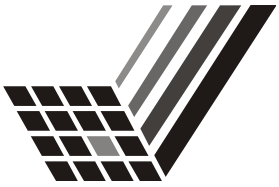
Projektgruppe
AutoLab

WS 07 / 08

Test und Diagnose



Universität Dortmund

Thomas Romanek
Eibenweg 11b
59423 Unna

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung und Begriffseinordnungen

Das Versagen von sicherheitskritischen Funktionen kann zu schweren Unfällen mit Toten führen. An die Sicherheit und Zuverlässigkeit von Fahrzeugfunktionen, wie zum Beispiel der Bremsen, werden daher hohe Anforderungen gestellt. Sowohl die Methode des Testens als auch die Diagnose bezeichnen Maßnahmen zur Sicherstellung dieser Anforderungen und zur Identifikation von Fehlern.

Testen ist eine Methode zur Identifikation von Fehlern während der Produktion. Eine besondere Bedeutung hat es somit im Rahmen der Qualitätssicherung. Es soll garantieren, dass das Produkt die gestellten Anforderungen erfüllt. Im Gegensatz zur Diagnose werden hier gezielt Spezifikations- und Implementierungsfehler nachgewiesen, die im Rahmen der Integration verschiedener Komponenten entstehen. Eine weitere Aufgabe des Testens ist möglichst frühzeitig Fehler, wie zum Beispiel Spezifikationsfehler, nachzuweisen, um so letztlich eine Beschleunigung des Entwicklungsprozesses und demzufolge Kostenersparnis zu bewirken.

Die Diagnosefunktion dient besonders der Realisierung der Überwachung sicherheitsrelevanter Systeme sowie in diesem Zusammenhang Werkstätten, um bei Fehlverhalten des Fahrzeugs eine gezielte Fehlersuche zu ermöglichen.

2 Tests zur Qualitätssicherung

2.1 Aufgaben

Tests identifizieren Fehler und tragen so zum kontinuierlichen Erreichen der Produktqualität bei. Aus diesem Grund sollten sie zum frühestmöglichen Zeitpunkt auf allen Systemebenen durchgeführt werden. Falls durch Tests keine Fehler gefunden werden, ist dadurch nicht nachgewiesen, dass keine Fehler vorhanden sind. Daher sollten weitere Maßnahmen, statische Methoden genannt, zur Qualitätssicherung durchgeführt werden. Im Gegensatz zu dynamischen Methoden wird bei statischen das Programm selbst nicht ausgeführt. Dazu zählen zum Beispiel Reviews.

Maßgeblich zur Organisation von Tests ist das sogenannte *V-Modell* aus dem Projektmanagement-Bereich, das in Abbildung 1 gezeigt ist. Folgende Testschritte werden unterschieden: Komponententest, Integrationstest und Systemtest, welche zu den Verifikationsmaßnamen zählen, sowie der Akzeptanztest als Validationsmaßnahme.

Im Bereich Automotive werden jedoch zusätzliche Testschritte verwendet, um den Entwicklungsablauf zu verkürzen. Schon während der Phase der Systemspezifikation wird z.B. versucht Spezifikationsfehler auszuschließen. Außerdem wird versucht durch geeignete Methoden Tests, die erst in einem späteren Stadium möglich wären, vorzuziehen.

