Praktikumsbericht

Bericht über Praktikum bei der BMW AG, Standort München,
Forschungs- und Innovationszentrum
(Abteilung EA-403 - Teilsystemtest/Absicherung Elektronik
Antriebsstrang)

Thema:
Erstellung eines CARMEN-Moduls zur BUS-Analyse

Zeitraum:
1. April 2007 bis 29. Februar 2008

Vorgelegt von:
Sven Richter

Betreuer:
Dipl.-Ing. Daniel Steffensky, EA-403

Inhaltsverzeichnis

KAPITEL 1

Einführung

1.1 Die Geschichte der BMW AG

Die Geschichte von BMW beginnt am 05. März 1912, als die „Flugwerke Deutschland GmbH“ in das Aachener Handelsregister eingetragen wird. Eine Zweigstelle für den Flugmotorenbau wird 1912 in München - Milbertshofen errichtet. Nur ein Jahr später beschließt die Gesellschafterversammlung die Auflösung der „Flugwerk Deutschland GmbH“. Am 20. August 1916 ist die Liquidation beendet, und die Firma wird aus dem Handelsregister gelöscht.

Auf dem Gelände der liquidierten „Flugwerke Deutschland GmbH“ wird die „Rapp Motorenwerke München GmbH“ gegründet. Ihr Ziel ist der "Bau und Vertrieb von Motoren aller Art, insbesondere Explosionsmotoren für Flugzeuge und Kraftfahrzeuge“, sowie der Erwerb der Motorenfabrik in München, die bisher der „Flugwerke Deutschland GmbH“ gehörte. Die Rapp Motorenwerke werden in „Bayerische Motoren Werke GmbH“ umbenannt. Nach der Gründung der „Bayerische Motoren Werke AG“ wird die GmbH am 19. August 1918 aufgelöst.

Das BMW Markenzeichen, welches einen stilisierten Propeller darstellt, wird für die Zeichenrolle des Kaiserlichen Patentamts angemeldet und am 10. Dezember 1917 eingetragen. Dieses war für folgende Waren bestimmt: "Land-, Luft- und Wasserfahrzeuge, Automobile, Fahrräder, Automobil- und Fahrradzubehör, Fahrzeugteile, stationäre Motoren für feste, flüssige und gasförmige Betriebsstoffe und deren Bestand- und Zubehörteile.

In einer außerordentlichen Hauptversammlung der „Bayerische Motoren Werke AG“ wird am 20. Mai 1922 der Motorenbau samt dem Namen BMW an die „Bayerischen Flugzeug Werke“ (BFW) verkauft. Die bisherige BMW AG wird in „Süddeutsche Bremsen AG“ umbenannt. Seit dieser von Camillo Castiglioni initiierten Transaktion hat BMW seinen Stammsitz an der Lerchenauer Straße östlich des Münchener Flugplatzes Oberwiesenfeld.

Abbildung 1.1: Das BMW Logo im Jahr 1917

Im Jahr 1923 verlässt das erste BMW Motorrad das Münchener Werk. BMW übernimmt im Jahr 1928 die Fahrzeugwerke Eisenach in Thüringen und mit ihnen die Lizenz zum Bau eines Kleinwagens namens Dixi (1929).

Abbildung 1.2: Der Dixi 3/15 PS DA 2 im Jahr 1929

Der BMW 3/15 Typ DA 2 ist das erste Automobil aus Eisenach mit dem blau - weißen Emblem. Der 3/20 (1932) ist die erste komplette automobile Eigenentwicklung aus München. Innerhalb der BMW AG wird 1934 der Produktionszweig Flugantriebe als „BMW Flugmotorenbau GmbH“ verselbständigt.

Die BMW Motorradfertigung wird kriegsbedingt schrittweise nach Eisenach verlagert, die Automobilproduktion nach 81.228 produzierten BMW Automobilen seit 1929 zunächst eingestellt. Durch den 2. Weltkrieg werden die Produktionsstandorte von BMW stark in Mitleidenschaft gezogen. So wird im Jahr 1944 das Münchner Werk fast vollständig zerstört, wohingegen das Werk Allach nahezu unversehrt bleibt. Mitte 1945 wird BMW die Genehmigung zur Reparatur von US-Armee-Fahrzeugen in Allach erteilt, wofür im Gegenzug wieder Ersatzteile, Ackergeräte und Fahrräder produziert werden dürfen.

Motorräder, welche ebenfalls vom Produktionsverbot ausgenommen werden, können aufgrund der nachkriegsbedingten Anlaufschwierigkeiten von BMW vorerst noch nicht wieder produziert werden. Drei Jahre später, anno 1948, ist es soweit: in München wird das einzylindrige Motorrad R24 vorgestellt.

1951 präsentiert BMW sein erstes Nachkriegs-Automobil, den 501. Nach Verlusten mit großen Limousinen soll BMW im Jahr 1959 verkauft werden. Kleinaktionäre, Belegschaft,

Abbildung 1.3: BMW Motorrad R24 im Jahr 1948

Händler sowie die Bayerische Staatsregierung können dies jedoch verhindern. Der BMW 1500, ein kompakter, dynamischer Mittelklassewagen, verhilft der Firma im Jahr 1961 zum Durchbruch.

Der 1500 begründet eine neue Klasse von BMW Automobilen. Fast 145.000 BMW 1800 leisten als Weiterentwicklung des 1500ers ab 1963 zehn PS mehr. 1966 werden in München fast 140.000 BMW 2000 gebaut. Im Jahr darauf wird der 1800 TI ausgeliefert, welcher im Motorsport für Furore sorgt. Im Jahr 1965 wird die „BMW Triebwerk GmbH“ verkauft, wodurch sich BMW für 25 Jahre aus der Sparte Flugmotoren zurückzieht.

Abbildung 1.4: BMW 1500

1972 wird die „BMW Motorsport GmbH“ - oder kurz „BMW M GmbH“ - und 1973 die erste europäische Tochtergesellschaft in Frankreich gegründet. Nach der Gründung der „M GmbH“ steigt BMW in die Formel 1, die Raylle Paris - Dakar und in das 24 Stunden Rennen von Le Mans ein. Hierbei erlangt BMW nach herausragenden Siegen seinen legendären Ruf als Sportwagenhersteller.

Ebenfalls im Jahr 1972 läuft in München die Produktion einer grundlegend neuen Modellgeneration an: die 5er Reihe, welche bald darauf in das Werk Dingolfing verlagert wird. Der 5er fährt BMW aus der überwiegend sportlichen Marktnische heraus. Die Variantenvielfalt vom sparsamen 518i bis zum bärenstarken M5 definieren die 5er Limousine als Alleskönner und machen sie zum grandiosen Verkaufserfolg.

Die Einführung der 3er Reihe im Jahr 1975 stellt ebenfalls einen Meilenstein in der

Abbildung 1.5: BMW 5er, 1972

Abbildung 1.6: BMW 5er touring, 2002

Firmengeschichte dar. Anfänglich nur in wenigen Modellvarianten produziert, werden es im Laufe der Jahre über 30 verschiedene Modelle, vom 316i bis zum M3 mit gegenwärtig 343 PS. Die 3er Reihe entwickelt sich weltweit zum Erfolg. Bis zum Jahrtausendwechsel werden ca. sieben Millionen 3er verkauft. Das Erfolgsrezept des 3ers sind kompakte Maße gepaart mit souveränem Fahrverhalten, sowie auf Wunsch eine üppige Motorisierung.

Abbildung 1.7: BMW M3 Cabrio, 2002

Elegante, komfortable Limousinen mit durchdachten Detaillösungen auf höchstem Niveau sollen die BMW Produktpalette nach oben hin abrunden. Aus diesem Grund stellt BMW 1977 die 7er Reihe als drittes Standbein neben dem 3er und 5er vor. Bis heute setzen die 7er - obwohl vom Design her nicht immer unumstritten - Maßstäbe und gehören zu den meistverkauften Oberklasseautomobilen weltweit.

Mitte 1990 gründen BMW und Rolls-Royce plc. eine gemeinsame Gesellschaft namens „BMW Rolls-Royce GmbH“ (BRR) mit Sitz in Oberursel bei Frankfurt am Main, in welcher bald 1.000 Beschäftigte an Gasturbinen und Komponenten für den Flugtriebwerkbau arbeiten. BRR beginnt die Entwicklung einer Triebwerksfamilie für eine neue Generation von Regionalflugzeugen.1994 wird die Rover Group gekauft, was gleichzeitig die Geburtsstunde der „BMW Group“ ist.

Vielen Dank an Wolfgang Wallner für seine umfangreiche Recherche zu der historischen

Entwicklung von BMW.

1.2 Die BMW Group

Die Geschichte der BMW Group beginnt im Jahre 1994, als die Rover Group mir ihren vier Marken Rover, Mini, MG und Land Rover der BMW AG einverleibt wird. Durch diese Fusion steigt die BMW Group zu einem der weltweit größten Automobilhersteller auf. Doch die Freude über die Fusion währt nur kurz, da sich die marode Rover Group als finanzielles Fiasko für BMW erweist. Seit der Übernahme von Rover ist der Kurs der britischen Währung um fast die Hälfte gestiegen, und die Rover Sanierungskosten belasten zunehmend den gesunden Teil des Konzerns, die BMW AG, welche hervorragende Geschäftsergebnisse vorzuweisen hat.

Da die Rover Group so keine ökonomische Chance hat, verkauft die BMW Group die Marken Rover und MG für einen symbolischen Preis von zehn Pfund. Kurz darauf wird Land Rover ebenfalls veräußert. Lediglich die Marke MINI verbleibt innerhalb des Konzerns.

