufgezeichnet wurde, arbeiten zu können, musste zunächst ein Parser entwickelt werden. Dabei wurde ein Java Programm implementiert, das die Logdatei einliest, die Zeilen mit den GPS Daten entsprechend zerschneidet und die Werte durch einen Separator getrennt in eine Datei schreibt. Diese Datei kann dann in Matlab eingelesen und mit der Matrix des SirfIII Empfängers verglichen werden.

Das mir zur Verfügung stehende Fahrzeug war ein Vorserienmodell und konnte nach einem Test leider nicht ausgelesen werden. Nach dem gescheiterten Versuch ein Fahrzeug aus dem Fuhrpark aus Untertürkheim auszulesen, wurde die HeadUnit im Fahrzeug

ausgetauscht. Da leider sehr viel Zeit dabei verloren ging, musste die Auswertung an den letzten drei Tagen des Praktikums stattfinden.

9.3. Ergebnis

Das Ergebnis der Messung wurde in dem Bericht [10] ausführlich dokumentiert. Die Befürchtung, dass der fest eingebaute Empfänger im Mercedes schlechter sein könnte, als der SirfIII Empfänger hat sich nicht bestätigt. Eher im Gegenteil: der fest eingebaute Empfänger liefert bessere Positionen und eine Querabweichung von 4m. Im Vergleich dazu hat der SirfIII Empfänger eine Querabweichung von 5,2m. Der SirfIII Empfänger hat dafür eine kürzere Ausfallzeit um durchschnittlich 1,4sec.

9.4. Fazit

Dieses letzte Projekt empfand ich als sehr spannend, da ich viel Einblick in Bereiche hatte, die mir davor noch unbekannt waren. Wie z.B. das Auswerten mit Matlab und allgemein mit dem Arbeiten von GPS Positionen. Schade, dass zum Ende hin die Zeit leider sehr schnell zu Ende ging.

10. Fazit

Im ersten Projekt (TMC) konnte ich das zuvor erlernte Wissen aus der Vorlesung SoftwareEngineering, Softwareentwicklung 1-3 und Datenbanken in der Praxis anwenden. Die anstehenden Tests im Testträger waren stets Abwechslung und Motivation zugleich. Vor allem war die Testfahrt in dem Testträger Mercedes S500 (W221) Vollausstattung war sehr eindrucksvoll. Sie diente zum Testen des Navigationssystems (Verhalten, Eingabe, Screendesign, ...) Während der Entwicklung gab es hier und da ein paar harte Nüsse zu knacken, aber schlussendlich bin ich sehr zufrieden mit dem Endergebnis, das jetzt den Fahrern des Testträgers zur Verfügung steht.

Bei dem zweiten Projekt (Willwarn) fand ich es sehr interessant, in ein bereits bestehendes Softwareprojekt Einblick zu erhalten. Vor allem zu entdecken, wie manche Probleme, mit denen ich bereits in meinem ersten Projekt konfrontiert wurde, gelöst wurden. Bereits die Idee, ein Adhoc Drahtlosnetzwerk zwischen Fahrzeugen aufzubauen, um darüber Warnungsnachrichten zu verschicken, finde ich sehr spannend. Auch das Mitverfolgen einer finalen Demonstration in der Industrie mit Teilnehmern aus Griechenland, Frankreich, Niederlande, Norwegen (EU-Kommissarin) und ganz Deutschland war ein großes Erlebnis. Bei dem Einrichten der Router Software auf einem Notebook habe ich sehr viel über Linux gelernt und jetzt kann ich auch meinen eigenen Kernel erfolgreich kompilieren.

Das dritte Projekt (GPS) fand ich sehr spannend. Einerseits war mir die Funktion der Positionsbestimmung bei GPS völlig unbekannt, andererseits konnte ich mein Wissen auf dem Gebiet wissenschaftlicher Messungen und Auswertungen erweitern. Auch den kurzen Einblick in das MTC (Mercedes Technology Center) in Sindelfingen war sehr interessant, zumal für den ca. 15 Minütigen Aufenthalt im MTC ein neuer Werksausweis für mich ausgestellt werden musste... Auch das Fahren von Oberklasse Fahrzeugen wie S 420 CDI und S 500 war eine sehr luxuriöse Erfahrung.