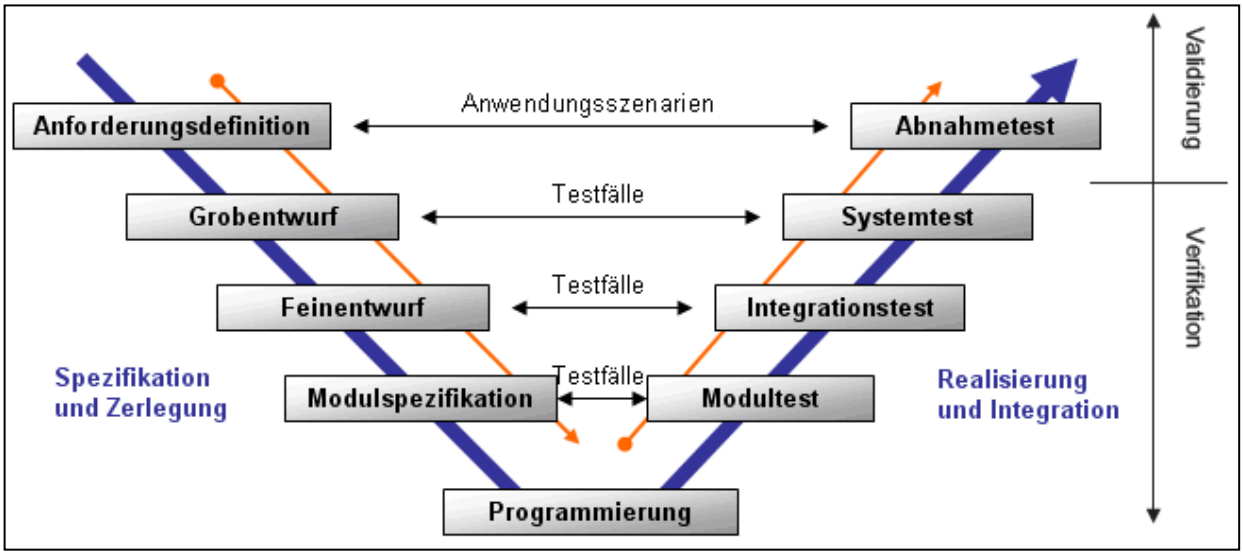


Abbildung 1: Das allgemeine V-Modell

Quelle: http://www.jprofil.de/Dokumentation/SoftwareEngineering/Vorgehensmodelle/images/vModell-02.gif

2.2 Spezielle Verfahren in Automotive

2.2.1 Model in the loop

Ziel dieser Methode ist das frühzeitige Erkennen von Spezifikationsfehlern. Es werden sowohl Software-Funktionen simuliert als auch Umgebungskomponenten. Dazu zählen etwa Hardware, Sollwertgeber, Sensoren und Aktuatoren. Es entsteht ein Modell, das auf einem Simulationssystem, etwa einem PC, ausgeführt werden kann. Nach Abschluss liegt so schon in der Spezifikationsphase ein verifiziertes Modell vor.

2.2.2 Rapid Prototyping

Unter *Rapid Prototyping* versteht man in diesem Zusammenhang Methoden zur Spezifikation und Ausführung von Software im realen Fahrzeug. Wie bei der Simulation (Model in the loop) liegt das zu entwickelnde System auch hier als Modell vor. Jedoch existiert beim Rapid Prototyping eine Schnittstelle zum realen Fahrzeug. Die Umgebung wird somit nicht simuliert. Dadurch muss die Ausführung der Software-Funktionen auf dem Experimentiersystem unter Echtzeitanforderungen erfolgen und das System muss in der Lage sein alle in der Umgebung vorhandenen Sensordaten zu verarbeiten und Aktuatoren anzusteuern.

Rapid Prototyping ermöglicht die frühe Ausführung des Programms und damit den Einsatz von dynamischen Prüfmethoden, die sonst erst nach dem Integrationsschritt der Software mit der Hardware möglich wären.

2.2.3 Software in the loop

Mit *Software in the loop* wird die Vorgehensweise bezeichnet, bereits realisierte Software-Komponenten in einer simulierten Umgebung auszuführen. Dadurch kann der Entwicklungsprozess erheblich beschleunigt werden, da die Software getestet werden kann ohne auf die zu diesem Zeitpunkt unter Umständen noch nicht verfügbare Zielhardware

angewiesen zu sein. So können frühzeitig dynamische, also automatisierte Software-Tests durchgeführt werden

2.2.4 Hardware in the loop

Hardware in the loop bezeichnet eine Klasse von Methoden und Werkzeugen, die zur Verifikation und Validation eingesetzt wird, sobald Hard- und Software eines Steuergeräts zur Verfügung stehen. Werden Regelungsfunktionen getestet, so kann das Steuergerät als Komponente im Regelkreis betrachtet werden, wodurch die Namensgebung begründet ist.

Die Vorgehensweise ist allerdings nicht nur auf Regelungsfunktionen beschränkt, so dass viele Prüfschritte, die ohne Testumgebung nur im Fahrzeug durchgeführt werden können, ins Labor verlagert werden. Dazu zählen Laborfahrzeuge und Prüfstände.