Abbildung 1.8: Corporate Identity der BMW Group

Nachdem das Jahr 2001 das bislang erfolgreichste in der Firmengeschichte war, setzt die BMW Group diesen Weg auch im Jahr 2002 konsequent mit insgesamt 99.464 Mitarbeitern und einem Gesamtumsatz von 11.599 Mio. Euro allein im 2. Quartal 2002 fort.Dies entspricht einer Steigerung von 8,9% gegenüber dem Vorjahresquartal bei einer insgesamt stagnierenden Weltmarktsituation. Somit erreicht der BMW Konzern als einziger Mehrmarken Automobilhersteller, welcher sich ausschließlich auf die Premium Segmente der weltweiten Automobil- und Motorradmärkte konzentriert, erneut Höchstwerte bei Umsatz und Ertrag. Zusätzlich rundet ab 2003 die prestigeträchtige Marke Rolls Royce das Produktportfolio der BMW Group im absoluten Top Segment ab. In Abbildung 1.9 und 1.10 sind zwei Automobile der aktuellen Fahrzeuglinien der Marken BMW bzw. MINI abgebildet.

Weitere Informationen über das Unternehmen seine Produkte sind im Internet auf den Seiten `http://www.bmwgroup.com` oder `http://www.bmw.de` zu finden.

Abbildung 1.9: 1er-BMW, 2007

Abbildung 1.10: MINI, 2007

1.3 Abteilung EA-403

Die Abteilung EA-403 des Forschungs- und Innovationszentrum der BMW AG beschäftigt sich vorwiegend mit der Absicherung der Elektronik des Antriebsstrangs. Dieses Teilsystem Antriebselektronik beinhaltet hauptsächlich die Motor- und Getriebesteuerung, sowie die Schnittstellen zu anderen beteiligten Systemen wie Zugangskontrolle, Anzeigeinstrument, Klemmensteuerung etc.

Die Funktionen dieser einzelnen, Komponenten und vor allem ihr Zusammenspiel wird hier ausgiebig getestet. Dafür wird auf Versuchsfahrzeuge und HiL (Hardware in the Loop) Prüfstände zurückgegriffen. Diese Prüfstände ermöglichen es dabei nicht nur die normale Fahrsituation darzustellen, sondern auch die Simulation von Extremsituationen und Fehlbedienungen unter realen Umgebungsbedingungen ist ohne größeren Aufwand realisierbar. Dies erlaubt eine nahezu vollständige Verifikation der Funktion, zur Erfüllung der an die Systeme gestellten Anforderungen, sowie detaillierte Rückschlüsse auf Ursachen bei Fehlverhalten. Diese Erkenntnisse können dann wieder in den Entwicklungsprozess einfliessen.

Teamleiter: Werner Wolf

1.4 Themenbereich und Aufgabenstellung des Praktikums

Mittels ECU-Test soll ein weitgehend automatischer Test der Kommunikation auf dem CAN und FlexRay Bus stattfinden. Dabei soll für das BMW Tool CARMEN/TopEd ein Modul erstellt werden das in der Lage ist, den Datenstrom zu analysieren und die realen Nachrichten mit der in der Bus-Datenbank vorgegebenen Parameter zu vergleichen. Zum Steuern der Analyse innerhalb von CARMEN und Auswertung der Ergebnisse wird die bereits vorhandene CARMEN-ECU-TEST Schnittstelle genutzt.

KAPITEL 2

Projekte

2.1 Einführung

2.1.1 Testautomatisierungstool ECU-TEST

> „ECU-TEST ist die Testautomatisierungs-Software für die Validierung eingebetteter Systeme im Automotive-Umfeld. ECU-TEST findet Anwendung beim Design, der Realisierung, der Durchführung und der Auswertung von Tests einschließlich der Generierung von intuitiven Testreports.“[Gmb]

Daher überrascht es nicht, dass im ECU-TEST Testklassen für die im Fahrzeug vorherrschenden Bus-Systeme wie CAN und FlexRay und Unterstützung für Simulationswerkzeuge wie Matlab-Simulink und CANOE implementiert sind. Durch die modulare Architektur lassen sich auch problemlos weitere Systeme unterstützen.

In ECU-TEST werden Versuche durch sequentielle Abfolge verschiedener Testschritte generiert. Natürlich ist es auch möglich, Schleifen und bedingte Anweisungen zu erstellen um komplexe Szenarien zu realisieren. Die Testsequenzen werden in einer grafischen Umgebung durch einfaches Drag und Drop erstellt. Dadurch ist eine vielfältige Funktionalität und einfache Bedienung, die nur minimale Einarbeitung benötigt, gewährleistet.

Außerdem enthält ECU-TEST zusätzlich einen Applikationsserver, womit es möglich wird Softwaretools und Schnittstellen auf entfernten Prüfstandsrechnern über das lokale Netzwerk anzusprechen.

In der Abteilung EA-403 wird ECU-TEST zur Absicherung an den HiL-Systemen und in Fahrzeugen genutzt. Dabei ist die gemeinsame Steuerung der verschiedenen Tools unter der einheitlichen Oberfläche von ECU-TEST der größte Vorteil. Dadurch ergibt sich auch

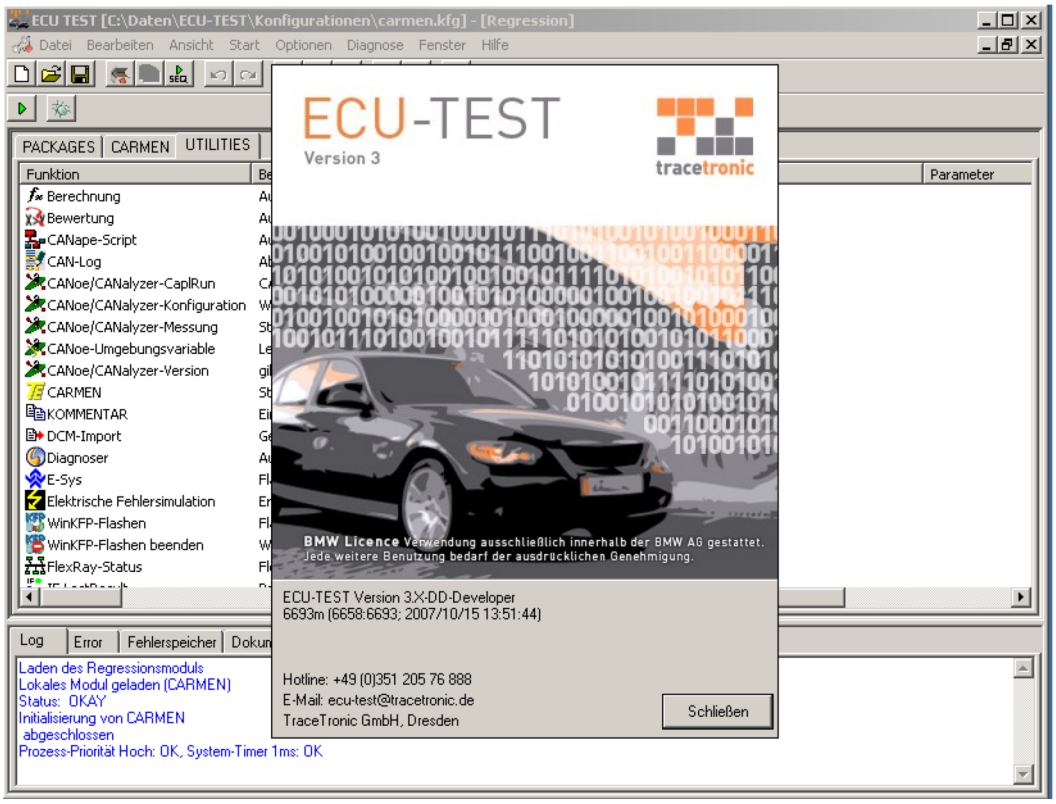

Abbildung 2.1: ECU-TEST Anwendung

die Möglichkeit der zentralisierten Auswertung, die den erforderlichen Arbeitsaufwand für den verantwortlichen Ingenieur stark verringert.

2.1.2 Buswerkzeug CARMEN

> "CARMEN steht für Car Measurement Environment. CARMEN ist ein offenes, erweiterbares Werkzeug (genauer: eine Sammlung von Werkzeugen), um Messungen am Bussystem eines Fahrzeugs durchzuführen. Auch die Stimulation wird unterstützt, so dass auch Simulationsaufgaben abgedeckt werden."[Ing]

Mit CARMEN -TopEd kann man, ähnlich Simulink, die verschiedenen Module in einem Netzplan anordnen. Standartmäßig sind zahlreiche Module wie ein Paketzähler, Paketlogger, Filter und so weiter vorhanden. Diese werden so angeordnet, dass entsprechend ihrer Funktion die Ein- und Ausgänge verbunden werden können.

Um spezielle Funktionen zu realisieren, ist es möglich CARMEN durch eigene Module zu erweitern. Diese können C++ oder Visual Basic implementiert werden. Außerdem