Worüber ich zunächst ein wenig erstaunt war ist, dass ich alle Themen alleine bearbeiten durfte. Eigentlich habe ich gedacht, dass ich während des Praktikums im Team

arbeiten werde. Dies war aber kein Nachteil, denn so konnte ich mich völlig frei entfalten. Durch das Arbeiten im Studentenzimmer, war stets die Hilfe von anderen Studenten möglich.

Allgemein empfand ich die Zeit bei DaimlerChrysler als sehr kurzweilig, da alle Projekte sehr praxisnah durch die Tests und Meßfahrten auf der Straße waren. Auch das Arbeiten im Studentenbüro war sehr locker und es herrschte ein entspanntes Klima. Durch mein bisheriges Studium habe ich das notwendige Wissen erworben, um die Aufgaben lösen zu können.

Schlussendlich bin ich sehr zufrieden mit dem Praktikum im Unternehmen und kann jedem anderen Studenten, der sich für dieses Themenumfeld interessiert, die Stelle nur wärmstens empfehlen!

11. Anhang

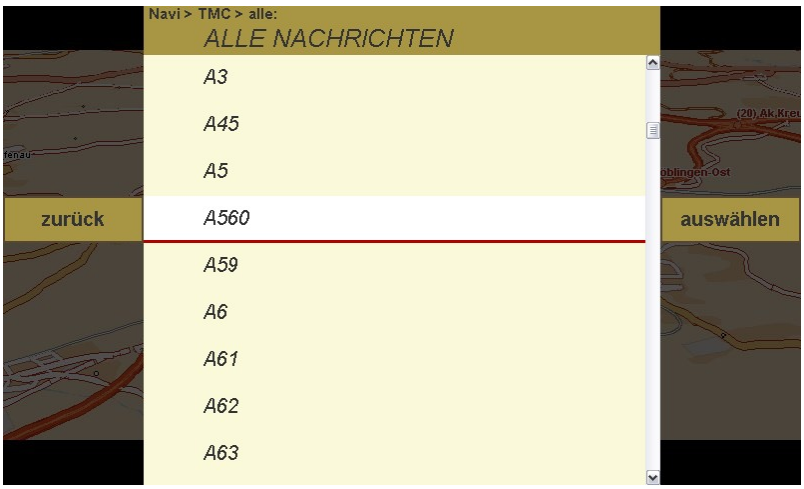


Abbildung 11.1.: Straßenansicht aller Nachrichten

Abbildung 11.2.: Eventansicht pro Straße

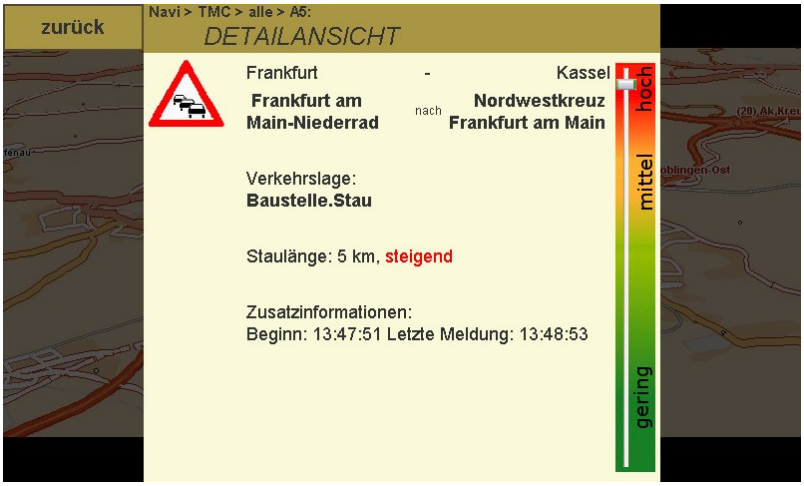


Abbildung 11.3.: Eventdetailansicht