Bei einem Aufbau eines Laborfahrzeugs wird das Steuergerät als Black-Box betrachtet, so dass das funktionale Verhalten nur anhand seiner Ein- und Ausgangssignale bewertet werden kann.

Der Übergang von einem Laborfahrzeug zu einem Prüfstand ist fließend. Es werden weitere reale Komponenten des Gesamtsystems als Prüflinge in die Prüfumgebung integriert. Zum Beispiel reichen elektrische Signale zum Prüfen elektrohydraulischer Aktuatoren nicht mehr aus.

2.3 Komponententest / Modultest

Eine Komponente wird gegen seine Spezifikation getestet. Dabei können verschiedene statische Tests durchgeführt werden. Dynamische Methoden können durch Einsatz von *Software in the loop* angewendet werden.

2.4 Integrationstest

Das System wird gegen die Spezifikation der technischen Systemarchitektur getestet.

2.4.1 Integrationstest der Software

Bei der Integration von Software werden Software-Komponenten zu einem Software-Stand zusammengeführt. Zum Integrationstest zählen dabei Prüfungen, ob Schnittstellenspezifikationen sowie der Namensraum für Variablen eingehalten wurden, oder ob ein einheitliches Speicherlayout verwendet wird. Hierbei handelt es sich um statische Tests gegenüber Implementierungsrichtlinien. Diese werden vor der Übersetzung manuell durchgeführt oder automatisiert, etwa vom Compiler-Tool-Set.

2.4.2 Integrationstest des Systems

Um auch Integrationstests des Systems durchführen zu können, ohne dass alle Komponenten, Subsysteme und Komponenten der Systemumgebung vorhanden sind, werden nicht vorhandene Komponenten als *virtuelle Komponenten* nachgebildet. Die Testumgebung für eine

reale Komponente ist mit einer virtuellen Integrationsplattform verbunden, die fehlende Komponenten nachbildet. Folglich ist dies eine *Hardware in the loop* Methode.

2.5 Systemtest

Das System wird gegen die Spezifikation der logischen Systemarchitektur getestet. Die Modellierung von Komponenten ist nie vollständig. Die Simulation nicht vorhandener Komponenten kann nur Fragen beantworten, die vorher auch gestellt wurden. Durch den Systemtest im Fahrversuch sind Risiken durch Vernachlässigung in der Modellbildung nicht mehr vorhanden.

2.6 Akzeptanztest / Abnahmetest

Das System wird gegen die Benutzeranforderungen in ihrer realen Systemumgebung und aus der Benutzerperspektive getestet.

3 Das Diagnosesystem

3.1 Begriffe

Fehler

Ein Fehler ist eine unzulässige Abweichung mindestens eines Merkmals einer Betrachtungseinheit. Der Fehler ist ein Zustand. Die unzulässige Abweichung ist der über den Toleranzbereich hinausgehende Unterschied zwischen dem Istwert und dem Sollwert.

Ein Fehler kann, muss aber nicht, die Funktion der Betrachtungseinheit beeinträchtigen. Ein Fehler kann einen Ausfall oder eine Störung zur Folge haben.

Ausfall

Ein Ausfall ist ein nach Beanspruchungsbeginn entstandenes Aussetzen der Ausführung einer Aufgabe einer Betrachtungseinheit aufgrund einer in ihr selbst liegenden Ursache und im Rahmen der zulässigen Beanspruchung. Der Ausfall ist ein Ereignis. Der Ausfall entsteht durch einen oder mehrere Fehler.

Störung

Eine Störung ist ein nach Beanspruchungsbeginn entstandener, vorübergehender Ausfall. Eine Funktionsstörung ist die vorübergehende Unterbrechung oder Beeinträchtigung der Funktion.

3.2 Überwachung

Falls ein sicherheitsrelevantes System nicht mehr funktionsfähig ist, droht potentiell Gefahr. Für entsprechende Reaktionen müssen aus diesem Grund Störungen, Ausfälle oder auch Fehler zuverlässig erkannt werden. Zu diesem Zweck werden sicherheitsrelevante Systeme überwacht. Die Überwachung folgt der Zielsetzung Fehler frühzeitig, möglichst vor einer Störung oder einem Ausfall, zu erkennen sowie mögliche Fehler zu behandeln, um möglichst Störungen oder Ausfälle ganz zu verhindern.

