DAS Departamento de Automação e Sistemas
CTC Centro Tecnológico
UFSC Universidade Federal de Santa Catarina

Development of an Electrical Power Grid Simulation Environment for Commercial Vehicles

Report submitted to the Federal University of Santa Catarina
as requirement for approval in the discipline
DAS 5511: Final Project Work

Ricardo de Oliveira Gonzalez Aldeyturriaga

Rüsselsheim, July 2017

Development of an Electrical Power Grid Simulation Environment for Commercial Vehicles

Ricardo de Oliveira Gonzalez Aldeyturriaga

This monograph was judged in the context of the
DAS5511: Final Project Work
and approved in its final form by the
Control and Automation Engineering Course

Prof. Hector Bessa Silveira

Supervisor stamp

Acknowledgment

I would like to thank the Electric Power System & Control Integration team in the International Technical Development Center (ITDC) for the support and help in all my questions and solutions during the internship period at Adam Opel GmbH .

My thanks to the engineers Ralf Weichert, Markus Demmerle, Christoph Ferber, Bernd Nolte and Jochen Lenz in the Electrical Power Systems and Controls Integration department for sharing with me their vast knowledge in engineering, explaining to me how the several electrical systems in the vehicle works and for helping me during my internship work.

My gratitude to the engineers Dirk Frerichs, Jochen Schaffnit and Andreas Erbes for sharing with me their knowledge in modeling and simulations environment, giving me good ideas during my internship to build the simulation environment for the electrical power grid.

I would like to express my sincere gratitude to my company supervisor Dr. Dirk Balzer for giving me the opportunity to work in a worldwide company like Adam Opel GmbH , helping me always when I needed and sharing with me his vast knowledge about the automaker industry.

My thanks to my university supervisor Prof. Hector Bessa Silveira for the support and his academic competence during the internship.

Finally I would like to thank my parents for the support during my time in Germany doing my Internship and for making all this possible.

Abstract

This work consists in the development of an electrical power grid simulation environment for commercial vehicles in the company Adam Opel GmbH. The company designs, assembles and distributes light and cargo vehicles to Africa, Asia, Europe and South America. The work was developed in the Electric Power System & Controls Integration department of Adam Opel GmbH.

The present work extends the activities and results obtained in a prior internship period in the company. During the internship, analysis, optimization and new implementations were performed in the charge balancing tool (CHABAL). The purpose of the CHABAL tool is to simulate the charges and discharges in the vehicle's batteries on various driving conditions and use of electrical energy consumers. It was noted in the internship the difficulty of adding improvements and new systems in the CHABAL tool due to the way the models are connected. In the present work, a broader and more modular tool was created with the aim of archiving a more transparent interaction between the models used as well as of including more realistic features in the simulations.

The new developed simulation environment allows analysis of transient responses in the electrical power grid, vehicle response using electro-mechanical components, optimization of controls, fuel consumption and pollutants emissions for offline simulations. In each of these cases, the user needs a corresponding model. The simulation environment was designed so that the engineers searches for the desired model in the database and connects it to the simulation environment in a user-friendly manner.

The developed simulation environment was developed in Matlab/Simulink environment and integrates with the independent standard Functional Mock-up Interface (FMI) and the CarMaker software package from IPG Automotive. With CarMaker and FMI the tool has more capability for model integration and allows the user to create an accurately real-world test scenarios including the entire surrounding environment. Such compatibility allows for the tool to easily integrate different models as well as for the user to create accurate real world test scenarios that include complex interactions with the surroundings environment.

The results obtained have shown that the tool designed and implemented in the present work is capable of simulating more realistic driving situations than previous tools proposed in the company. The electrical power grid simulation environment here developed allows for the engineers in the company to connect models more transparently in the simulations, identify problems in the early stages of vehicle design and improve the quality of the various subsystems that compose a commercial vehicle.
Keywords : Simulation, Vehicle Electric power Grid, FMI, Model-Based Engineering

Resumo

Este trabalho consiste no desenvolvimento de um ambiente de simulação da rede elétrica para veículos comerciais na empresa Adam Opel GmbH. A empresa projeta, monta e distribui veículos ligeiros e de carga para África, Ásia, Europa e América do Sul. O trabalho foi desenvolvido no departamento de Electric Power System & Controls Integration da Adam Opel GmbH.