Abbildung 11.4.: E-Klasse Testträger

Abbildung 11.5.: E-Klasse Headunit

Abbildung 11.6.: Smart Testträger

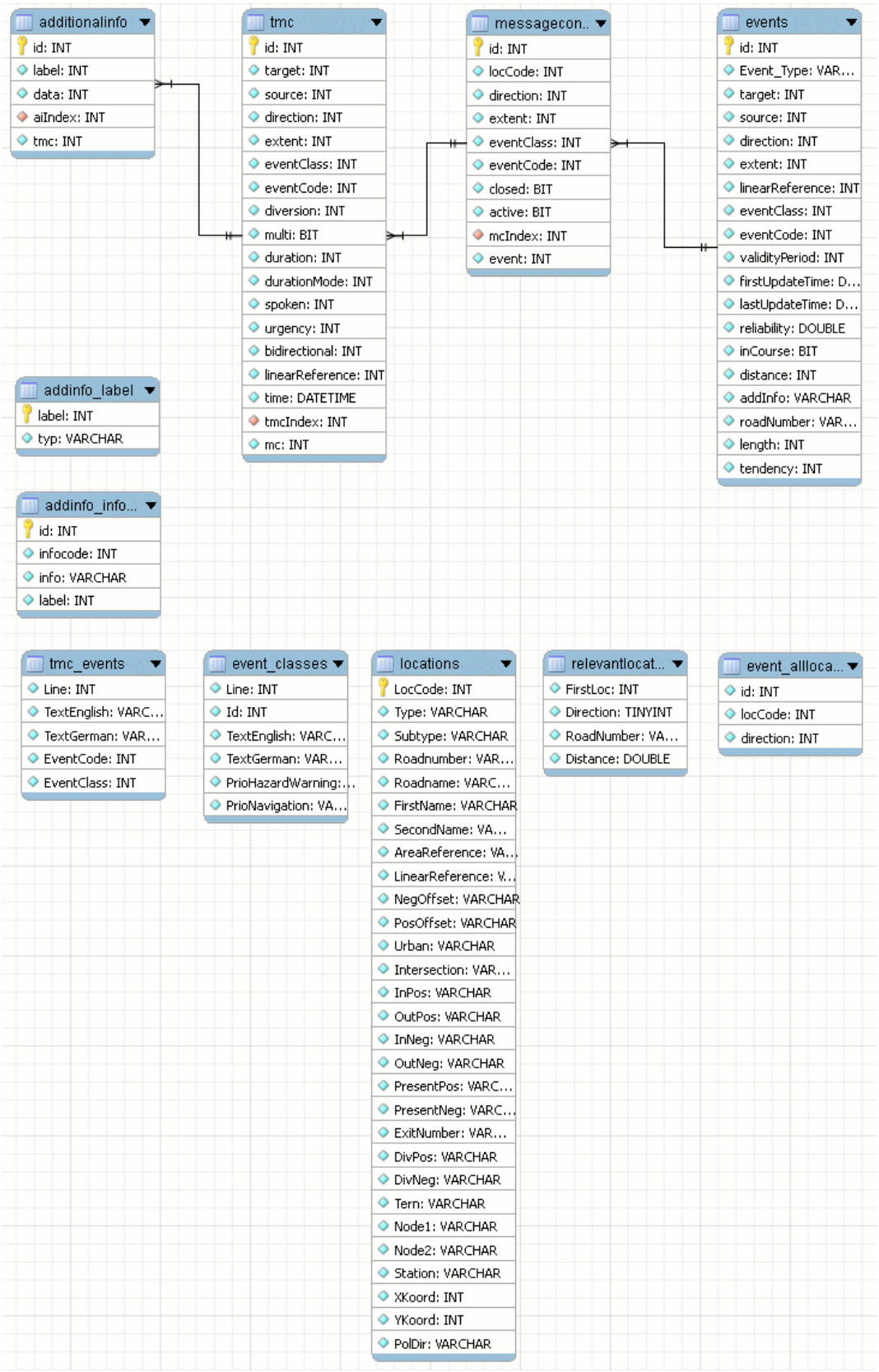


Abbildung 11.7.: ERM Diagram

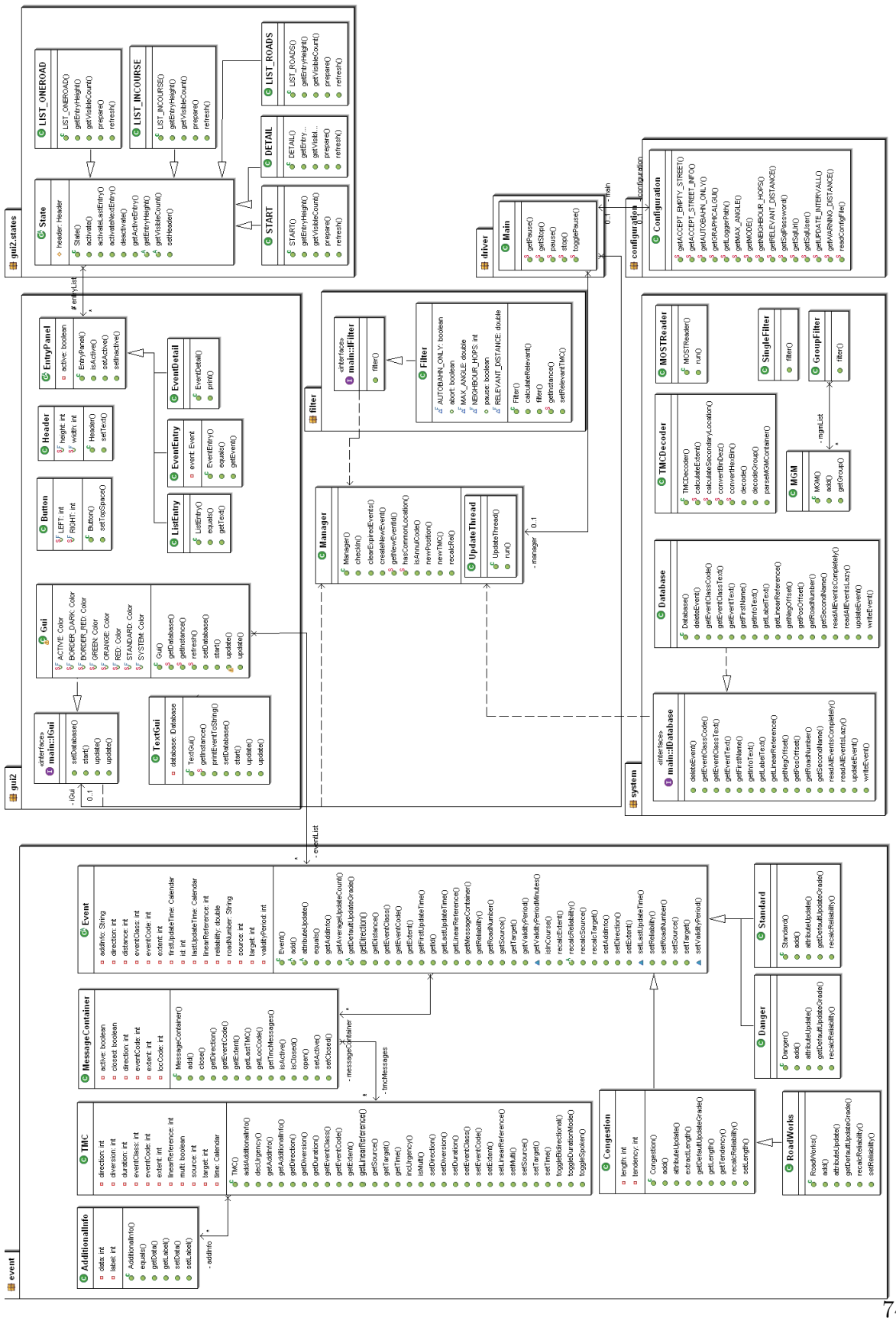


Abbildung 11.8.: Class Diagram

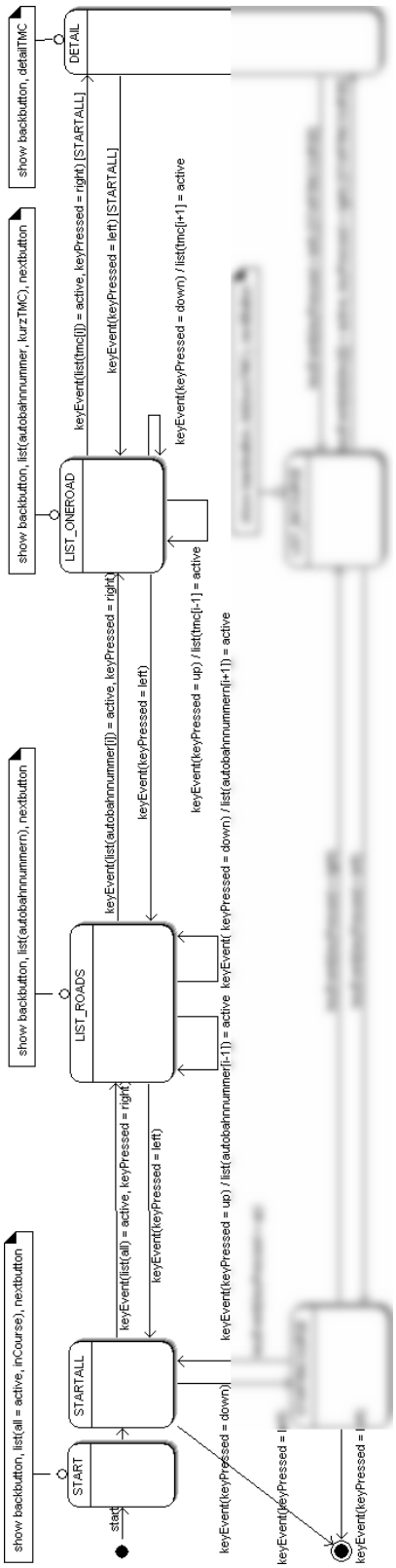
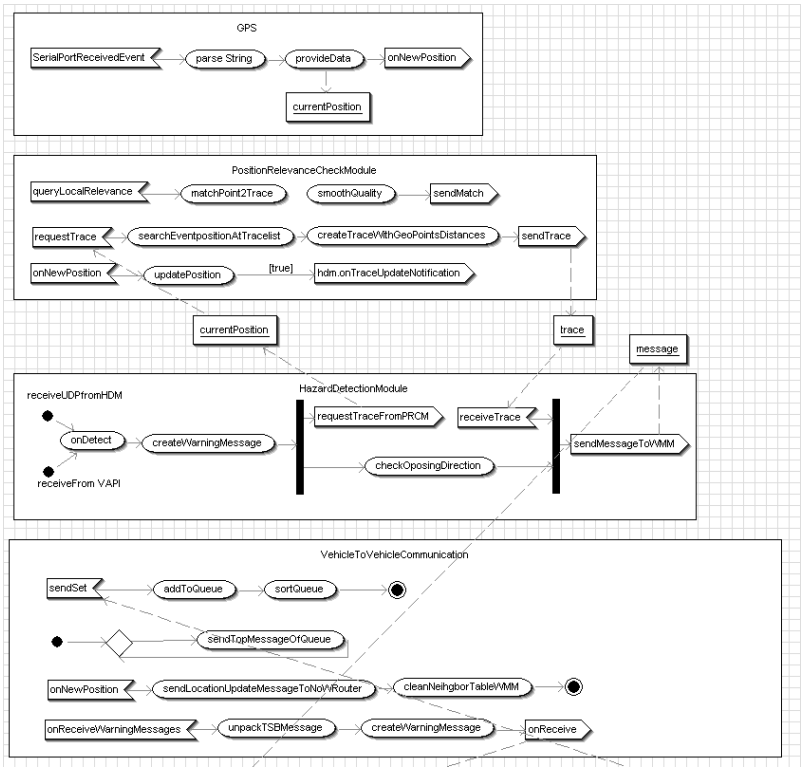
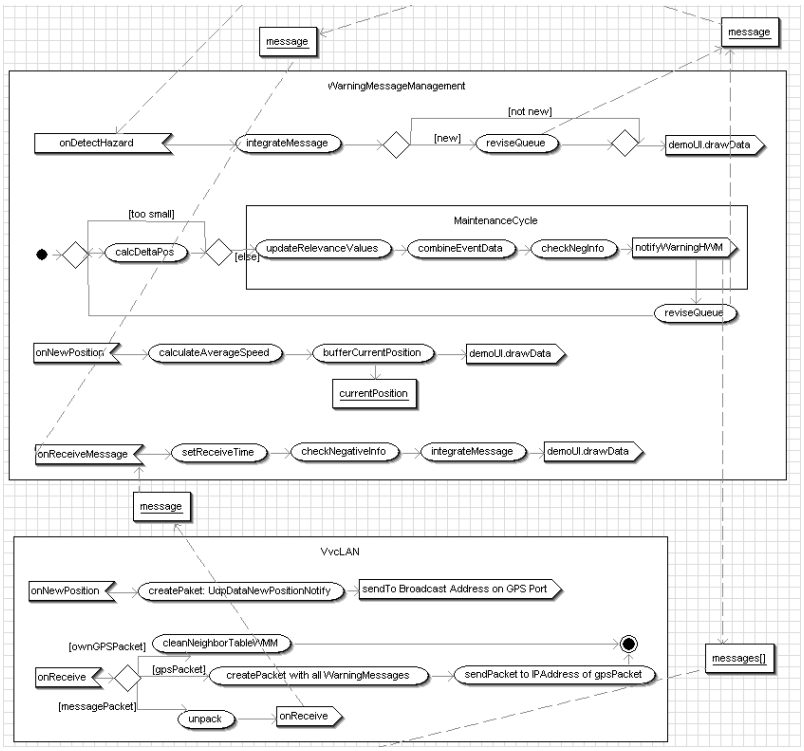


