

---

# Praktikumsbericht

---

**Samir Alexis Revelo Cordoba**

Praktikumsbericht

Betreuer: Dipl.-Ing. Helmut Hörner, Dipl.-Ing. Markus Schwarz

Unternehmen: Vector Informatik GmbH

Abteilung: PES1



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT





---

## Aufgabenstellung

---

Moderne Dieselmotoren werden bei der Applikation hinsichtlich verschiedenster Kriterien optimiert. Neben den Schadstoff-Emissionen zählen hierzu sowohl das Motordrehmoment, der Wirkungsgrad als auch die Geräuschemissionen.

Im Rahmen dieser Bachelorarbeit soll zunächst auf Basis einer Literaturrecherche untersucht werden, wie aus dem Zylinderdruckverlauf Kenngrößen generiert werden können, die es ermöglichen, das Drehmoment, den Wirkungsgrad als auch die Geräuschemissionen am Beispiel des Dieselmotors zu quantifizieren. Diese gewonnenen Größen sollen hierauf aufbauend modellbasiert und rechnergestützt mono- als auch multikriteriell lokal optimiert werden. Für ausgesuchte Stellgrößen soll zusätzlich analysiert werden, in welchem Maße sie das Pareto-Problem entschärfen können. Als Basis für diese Optimierung muss zunächst ein Gesamtmodell des Motors erstellt werden, dessen beide Teilmodelle, das Luftpfad- und Verbrennungsmodell, bereits zur Verfügung stehen.

Beginn: 01. April 2016  
Ende: 31. September 2016  
Seminar: 27. November 2013

---

Prof. Dr.-Ing. Dr. h.c. Rolf Isermann

---

Dipl.-Ing. Helmut Hörner  
Dipl.-Ing. Markus Schwarz

Technische Universität Darmstadt  
Institut für Automatisierungstechnik  
und Mechatronik  
FG Regelungstechnik und Prozessautomatisierung  
Prof. Dr.-Ing. Dr. h.c. Rolf Isermann

Landgraf-Georg-Straße 4  
64283 Darmstadt  
Telefon 06151/16-5465  
<http://www.rtm.tu-darmstadt.de/rtp.html>

REGELUNGSTECHNIK UND  
PROZESSAUTOMATISIERUNG **rtp**



---

## **Erklärung**

---

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt habe. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Stuttgart, den 31. September 2016

---

Samir Alexis Revelo Cordoba

---

## Kurzfassung

---

Im Rahmen dieser Bachelorarbeit wird zunächst eine Literaturrecherche über bestehende Analysemethoden durchgeführt, mit deren Hilfe die Lösung einer multikriteriellen Optimierungsaufgabe hinsichtlich der Drehmoment- und Geräuschbildung am Dieselmotor realisiert werden kann.

Aufbauend wird ein ausgewähltes Verfahren benutzt, um ein Optimum des Motordrehmoments bei möglichst niedrigen Geräuschemissionen zu finden. Die Optimierung wird anhand eines am Institut für Automatisierungstechnik entwickelten Simulationsmodells eines Dieselmotors durchgeführt und am institutseigenen Motorenprüfstand validiert.

**Schlüsselwörter:** Dieselmotor, Drehmomentbildung, Geräuschemissionen, Steuerung, multikriterielle Optimierung, Wirkungsgrad.

---

## Abstract

---

As part of this bachelor thesis a literature research about existing analytical methods is initially performed. The latter enables one to realize the solution of a multi-criterial optimization task in regards to torque and noise generation.

On the basis of this a selected procedure is used to identify the optimum of the engine torque at lowest noise generation. The optimization is performed for a diesel engine with the help of a simulation model developed by the Institute of Automatic Control and Mechatronics at the TU Darmstadt ( Institut für Automatisierungstechnik) and is being validated on a engine test bed provided by the institute.

**Keywords:** Dieselmotor, motortorque, motornoise, optimization, control, multi-criterial optimization, motor effectiveness.

---

# Inhaltsverzeichnis

<b>Symbole und Abkürzungen</b>	<b>vii</b>
<b>1. Firmenprofil</b>	<b>1</b>
<b>2. Einführung</b>	<b>3</b>
2.1. Motivation . . . . .	3
<b>3. Wochenübersicht und Arbeitsberichte</b>	<b>5</b>
3.1. 1. Woche . . . . .	5
3.1.1. Wochenübersicht . . . . .	5
3.1.2. Arbeitsbericht . . . . .	5
3.2. 2. Woche . . . . .	8
3.2.1. Wochenübersicht . . . . .	8
3.2.2. Arbeitsbericht . . . . .	8
3.3. Dritte Woche . . . . .	10
3.3.1. Wochenübersicht . . . . .	11
3.3.2. Arbeitsbericht . . . . .	11
3.4. Vierte Woche . . . . .	12
3.4.1. Wochenübersicht . . . . .	12
3.4.2. Arbeitsbericht . . . . .	12
3.5. Fünfte Woche . . . . .	14
3.5.1. Wochenübersicht . . . . .	14
3.5.2. Arbeitsbericht . . . . .	14
3.6. Sexte Woche . . . . .	14
3.6.1. Wochenübersicht . . . . .	14
3.6.2. Arbeitsbericht . . . . .	14
3.7. Siebte Woche . . . . .	17
3.7.1. Wochenübersicht . . . . .	17
3.7.2. Arbeitsbericht . . . . .	17
3.8. Achte Woche . . . . .	17
3.8.1. Wochenübersicht . . . . .	18
3.8.2. Arbeitsbericht . . . . .	18
3.9. Neunte Woche . . . . .	18
3.9.1. Wochenübersicht . . . . .	18
3.9.2. Arbeitsbericht . . . . .	19

---

3.10. Zehnte Woche . . . . .	19
3.10.1. Wochenübersicht . . . . .	19
3.10.2. Arbeitsbericht . . . . .	19
3.11. Elfte Woche . . . . .	19
3.11.1. Wochenübersicht . . . . .	22
3.11.2. Arbeitsbericht . . . . .	22
<b>4. Zusammenfassung und Ausblick</b>	<b>23</b>
<b>A. Motorprüfstand des IAT</b>	<b>25</b>
<b>B. Verwendeter Dieselmotor und entsprechendes Simulationsmodell</b>	<b>27</b>
B.1. Motordaten . . . . .	27
B.2. Verwendetes Simulink-Modell des Dieselerbrennungsmotors . . . . .	27
<b>C. Softwaredokumentation</b>	<b>29</b>
C.1. Berechnung der Schwerpunktlage der Verbrennung $\phi_{Q50}$ . . . . .	29
C.2. Berechnung des Druckgradienten . . . . .	32
<b>D. Prüfstandsmessdaten</b>	<b>35</b>
<b>Abbildungsverzeichnis</b>	<b>37</b>
<b>Tabellenverzeichnis</b>	<b>39</b>
<b>Literaturverzeichnis</b>	<b>41</b>



# Symbole und Abkürzungen

## Lateinische Symbole und Formelzeichen

Symbol	Beschreibung	Einheit
$c_K$	Kolbengeschwindigkeit	m/s
$dp_Z$	Zylinderdruckgradient	bar/°KW
$dp_{Z,max}$	maximales Zylinderdruckgradient	bar/°KW
$dQ_B$	Brennverlauf	W
$dQ_W$	Wandwärme	W
$g$	Verhältnisfunktion	m/°KW
$f$	einzelne Zielfunktion	—
$k$	Gewichtungsfaktor	—
$l_{pl}$	Pleuellänge	m
$m_B$	Brennstoffmasse	kg
$m_{osz}$	oszillatorische Masse	kg
$n_{mot}$	Motordrehzahl	min <sup>-1</sup>
$p_Z$	Zylinderdruck	bar
$p_{mi}$	Indizierter Mitteldruck	bar
$q_{inj}$	Kraftstoffmenge	mm <sup>3</sup> /inj
$r_{KW}$	Kurbelradius	m
$s_K$	Kolbenweg	m
$u_{HDAGR}$	Stellgröße Ventil für Hochdruck-AGR	—
$u_{NDAGR}$	Stellgröße Ventil für Niederdruck-AGR	—
$u_{VTG}$	Stellgröße Leitschaufel Turbolader	—
$x$	Brennstoffumsetzungsrate	—
$x_{AGR}$	Abgasrückführrate	—
$x_{2EB}$	Gaszusammensetzung-Einlassbehälter	—
$x_3$	Gaszusammensetzung-Auslassbehälter	—
$A_K$	Wirksamekolbenfläche	m <sup>2</sup>
$F$	Zielfunktion	—
$F_S$	Pleuelstangenkraft	N

Symbol	Beschreibung	Einheit
$F_T$	Tangentialkraft	N
$F_K$	Kolbenkraft	N
$F_{M,osz}$	oszillatorische Trägheitskräfte	N
$H_u$	Unterer Heizwert	J/kg
$M_{mot}$	Gesamtmotordrehmoment	Nm
$M_Z$	Zylinder-Drehmoment	Nm
$\overline{M}_Z$	Zylinder-Durchschnittsdrehmoment	Nm
$M_R$	Motorreibmoment	Nm
$Q_B$	Integraler Brennverlauf	J
$P_{2EB}$	Einlassbehälterdruck	Pa
$P_3$	Auslassbehälterdruck	Pa
$T_{2EB}$	Einlassbehältertemperatur	K
$T_3$	Auslassbehältertemperatur	K
$T_3$	Stellgrößenvektor	K
$V_C$	Kompressionsvolumen	m <sup>3</sup>
$W_e$	Effektive Arbeit	J
$W_{BV}$	Arbeitsinhalt des Verbrannten Stoffs	J
$W_B$	Arbeitsinhalt des eingespritzten Brennstoffs	J
$W_{th}$	Arbeit aus dem Gleichraumprozess	J
$W_{iHD}$	die am Kolben indizierte Arbeit der Hochdruckschleife	J
$W_i$	die gesamte am Kolben indizierte Arbeit	J
$\lambda_{pl}$	Pleuelstangenverhältnis	—
$\omega_{mot}$	Kreisfrequenz Motor	rad/s
$\eta_e$	Effektiver Wirkungsgrad	—
$\eta_B$	Brennstoffsumsetzungsgrad	—
$\eta_{th}$	thermischer Wirkungsgrad	—
$\eta_{LW}$	Ladungswechselverluste	—
$\eta_v$	Verbrennungsgüte	—
$\eta_m$	mechanischer Wirkungsgrad	—
$\eta_i$	Innenwirkungsgrad	—
$\eta_A$	relevanter Wirkungsgrad	—
$\phi_{HE}$	Stellgröße Winkel der Haupteinspritzung	°KW
$\phi_{VE}$	Stellgröße Winkel der Voreinspritzung	°KW