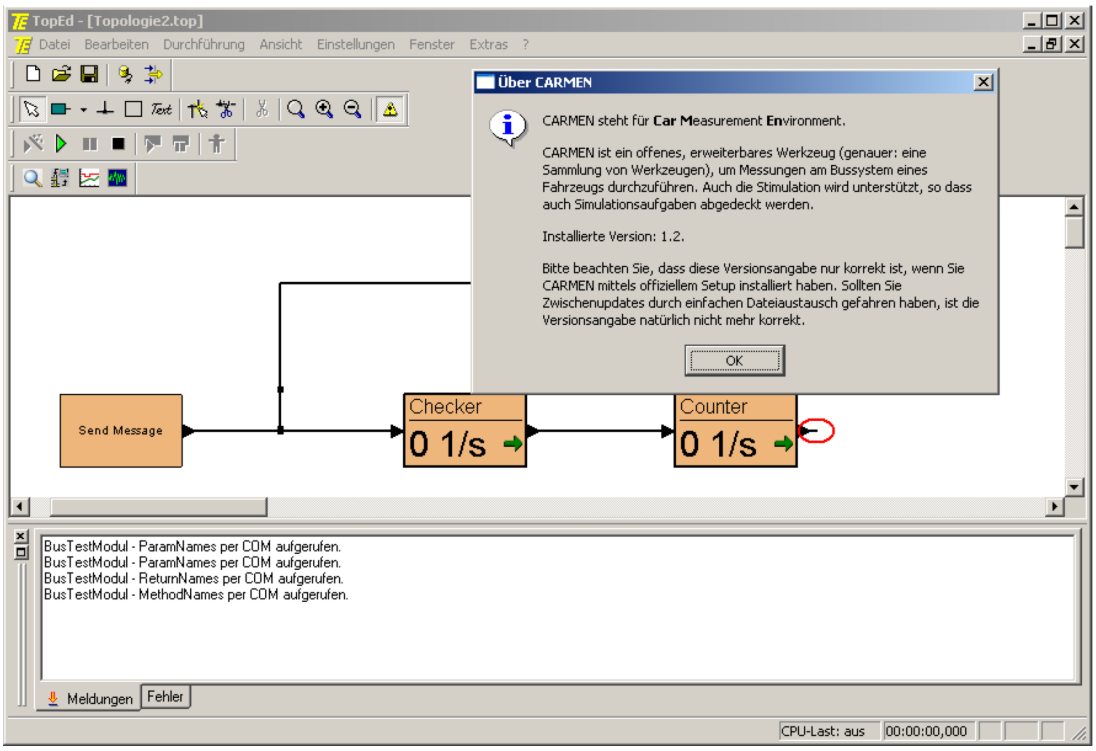

Abbildung 2.2: CARMEN - TopEd Anwendung

besitzt CARMEN eine COM- Schnittstelle zur Automatisierung, womit grundlegende Programmfunktionen, wie zum Beispiel ”Konfiguration laden”, ”Test starten” et cetera, aufgerufen werden können.

Vom Benutzer erzeugte Module müssen vorgegebene Methoden integrieren und arbeiten eventgesteuert. So wird bei jedem Paket eine entsprechende Methode aufgerufen, die dieses anschließend verarbeitet. Des Weiteren besteht die Möglichkeit Timer zu definieren, welche dann gemäß ihrem Intervall eine Methode aufrufen. Im Rahmen meiner Studienarbeit habe ich bereits eine Schnittstelle zwischen ECU-TEST und CARMEN geschaffen. Diese nutzt die Automatisierungs Schnittstelle von CARMEN und erlaubt den Zugriff auf Methoden und Attribute entsprechender Module. Diese Module können durch festgelegte Methoden ihre bereitgestellten Schnittstellen publizieren, die der Nutzer dann in ECU-TEST dargestellt bekommt.

2.1.3 Einführung „Hardware in the Loop“ (3 Wochen)

Im Teilsystemtest wird reale Hardware und Software in einer Umgebung aus simulierten und realen Komponenten erprobt (HiL). Diese Umgebung gestattet vielfältige, gezielte Eingriffe in das System, vom simulierten Fahren über veränderte Sensorwerte bis hin

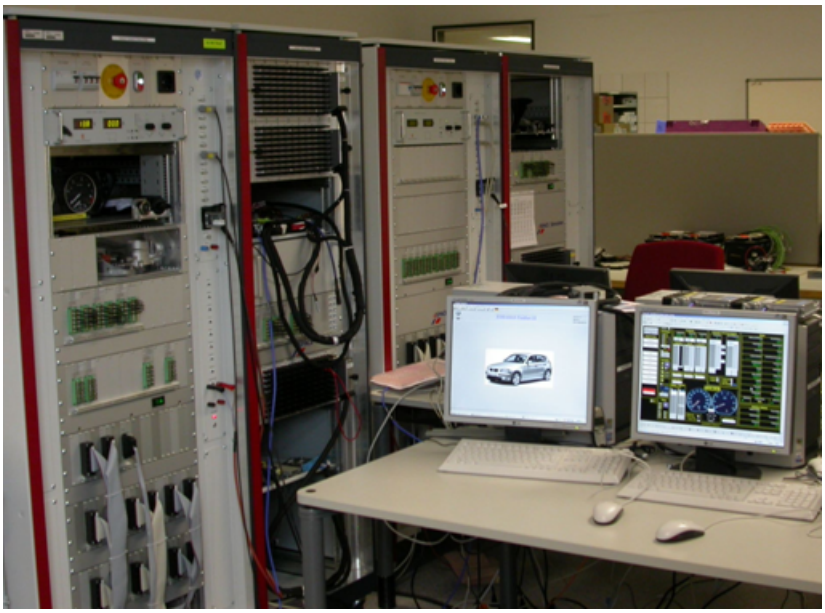

Abbildung 2.3: HiL-Arbeitsplatz

zu verändertem Teilsystemverhalten. Bei der Antriebsabsicherung werden hauptsächlich Motorsteuergeräte und Getriebesteuerung im Verbund erprobt. Die HiL-Testumgebung enthält dabei folgende Komponenten:

- reale Steuergeräte (Bordnetzträger mit Kombi-Intrument, CAS, Gateway)
- Realkomponenten wie Einspritzventile, Drosselklappe, VVT-Steller, etc.
- Simulationsrechner mit physikalischen Modellen
- CAN Rechner mit Restbussimulationen
- Messrechner mit Steuer und Messsoftware(ECU-TEST , CARMEN , etc.)

Das Testen von Systemen erfolgt durch die Anreizung mit bestimmten Parametern (Realbedienung, Umweltbedingungen, etc.) sowie der stetigen Messung der beeinflussten Systemgrößen. Damit ist es möglich, die Funktionalität der Antriebselektronik für die jeweiligen Anwendungsfälle sicherzustellen. Ein Beispiel für einen solchen Anwendungsfall ist die Abschaltung des Verbrennungsmotors bei dreifacher Betätigung des Start-Stopp Tasters (SST). Weiterhin bietet der HiL die Möglichkeit, entsprechende Messsoftware (CANoe, INCA, CARMEN, EDIABAS, etc.) vorausgesetzt, schon im Rahmen des Testsvorgangs mögliche Fehlerursachen aufzudecken.

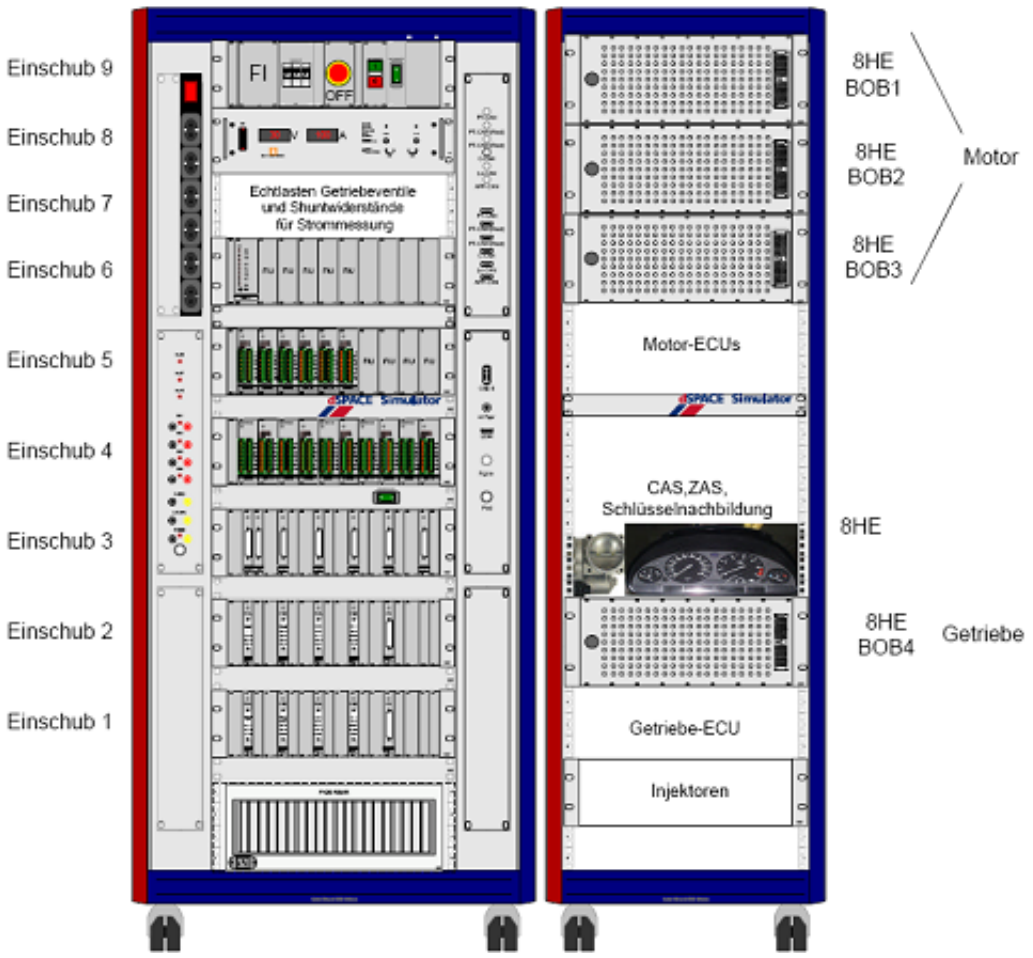

Abbildung 2.4: dSpace Simulator

Die Einarbeitung begann mit einfachen Tätigkeiten am HiL. Bei diesen Hilfstätigkeiten erfolgte das Kennenlernen der Systemstruktur, des Systemaufbaus sowie der folgenden Programme und Vorgänge:

- Steuerung des simulierten Fahrbetriebs mit Controldesk
- Messen in der Restbussimulation und mit CARMEN
- Modellwechsel am HiL
- Automatisierung mit ECU-TEST

2.2 CARMEN -Modul (20 Wochen)

2.2.1 Anforderungen

Es sollte ein MODUL für das Buswerkzeug CARMEN realisiert werden, um den CAN und FlexRay Bus zu überprüfen. Dabei interessiert zum einem die Aufnahme des Timings und des DLC's konfigurierter Nachrichten, sowie die Fehlerkontrolle eines vorhandenen Alive-Counters oder einer CRC-Prüfsumme bzw. beim CAN Bus sollte auch das Testen der BMW-Checksumme möglich sein.