Abbildung 11.9.: Statemachine Diagram


GPS
SerialPortReceivedEvent
parse String
provideData
onNewPosition
currentPosition
PositionRelevanceCheckModule
queryLocalRelevance
matchPoint2Trace
smoothQuality
sendMatch
requestTrace
searchEventpositionAtTracelist
createTraceWithGeoPointsDistances
sendTrace
onNewPosition
updatePosition
[true]
hdm.onTraceUpdateNotification
currentPosition
trace
message
HazardDetectionModule
receiveUDPfromHDM
onDetect
createWarningMessage
receiveFrom VAPI
requestTraceFromPRCM
receiveTrace
checkOposingDirection
sendMessageToVMM
VehicleToVehicleCommunication
sendSet
addToQueue
sortQueue
sendTopMessageOfQueue
onNewPosition
sendLocationUpdateMessageToNoWRouter
cleanNeihgborTableVMM
onReceiveWarningMessages
unpackTSBMessage
createWarningMessage
onReceive


message
message
vWarningMessageManagement
onDetectHazard
integrateMessage
[new]
reviseQueue
[not new]
demoUI.drawData
[too small]
calcDeltaPos
[else]
MaintenanceCycle
updateRelevanceValues
combineEventData
checkNegInfo
notifyWarningHVM
reviseQueue
onNewPosition
calculateAverageSpeed
bufferCurrentPosition
demoUI.drawData
currentPosition
onReceiveMessage
setReceiveTime
checkNegativeInfo
integrateMessage
demoUI.drawData
message
VvcLAN
onNewPosition
createPaket: UdpDataNewPositionNotify
sendTo Broadcast Address on GPS Port
[ownGPSPacket]
cleanNeighborTableVMM
onReceive
[gpsPacket]
createPacket with all WarningMessages
sendPacket to IPAddress of gpsPacket
[messagePacket]
unpack
onReceive
messages[]