3.2.1 Fehlererkennungs- bzw. Fehlerdiagnoseverfahren

In einem Fehlerdiagnoseverfahren wird geprüft, ob zwischen mindestens zwei Werten, die zwischen diesen Werten bestehenden Zusammenhänge erfüllt sind. Entsprechende Verfahren sind beispielsweise:

- **Referenzwertüberprüfung:** Die Antwort des Systems zu einer bestimmten Eingabe wird mit der bekannten Soll-Antwort verglichen.
- **Überprüfung anhand redundanter Werte:** Dies ist in Software realisierbar, indem man mindestens zwei prinzipverschiedene (Diversität vorhanden) Algorithmen auf dieselben Eingangswerte anwendet.
- **Beobachtung von Kommunikationsverbindungen:** Zum Beispiel durch Paritätsprüfungen realisierbar.
- **Senden einer Bestätigung bei Nachrichteneingang.**
- **Beobachtung von physikalischen Eigenschaften:** Etwa durch Beobachtung der Temperatur.
- **Beobachtung der Programmausführung:** Beispielsweise durch eine Watch-Dog-Schaltung. Diese löst Fehlerreaktionen im Falle von unzulässig langen Ausführungszeiten aus.
- **Modellbasierte Fehlererkennung:** Das Verhalten der realen Komponenten wird mit dem Verhalten von modellierten Komponenten verglichen.

3.2.2 Fehlerbehandlung

In einem Verfahren zur Fehlerbehandlung wird festgelegt, wie auf Fehlersymptome reagiert wird. Reaktionen sind beispielsweise:

- Verwendung redundanter Werte
- Abschaltung von Subsystemen oder des Gesamtsystems
- Verharren im Fehlerzustand oder Strategiewechsel
- Fehlerspeicherung (z.B. im Fehlerspeicher)
- Fehlerbeseitigung (z.B. durch Reset des Mikroprozessors bei Watch-Dog-Schaltungen)

Für sicherheitsrelevante Systeme gibt es außerdem eine eigene Sicherheitslogik, die Fehlerbehandlungsmaßnahmen beschreibt. Diese wird hier nicht näher betrachtet.

3.3 Diagnosesystem

Durch das Diagnosesystem wird die Überwachung realisiert. Es gehört zum Grundumfang eines Seriensteuergeräts. Es wird zwischen Onboard- und Offboard-Diagnosefunktionen unterschieden.

3.3.1 Onboard-Diagnosefunktion

Onboard-Diagnosefunktionen werden die Fehlererkennungsfunktionen genannt, die im Steuergerät selbst ausgeführt werden. Es werden beispielsweise die Ein- und Ausgänge des Steuergeräts überprüft. Meistens nehmen diese auch einen Eintrag im Fehlerspeicher vor. Sie können je nach Anwendungsfall nur beim Systemstart, nach dem Betrieb oder auch zyklisch während des Betriebs durchgeführt werden.

3.3.2 Offboard-Diagnosefunktion

Als Offboard-Diagnosefunktionen werden Fehlererkennungsfunktionen bezeichnet, bei denen das Steuergerät an einen Diagnosetester angeschlossen wird. Dabei wird der Diagnosetester an eine zentrale Fahrzeugdialogschnittstelle angeschlossen, den „Diagnosestecker“, der mit den verschiedenen Steuergeräten des Fahrzeugs verbunden ist. So kann auf alle diagnosefähigen Steuergeräte zugegriffen werden.

3.3.3 Sollwertgeber- und Sensordiagnosefunktion

Die Sollwertgeber, Sensoren und die Verbindungsleitungen zum Steuergerät können anhand der ausgewerteten Eingangssignale überwacht werden.

Die Diagnose wird ermöglicht durch Messung von Steuergeräteeingangssignalen und steuergeräteinternen Größen. Diese Signale werden über die Offboard-Diagnosekommunikation an den Diagnosetester übertragen und können dort dargestellt werden. Für Plausibilitätsprüfungen können auch Signale von den Bussen mitgemessen werden. Ergänzend können zusätzliche Diagnosemodule ins Fahrzeug eingebaut werden, die weitere Signale für Plausibilitätsprüfungen liefern.