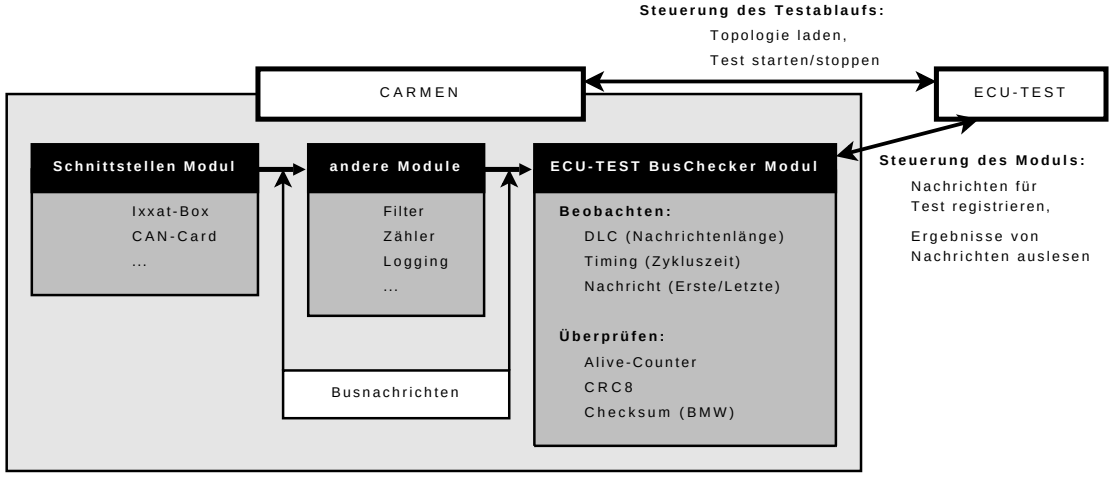

Abbildung 2.5: schematische Darstellung des Moduls und der Anbindung an ECU-TEST

Die Konfiguration des Moduls, also der zu beobachtenden bzw. zu testenden Nachrichten, soll über die Schnittstelle zu ECU-TEST erfolgen, damit ein automatisierter Test möglich ist. Damit entfällt die Generierung des Reports und Auswertung der einzelnen ermittelten Werte in den Aufgabenbereich von ECU-TEST und muss nicht vom CARMEN Modul selbst erledigt werden.

2.2.2 Entwurf

Das Modul besteht aus einer zentralen Klasse die für die Kommunikation nach aussen verantwortlich ist. Sie realisiert daher die API-Schnittstelle zu CARMEN TopEd und die COM-Schnittstelle zu ECU-TEST . Außerdem verwaltet sie eine Liste aller registrierten Nachrichten und hält eine Referenz auf die Klassen, die die einzelnen Nachrichtentest/-beobachtungen durchführen.

Die Nachrichtenliste beinhaltet die Informationen für alle Busnachrichten die einer Überprüfung unterzogen werden sollen. Außerdem sind auch für manche Tests zusätzliche

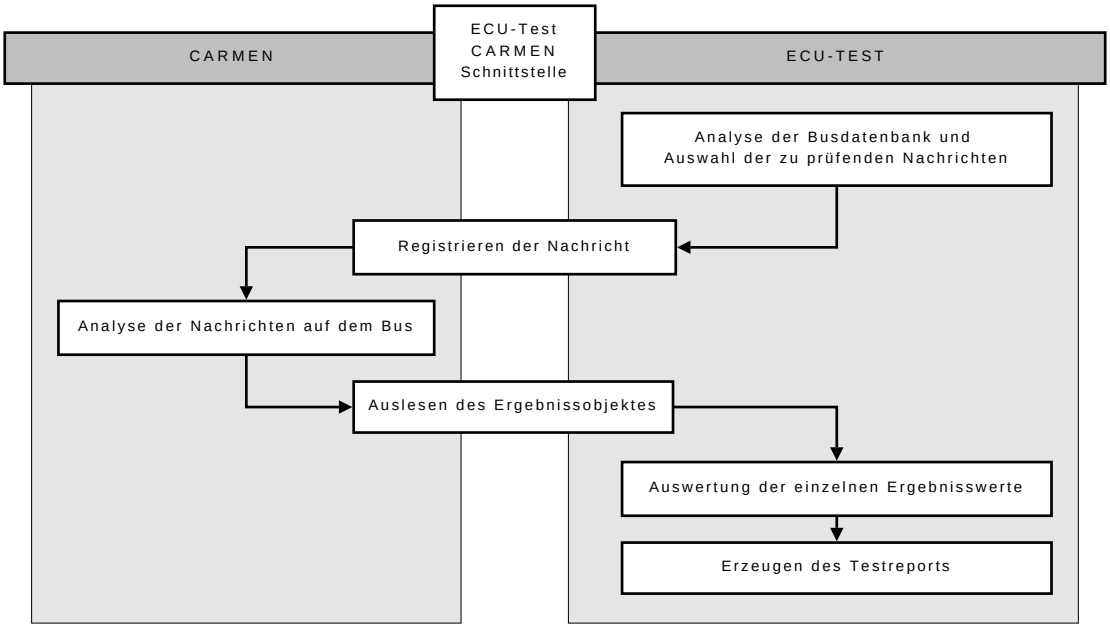

Abbildung 2.6: Aufgabenverteilung zwischen CARMEN und ECU-TEST bei der Konfiguration der Busanalyse

Informationen, wie die Position der Checksumme oder der Typ des Alive-Zählers nötig, diese werden ebenfalls mit in der Liste gespeichert.

Nachdem die zu prüfenden Nachrichten und ihre angesetzten Tests über ECU-TEST im Modul registriert wurden, also in der Nachrichtenliste gespeichert sind, laufen die in Abbildung 2.8 dargestellten Aktionen ab, wenn eine Bus-Nachricht am Moduleingang eintrifft. Grundsätzlich funktioniert dies eventbasiert, das bedeutet, dass immer wenn eine Nachricht den Eingabeport erreicht, die entsprechende Event-Methode im Modul gerufen wird, welche gegebenenfalls die Verarbeitung in Gang setzt.

Um die Funktionalität des Moduls, also die Bandbreite der Angebotenen Checks, leicht erweitern zu können, wird für jeden Test eine eigene Test-Klasse implementiert. Diese erben von einer zentralen Check Klasse die sinnvollen Unterstützungsfunktionen f“ur die Checks.

Die Performance der Datenübertragung über COM ist im Vergleich mit der restlichen Datenverarbeitung, aufgrund der damit zusammenhängenden komplexen Kommunikationsvorgänge eher gering, daher sollte die nötige Kommunikation möglichst mit so wenig wie möglich Anfragen auskommen. Deshalb werden die Testergebnisse in Klassen zusammengefasst. Dies hat zwar den Nachteil, dass dem Empfänger die Struktur der Result-Klasse bekannt sein muss, aber er erhält auch alle wichtigen Daten auf einmal. Die Typebibliothek die diese Struktur beschreibt, wird nach der Compilierung auch mit in die entstehende DLL integriert.