## Griechische Symbole und Formelzeichen

Symbol	Beschreibung	Einheit
$\phi$	Kurbelwinkel	°KW
$\eta$	Wirkungsgrad	
$\omega$	Kreisfrequenz	s <sup>-1</sup>
$\lambda$	Pleuelstangenverhältnis	

## Abkürzungen

Kürzel	vollständige Bezeichnung
BTE	Basis Test Enviroment
EB	Einlassbehälter / Einspritzbeginn
ED	Einspritzdauer
GODLIKE	engl. Global Optimum Determination by Linking and Interchanging Kindred Evaluators
HE	Haupteinspritzung
HDAGR	Hochdruck-Abgasrückführung
inj	engl. injection (Einspritzung)
IAT	Institut für Automatisierungstechnik
KW	Kurbelwinkel
mot	Motorisch
NDAGR	Niederdruck-Abgasrückführung
NO <sub>x</sub>	Stickoxid
OT	Oberer Totpunkt
PA	Partikel
SPL	Schwerpunktlage der Verbrennung
UT	Unterer Totpunkt
VE	Voreinspritzung
VTG	Variable Turbinengeometrie
x	Gaszusammensetzung
Z	Zylinder



---

# 1 Firmenprofil

Beschreibung, was in Vector gemacht wird.



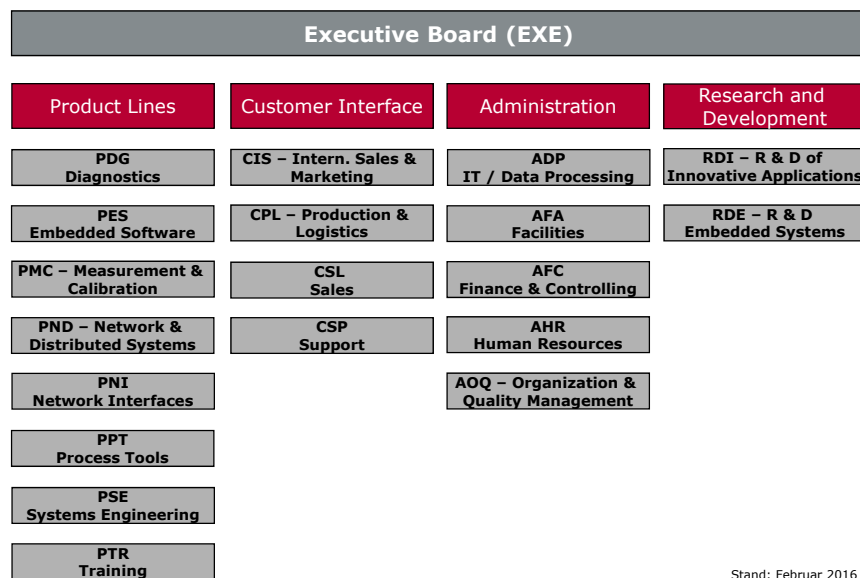
## 2 Einführung

Die Abteilung PES stellt Erstausrüstern (OEMs) und Zulieferern der Automobilindustrie und verwandter Branchen Komponenten und Dienste bereit, um eingebettete Systeme zu erstellen. Die genannte Abteilung fokussiert somit auf fundamentale Softwaretools zur Ergänzung der eigenen Applikationssoftware der Kunden.

### 2.1 Motivation

Aufbau-Organisation

VECTOR 



Stand: Februar 2016

1

Abbildung 2.1.: Vectorabteilungen1





## 3 Wochenübersicht und Arbeitsberichte

### 3.1 1. Woche

#### 3.1.1 Wochenübersicht

Tabelle

#### 3.1.2 Arbeitsbericht

Am ersten Tag meiner Tätigkeit bei Vector Informatik bekam ich die Gelegenheit die Firma und ihre unterschiedlichen Abteilungen näher kennen zu lernen. Es war erforderlich, an einer ausführlichen Präsentation über die Firma und die unterschiedlichen Abteilungen teilzunehmen. Dabei konnte ich ebenfalls Mitarbeiter kennen lernen, für die dieser auch ihr erster Tag bei Vector war.

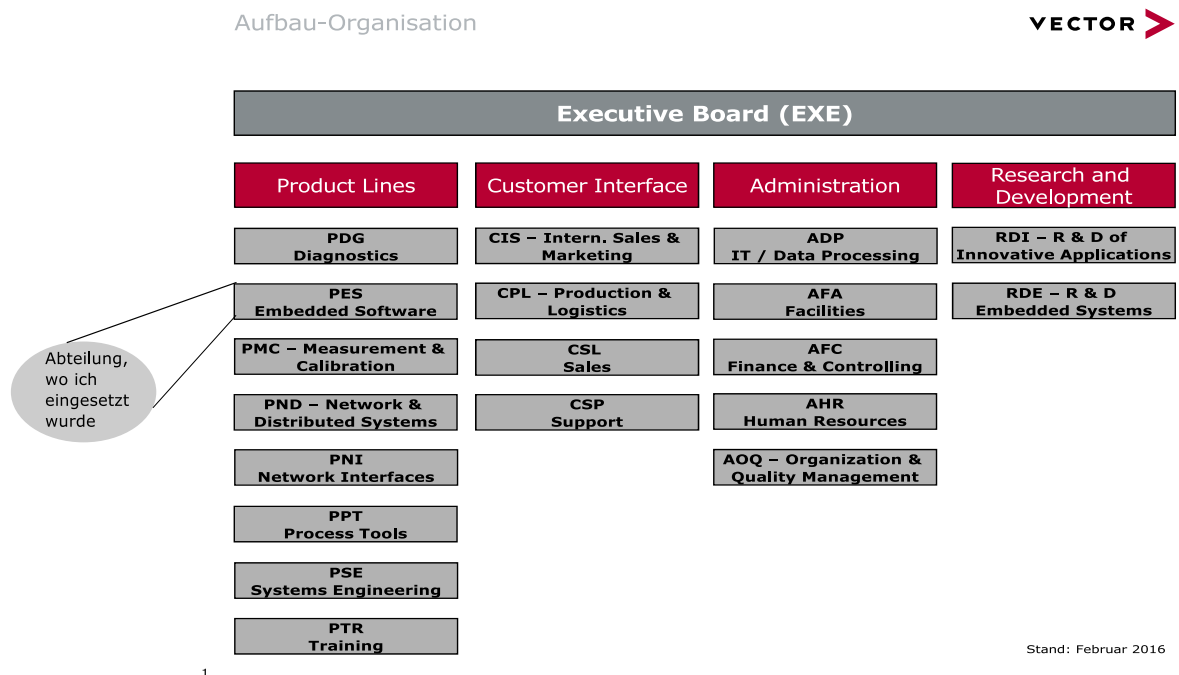


Abbildung 3.1.: Zeichnung1

Die Softwareabteilung (ADP) gab uns u.a. zahlreiche Einblicke in Themen wie vorhandene Software-Pakete, Software-Sicherheit und den richtigen Umgang mit den von Vector zur Verfügung gestellten Programmen und Anwendungen.

Mein interner Betreuer, der Herr Dipl. Ing. Markus Schwarz, stellte mir eine die Abteilung vor und begleitete mich an meinen Arbeitsplatz.

In den ersten Tagen galt meine Arbeit der Einarbeitung und Verwendung der ersten Software Tools, mit denen ich während meiner Tätigkeit umgehen musste. Unter anderem wird bei Vector eine eigene Umgebung verwendet, mit deren Hilfe man die Arbeitsstunden dokumentiert.

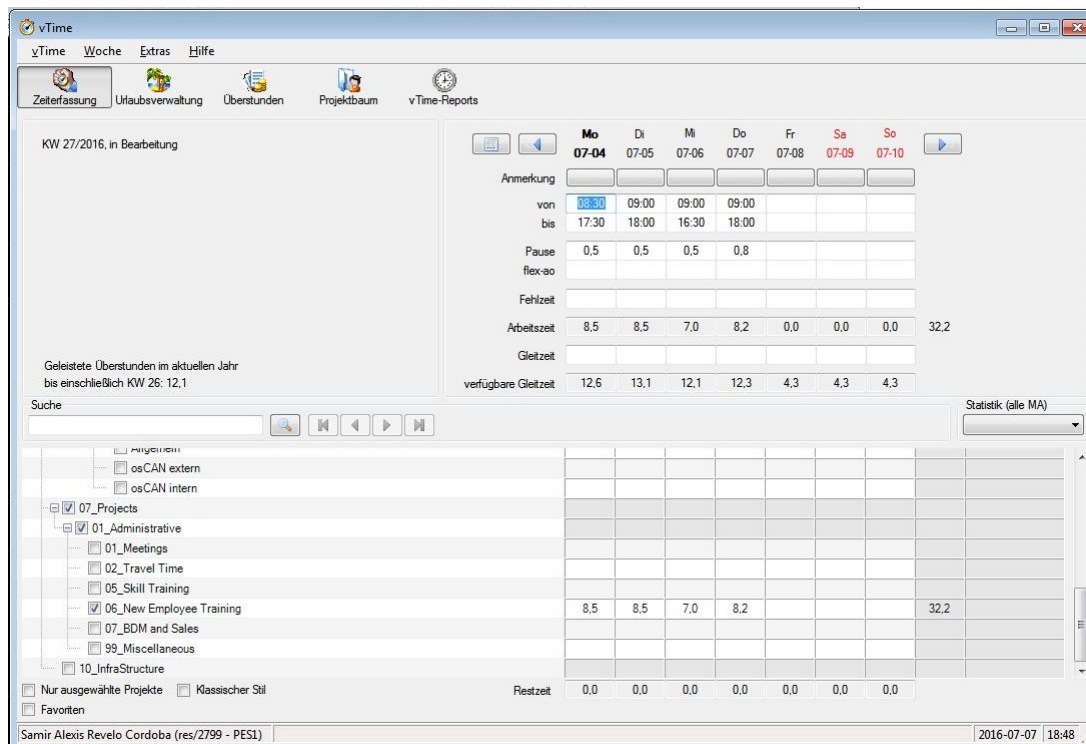


Abbildung 3.2.: viTime

Mir wurde ebenfalls die Aufgabe gestellt, mich mit den Konzepten des MISRA-C-Kodierungsstandards auseinanderzusetzen, welcher zur Entwicklung robusterer Software beiträgt.