3.3.4 Aktuatordiagnosefunktion

Über den Diagnosetester können gezielt einzelne Aktuatoren des Steuergeräts aktiviert werden, um so ihre Funktionsfähigkeit zu überprüfen. In der Regel funktioniert dies nur bei stehenden Fahrzeugen und unter festgelegten Bedingungen (z.B. Motordrehzahl). Die Ausgangsgrößen der Steuerung oder des Reglers werden dabei online durch den Diagnosetester vorgegeben.

3.3.5 Fehlerspeichermanager

Durch die Onboard-Diagnosefunktion erkannte Fehlersymptome werden in der Regel im Fehlerspeicher des Steuergeräts gespeichert. Dieser Speicher wird meist im EEPROM abgelegt, so dass die Dauerhaftigkeit gewährleistet ist. Gesetzliche Anforderungen legen fest, dass für jeden Fehlereintrag neben dem Fehlercode (engl. Diagnostic Trouble Code, DTC)

zusätzliche Informationen gespeichert werden müssen. Dazu gehören unter anderem Betriebs- und Umweltbedingungen, die beim Auftreten des Fehlersymptoms herrschten. Beispielsweise werden Motordrehzahl, Motortemperatur und der Kilometerstand gespeichert.

Im Motormanagement sind oft länderspezifische Abgasgesetze zu beachten. Diese schreiben für Fehler, die Einfluss auf die Abgasemission haben, die Fehlercodes sowie die Blinkcodes der sogenannten Fehlerlampe zur Information des Fahrers vor.

Der Fehlerspeichermanager übernimmt das Ein- und Austragen von Fehlersymptomen in den Fehlerspeicher und wird meist als eigenständige Software-Komponente realisiert. Des weiteren übernimmt der Manager die Aufgaben mit dem Diagnosetester. Durch diesen können die gespeicherten Informationen, etwa in der Servicewerkstatt, ausgelesen und dargestellt werden. Nach Beseitigung der Fehler können sie aus dem Fehlerspeicher gelöscht werden.

3.3.6 Offboard-Kommunikation

Für die Kommunikation zwischen Steuergerät und Diagnosetester wurden Standards definiert. In der Regel wird die Offboard-Kommunikation unter Berücksichtigung dieser Standards eindeutig festgelegt. Der Weg zur Standardisierung wurde unter anderem getrieben durch gesetzliche Vorschriften, die für die Überprüfung der abgasrelevanten Systemfunktionen eine relativ einheitliche Schnittstelle erzwangen (On Board Diagnose).

Das Keyword 2000 Protokoll ist derzeit der am weitesten verbreitete Standard in europäischen Fahrzeugen. Es wurde zunächst mit K-Line, einem bidirektionalen Ein-Draht-Bus, und später mit CAN-Bussystemen realisiert. Normiert sind sie in ISO 14230-3 für K-Line und ISO 15765-3 für CAN. Im nächsten Kapitel wird dieses näher betrachtet.

Mit ISO 14229 wird der Versuch unternommen, die mit der KWP 2000-Diagnose eingeführten Prinzipien zu verallgemeinern und vom darunter liegenden, realen Busprotokoll unabhängig zu machen. So sollen neue Bussysteme wie LIN oder FlexRay leichter integriert werden können.

ISO 15765-3 referenziert sehr oft ISO 14230-3 sowie andere Standards, wodurch die Spezifikation unübersichtlich wird. Die in ISO 14229 spezifizierten Dienste sind funktional und im Grundaufbau mit den KWP 2000-Diensten identisch, jedoch wurden Dienste teilweise neu gruppiert, neue Identifier vergeben und der Parameteraufbau in vielen Details verändert. Das Ergebnis, Unified Diagnostic Services UDS, ist ein zu KWP 2000 nah verwandtes Protokoll, welches allerdings als eigenständig betrachtet werden muss.

3.4 KWP2000

3.4.1 Kommunikation

Die gesamte Kommunikation über dieses Protokoll geht vom Testgerät aus. Die Diagnose-Anwendung des Testers sendet eine Botschaft mit Diagnose-Anfrage (Request) über das Netz an das Steuergerät. So wird die Steuergeräte-Anwendung durch den Appllication Layer informiert (Indication). Die Antwort (Response) des Steuergeräts wird zurück an den Tester übertragen, dessen Application Layer die Antwort (Confirm) an die Tester-Anwendung übergibt.