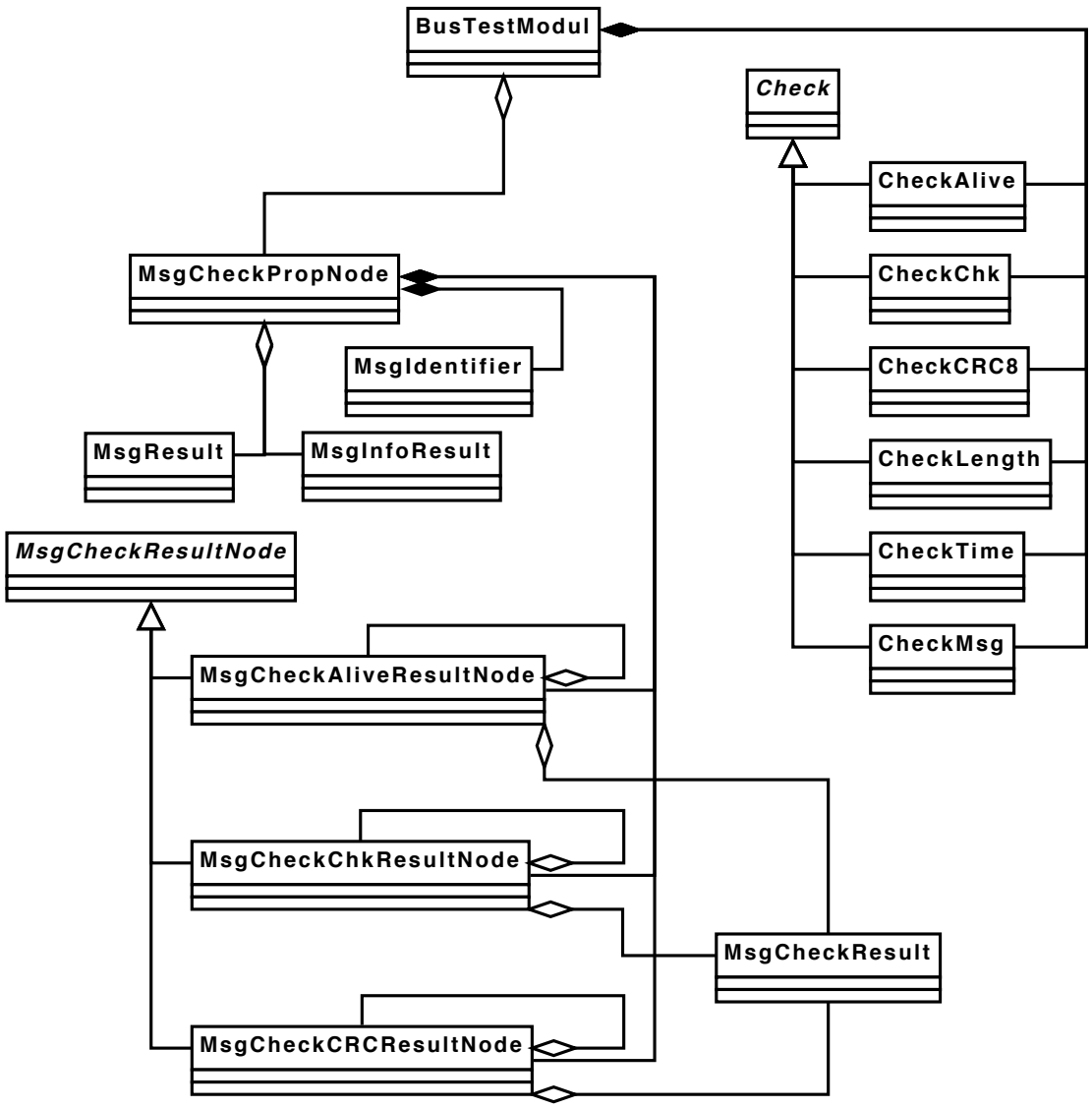

Abbildung 2.7: UML des CARMEN Moduls

2.2.3 Realisierung

Die Realiserung des Modules wurde in C++ (Microsoft Visual Studio 2005) vorgenommen, da CARMEN selbst mit Visual Studio erstellt wurde. Alternativ ist es möglich Module in Visual BASIC zu programmieren, allerdings geschieht die Anbindung dann über COM. CARMEN schickt die Daten über COM an das Modul, da jedoch die Auswahl der Nachrichten erst nach dem Event in dem Modul stattfindet, wäre dies von der Performance problematisch geworden, da die Übergabe der Nachrichten-Objekte über COM im Vergleich zu der restlichen Verarbeitung sehr langsam ist. Bei mehreren Busnachrichten mit einer

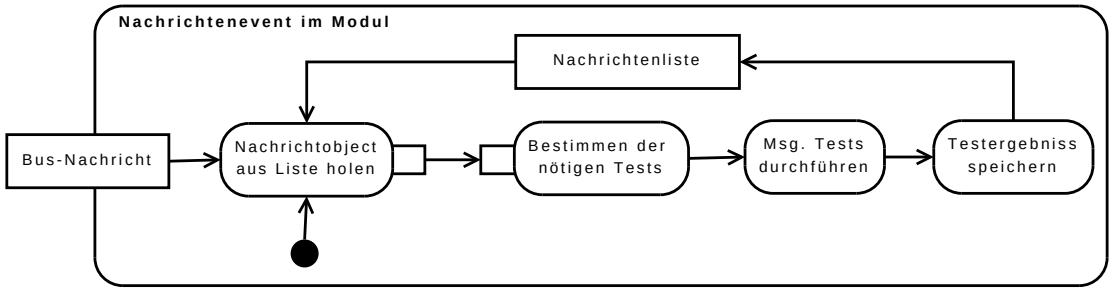

Abbildung 2.8: Ablauf beim Eintreffen einer Nachricht am Eingang des CARMEN Moduls

Zykluszeit von ein paar Millisekunden ist leicht zu erkennen, dass dies nicht realisierbar gewesen wäre.

Im folgenden werden die grundsätzlichen Elemente der Realisierung näher beschrieben.

2.2.3.1 Nachrichtenliste

Die Nachrichtenliste ist als Array implementiert, wobei die Nachrichten-ID dem Array Index entspricht. Dadurch ist ein zügiger Zugriff auf das richtige Object gewährleistet. Da allerdings mehre Busse von einem einzigen Modul analysiert werden und auch Busse existieren, bei denen Nachrichten nicht nur an der ID eindeutig zuzuordenen sind, besitzt ein Nachrichtenobject eine Instanz eines Nachrichtenidentifiers, der in der Lage ist, sich mit anderen Identifier-Instanzen zu vergleichen. Die Menge der Nachrichten mit einer identischen numerischen Nachrichten-ID ist durch eine einfache verknüpfte Liste realisiert, da ihre Menge in den meisten Fällen recht übersichtlich bleibt. Damit führen zu einem wenige Suchschritte zum Erfolg und zum anderen verhält sich die Anzahl sehr dynamisch was bei der Ablage berücksichtigt werden muss.

Die Anzahl der Tests pro Nachricht kann sehr unterschiedlich sein, daher werden die Daten dafür teilweise in einer einfachverlinkten Liste vorgehalten. Für die Beobachtung des Timings, DLC und der Nachricht selbst ist dies natürlich nicht nötig, daher werden die Ergebnisse direkt in dem Nachrichten-Daten-Objekt gespeichert.

2.2.3.2 BusTestModul

Die Klasse „BusTestModul“ ist die Hauptklasse des Moduls. Sie ist von der Standart-Modul-Klasse von CARMEN abgeleitet und stellt die Schnittstellen und Steuerung des Moduls bereit. Durch Vererbung erhält die Klasse alle nötigen Methoden, um auf Ereignisse von CARMEN TopEd zu reagieren. Darunter fällt das Eintreffen einer Busnachricht am Eingang genauso, wie die regelmäßige Abarbeitung von Timer-Events.

Auch die Methoden zur Kommunikation über COM, also in erster Linie mit ECU-TEST , sind in der BusTestModul-Klasse implementiert. Dafür existieren die Set- und Get-Methoden der Attribute und einzelne Methoden, zum Beispiel zum Abruf der Ergebnisobjekte der einzelnen Checks. All diese Schnittstellen mussten in der IDL[1] formal definiert sein, um korrekt über COM zur Verfügung gestellt zu werden.

Leider ist die Kommunikation über COM im Vergleich zur restlichen Datenverarbeitung des Moduls vergleichsweise langsam. Damit die Verzögerung gering bleibt, sind die Testergebnisse in Ergebnis-Klassen „MsgCheckResult“, „MsgInfoResult“ und „MsgResult“ zusammengefasst. So wird das gesamte Datenpaket mit einer Anfrage übertragen und es müssen nicht alle einzelnen Elemente separat abgefragt werden. In ECU-TEST wird die Datenklasse einfach in ein Dictionary, ein assoziatives Array, umgewandelt.

2.2.3.3 Checks

Damit das Modul unkompliziert um weitere Checks erweitert werden kann, wurden diese jeweils als einzelne Klassen implementiert, die alle Basisfunktionen von der abstrakten Klasse „Checks“ erben. Im folgenden wird kurz beschrieben was die einzelnen Checks beinhalten. Obwohl alle Klassen mit „Check..“ beginnen sind streng genommen nicht alles Überprüfungen, sondern teilweise auch Beobachtungen. Diese beiden Arten unterscheiden sich eigentlich nur in der Ergebnisauswertung. Bei Überprüfungen wird eine Eigenschaft der Nachrichten getestet und die Anzahl der Fehlerfälle gezählt. Bei Beobachtungen wird nur der Maximal- und Minimalwert aufgenommen.

CRC-Überprüfung Für die Überprüfung des CRC wird der allgemein bekannte und standardisierte CRC-8 Allgorithmus genutzt. Dabei wird die Nachricht durch ein Generatorpolynom dividiert. Um diese zu beschleunigen, wird auch hier eine Lookup-Tabelle genutzt. Diese enthählt für jedes der möglichen 256 Bytes den zugehörigen CRC-Wert.

Zusätzlich wird bei BMW für jede Nachricht eine ApplicationID festgelegt, diese wird als Ausgangspunkt für die CRC-Berechnung genutzt.

Checksum-Überprüfung Zur Berechnung der Checksumme, werden zuerst in der Nachricht die Bits, die die Checksumme bilden durch 0 ersetzt, damit diese keinen Einfluss auf die Berechnung haben. Danach werden die Nachrichten-Bytes aufaddiert. Diese 16Bit Summe wird nun in eine 8Bit Checksumme gewandelt, indem sie in zwei 8Bit Teile zerteilt (jeweils die vorderen und hinteren 8Bit) wird und diese dann addiert werden. Bei Erzeugung der 4Bit Checksumme wird das Vorgehen noch einmal wiederholt, nur diesmal addiert man die beiden 4Bit Teile der 8Bit Summe.

1 Interface Description Language (eine Programmiersprachen unabhängige Interface Beschreibung)

Das Problem der Checksumme liegt an ihrer Position in der Gesamtnachricht, denn es ist zulässig, dass die Summe so positioniert ist, dass sie nicht mit den Bytegrenzen abschließt, was den Aufwand zu ihrer Extraierung ein wenig erhöht.

Alive-Überprüfung Es gibt Nachrichten in denen ein 8bit oder 4bit Signal als Alive-Zähler fungiert. Dieser wird bei jedem Senden der Nachricht auf dem Bus um eins incrementiert. Allerdings ist der Wert 0xF (bei 4bit) beziehungsweise 0xFF (bei 8bit) als Fehlerfall definiert und kein gültiger Signalwert.

In der Nachricht wird die Position, an der sich der Zähler befindet extrahiert und mit dem Increment des Zählerstandes der vorhergehenden Nachricht verglichen. Im Fehlerfall wird dieser registriert und eine Fehlermeldung gespeichert, ausserdem wird die Anzahl der Fehler um eins erhöht. In beiden Fällen wird natürlich der aktuelle Zählerstand für die folgende Nachricht gespeichert.

Timing-Beobachtung Vorwiegend bei zyklischen Nachrichten muss geprüft werden, ob bei allen Nachrichten die Zykluszeit korrekt eingehalten wurde. Zusätzlich kann dieser Check auch genutzt werden, um das Zeitverhalten ereignisgesteuerter Nachrichten zu kontrollieren, wenn das Resultat vor dem Auslösen des Ereignisses resettet wird.

Das Funktionsprinzip dieses Moduls besteht darin, dass der von CARMEN genutzte Timestamp zur Berechnung der Zeitdifferenz genutzt wird und beim Unter- oder Überschreiten der Extrema diese neu gesetzt werden. Es ist zu beachten, dass CARMEN mit Zeitstempeln in Mykrosekunden arbeitet und daher auch die Differenzen in Mykrosekunden gespeichert werden.