Wie in der Einführung angegeben ist der Schwerpunkt bei der Arbeit der PES-Abteilung die Erstellung eingebetteter Software, deren Programmierung überwiegend in der Programmiersprache C erfolgt. Aufgrund der Tatsache, dass C keine sichere Programmierung von Software gewährleistet, wird seit mehreren Jahren bei Vector Informatik der C-Programmiersstandard MISRA-C: 2004 eingesetzt. Der genannte Standard wies bisher noch bestimmte Nachteile auf, welche dazu führten, dass im Jahr 2013 eine neue Version ( MISRA-C: 2012 ) verabschiedet wurde. Diese Version soll Verbesserungen und Ergänzungen zu der alten Version hervorbringen. Vektor und somit die Abteilung PES sind aus diesem Grund darauf angewiesen, einen Umstieg in die neue Version zu ermöglichen. Da dies sich nicht ohne großen Aufwand umsetzen lässt, muss dazu eine detaillierte Analyse vor allem über die Kompatibilität zwischen den beiden Versionen durchgeführt werden. Hier soll eine ausführliche Erklärung über die Aufgabe angegeben werden, bei der der Vergleich der beiden MISRA-Versionen durchgeführt wird.

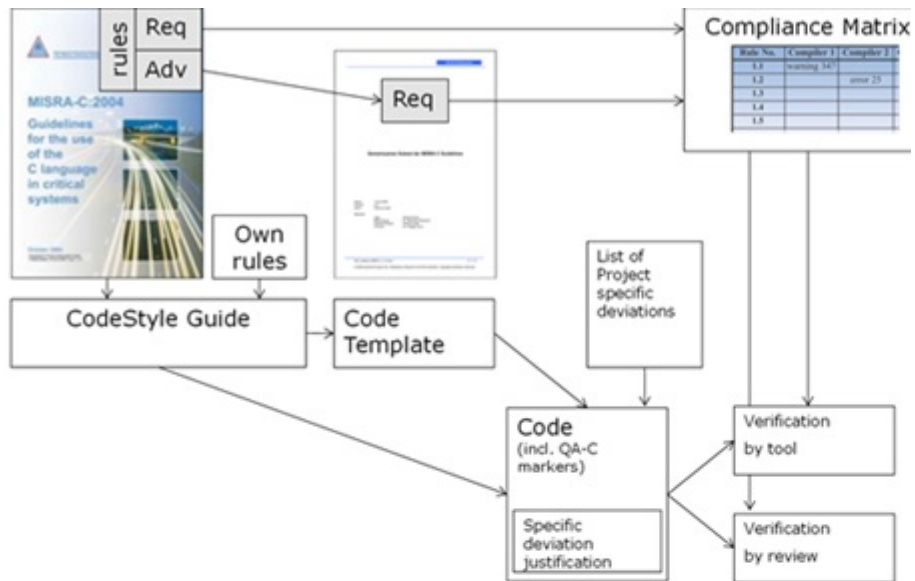


Figure 1-1 Overall MISRA process: MISRA standard - code style guides - code - compliance matrix - ...

Abbildung 3.3.: MISRA

## QAC

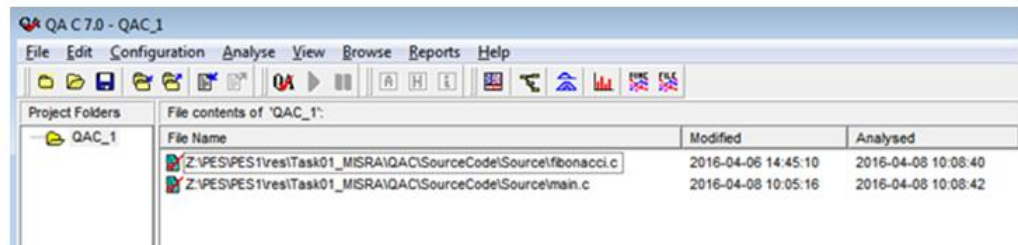


Abbildung 3.4.: QAC

---

## 3.2 2. Woche

---

### 3.2.1 Wochenübersicht

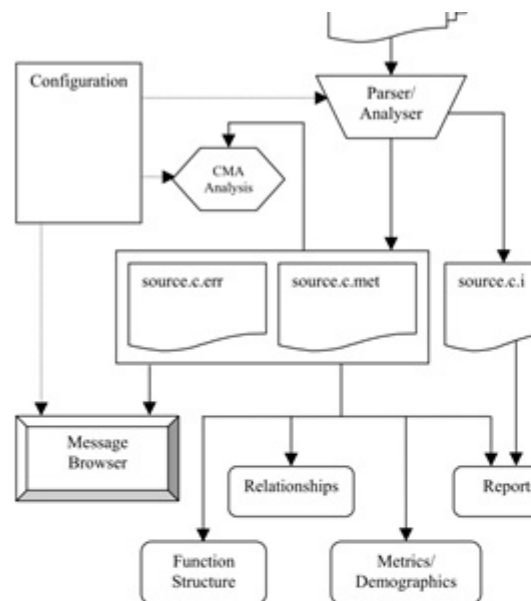
---

Tabelle

### 3.2.2 Arbeitsbericht

---

Meine Tätigkeiten bestanden vorwiegend darin, die richtigen Einstellungen zur statischen Code-Analyse einer in der Abteilung erstellten eingebetteten Beispiel-Anwendung durchzuführen. Die Analyse sollte mit Hilfe der Version 7.0 der Analyse-Software QAC erfolgen, deren Funktionalität im Beschreibungskapitel der 1. Woche angegeben wird (Dokument QAC User Guide verwenden). Diese Version wird momentan in Vector angewendet. Die folgende Graphik zeigt einen Überblick über die funktionalen Beziehungen der Analyse-Software und die unterschiedlichen Ergebnisse einer beliebigen Beispielanwendung.



**Abbildung 3.5.: QAC**

Es ist notwendig richtige Einstellungen für jedes QAC-Projekt durchzuführen, welches der Struktur eines entsprechenden Entwicklungsprojekt widerspiegelt. Ein entsprechendes Projekt beinhaltet ein oder mehrere Ordner, Quellcode-Dateien bzw. Einstellungsinformationen, die als personalities bezeichnet werden. In den personalities werden laut application note jeweils folgende Einstellungen durchgeführt: ..... Beschreibung der einzelnen Files ( $*.p_a, *.p_c, *.p_s$ ).

Inerhalb der Datei  $*.p_c$  musste ich die Definition  $_WIN32\_WINT$  anpassen, da auf dem Computer, auf dem die statische Codeanalyse durchgeführt werden soll, ein 64-bit-Betriebssystem läuft. Somit sollen manche Makros angepasst oder sogar außer Betracht gelassen werden .

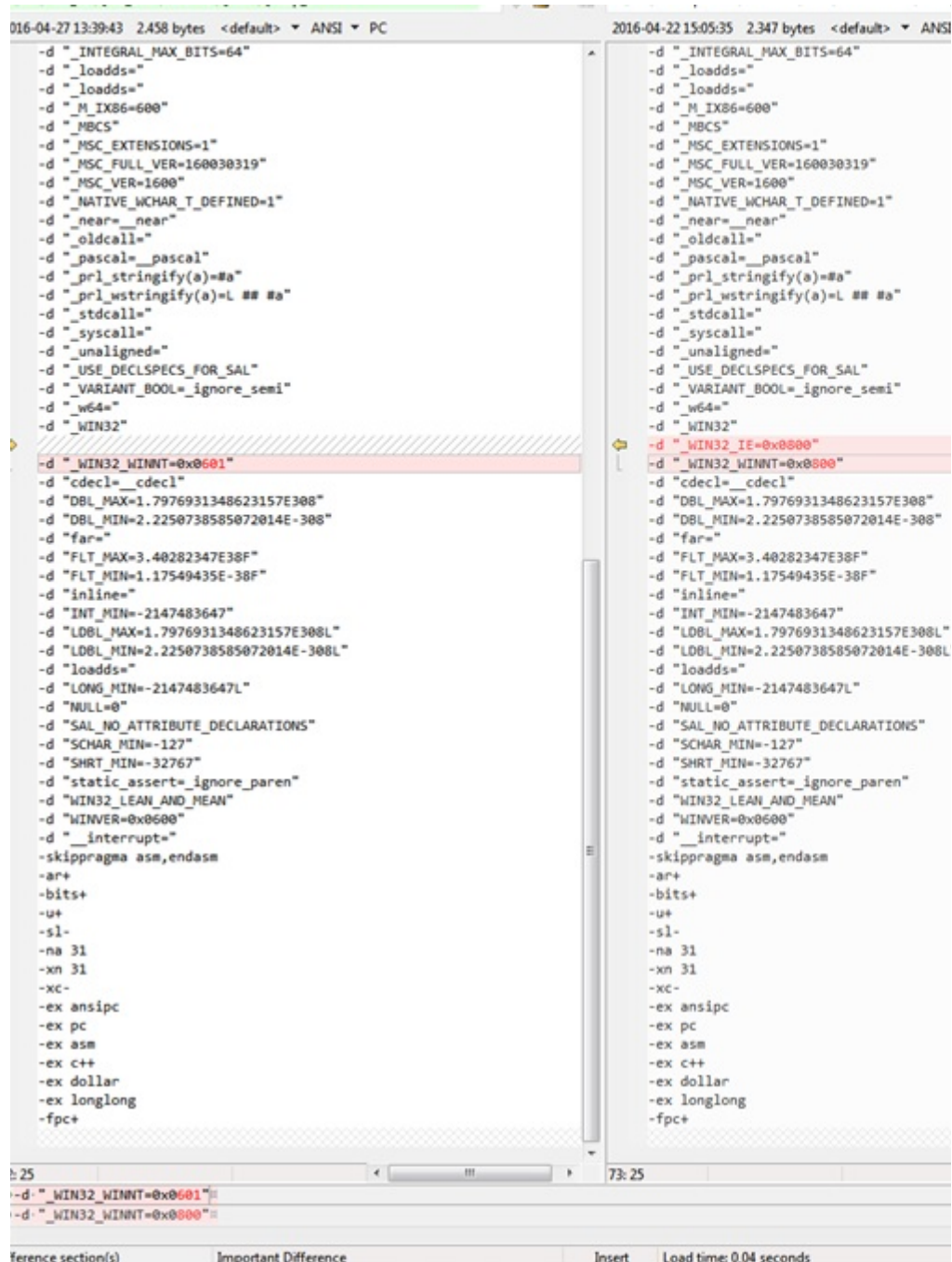


Abbildung 3.6.: QAC

Nachdem die richtigen Einstellungen für das Projekt ausgewählt und vorgenommen wurden, haben die ersten Analysen stattgefunden, welche Hinweise auf eventuell fehlerhafte Teile beim Code ausgaben.