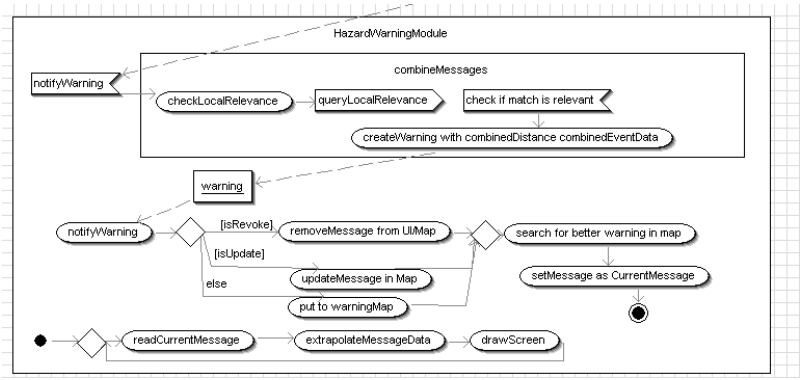


Abbildung 11.10.: Activity Diagram

12. Glossar

TMC Traffic Message Channel. Digitaler Radiodienst, der kontinuierlich über einen Radiokanal Verkehrsstörungen in encodierter Form ausstrahlt.

E-/U-Musik Mit E-Musik ist die ernste Musik gemeint. Dazu gehört unter anderem klassische Musik. Unter U-Musik versteht man die allgemeine Unterhaltungsmusik.

WILLWARN Wireless Local Danger Warning. Teilprojekt des PReVENT Projekts, das unter Verwendung der Car2Car Communication den Fahrer vor bevorstehenden Gefahren informiert.

Neighbortable Tabelle, die jedes Fahrzeug speichert. Darin sind alle sichtbaren Nachbarfahrzeuge enthalten inklusive Zeitstempel und deren aktuelle Position.

OSGi Open Services Gateway Initiative. Ein Konsortium, das eine hardwareunabhängige SOA Softwareplatfrom entwickelt hat. Dies setzt eine Java Virtual Machine voraus. OSGi definiert lediglich die APIs und Testcases.

Knopflerfish Dies ist ein Framework, welche die von OSGi definierten APIs und Testcases implementiert.

NoW Network on wheels. Ein Projekt, das ein Protokoll und einen Router für die Fahrzeug Fahrzeug Kommunikation entwickelt hat. Die Kommunikation basiert auf WLAN AdHoc Netzwerken. Zur Zeit wird noch das 5GHz Band verwendet, bald soll aber ein nicht-kommerziell benütztes Band verwendet werden.

Literaturverzeichnis

[1] Wolfgang Becker. *Standortbestimmung basierend auf kartographischen Daten mittels GPS und zusätzlichen physikalischen Ortungssensoren.* Fachhochschule Karlsruhe, WS 2003/04.

[2] CEN. *TMC Compendium - Alert-C Coding Handbook*, f02.1 edition, 2 1999.

[3] Gesellschaft für Verkehrsdaten mbH ddg. *Von der Meßwerterfassung bis zur automatischen generierten Verkehrsmeldung.* http://www.ddg.de/pdf-dat/datenerf.pdf, 2007.

[4] Arno Hinsberger Hans-Josef Hilt. *D22.61 Willwarn System Integration.* HTW.

[5] Andreas Hiller. *D22.41/II Willwarn Warning Message Management.* Daimler-Chrysler AG.

[6] Michael Köhne, Anja; Wößner. *GPS-Info, GPS - Global Positioning System.* http://www.kowoma.de/gps/, 2003.

[7] *Meyers großes Taschenlexikon*, chapter Band 18. 2006.

[8] Bernd Weiskopf. *ARI-Technik - Hilfe beim DX.* http://www.ukwtv.de/de/artikel/technik/ARI-Technik.pdf, 2001.

[9] Benjamin Zaiser. *Dokumentation: Bewertung von RDS/TMC gemeldeten Störungen auf Basis ihrer Historie*, chapter 6 Statistische Analyse. nicht veröffentlicht, 2006.

[10] Benjamin Zaiser. *Dokumentation: Sirf III und eingebauter Empfänger im Mercedes W221.* nicht veröffentlicht, 2006.

13. Erklärung

Es wird bestätigt, dass der von Herrn Benjamin Zaiser vorgelegte Tätigkeitsbericht in allen Teilen freigegeben ist und dass der Inhalt des Berichts der von ihm ausgeübten Tätigkeit entspricht.

Böblingen, den 1. März 2007