DLC-Beobachtung Beim DLC, also der Länge der Nachrichten, wird ähnlich wie beim Timing der von CARMEN weitergereichte Parameter gespeichert und gegebenenfalls das Minima oder Maxima überschrieben.

Nachricht-Beobachtung Für das Auslesen von einzelnen Nachrichten interessiert der erste und der letzte Wert der Nachricht. Daher wird nach dem Reset dieses Checks die aktuelle Nachricht als Hex-String einmal als Initialwert und als aktueller Wert gespeichert. Bei jedem weiteren empfangen der Nachricht wird der aktuelle Wert überschrieben.

2.2.4 Test

Um die korrekte Funktionalität zu überprüfen, wurde ein spezielle DBC-Datei[1] erstellt und mit dem „SendMessage“ Modul von CARMEN entsprechende Nachrichten auf dem Bus generiert. Dies hat den Vorteil, dass alle Parameter sehr einfach manipuliert werden können. So ist es es zum Beispiel möglich, die Position des Alive Counters oder CRC an die ungüngstigsten Position im Nachrichtenframe zu setzen.

In dem „SendMessage“ Modul können ausschließlich CAN Nachrichten erzeugt werden. Für die Überprüfung der Korrektheit der FlexRay Nachrichten konnte das Simulationsmodul der IXXAT[2] Software verwendet werden. Zwar kann man mit diesem nur bedingt die gesendeten Nachrichten beeinflussen, aber die Allgorithmen für den Test der Nachrichten beziehen sich ausschließlich auf deren Inhalt und sind damit identisch zur Implementierung bei CAN. Es unterscheidet sich nur die Bestimmung der vorgesehenen Tests für die Nachricht, da bei Flexray zusätzlich zur Nachrichten ID die Slot ID zur eindeutigen Differenzierung der Nachrichten herangezogen wird.

2.2.5 Anwendung

Die Steuerung des CARMEN Moduls erfolgt für den Endnutzer über ECU-TEST . Dabei ist die Schnittstelle für die Version 3 und 4 sehr unterschiedlich implementiert.

ECU-TEST 3 Hier wird das Modul recht direkt gesteuert da der Nutzer unmittelbar Zugriff auf die Methoden und Attribute erhält. Diese werden in einer Liste dargestellt und können als Lese- beziehungsweise Schreibschritt in den Test eingefügt werden. Auch Methoden werden mit aufgelistet und können als Testschritte ausgeführt werden. Beim Auslesen von Attributen als Rückgabewert und Parameter sind zwar nur einfache Datentypen wie Integer oder Strings möglich, aber ein Methodenaufruf kann sogar Klassen als Ergebnis haben.

Da der Nutzer damit zwar einen sehr transparenten Zugang zum CARMEN Modul hat, erhält er allerdings auch viele Freiheiten und kann damit viel falsch machen. Um dieses Problem zu verringern, sind im Modul ebenso Methoden implementiert die einen Hilfetext ausgeben, der von ECU-TEST dargestellt wird.

ECU-TEST 4 In dieser Version von ECU-TEST gibt es eine spezielle Schnittstelle zum Ansprechen von Tests zur Überprüfung von Busnachrichten. Auch das Ansprechen externer Tools wurde grundlegend geändert. Sie werden nun als Tools angesprochen, die

1 Busdatenbank für CAN-Busse
2 `http://www.ixxat.com`

unterschiedliche Ports besitzen, die wiederum unterschiedlichen Typs sein können. Für die Analyse der Busnachrichten wird ein Port des Typs „24SACCESS“ benötigt. Danach muss dieser Port nur noch einem Bus zugeordnet werden, damit die Optionen zur Analyse der Nachrichten in der Ansicht verfügbar sind.

Bei beiden Versionen von ECU-TEST ist die Handhabung der Busanalyse sehr ähnlich. Zuerst müssen die Nachrichten für die Checks im Modul registriert werden. Wenn genügend Nachrichten in CARMEN durchgelaufen sind, kann das Ergebnis-Objekt mit der entsprechenden Methode ausgelesen werden. Entweder durch das direkte Aufrufen der Methode in ECU-TEST 3 oder durch das Einfügen des Lese-Ergebnis-Schrittes in ECU-TEST 4.

2.3 Packagegenerator (4 Wochen)

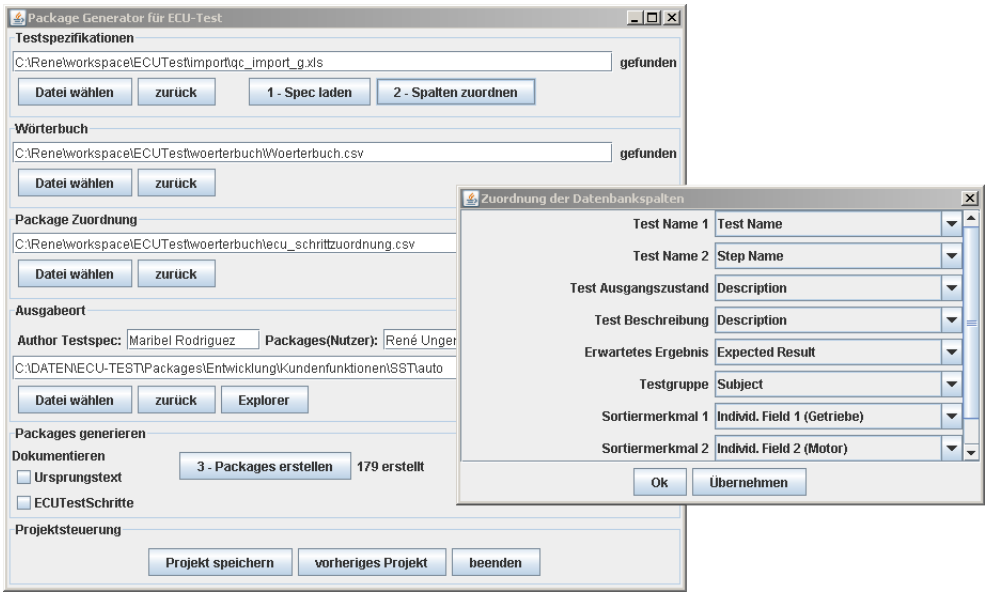

Abbildung 2.9: Package Generator und Zuordnung der Eingabedatenbank

Der Packagegenerator wird genutzt, um aus einer Testfallspezifikation eine Testpackage für ECU-TEST zu erstellen. Die Testspezifikation muss dafür als Excel Datei oder CSV[1] vorliegen. Mit Hilfe eines Wörtebuches, das ebenfalls in Excel editiert werden kann, wird diese in ein Package umgewandelt, dass in ECU-TEST eingelesen und ausgeführt wird.

Motivation für das Vorgehen ist die Tatsache, dass bei neuen Baureihen immer wieder ähnliche Funktionen überprüft werden müssen. Daher erspart diese automatische Generierung von Testpackages eine grosse Menge Arbeit, die sonst immer wieder neu geleistet werden müsste. Auch bei Änderung der Testfallspezifikation muss gegebenenfalls nur

1 Comma seperated file

das Wörterbuch geringfügig aktualisiert werden, um die Vielzahl an Packages erneut zu erstellen.

2.3.1 Funktionsprinzip

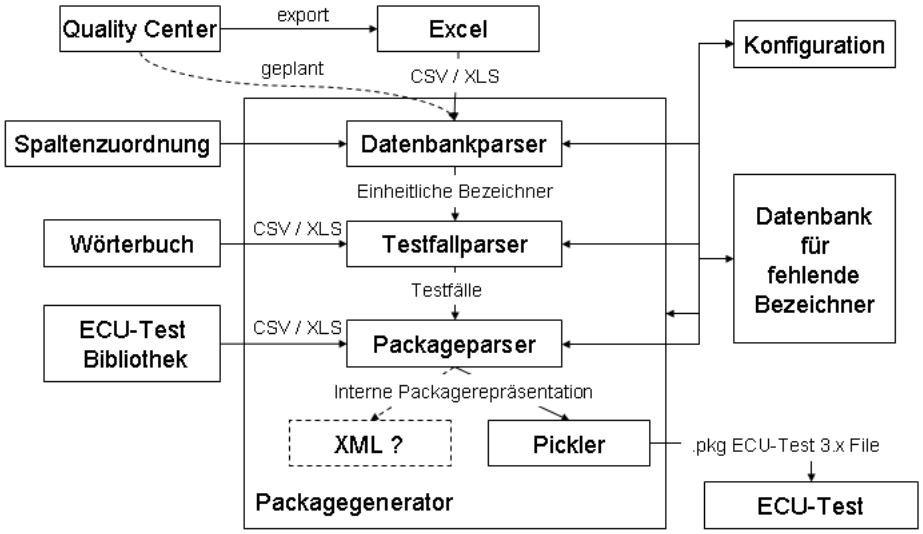

Abbildung 2.10: Die wichtigsten Funktionsmodule des Packagegenerators und der Informationsfluss vom Testfall in Quality Center zum ECUTest-Script (.pkg)

In Abbildung 2.11 ist der funktionelle Aufbau des Package-Generators dargestellt. Dabei bedarf die Verarbeitung der Testspezifikation einer näheren Erläuterung. Die Testspezifikation ist in einer Tabelle vorhanden, deren Spalten zugeordnete Informationen wie Bezeichnung, Ausgangszustand, Anweisungen, erwarteter Endzustand und so weiter jeweils als Listen gespeichert sind.

In einem ersten Schritt wird der Prosa in den jeweiligen Spalten mit Hilfe der Wörterbuch-Datei ein eindeutiger Bezeichner zugeordnet. Je eindeutiger die Bezeichnung in der Prosa schon vorgenommen worde, desto einfacher ist die Übersetzung. Im optimalen Fall sollte die Beschreibung schon mit Hilfe dieser eindeutigen Bezeichner vorgenommen werden, damit eine erste Umwandlung eigentlich nicht mehr nötig ist. Ausserdem ist im Wörterbuch jedem dieser Bezeichnern ein ECU-TEST Package zugeordnet. Die Packages sollten möglichst Basisfunktonen beinhalten, damit sie einfacher zu den in der Testfallspezifikation beschriebenen Bedingungen und Aktionen zugeordnet werden können. Danach setzt der Packagegenerator aus diesen Basispaketen ein Paket pro Testfall zusammen, was damit entsprechend der Spezifikation den Testfall beschreibt.