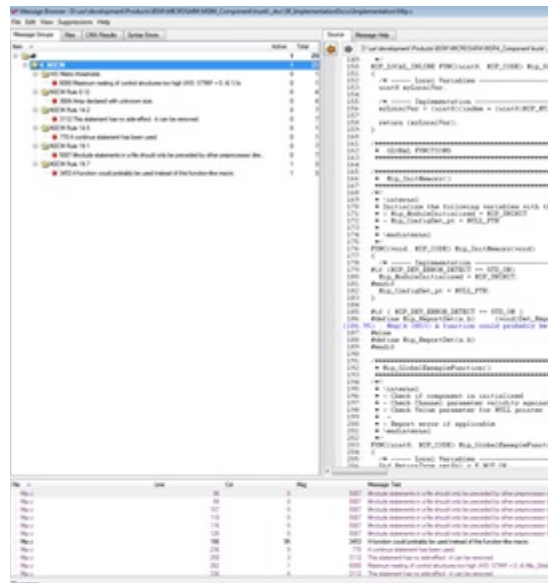


Abbildung 3.7.: QAC

Bei Vector werden unterschiedliche Programme eingesetzt, welche von jedem Mitarbeiter beherrscht werden sollen und u.a. Informationen über die Arbeitstätigkeit in der Firma der Personalabteilung weiterleiten. Diese sind unter anderem (diese Version richtet sich nach den Richtlinien der Universität. In der Version von Vector muss ich eine richtige Themengliederung angeben. Ein richtiger Bericht muss es werden): TortoiseSVN: Hier bitte eine kurze Beschreibung von Tortoise geben. Du kannst auch dazu irgendeine Präsentation von ADP benutzen, suche im Intranet danach.

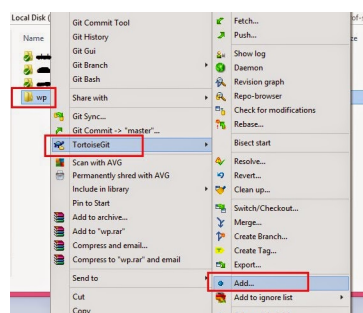


Abbildung 3.8.: Tortoisegit

### 3.3 Dritte Woche



### 3.3.1 Wochenübersicht

Tabelle

### 3.3.2 Arbeitsbericht

Während der dritten Woche im Unternehmen bekam ich den Auftrag, die release notes vom QAC Programm zu sammeln und untereinander zu vergleichen. Das Ziel war dabei, die Unterschiede zwischen der aktuell der PES-Abteilung zur Verfügung stehenden QAC-Version 7.0 und den höheren Versionen des verwendeten Programms festzustellen. Es war somit notwendig eine Methodik auszuwählen, um die genannten Unterschiede darzustellen. Es hat sich dementsprechend als sinnvoll erwiesen eine Excel-Tabelle zu erstellen, damit sich jeder einen möglichst guten Überblick über das Thema verschafft. Die Struktur der genannten Tabelle, welche in Abbildung 3.9 zu sehen ist, stellt idealerweise einen Teil der Entwicklungslinie der neuen QAC-Versionen. Insgesamt sind 6150 Nachrichten behandelt worden, die innerhalb des Entwicklungsprozesses der statischen QAC-Analysesoftware eingeführt worden sind. Unter diesen Nachrichten befinden sich jedoch auch welche, die gelöscht, angepasst oder völlig umgeschrieben wurden. Dementsprechend würde sich ohne eine geeignete Methode nur schwer erkennen lassen, welche relevanten Modifikationen die Analysesoftware erfahren hat.

	D	E	H	K	N	Q	T	V
	QAC 7.0 [actual]	QAC 7.1	QAC 7.2	QAC 8.0	QAC 8.1	QAC 9.0		
Rules	1323	1369	1399	1416	1658	1705		
N	0	46	30	23	207	8		
D	0	2	0	6	6	0		
C	0	3	4	1	7	42		
G/L/GL	0	49	47	0	0	0		
F	0	23	39	33	229	18		
new Rules								
QAC Rule	Message	checked by Vector	state	state	state	state	state	Status to
626	1332 Type or number of arguments doesn't match prototype found later.	X	X	X	X	X	X	equal
627	1333 Type or number of arguments doesn't match function definition found later.	X	X	X	X	F	X	changed
628	1334 The parameter identifiers in the prototypes of these functions/function pointers are different.	X	X	X	X	X	X	equal
629	1335 Parameter identifiers missing in function prototype declaration.		N	X	F	X	X	new
630	1336 Parameter identifiers missing in declaration of a function type.		N	X	F	X	X	new
631	1337 Function defined with a variable number of parameters.					N	X	new
632	1400 Equality comparison with enum constant from enum of different type.		X	X	X	X	X	equal
633	1401 Argument passed is constant enum not in enum type.	X	X	GL	X	X	X	changed
634	1402 Assignment of constant enum not in enum type.	X	X	GL	X	X	X	changed
635	1403 Return of constant enum not in enum type.	X	X	GL	X	X	X	changed
636	1404 Comparison with constant enum not in enum type.	X	X	X	X	X	X	equal
637	1405 Equality comparison of an enum with a numeric constant.		X	X	X	X	X	equal
638	1411 Argument passed is a numeric constant but an enum is expected.	X	X	GL	X	F	X	changed
639	1412 Constant expression assigned to an enum.	X	X	GL	X	F	X	changed
640	1413 Return of numeric constant when enum expected.	X	X	GL	X	F	X	changed
641	1414 Relational comparison of an enum with a numeric constant.		X	X	F	X	X	changed
642	1420 Equality comparison with enum object of different type.		X	X	X	X	X	equal
643	1421 Argument passed is enum object of different type.	X	X	GL	X	X	X	changed
644	1422 Assignment of enum object of different type.	X	X	GL	X	X	X	changed
645	1423 Return of enum object of different type.	X	X	GL	X	X	X	changed
646	1424 Comparison with enum object of different type.		X	X	X	X	X	equal
647	1431 Argument passed is enum when non enum type expected.	X	X	X	X	X	X	equal
648	1432 Assignment of enum to non enum type.		X	X	X	X	X	equal
649	1433 Return of enum to non enum type.	X	X	X	X	X	X	equal
650	1434 This enum constant is not representable in a 36 bit integer type.		X	X	X	X	X	equal
651	1440 Equality comparison between enum and non enum types.		X	X	X	X	X	equal
652	1441 Argument passed is non enum type when enum expected, and may be out of range.	X	X	GL	X	F	X	changed

Abbildung 3.9.: QAC

Bei der Untersuchung musste ich überprüfen...

Danach war es notwendig, nach dem Neuerwerb der QAC Version 9.0 für eine kleine Anzahl an Sourcecode (simple component) ein QAC-Projekt zu erstellen und deren Einstellungen anzupassen. Gleichzeitig war es notwendig, die Quelldateien mit der QAC Version 7.0 zu analysieren, um zu untersuchen, welche Änderungen vorkommen und was gleich geblieben ist. Folgende Tabelle wurde erstellt, beschreiben...

Datei	MISRA 2004						MISRA 2012		
	QAC 7.0			QAC 9.0			QAC 9.0		
	QAC Nachricht	MISRA Regel	Anzahl der Vorkommen	QAC Nachricht	MISRA Regel	Anzahl der Vorkommen	QAC Nachricht	MISRA Regel	Anzahl der Vorkommen
Mip.c	770	14.5	1	770	14.5	1			
	1503	14.1							
				2982	21.1	1	2982	2.2	1
	3112	14.2	7						
				3138	14.3	2			
	3453	19.7	3	3453	19.7	1	3453	Dir-4.9	1
	3684	8.12	4						
	5087	19.1	7				5087 (suppressed diagnostic)	20.1 (suppressed diagnostic)	7
Mip_Cfg.c	6080		1						
	3453	19.7	2						
Mip_Lcfg.c	3684	8.12	4						
				1290	10.1	16			
	3453	19.7	2						
	3684	8.12	4						
Mip_Pbcfg.c	5087	19.1	1				5087 (suppressed diagnostic)	20.1 (suppressed diagnostic)	1
	1504	8.1							
				1290	10.1	3			
	3453	19.7	1						
	3684	8.12	4						
	5087	19.1	1				5087 (suppressed diagnostic)	20.1 (suppressed diagnostic)	1