Das Steuergerät wird dabei normiert als Server, der Diagnosetester als Client und die vordefinierten Anfragen als Services (Dienste) bezeichnet.

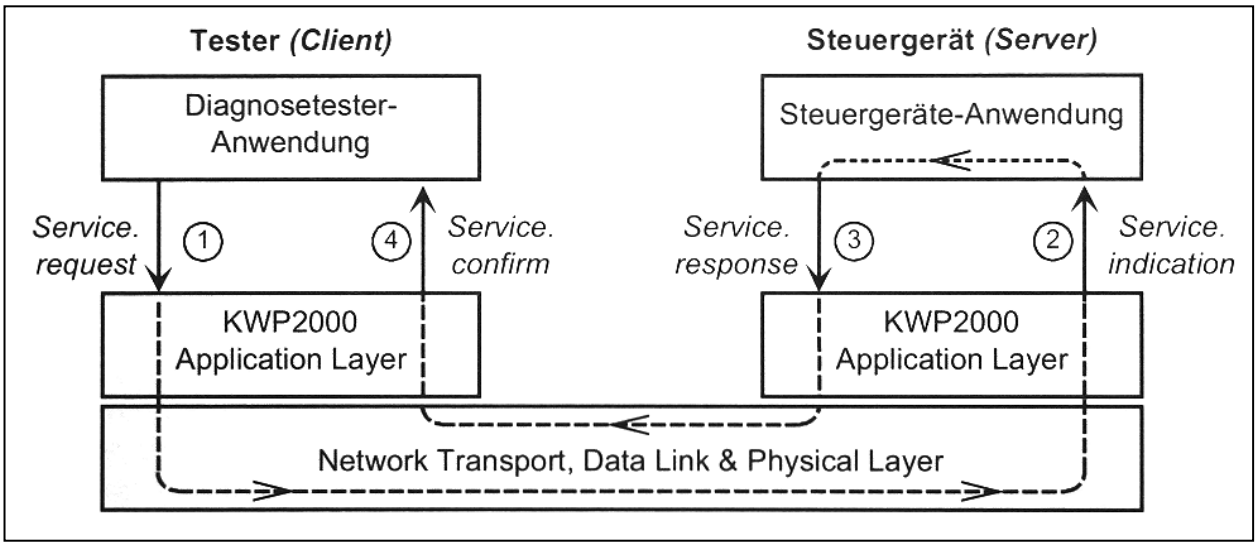


Abbildung 2: Kommunikationsmodell der KWP 2000 - Anwendungsschicht

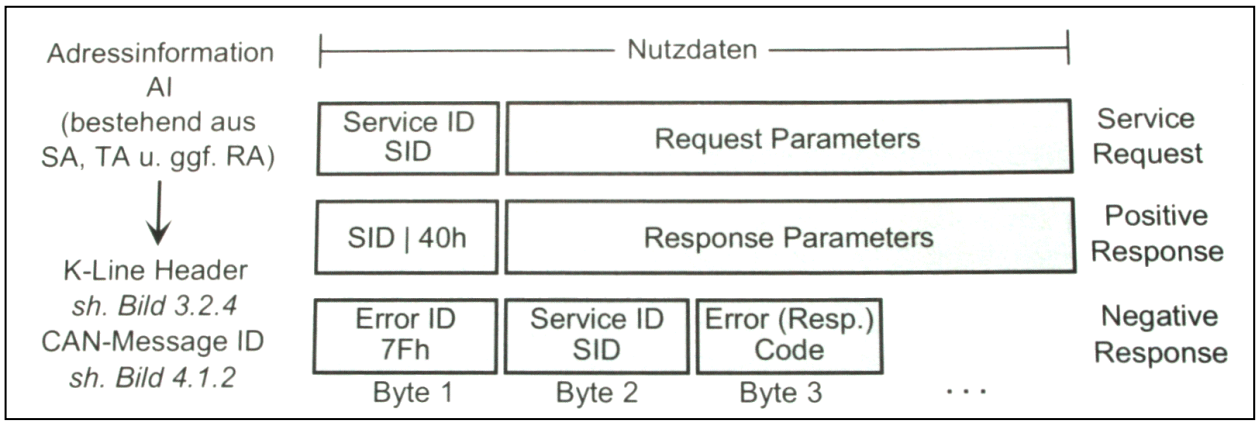


Abbildung 3: Aufbau der KWP 2000 Nachrichten der Anwendungsschicht

Die Dienste der Anwendungsschicht bestehen, wie in Abbildung 3 gezeigt, aus folgenden Teilen:

- Adressinformation AI bestehend aus Source Address SA, Target Address TA sowie gegebenenfalls Remote Address RA, jeweils 1 Byte
- *Service Identifier SID*, der den ausgewählten Dienst kennzeichnet
- Parameter, Anzahl abhängig vom jeweiligen Dienst

Die Adressinformationen werden im jeweiligen Nachrichten-Header (K-Line bzw. CAN) übergeben. Der Service Identifier wird im ersten Nutzdatenbyte, die Parameter in den folgenden Nutzdatenbytes übertragen. Einen Überblick über Services Identifier gibt Tabelle 1.

Bei Antwortnachrichten wird unterschieden: Bei positiven Antworten entsprechen die Service Identifier denen der Anfrage, mit dem einzigen Unterschied, dass das Bit 6 auf 1 gesetzt wird

(was einem logischen ODER mit 40h entspricht). Negative Antworten enthalten als Service Identifier stets 7Fh und es wird anschließend der SID des fehlgeschlagenen Dienstes sowie ein Fehlercode der Länge 1 Byte gesendet (siehe Tabelle 2).

	Request	Positive Response	Negative Response	Definiert durch
OBD kompatible Dienste	00 ...0Fh	40 ... 4Fh	7F 00 ...7F 0Fh	SAE J1979 (ISO 15031-5)
Allgemeine Dienste (für K-Line und CAN)	10 ...3Eh	50 ... 7Eh	7F 10 ... 7F 3Eh	ISO 14230-3
Escape Codes (ESC)	80h	C0h	7F 80h	Dito
K-Line-spezifische Dienste	81 ... 83h	C1 ... C3h	7F 81 ... 7F 82h	ISO 14230-2
CAN-spezifische Dienste	84 ... 85 h	C4 ... C5h	7F 84 ... 7F 85h	ISO 15765-3
Herstellerspezifische Dienste	A0 ... BEh	E0 ... FEh	7F A0 ... 7F BEh	Fahrzeug- bzw. Systemhersteller

Tabelle 1: Überblick Service Identifier

10h	General Reject
11h	Service not supported
12h	Subfunction not supported
21h	Busy, repeat Request
78h	Response Pending
33h	Security access denied
35h	Invalid Key
...	...

Tabelle 2: Typische Response Error Codes

In den restlichen Protokollen des Protokollstapels, Transport bzw. Data-Link Layer werden die Botschaften mit entsprechenden Header- und Trailerinformationen versehen.

Steuergeräten, die neuprogrammierbar (*flashbar*) sind, kann über die Nachrichten auch eine neue Programmversion zugesendet werden.

3.4.2 Diagnosesitzungen

Aus Sicherheitsgründen sind nicht stets alle Diagnosefunktionen verfügbar. Unter einer Diagnosesitzung versteht man einen Betriebszustand des Steuergeräts, in dem ein bestimmter Satz von Diagnosefunktionen ausführbar ist. Im normalen Betrieb befindet sich das Steuergerät in der *Default Diagnostic Session*. Von dieser ausgehend, kann der Diagnosetester das

Steuergerät durch eine *Start Diagnostic Session* Botschaft auffordern, eine spezielle Diagnosesitzung zu eröffnen.