Im Rahmen des Praktikums wurde der Packagegenerator um weitere kleine Funktionen erweitert. So ist es jetzt auch möglich, Basispakete mit mehr als einem Paramter aufzurufen oder einem Bezeichner mehrere Packages zuzuordnen. Ebenso wurde ein Grossteil

der Basispakete geändert und damit war es nötig, das bestehende Wörtebuch für die Testspezifikation des Start-Stopp-Tasters anzupassen und weitere Basispakete zu erstellen, um benötigte Funktionen bereitzustellen. Dies zog natürlich mehrere Test am HiL nach sich um die Funktion der neuen Pakete und Features zu überprüfen.

2.4 Filterviewer (3 Wochen)

Der FilterViewer ist ein kleines Tool, das ursprünglich von Rene Unger für EA-403 entwickelt wurde, um in größeren Text-Dateien nach Zeichenfolgen zu suchen und ein entsprechend gefiltertes Ergebnis zu erzeugen und anzuzeigen. Es zeichnet sich besonders durch seine einfache Bedienung aus.

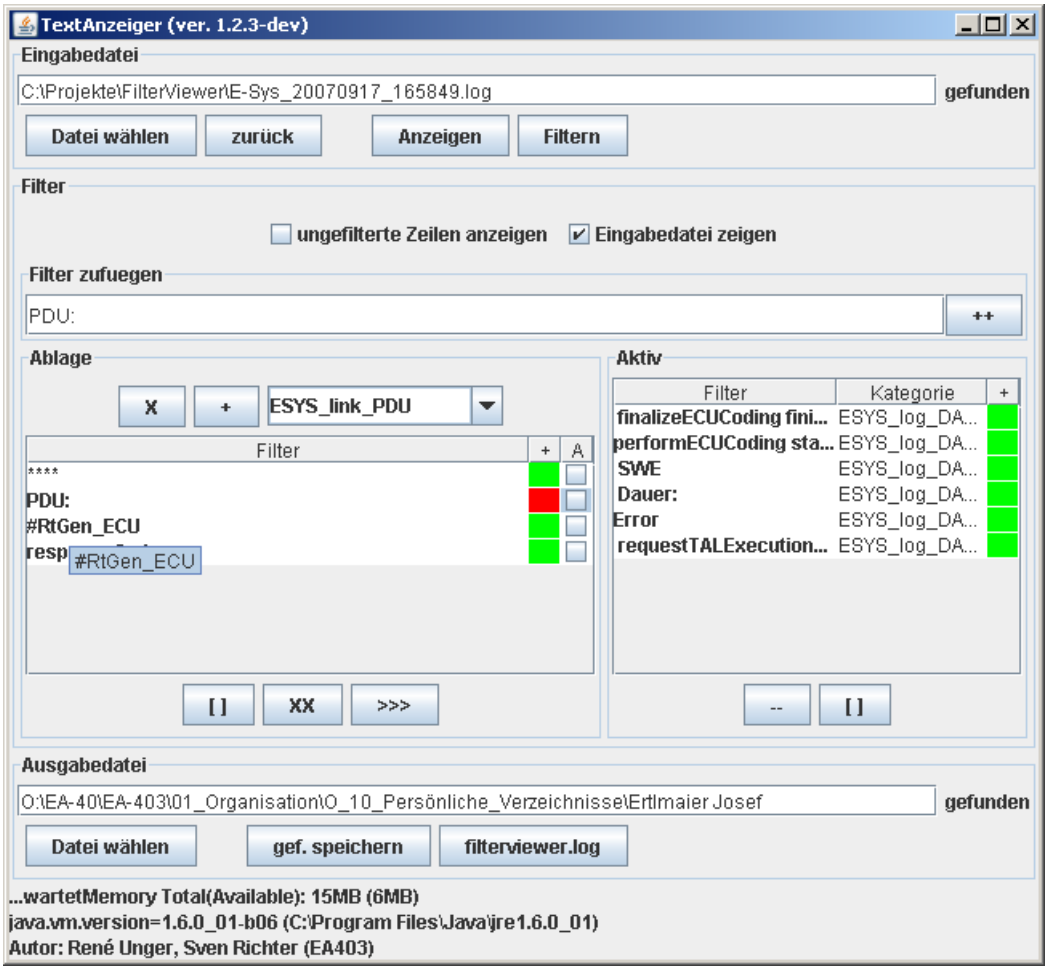

Abbildung 2.11: FilterViewer - Hauptfenster

Allerdings entstand schnell der Bedarf nach weiteren Funktionen, deren Implementierung im Folgenden beschrieben werden soll.

Anforderungen:

- Filter in Gruppen zusammenfassen
- aktivieren/deaktiviern von Filtern ohne sie zu löschen
- einfaches aktivieren/deaktivieren von ganzen Filtergruppen
- Verbindung zwischen Darstellung der Ursprungsdatei und dem Filterergebnis, doppelklicken einer Zeile in dem einen Fenster führt zur Zentrierung dieser Zeile im anderen Fenster
- kleine kosmetische Eingriffe an der Bedienoberfläche, wie zum Beispiel ein größeres Feld zur Eingabe bzw. Darstellung des Filter-Strings etc.

2.4.1 Realisierung

Damit die komplexeren Anforderungen, die an die Filter gestellt werden realisierbar wurden mussten diese als komplett neue Klassen implementiert werden. Damit können diese auch gleich serialisiert gespeichert werden und wissen nun selbst, zu welcher Gruppe sie gehören und ob sie aktiviert sind. Jeder Filter ist nun fest in einer Gruppe und damit ist es möglich, ganze Gruppen auf einmal anzuwenden. Somit besteht die Möglichkeit der Deaktivierung eines Filters, ohne ihn löschen zu müssen. Der Vorteil offenbart sich vor allem wenn nach längere Zeihenfolgen gesucht wird.

Die Anzeige der Ausgangsdatei und des Ergebnisses der Filterung wird nun als Tabelle, anstatt einer Liste realisiert, um die Verlinkung zwischen beiden Fenstern zu ermöglichen. Damit kann die jeweils andere Darstellung auf die gewählte Zeile der primären Ansicht zentriert werden.

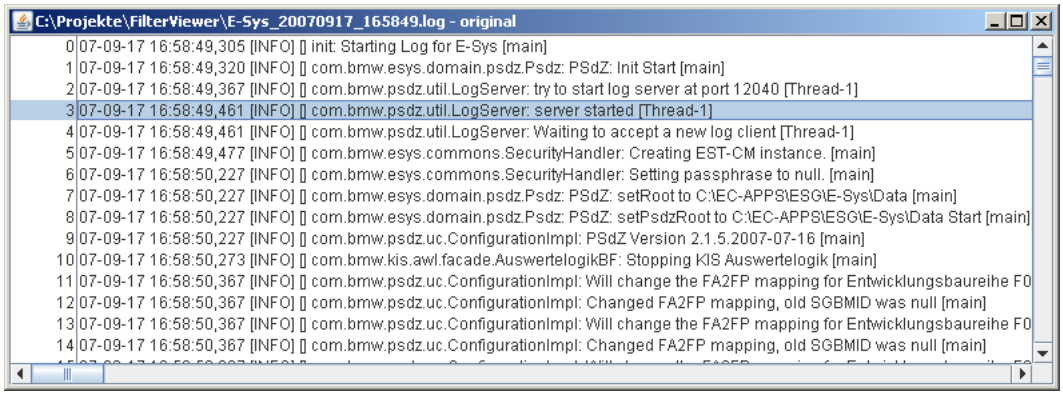

Abbildung 2.12: FilterViewer - Ausgabefenster

Die Fensteraufteilung wurde an die neuen Funktionen angepasst. So wurde die Darstellung der einzelnen Zeichenfolgen der Filter um ihre Eigenschaften erweitert. Eine einfache Veränderung in der List durch den Nutzer ist selbstverständlich ebenso möglich. Beide Filterarten (negative und positive) werden nun in der gleichen Liste dargestellt und farblich markiert. Der gewonnene Platz ermöglicht die Darstellung der einzelnen Filtergruppen. In den Listen kann auch ein oder mehrere Filter markiert werden. Mit Hilfe von Knöpfen am unteren Rand sind die Attribute veränderbar.

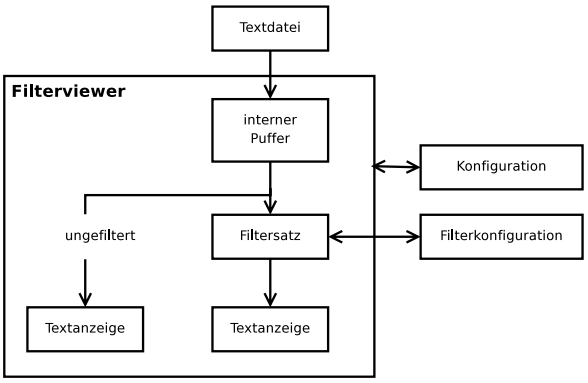

Abbildung 2.13: FilterViewer - Aufbau/Funktionsprinzip

An dem grundsätzlichen Funktionsprinzip wurde ansonsten nichts weiter geändert. Es wird eine Quelldatei in einen Puffer eingelesen und danach zeilenweise alle aktivierten Filter angewendet. Je nach Wahl des Nutzers wird eine Zeile zur Ergebnisdarstellung überführt, wenn mindestens ein (positive Filter) oder kein einziger (negative Filter) Filterstring positiv angewendet werden kann.