Abbildung 3.10.: QAC

### 3.4 Vierte Woche

#### 3.4.1 Wochenübersicht

Tabelle

#### 3.4.2 Arbeitsbericht

Während der 4. Woche musste ich mich mit den weiteren Einstellungen eines QAC-Projekts beschäftigen. Diesmal handelte es sich um ein relativ umfangreicheres Projekt, so dass es aufwendiger war, die Konfigurationsdateien anzupassen. Das Ziel war, beide Versionen am Beispiel eines umfangreicheren Projekt zu analysieren. Dieses bestand aus einer großen Anzahl an Quellcode. Die zu analysierende Dateien mussten in ein bestimmtes Verzeichnis übertragen werden,



wobei sich diese ursprünglich zur besseren Übersicht für den Kunden in sehr vielen Unterordner organisiert waren. Dies hat dazu geführt, dass ich mir eine Methode aussuche, um die Dateien nicht einzeln und per Hand in die Zieladressen zu kopieren. Es hat sich somit als sinnvoll erwiesen, eine dazu geeignete Programmiersprache zu verwenden, die mir diese Aufgabe erleichterte. Ich konnte feststellen, dass sich zum Lösen u.a. der genannten Aufgabe einige Skriptsprachen wie Perl oder Python eignen. Da bisher in der Abteilung die Mitarbeiter meistens mit Perl große Projekte erstellten und somit große Erfahrung dabei haben, habe ich mich entschieden Perl einzusetzen.



```
1 #!/D:/wimperl/perl/bin/perl.exe -w
2 use strict;
3 use File::Copy;
4 use File::Find::Rule;
5
6 my $stocopy = "D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant";
7 my $from = "D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant/BSu";
8
9 my $working = "C:/Users/visres/Desktop/PERL";
10 my $hilfsvar;
11 my @fileshelp_c;
12 my @fileshelp_h;
13
14 opendir( my $DIR, $from ) || die "can't opendir : $!";
15 my @files = grep{ $_ ne '.' and $_ ne '..' } readdir($DIR);
16 closedir $DIR;
17
18 foreach(@files) {
19
20     @fileshelp_c = <$from/$_.c>;
21     @fileshelp_h = <$from/$_.h>;
22
23     foreach(@fileshelp_c){
24
25         copy($_, $stocopy) or die "File cannot be copied.";
26         print "$_\n";
27     }
28
29     foreach(@fileshelp_h){
30
31         copy($_, $stocopy) or die "File cannot be copied.";
32         print "$_\n";
33     }
34
35     # copy(@fileshelp_c, $stocopy) or die "File cannot be copied.";
36     # print "@fileshelp1\n";
37
38 }
39
40 # foreach(@files) {
41
42     ## print "$_\n";
43
44     ## $hilfsvar=$from."/".$_."";
45     ## $hilfsvar=$from."/".$_."";
46     ## opendir( my $DIR, $hilfsvar ) || die "can't opendir : $!";
47     ## @fileshelp1 = grep{ $_ ne '.' and $_ ne '..' } readdir($DIR);
48     ## print "@fileshelp1\n";
49     ## closedir $DIR;
50
51     ## my @fileliste = glob("*.prj");
52 }
```

Abbildung 3.11.: QAC

Die Übertragung der Dateien war nicht die einzige Aufgabe, die zum Erstellen des QAC 9.0-Projektes durchgeführt werden musste. Es war außerdem notwendig die Pfade der einzelnen zu analysierenden Dateien anzugeben. Wie im Bild XX zu erkennen ist, bedient sich QAC9.0 einer XML-Konfigurationsdatei, aus der das Programm nicht nur die Pfade der Dateien entnimmt, sondern auch weitere wichtige Einstellungen, die im Prinzip sich auch über die GUI angeben

---

lassen. Diese Methode bzw. die XML-Datei zu bearbeiten, ist jedoch hilfreich, um das Aufrufen des Programms implizit beispielsweise über die Console zu ermöglichen. Dies wird in einem späteren Szenario meiner Tätigkeit nötig sein, damit man eine statische Code-Analyse für die verschiedenen Abteilungen automatisieren und somit vereinfachen kann.

---

### 3.5 Fünfte Woche

---

---

#### 3.5.1 Wochenübersicht

---

Tabelle

---

#### 3.5.2 Arbeitsbericht

---

Nachdem die Einstellungen für das oben erwähnte Projekt durchgeführt worden sind, lässt sich die statische Codeanalyse mit Hilfe der neuen QAC-Version (9.0) durchführen. Es muss dabei untersucht werden, inwiefern die aktuellen Projekte sich mit Hilfe der neuen QAC-Version analysieren lassen, ohne dass dabei beispielsweise eine große Menge von falsch positiven bzw. negativen Nachrichten ausgegeben werden.

Eine Möglichkeit, dies zu realisieren, besteht darin, ein mit Hilfe der 7.0-QAC-Version bereits analysiertes Projekt ebenfalls mit Hilfe der 9.0-QAC-Version zu analysieren und anschließend die entsprechenden Reports zu miteinander zu vergleichen.

---

### 3.6 Sexte Woche

---

---

#### 3.6.1 Wochenübersicht

---

Tabelle

---

#### 3.6.2 Arbeitsbericht

---

Die Aufgabe der letzten Woche musste fortgesetzt werden. Die Daten aus den Analyseergebnissen mussten extrahiert werden. Einerseits wird dies vom QAC-Programm selbst dadurch erledigt, dass in Form von \*.xml-Dateien die verschiedenen verletzten MISRA-Regel bzw. QAC-Warnungen ausgegeben werden. Diese Form der Datendarstellung ist für die Datenübertragung geeignet, jedoch nicht dafür, dass Trends und Zusammenhänge erkannt werden. Diese Daten müssen erstmal extrahiert und so bearbeitet werden, dass ein Mensch diese verstehen und verwenden kann.

```
1 <?xml version='1.0' encoding='UTF-8'?>
2 <prqaproject xmlversion="2" qafversion="2.0.1">
3   <!-- Config file information... -->
4   <configuration>
5     <root_paths>
6       <root_path path="{PROJECT_ROOT}" name="SOURCE_ROOT"/>
7     </root_paths>
8     <languageFamily name="C_CPP"/>
9     <baselinedSourceRoot baseline="OFF"/>
10    <baselineSnapshot id="" name="" date=""/>
11    <processMonitor>
12      <includeOptionSyntax>
13        <option string="-I"/>
14        <option string="/I"/>
15      </includeOptionSyntax>
16      <defineOptionSyntax>
17        <option string="-D"/>
18        <option string="/D"/>
19      </defineOptionSyntax>
20      <settingsOptionSyntax>
21        <option compiler="Visual Studio" string="@> quotes_escaped="yes" extension="rsp"/>
22        <option compiler="Windriver" string="-@" quotes_escaped="yes" extension=""/>
23        <option compiler="Greenhills" string="--Via" quotes_escaped="yes" extension="__i"/>
24      </settingsOptionSyntax>
25      <excludeStrings/>
26      <fileFilterStrings/>
27    </processMonitor>
28    <suppressIncludes active="no"/>
29    <acf name="default.acf"/>
30    <rcf name="m2cm-3_3_0-en_US.rcf"/>
31    <ccts>
32      <cct target="C" active="yes" name="MS_VC+_CL_16_x64_C.cct"/>
33    </ccts>
34  </configuration>
35  <!-- Mapping of extension to target_language... -->
36  <file_extensions>
37    <language target="C">
38      <extension ext=".c"/>
39    </language>
40    <language target="C">
41      <extension ext=".C"/>
42    </language>
43    <language target="C++">
44      <extension ext=".cpp"/>
45    </language>
46    <language target="C++">
47      <extension ext=".cxx"/>
48    </language>
49    <language target="C++">
50      <extension ext=".cc"/>
51    </language>
52    <language target="C++">
53      <extension ext=".CPP"/>
54    </language>
55    <language target="C++">
56      <extension ext=".CXX"/>
57    </language>
58    <language target="C++">
59      <extension ext=".CC"/>
60    </language>
61  </file_extensions>
62  <!-- Files in project... -->
63  <files>
64    <!-- Explicit files... -->
65    <file target="C" name="Adc.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
```

Abbildung 3.12.: QAC



```
C:\Users\visres\Desktop\PERL\prqproject_copy.xml - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?

prqproject_copy.xml
71 <file target="C" name="CanSM.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
72 <file target="C" name="CanTp.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
73 <file target="C" name="CanTcv_30_GenericCan.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
74 <file target="C" name="CanTsyn.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
75 <file target="C" name="CanXcp.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
76 <file target="C" name="Com.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
77 <file target="C" name="ComM.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
78 <file target="C" name="Crc.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
79 <file target="C" name="Cry.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
80 <file target="C" name="Cry_AesDecrypt128.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
81 <file target="C" name="Cry_AesEncrypt128.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
82 <file target="C" name="Cry_Fips186.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
83 <file target="C" name="Cry_HmacSha1.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
84 <file target="C" name="Cry_RsaDecrypt1024.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
85 <file target="C" name="Cry_RsaSha1SigVer.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
86 <file target="C" name="Csm.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
87 <file target="C" name="Dbg.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
88 <file target="C" name="Dcm.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
89 <file target="C" name="Dcm_Ext.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
90 <file target="C" name="Dcm.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
91 <file target="C" name="Det.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
92 <file target="C" name="Dio.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
93 <file target="C" name="Dlt.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
94 <file target="C" name="DoIP.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
95 <file target="C" name="E2E.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
96 <file target="C" name="E2E_P01.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
97 <file target="C" name="E2E_P02.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
98 <file target="C" name="E2E_P04.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
99 <file target="C" name="E2E_P05.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
100 <file target="C" name="E2E_P06.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
101 <file target="C" name="E2E_SM.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
102 <file target="C" name="Ea.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
103 <file target="C" name="Ea_BlockHandler.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
104 <file target="C" name="Ea_DatasetHandler.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
105 <file target="C" name="Ea_EepCoordinator.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
106 <file target="C" name="Ea_InstanceHandler.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
107 <file target="C" name="Ea_Layer1_Erase.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
108 <file target="C" name="Ea_Layer1_Invalidate.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
109 <file target="C" name="Ea_Layer1_Read.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
110 <file target="C" name="Ea_Layer1_Write.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
111 <file target="C" name="Ea_Layer2_InstanceFinder.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
112 <file target="C" name="Ea_Layer2_InvalidateInstance.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
113 <file target="C" name="Ea_Layer2_WriteInstance.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
114 <file target="C" name="Ea_Layer3_ReadManagementBytes.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
115 <file target="C" name="Ea_PartitionHandler.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
116 <file target="C" name="Ea_TaskManager.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
117 <file target="C" name="EcuM.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
118 <file target="C" name="Eep.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
119 <file target="C" name="Eth.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
120 <file target="C" name="EthIf.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
121 <file target="C" name="EthIf_ExtndTrafficHndl.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
122 <file target="C" name="EthIf_Ptp.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
123 <file target="C" name="EthIf_Switch.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
124 <file target="C" name="EthSM.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
125 <file target="C" name="EthTcv_30_Canoeemu.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
126 <file target="C" name="EthTsyn.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
127 <file target="C" name="Fee.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
128 <file target="C" name="Fee_ChunkInfo.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
129 <file target="C" name="Fee_Partition.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
130 <file target="C" name="Fee_Processing.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
131 <file target="C" name="Fee_Sector.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
132 <file target="C" name="FilM.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
133 <file target="C" name="Fls.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
134 <file target="C" name="Fr.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
135 <file target="C" name="Fr_Irq.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
136 <file target="C" name="Fr_Stack.c" folder="D:/usr/Task_01/QAC_Eval/Baseline/Code/Relevant"/>
```

Abbildung 3.13.: QAC

---

Eine \*.xml-Datei lässt sich mit Hilfe einer xslt-Transformation so bearbeiten, dass die enthaltenen Daten als eine Tabelle und beispielsweise als eine \*.txt-Datei dargestellt werden kann. (Hierbei muss ich eine genaue Beschreibung der XSLT-Transformation angeben)

Eine Art, die Daten besser darzustellen sind Excel-Tabellen, wie in der unteren Tabelle zu erkennen ist. Die in den beschriebenen \*.xml-Dateien enthaltenen Daten können dadurch besser miteinander verglichen werden.