Die Bedingungen zur Öffnung einer Sitzung werden vom Fahrzeughersteller festgelegt. Üblich sind:

- Fahrzeug und Motor müssen sich in einem bestimmten Zustand befinden. Beispielsweise muss beim *Flashen* das Fahrzeug stehen und der Motor abgestellt sein.
- Der Diagnosetester muss sich beim Steuergerät mit Hilfe eines Schlüsselaustausches (Seed and Key) anmelden, um Zugriff auf bestimmte Diagnosedienste zu erhalten. Dazu sendet der Tester eine *Security Access Request Seed* Botschaft, auf die das Steuergerät mit einem Initialisierungswert (*Seed*) antwortet. Dies ist oft eine Zufallszahl. Aus ihr berechnet der Tester einen Schlüsselwert (*Key*) und sendet diesen mit einer *Security-Access – Send Key* Botschaft an das Steuergerät zurück. Stimmt die Antwort mit dem vom Steuergerät erwarteten Wert überein, sendet dieses eine positive Antwort und schaltet in die Diagnosesitzung um. Die Länge der Schlüssel sowie die Algorithmen zur Berechnung können vom Hersteller definiert werden.

In einer speziellen Diagnosesitzung wird ein *Time-out-Mechanismus* aktiviert. Wenn das Steuergerät nicht spätestens alle fünf Sekunden eine Diagnose-Botschaft vom Tester erhält, stoppt es die Diagnosesitzung und kehrt zur Default-Session zurück. Aus diesem Grund sendet der Tester spätestens alle vier Sekunden eine *Tester-Present*-Botschaft*,* falls er keine echten Diagnosebotschaften zu senden hat. Darüber hinaus gibt es auch Botschaften zum Beenden einer Sitzung. An dieser Stelle sei auf die Literatur verwiesen.

4 Einordnung in das Vorhaben der Projektgruppe

Mit dem Vorhaben den Grundstein für ein neues Labor zu legen entspricht unser Vorhaben dem, ein Laborfahrzeug aufzubauen. Wir werden im Rahmen der Projektgruppe deshalb nicht umhin kommen die vorgestellten Mittel zum Testen von Komponenten, der Integration sowie des Gesamtsystems einzusetzen.

Da das Labor langfristig für Analysen unter realen Lastbedingungen eingesetzt werden soll, wird in diesem Zusammenhang die Diagnose eine immer größere Rolle spielen.

4.1 CANoe 7.0

Im Rahmen unserer Projektgruppe werden wir das Tool CANoe 7.0 nutzen. Dabei handelt es sich um ein Werkzeug zur Entwicklung, dem Test und der Analyse von Netzwerken und Steuergeräten. Folgende wichtige Eigenschaften des Programms kann ich identifizieren beziehungsweise der Ausarbeitung zuordnen:

- Zu Anfang des Entwicklungsprozesses werden Simulationsmodelle erstellt, die das Verhalten der Steuergeräte nachbilden (⇒ Model in the loop)

Abbildung 4: CANoe Bedienoberfläche

- Diese Modelle bleiben während der gesamten Steuergeräteentwicklung Grundlage für die Analyse, den Test und die Integration von Bussystemen und Steuergeräten (⇒ Software in the loop, Hardware in the loop)
- Simulation kompletter Systeme ist möglich sowie Simulation von Nachrichten (noch) nicht vorhandener Steuergeräte (Restbussimulation)
- Ermöglicht automatisierte Durchführung von Tests (⇒ dynamische Tests)
- Diagnosekommunikation nach KWP2000 und UDS sowie Einsatz als vollständiger Diagnosetester
- Unterstützung von Modellentwicklung in MATLAB und Statemate
- Unterstützung der Bussysteme CAN, LIN, MOST, FlexRay, J1587 sowie weiterer CAN-basierter Protokolle

Weitere wichtige Eigenschaften des Programms sind:

- Erstellung benutzerdefinierter Oberflächen zur Steuerung der Simulation und Tests oder der Anzeige der Analysedaten
- Freie Programmierbarkeit durch die integrierte C-ähnliche Programmiersprache CAPL zur Unterstützung von Simulation und Analyse
- Arbeitet auf zwei PCs verteilt (für kritische echtzeitrelevante Simulationen)

Literaturverzeichnis

Zimmermann, Werner; Schmidgall, Ralf: Bussysteme in der Fahrzeugtechnik, 2. Auflage, ATZ/MTZ-Fachbuch, Wiesbaden, 2007

Schäufelle, Jörg: Automotive Software Engineering (ATZ-MTZ Fachbuch), 3. Auflage, ATZ/MTZ-Fachbuch, Wiesbaden, 2007

Vector Informatik GmbH: http://www.vector-worldwide.com/portal/medien/cmc/datasheets/CANoe_DataSheet_DE.pdf