2.5 Weitere Aufgaben (2 Wochen)

2.5.1 Durchführung einer Schulung CARMEN und ECU-TEST

Damit die Benutzung von CARMEN und dem Bustes-Modul mit ECU-TEST für alle Mitarbeiter erleichtert wird, wurde eine Schulung durchgeführt. Dabei wurden im speziellen folgende Themen angesprochen.

- Einführung in das Buswerkzeug CARMEN . Erstellung eine Topologie und der zur Analyse nötigen Module. Möglichkeiten zur Beobachtung der Busnachrichten innerhalb von CARMEN .

- Kennenlernen von ECU-TEST 4 und der Verwaltungsmethodik von externen Softwarekomponenten sowie deren Konfiguration.

- Konfiguration der CARMEN Toollib und Verknüpfung mit dem Bustest-Modul in CARMEN . Erstellung der Testpackages und Bewertung der erzeugten Dokumentation.

2.5.2 PDXScanner

Bei dem PDX Scanner handelt es sich um ein Phython Script mit GUI, um ein Verzeichnis rekursiv auf Veränderung zu prüfen und das Ergebnis grafisch darzustellen. Das Script benötigt Python (`http://www.python.org/`) in der Version 2.5 und wxPython (`http://www.wxpython.org/`) um ausgeführt zu werden. Es ist auf dem SVN Server unter `http://WM19N590.muc:8081/svn/EA-403/Tools/FilterViewer` verfügbar.

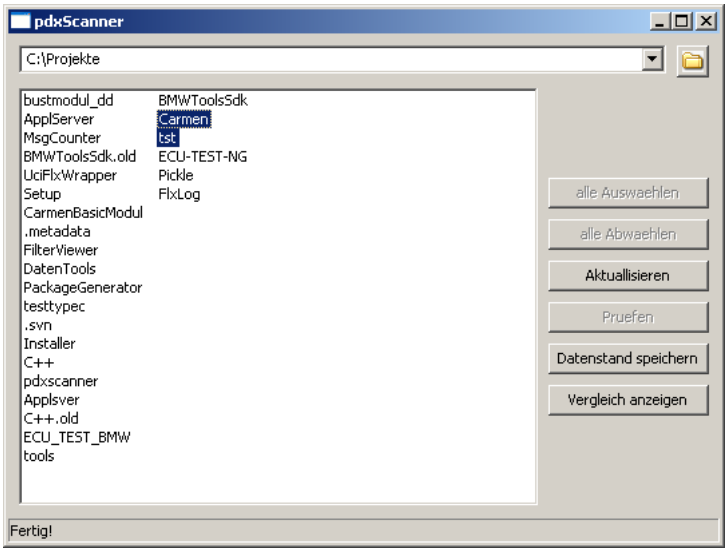

Abbildung 2.14: PDXScanner - Hauptfenster

Die zum Flashen der Steuergeräte nötigen PDX Container sind in einer komplexen Verzeichnisstruktur organisiert, um zum Beispiel die Baureihe, den Entwickler und das Steuergerät et cetera differenzieren zu können. Dies hat zwar den Vorteil, dass die nötigen Dateien nach logisch abgelegt sind, aber es ergibt sich eine recht tiefe Hirachie in der man sehr leicht den Überblick verlieren kann, wenn sich einzelne Pakete ändern.

Dabei soll das in Abbildung 2.14 dargestellte Programm helfen, denn es scannt ausgehend von einem gewählten Verzeichnis alle Unterverzeichnisse. Wenn bereits ein alter Datenstand gespeichert wurde, kann von selektierten Verzeichnissen die Unterschiede zum letzten Stand angezeigt werden. Wie man in Abbildung 2.15 sehen kann, ist es möglich die Anzeige nach verschiedensden Kriterien zu Filtern, um eine optimale Darstellung zu erreichen.

Das Ergebnis des Vergleiches wird in einer Datei mit dem Datenstand gespeichert. Um die Information leicht wiederverwertbar zu machen, wurde die Datenspeicherung als Python-

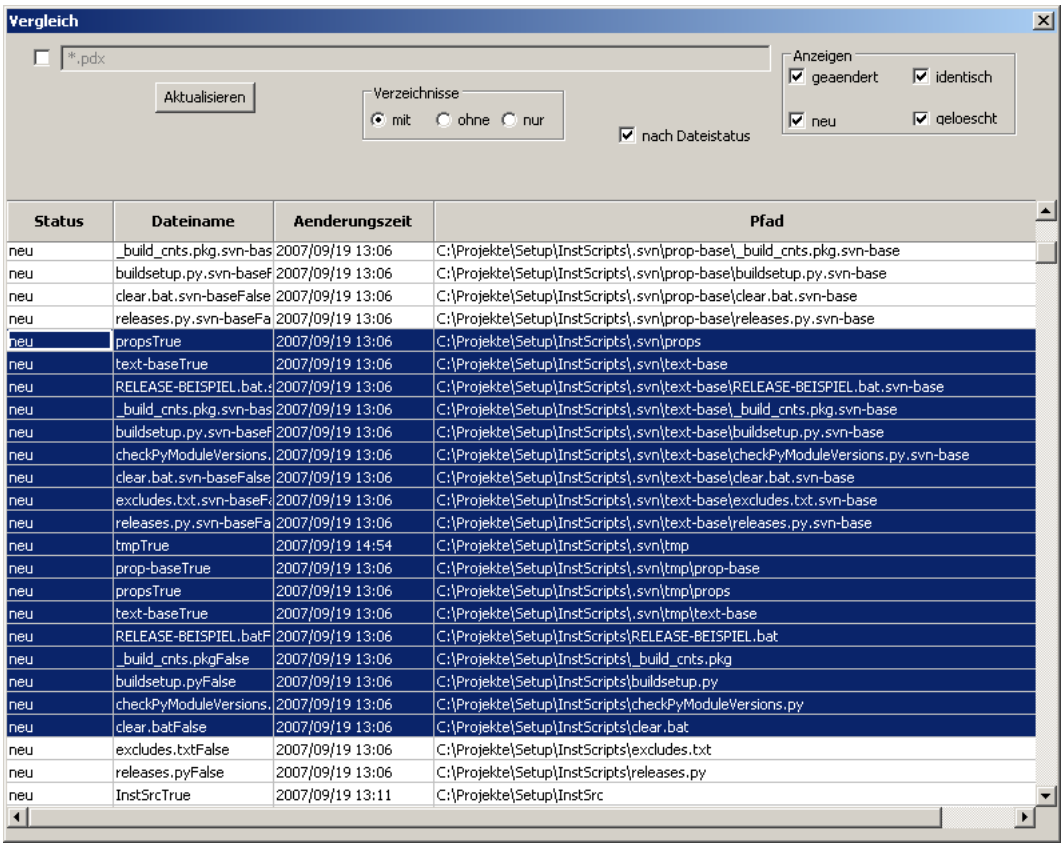

Abbildung 2.15: PDXScanner - Ergebnisanzeige

Code vorgenommen. Das bedeutet, dass eine Datei erzeugt wird, die wenn sie mit „eval“ zeilenweise innerhalb eines anderen Phyton Scriptes ausgeführt wird, ein Dictionary mit den Verzeichnisinformationen füllt, was dann sofort verwendet werden kann.

Ausblick Es wäre denkbar, sofort aus dem Scanner heraus eine anderes Programm/Script für die gewählten Dateien in der Ergebnisliste auszuführen. Dieses könnte für jede Datei separat aufgerufen werden und als Parameter den Dateinamen erhalten oder ein einziges mal mit der Liste der Dateien als Parameter. Desweiteren wäre es auch denkbar, eine Datei mit der Liste der Ergebnisse zu erstellen.

Für eine Anwendung aus einer Script oder Batch-Datei heraus wäre es ausserdem sinnvoll, das Tool in die Lage zu versetzen, Parameter aus der Kommandozeile zu verarbeiten, damit eine einfache Automatisierung möglich wird.

KAPITEL 3

Zusammenfassung

Im Praktikum wurden im Rahmen der Aufgabenstellung folgende Teilprojekte bearbeitet:

- Entwurf, Implementierung und Test des CARMEN Moduls „ECU-Test-BusChecker“
- Schulung der Mitarbeiter in der Benutzung von ECU-TEST 4 zur Busanalyse mit Hilfe von CARMEN
- Erweiterung des Filtervierwers zur komplexen Verwaltung von verschiedenen Filtergruppen und größere Nutzerfreundlich bei der Handhabung der Vergleichsergebnisse
- Erweiterung des Packagegenerators zum automatisierten Erzeugen von Testscripten für ECU-Test.

KAPITEL 4

Bewertung

Während meines Praktikums konnte ich viel über die Elektronik im Antrieb eines Kraftfahrzeuges lernen. Vorallem die Probleme und Möglichkeiten durch Vernetzung der zahlreichen Elemente wurden deutlich. Dabei gewann ich einen guten Überblick über eine Vielzahl der Spezialwerkzeuge die nötig sind, um eine Testumgebung bereitzustellen und die Funktionalität zu überprüfen (Controldesk, INCA, ECU-TEST, CAMEN, EDIABAS)

Die Erstellung des CARMEN-Moduls gab mir die Möglichkeit, eigenständig ein komplexes in sich abgeschlossenes Projekt zu bearbeiten.

Die Mitarbeiter waren alle freundlich und halfen bei Problemen gern weiter. Ich würde jederzeit gern wieder in diesem Team arbeiten.

Für mich war das Praktikum daher ein voller Erfolg und ich konnte viele positive Erfahrungen mitnehmen. Dafür möchte ich mich nochmals bei allen Beteiligten aufs herzlichste bedanken.

Sven Richter

Daniel Steffensky (Betreuer)

Literaturverzeichnis

[Gmb] GmbH, TraceTronic: Homepage TraceTronic GmbH, URL `http://www.tracetronic.de` (Zitiert auf Seite 7)

[Ing] Ingenieurbüro Matthias Puchner: *Anwendungshilfe zu Topologie Editor* (Zitiert auf Seite 8)