Ich musste auch anfangen, mich mit den qaccli-Befehlen zu beschäftigen, da diese auch eine wichtige Rolle spielen, wenn man über die Konsole eine automatische statische Analyse ermöglichen möchte. (Hier sollte ich versuchen, Markus Bilder wiederzugeben). Die wichtigsten QACLI befehle auch hier angeben.

Als letztes habe ich die erstellten PERL- bzw. Batch-Dateien zum Aufrufen letzterer in dem Netzwerkordner gespeichert!

---

## 3.7 Siebte Woche

---

---

### 3.7.1 Wochenübersicht

---

Tabelle

---

### 3.7.2 Arbeitsbericht

---

Markus hat mir am Dienstag (Montag war ja Feiertag) die zweite Aufgabe vorgestellt. Dabei handelte es sich um das Modifizieren bzw. das Anpassen eines Frameworks zum automatisierten Testen von Software-Module. Ein sogenanntes Basic Test Environment (BTE) gehört zu dem genannten Testframework (Bilder hinzufügen), welches hardware unabhängig Software-Komponenten auf einem beliebigen PC testet. Bei den Tests werden wie in der Abbildung gezeigt von der BTE Nachrichten aus den Testergebnissen erstellt und in verschiedenen Report-Dateien aufgelegt. Auf diese Art und Weise kann der Software-Tester oder der Anwender einfacher überprüfen, in welchen Stellen des getesteten Codes es fehlerhafte Module gibt oder Warnungen auftauchen.

Zu meiner Aufgabe gehört, diese Möglichkeit so zu erweitern, dass die BTE auch auf einer beliebigen Hardware läuft, d.h. hardwareunabhängig, wobei die Möglichkeit geboten sein muss, entsprechende Reports zu erstellen. Die Reports können in diesem Fall nicht mehr als entsprechende Dateien auf dem Rechner abgelegt werden, viel mehr muss eine Methode gefunden werden, wie der Report auf der RAM temporär zu speichern ist.

---

## 3.8 Achte Woche

---

---

### 3.8.1 Wochenübersicht

---

Tabelle

---

### 3.8.2 Arbeitsbericht

---

In der BTE Software sind Methoden (sprintf, printf) verwendet, die die formatierte Ein- und Ausgabe von Strings ermöglichen. Zudem werden Methodenaufrufe wie fprintf oder fopen eingesetzt, die der Umgang mit Streams bzw. die Erstellung einer Datei ermöglichen.

Zwei Möglichkeiten können betrachtet werden, um mit den knappen Softwareressourcen der Embedded Platform umzugehen:

- Man bindet entsprechende Ersatzbibliotheken ein, bei denen der Verbrauch der Ressourcen in Grenzen gehalten wird.
- Man bietet dem Anwender die Möglichkeit an, durch entsprechende Routinen diejenigen Teile vom Code beim Kompiliervorgang auszublenden, die nicht einzusetzen sind.

Die zweite Lösung wird so angewendet, dass die nicht kompatiblen Codestellen mit vorhandenen und zusätzlich eingebauten Präprozessor-Direktiven #define "auskommentiert,,werden.

XXX Bild zur Architektur des Reports, welches auf der RAM gespeichert wird.

Die Stellen vom Speicher, wo die zu interessierenden Daten der durchgeführten Tests zum Erstellen eines Reports abgelegt wurden, lassen sich mit Hilfe eines memory mapping des entsprechenden Prozessors analysieren. Dies ist mit einem Debugger möglich und kann bestenfalls als binary file ausgegeben werden. Das ist die Beschreibung des Ansatzs, welcher von Markus und Timo erklärt worden ist.

Ein Ansatz, um aus den binary files ein Report zu erstellen, ist, das binary File mit Hilfe einer Skriptsprache zu analysieren, damit das ursprüngliche von der BTE auf dem PC erzeugte Report erneut ausgegeben werden kann. Die genaue Darstellung des Perl-Scripts (Perl-Listing) kann im Anhang C gefunden werden.

Die erzielten Lerneffekte sind vor allem der Umgang mit einer Skriptsprache, das Speicherplatz sparend Codieren in C.

---

## 3.9 Neunte Woche

---

---

### 3.9.1 Wochenübersicht

---

Tabelle

---

### 3.9.2 Arbeitsbericht

---

Da die Reports auf einer beliebigen Hardware zu implementieren sind, sollte eine bestimmte Hardware ausgesucht werden. Die Hardware die mir vorgestellt wurde, war das STM32F4-Entwicklungsboard. Mit dieser Hardware habe ich in der Vergangenheit schon zu tun gehabt, deswegen konnte ich ohne große Mühe die Einstellungen vornehmen, um ein Projekt zu starten.

Das genannte Board lässt sich über ST-Link-Treiber flaschen, welcher vom Hersteller zur Verfügung gestellt wird. Der auf dem Mikrocontroller erzeugte Report sollte auf einer Speicher sparenden Art und Weise erstellt werden. Man hat sich 2 Möglichkeiten überlegt, wie dies geschehen soll. Letzteres wird im Folgenden Bild wiedergeben:

Zu der Erkenntnis konnte man gelangen, dass die zweite Möglichkeit besser geeignet ist und auf einer seriellen Übertragung der Daten basiert.

---

## 3.10 Zehnte Woche

---

---

### 3.10.1 Wochenübersicht

---

Tabelle

---

### 3.10.2 Arbeitsbericht

---

In der vorliegenden Woche war meine Aufgabe das Perl Programm zu pflegen und ein wenig besser zu strukturieren. Außerdem wurde mir erklärt, dass die Messages, die über die BTE durchgeschaltet, jedoch nicht in die LogListe registriert werden, auch in dem Report vorkommen sollten. Dies geschieht auch bei einem normalen Report, welcher bei einem Test auf dem Computer erzeugt wird. Dies musste ebenfalls im Programm implementiert werden.

Die aktuelle Version meiner Anwendung erzeugt das auf dem linken Teil gezeigte File im xml-Format. Das rechts davon gezeigte File ist das Report, welcher wie bereits oben erwähnt bei einem Test auf dem Computer erzeugt wird. Die Unterschiede lassen sich dadurch erklären, dass manche Features, die im auf dem Computer erstellten Report in dem auf der RAM gespeicherten Report nicht relevant sind und somit nicht darzustellen sind. Das Bild zum UML Statechart sollte ich mal einfügen und von Markus checken lassen und endgültig hier einfügen.

Am Ende erfolgte eine Abgabe der letzten Version meiner programmierten PERL-Anwendung abgegeben werden. Markus hat manche Korrekturen und Verbesserungen durchgeführt und mir dann diesbezüglich Rückmeldung gegeben.

---

## 3.11 Elfte Woche

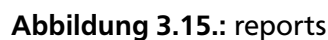
---





Abbildung 3.14.: eclipsegit





---

### 3.11.1 Wochenübersicht

---

Tabelle

---

### 3.11.2 Arbeitsbericht

---

Diese Woche wurde mir die dritte Aufgabe vorgestellt. Dabei ginge es darum mich mit der Arduino Anwendung und der Einstellung eines geeigneten SCI-Treiber zu beschäftigen. Die Programmierung von Win32-Anwendungen ist nicht trivial, dabei ist man auf die Verwendung von Funktionen angewiesen, die von Windows zur Verfügung gestellt werden.

Hierbei beschreiben, was Win32 ist und wieso ich nicht andere Module verwenden könnte.

Dabei hat die Programmierung der SCI-Anwendung in VS erfolgt, ich habe mich entschieden das Arduino Projekt nicht wie Andreas vorgeschlagen hat, sondern in AtmelStudio umzusetzen. Die Atmelstudio Anwendung und die Einstellung eines Projektes mit den external configurations beschreiben.

Der programmierte Win32-Treiber arbeitet so, dass die aufzurufende Funktion so lange wartet bis die aufgerufene fertig ist.(waiting rendezvous) nach [1]. Die dabei verwendete Methode(...) arbeitet als "timed rendezvous", wartet also nur eine bestimmte Zeit bis benötigte Task antwortet, ansonsten bricht sie ihren Methodenaufruf ab.

---

## 4 Zusammenfassung und Ausblick

Moderne Dieselmotoren werden bei der Applikation hinsichtlich verschiedenster Kriterien optimiert. Neben den Schadstoff-Emissionen zählen hierzu sowohl das Motordrehmoment, der Wirkungsgrad als auch die Geräuschemissionen. Häufig stehen diese Anforderungen zueinander im Widerspruch und eine Reduzierung der Geräuschemissionen zieht in der Regel eine Drehmomentsenkung nach sich, weshalb geeignete Kompromisse gefunden werden müssen.

Bei der vorliegenden Arbeit wird eine stationäre, modellbasierte Optimierung durchgeführt. Die Optimierung hat als Ziel, die Motorstellgrößen  $u_{VTG}$ ,  $u_{HDAGR}$ ,  $u_{NDAGR}$ ,  $\phi_{HE}$  und  $\phi_{VE}$  in einem vorgegebenen Betriebspunkt so zu wählen, dass sowohl das innermotorische Drehmoment  $M_Z$ , der Motorwirkungsgrad<sup>1</sup>  $\eta_A$  und die Geräuschemissionen<sup>2</sup> optimal werden.

Die stationäre Optimierung wird mit Hilfe von zwei am Institut für Automatisierungstechnik (IAT) der TU-Darmstadt entwickelten Simulationsmodellen eines Z19DTH (4V) - Dieselmotors<sup>3</sup> durchgeführt. Mit den genannten Modellen können sowohl die Zustandsgrößen vom Luftpfad als auch von der Verbrennung<sup>4</sup> berechnet werden. Zunächst werden zwecks des Aufbaus eines gesamten Motormodells und somit der modellbasierten Optimierung beide zur Verfügung gestellten Simulationsmodelle zusammengefügt.

Weiterhin wird im Rahmen dieser Arbeit auf Basis einer Literaturrecherche untersucht, wie aus dem Zylinderdruckverlauf Kenngrößen generiert werden können, die es ermöglichen, sowohl das Drehmoment, den Wirkungsgrad als auch die Geräuschemissionen am Beispiel des Dieselmotors zu quantifizieren.

Daran anschließend wird zur Analyse übergegangen, inwieweit sich die oben genannten Stellgrößen auf die zu optimierenden Zielgrößen auswirken.

Die stationäre Grundoptimierung wurde bisher meist von Prüfstandsingenieuren anhand der Erfahrung mit ähnlichen Motoren online am Prozess durchgeführt. Diese Vorgehensweise ist sehr zeit- und kostenaufwändig, da zur Optimierung der oben genannten Zielgrößen immer mehr Stellmöglichkeiten berücksichtigt werden müssen. In dem in dieser Arbeit gewählten Ansatz wird das Prozessverhalten, wie oben erwähnt, durch ein Simulationsmodell approximiert, so dass eine Offline-Optimierung im Rechner durchgeführt werden kann.

Um die Offline-Optimierung durchführen zu können, war es zweckmäßig, den Begriff der mono- bzw. multikriteriellen Optimierung einzuführen.

---

<sup>1</sup> vgl. ??

<sup>2</sup> Die Geräuschemissionen werden in dieser Arbeit durch eine ausführliche Untersuchung des Zylinderdruckgradienten berücksichtigt. Eine detaillierte Ausführung wird in Kapitel ?? gegeben

<sup>3</sup> Eine detaillierte Ausführung der Motorkenndaten erfolgt im Anhang B

<sup>4</sup> Für nähere Informationen wird auf Kapitel ?? hingewiesen.

---

Eine monokriterielle Optimierung wird zunächst hinsichtlich der Erhöhung des Durchschnittsdrehmoments  $\overline{M}_Z$  durchgeführt. Dabei stellt man fest, dass eine Verbesserung der genannten Kenngröße durch den Einsatz der VTG-Aufladung erfolgt, die weiteren Stellgrößen, im Speziellen  $u_{HDAGR}$  und  $u_{NDAGR}$ , können demgegenüber keine wesentliche Erhöhung von  $\overline{M}_Z$  erzielen. Letzteres widerspricht den im Kapitel ?? gewonnenen Erkenntnissen, denn bei der dort durchgeführten Analyse konnte eine Verbesserung von  $\overline{M}_Z$  durch den Einsatz einer AGR-Rate festgestellt werden.

Ein multikriterielles Optimierungsverfahren wird verwendet, um die Motorstellgrößen  $u_{VTG}$ ,  $u_{HDAGR}$ ,  $u_{NDAGR}$ ,  $\phi_{HE}$  und  $\phi_{VE}$  in einem vorgegebenen Betriebspunkt so zu wählen, dass sowohl das innermotorische Drehmoment  $M_Z$ , der Motorwirkungsgrad<sup>5</sup>  $\eta_A$  und die Geräuschemissionen<sup>6</sup> optimal werden. Es ist dabei zu erkennen, dass die Verwendung von rückgeführtem Abgas in bestimmten Arbeitspunkten zu einer Entschärfung des Zielkonfliktes zwischen  $\overline{M}_Z$  und  $\frac{dp_{Z,max}}{d\phi}$  führt.

Aus der im Kapitel ?? dargestellten Druckverlaufsanalyse der Prüfstandsmessdaten wurden die oben beschriebenen Ergebnisse bestätigt. Eine Erhöhung des Motordrehmoments und somit des innermotorischen Wirkungsgrades wird nämlich festgestellt, nachdem eine bestimmte, nicht zu vernachlässigende AGR-Rate stattgefunden hat. Das Potential, den innermotorischen Wirkungsgrad durch den Einsatz einer AGR-Rate  $x_{AGR}$  zu erhöhen, kann demzufolge durch die genannte Analyse der gemessenen Druckverläufe begründet werden.

---

<sup>5</sup> vgl. ??

<sup>6</sup> Die Geräuschemissionen werden in dieser Arbeit durch eine ausführliche Untersuchung des Zylinderdruckgradienten berücksichtigt. Eine detaillierte Ausführung wird in Kapitel ?? gegeben

---

# A Motorprüfstand des IAT

Die experimentelle Untersuchung des Versuchsmotors erfolgt am hochdynamischen Motorprüfstand des Instituts für Automatisierungstechnik der TU Darmstadt ???. Es handelt sich dabei um einen modern ausgestatteten Prüfstand bei dem die Versuchsträger auf Rollenwagen aufgebaut und mit Schnellkupplungen für die Kraftstoff- und Kühlwasserversorgung versehen sind. Das Rollenwagenkonzept ermöglicht einen Austausch der Versuchsmotoren in kürzester Zeit.

Eine Asynchronmaschine zählt zu den weiteren Bestandteile des Prüfstands, denn durch sie erfolgt die Belastung des Motors. Diese wird üblicherweise drehzahlregelt betrieben. Sie erzeugt abhängig vom gewählten Motorbetriebspunkt ein positives oder negatives Belastungsmoment (generatorischer bzw. motorischer Betrieb). Die Momentenanregelzeit des Systems aus Umrichter und Asynchronmaschine liegt unter 5ms. Der Prüfstand ist damit insbesondere für die Untersuchung dynamischer Fahrzustände geeignet.

Ein RCP-System<sup>1</sup> steuert den Frequenzumrichter der Asynchronmaschine, die Abgasanlage, die Messtechnik und regelt das Kühlsystem [2].

---

<sup>1</sup> engl. Rapid Control Prototyping



---

## B Verwendeter Dieselmotor und entsprechendes Simulationsmodell

---

### B.1 Motordaten

---

Der Versuchsmotor ist ein Opel/Fiat 1.9l Common-Rail Dieselmotor. Es ist mit einer Hochdruckabgasrückführung und einem Abgasturbolader mit variablen Turbinengeometrie ausgestattet. Eine Niederdruckabgasrückführung-Einrichtung wurde nachträglich eingebaut.

Hersteller	GM/Opel und Fiat
Motorbezeichnung	Z19DTH (4V)
Kreisprozess	Viertaktmotor
Zylinderzahl	4
Hubraum	$1910 \text{ cm}^3$
Verdichtungsverhältnis	17.5
Max. Leistung (bei $4000 \text{ min}^{-1}$ )	110 kW
Max. Drehmoment (bei $2000 \text{ min}^{-1}$ )	315 Nm
Leerlaufdrehzahl	$850 \text{ min}^{-1}$
Maximale Drehzahl	$5100 \text{ min}^{-1}$

---

### B.2 Verwendetes Simulink-Modell des Dieselerbrennungsmotors

---

In Abbildung B.1 ist der modellierte Dieselmotor dargestellt. Das links im Bild gelegene Luftpfadmodell liefert dem Verbrennungsmodell (rechte Seite) die entsprechenden Luftpfadgrößen Gaszusammensetzung  $x_{2EB}$  bzw.  $x_3$ , die Temperatur  $T_{2EB}$  bzw.  $T_3$  und die Drücke  $P_{2EB}$  bzw.  $P_3$  des jeweiligen Einlass- und Auslassbehälters. Das Verbrennungsmodell führt nach einem Arbeitsspiel die berechnete SPL zurück zum Luftpfadmodell.

**Abbildung B.1.:** gesamtes Simulink-Modell des Verbrennungsmotors





# C Softwaredokumentation

Im Rahmen dieser Arbeit sind folgende Software-Tools entwickelt worden:

- **Schwerpunktlage der Verbrennung**  
SPL = function\_SPL(phi\_mod,QB,abtastzeit\_s,nMot) berechnet.
- **Zylinderdruckgradient**  
[dp\_dphi,ddp\_ddphi] = Grad\_dGrad\_pZMax(phi,pZ)

Die oben genannten Tools werden im Folgenden näher erläutert.

---

## C.1 Berechnung der Schwerpunktlage der Verbrennung $\phi_{Q50}$

---

Um die Kenngröße SPL in Echtzeit ermitteln zu können, sind die momentanen Werte des simulierten integralen Brennverlaufs  $Q_B$  und des Kurbelwinkels  $\phi$  nötig. Diese Kenngrößen werden vom Verbrennungsmodell ermittelt und können dementsprechend benutzt werden. In Abbildung C.1 ist die entsprechende Embdedded MATLAB Function zu erkennen, mit deren Hilfe die SPL berechnet wird. Der jeweilige Quellcode wird im Listing C.1 aufgezeigt.

**Abbildung C.1.:** Embdedded-MATLAB Function zur Berechnung der Schwerpunktlage der Verbrennung  $\phi_{Q50}$

---

### Listing C.1: Embdedded MATLAB Function

---

```
function SPL = function_SPL(phi_mod,QB,abtastzeit_s,nMot)
%#codegen

5  abtastzeit_radKW=2*pi*nMot*abtastzeit_s/60; % Abtastzeit Sek. -> →
    ←rad KW
Anzahl_abtast_radKW=ceil(4*pi/abtastzeit_radKW); % Gesamte Anzahl →
    ←Abtastschritte in rad KW

persistent QB_toSave phi_toSave counter SPL_persistent hilfe_phi

10 if isempty(QB_toSave) % Initialisierung der persistent→
    ←Variablen
    QB_toSave=zeros(Anzahl_abtast_radKW+3,1);
```

```

    hilfe_phi=0;
    counter=1;
end
15
if isempty(phi_toSave)      % Initialisierung der persistent →
    ←Variablen
    SPL_persistent=0;
    probe=0;
    phi_toSave=zeros(Anzahl_abtast_radKW+3,1);
20
end

if phi_mod == 0 % Die relevanten Daten werden bei phi_mod==0 nur →
    ←einmal gespeichert.
    QB_toSave(counter,1) = QB;
25
    phi_toSave(counter,1) = phi_mod;
    hilfe_phi=phi_mod; % Diese Variable hilft dabei, das Ende →
        ←eines Arbeitsspiels zu erkennen.
    counter=counter+1;

elseif hilfe_phi < phi_mod % Ist ein Arbeitsspiel noch nicht →
    ←vorbei, dann werden die relevanten Daten weiter gespeichert.
30
    QB_toSave(counter,1) = QB;
    phi_toSave(counter,1) = phi_mod;
    hilfe_phi=phi_mod;
    counter=counter+1;

35 elseif hilfe_phi > phi_mod % Ist ein Arbeitsspiel vorbei, dann wird →
    ← aus den gespeicherten Daten die SPL berechnet.

    check=max(QB_toSave)/2; % Die Hälfte des integralen →
        ←Heizverlaufs wird bestimmt.

    for i=1:counter          % Der Index wird gesucht, bei dem die →
        ←Hälfte des int. Brennverlauf gespeichert wurde. →
        ←Anschliessend wird dieser Index benutzt, um die SPL zu →
        ←bestimmen.
40
        if(QB_toSave(i,1) > check) %
            SPL_persistent=(phi_toSave(i,1)-pi)*180/pi; % SPL wird →
                ←als °KW ausgegeben

```

---

```
        hilfe_phi=phi_mod;
        counter=1;
        break;
45    elseif (i==counter)      % Diese if-Anfrage, hilft dabei, →
        ←die Variablen neu zu setzen, falls bei der 1. if-→
        ←Anfrage die SPL nicht gefunden werden konnte.
        hilfe_phi=phi_mod;
        QB_toSave=zeros(Anzahl_abtast_radKW+3,1);
        counter=1;
        SPL_persistent=0;
50    phi_toSave=zeros(Anzahl_abtast_radKW+3,1);
        end
    end
end
SPL=SPL_persistent; % SPL wird ausgegeben
```

---

---

## C.2 Berechnung des Druckgradienten

---

Für die Echtzeitbestimmung der Kenngrößen Druckgradientenverlauf  $\frac{dp}{d\phi}$  und deren zeitlichen Ableitung  $\frac{d^2p}{d\phi^2}$  sind die momentanen Werte des simulierten Zylinderdrucks  $p_Z$  und des Kurbelwinkels  $\phi$  nötig. Diese Kenngrößen werden vom Verbrennungsmodell ermittelt und können dementsprechend verarbeitet werden. In Abbildung C.2 ist die entsprechende Embdedded MATLAB Function zu erkennen. Der jeweilige Quellcode wird im Listing C.2 aufgezeigt.

**Abbildung C.2.:** Embdedded-MATLAB Function zur Berechnung des Druckgradienten  $\frac{dp_Z}{d\phi}$

---

### Listing C.2: Embdedded MATLAB Function

---

```
function [dp_dphi,ddp_ddphi] = Grad_dGrad(phi,pZ)
%#codegen

persistent counter phi_toSave pZ_toSave dp_toSave

5
if isempty(counter) % persistent-Variablen werden initialisiert.
    counter=0;
    phi_toSave=phi;
    pZ_toSave=pZ;
10    dp_toSave=0;
end

if (counter == 1) && (phi > phi_toSave) % Ab hier beginnt die →
    % Gradientenbestimmung.
    dp_dphi=(pZ_toSave-pZ)/(phi_toSave-phi); % Alleresrte →
    % Druckgradientbestimmung.
15    dp_toSave=dp_dphi; % Vorherige relevante Werte werden →
    % gespeichert.
    pZ_toSave=pZ;
    phi_toSave=phi;
    counter=counter+1;
    return
20 elseif (counter == 2) && (phi > phi_toSave) % Ab hier beginnt die →
    % 2. zeitliche Ableitung des Drucks, gleichzeitig wird die →
    % Druckgradientbestimmung fortgefahen.
    dp_dphi=(pZ_toSave-pZ)/(phi_toSave-phi); % →
    % Druckgradientenbestimmung.
```

---

```
    ddp_ddphi=(dp_toSave-dp_dphi)/(phi_toSave-phi); % 2. zeitliche →  
        ←Ableitung des Drucks.  
    dp_toSave=dp_dphi;  
    pZ_toSave=pZ;  
25    phi_toSave=phi;  
    return  
elseif counter == 0 % Anfangswerte von Hilfsvariablen werden →  
    ←festgelegt.  
    counter=counter+1;  
    dp_dphi=0;  
30    ddp_ddphi=0;  
    return  
end  
dp_dphi=0; % Falls ein Fehler auftritt, betragen beide →  
    ←Kenngröße 0.  
ddp_ddphi=0;
```

---



## D Prüfstandsmessdaten

In folgender Tabelle sind sowohl die Luftpfadsgrößen des Einlassbehälters als auch der Winkel der Haupteinpritzung aufgelistet, die im Betriebspunkt ( $n_{mot} = 1500 \text{ min}^{-1}$ ;  $q_{inj} = 10 \text{ mm}^3/\text{inj}$ ) variiert und vermessen wurden.

Messung Nr.	$s\phi_{HE} [^\circ \text{KW v.OT}]$	$\lambda$	$T_{2EB} [\text{K}]$	$\frac{dm_L}{dt} [\frac{\text{kg}}{\text{s}}]$	$\frac{dm_{Gas}}{dt} [\frac{\text{kg}}{\text{s}}]$	$x_{AGR} [-]$
1	-1	4.8	291	0.024	0.024	0
2	1	3.7	291.4	0.024	0.024	0
3	3	3.6	291.4	0.024	0.024	0
4	5	3.58	291.6	0.024	0.024	0
5	7	3.55	291.3	0.024	0.024	0
6	9	3.52	291.2	0.024	0.024	0
7	11	3.56	291.5	0.024	0.024	0
8	1	3.35	291.5	0.02	0.025	0.2
9	3	3.1	292.3	0.02	0.025	0.2
10	5	3.03	292.5	0.02	0.025	0.2
11	7	3	292.2	0.02	0.025	0.2
12	9	3	291.8	0.02	0.025	0.2
13	11	3	291.9	0.02	0.025	0.2
14	1	2.3	338	0.013	0.022	0.041
15	3	2.09	340.2	0.013	0.022	0.041
16	5	2.06	340.8	0.013	0.022	0.041
17	7	2.03	330	0.013	0.022	0.041
18	9	1.98	338	0.013	0.022	0.041
19	11	2.02	337	0.013	0.022	0.041
20	13	2.04	336	0.013	0.022	0.041
21	15	2.06	335.5	0.013	0.022	0.041
22	5	1.92	315	0.01	0.023	0.565
23	7	1.71	315.5	0.01	0.023	0.565
24	9	1.62	315	0.01	0.023	0.565
25	11	1.62	315	0.01	0.023	0.565

Messung Nr.	$s\phi_{HE} [^\circ\text{KW v.OT}]$	$\lambda$	$T_{2EB} [\text{K}]$	$\frac{dm_L}{dt} [\frac{\text{kg}}{\text{s}}]$	$\frac{dm_{Gas}}{dt} [\frac{\text{kg}}{\text{s}}]$	$x_{AGR} [-]$
26	13	1.64	313	0.01	0.023	0.565
27	15	1.65	313	0.01	0.023	0.565
28	17	1.68	312.5	0.01	0.023	0.565
29	19	1.68	313	0.01	0.023	0.565
30	21	1.7	313	0.01	0.023	0.565
31	19	1.33	338	0.008	0.021	0.62
32	21	1.34	338	0.008	0.021	0.62
33	23	1.36	338	0.008	0.021	0.62
34	25	1.36	338	0.008	0.021	0.62
35	27	1.36	336	0.008	0.021	0.62
36	29	1.37	337	0.008	0.021	0.62

**Tabelle D.1.:** Luftpfadsgrößen des Einlassbehälters und Winkel der Haupteinpritzung, bei denen die Druckverläufe am Prüfstand gemessen wurden. Betriebspunkt ( $n_{mot} = 1500\text{min}^{-1}$ ;  $q_{inj} = 10\text{mm}^3/\text{inj}$ ).



---

# Abbildungsverzeichnis

2.1. Vectorabteilungen1 . . . . .	3
3.1. Zeichnung1 . . . . .	5
3.2. viTime . . . . .	6
3.3. MISRA . . . . .	7
3.4. QAC . . . . .	7
3.5. QAC . . . . .	8
3.6. QAC . . . . .	9
3.7. QAC . . . . .	10
3.8. Tortoisegit . . . . .	10
3.9. QAC . . . . .	11
3.10.QAC . . . . .	12
3.11.QAC . . . . .	13
3.12.QAC . . . . .	15
3.13.QAC . . . . .	16
3.14.eclipsegit . . . . .	20
3.15.reports . . . . .	21
B.1. gesamtes Simulink-Modell des Verbrennungsmotors . . . . .	27
C.1. Embendded-MATLAB Function zur Berechnung der Schwerpunktlage der Ver- brennung $\phi_{Q50}$ . . . . .	29
C.2. Embendded-MATLAB Function zur Berechnung des Druckgradienten $\frac{dp_z}{d\phi}$ . . . . .	32



---

# Tabellenverzeichnis

D.1. Luftpfadgrößen des Einlassbehälters und Winkel der Haupteinpritzung, bei denen die Druckverläufe am Prüfstand gemessen wurden. Betriebspunkt ( $n_{mot} = 1500min^{-1}$ ; $q_{inj} = 10mm^3/inj$ ). . . . .	36
--	----



---

# Literaturverzeichnis

- [1] DOUGLASS, BRUCE POWEL: *Doing Hard Time*. Addison-Wesley.
- [2] ZAHN, S.: *Arbeitsspielaufgelöste Modellbildung und Hardware-in-the-Loop-Simulation von Pkw-Dieselmotoren mit Abgasturboaufladung*. Doktorarbeit, Technische Universität Darmstadt, Darmstadt, Deutschland, April 2012.