O presente trabalho amplia as atividades e os resultados obtidos durante o período de estágio na empresa. Durante o estágio, análise, otimização e novas implementações foram realizadas na ferramenta de balanceamento de carga (CHABAL). O objetivo da ferramenta CHABAL é simular as cargas e descargas nas baterias do veículo em diversas condições de condução e uso de consumidores de energia elétrica. Observou-se no estágio a dificuldade de adicionar melhorias e novos sistemas na ferramenta CHABAL devido à forma como os modelos estão conectados. No presente trabalho, foi criada uma ferramenta mais ampla e modular com o objetivo de deixar uma interação mais transparente entre os modelos utilizados, além de incluir características mais realistas nas simulações.

O novo ambiente de simulação desenvolvido permite a análise de respostas transitórias na rede de energia elétrica, resposta do veículo usando componentes eletromecânicos, otimização de controles, consumo de combustível e emissões de poluentes. Em cada um desses casos, o usuário precisa de um modelo correspondente. O ambiente de simulação foi projetado para que os engenheiros busquem o modelo desejado em um banco de dados e o conecte ao ambiente de simulação.

A ferramenta desenvolvida é compatível com a Function Mock-up Interface (FMI) e o pacote de software CarMaker da IPG Automotive. Com o CarMaker e o FMI, a ferramenta possui mais capacidade para a integração do modelos e permite que o usuario cire ambientes de simulações mais reais. Essas compatibilidades permitem que o ambiente de simulação crie cenários de teste mais proximos do mundo real que incluam complexas interações entre os componentes e o meio envolvente.

Os resultados obtidos mostraram que a ferramenta projetada e implementada no presente trabalho é capaz de simular situações de condução mais realistas do que as ferramentas anteriores propostas na empresa. O ambiente de simulação de rede de energia elétrica aqui desenvolvido permite que os engenheiros da empresa conectem modelos de forma mais transparente nas simulações, identifiquem problemas nos estágios iniciais do projeto do veículo e melhorem a qualidade dos vários subsistemas que compõem um veículo comercial.

Palavras-Chaves: Simulacao, Rede eletrica no vehiculo, FMI, Engenharia de modelos

Contents

Acronyms

AGM - Absorbent glass mat battery
API - Application Programming Interface
BCM - Body Control Module
CCM - Central Control Module
CTM - Central Timing Module
CHABAL - Charge Balance
EBCM - Brake Control Module
ECU - Electrical Control Unit
ECM - Engine control module
EMC - Electromagnetic Compatibility
EPM - Electrical Power Management
EPS - Electrical Power Steering
EFB - Enhanced flood battery
EV - Electrical Vehicle
FFBD - Functional Flow Block Diagram
FLA - Flooded lead acid
FMI - Functional Mock-up Interface
FMU - Functional Mock-up Unit
GEM - General Electronic Module
GUI - Graphical User Interface
HIL - Hardware in the Loop
HVAC - Heating, Ventilation and Air Conditioning
ICE - Internal Combustion Engine
IP - Intellectual Property
LIN - Local Interconnected Network
MBE - Model-Based Engineering
MIL - Model in the Loop
OCV - Open Circuit
OEM - Original Equipment Manufacturer
OSIMOT - Object-Oriented Simulation Model Tree
PCM - Powertrain Control Module
PSP - Pilot Support Package
PWM - Pulse Width Modulation
RVC - Remote Voltage Control
RMS - Root Mean Square
SCM - Suspension Control Module

SIL - Software in the loop	
SoC - State of Charge	
SoF - State of Function	
SoH - State of Healthy	
TCM - Transmission Control Module	
TPC - Transient Profile Creator	
WLTP - Worldwide harmonized Light vehicles Test Procedure	
VCU - vehicle control unit	
VDP - Vehicle Development Process	

List of Figures

List of Tables

Chapter 1

Introduction

This work describes the electrical power grid simulation environment for commercial vehicles developed in the Electrical Power Systems & Controls Integration department of Adam Opel GmbH.

1.1 The Company

Adam Opel GmbH is a German automobile manufacturer headquartered in Rüsselsheim, Hessen, Germany, and is subsidiary of General Motors since 1929. In March 2017, the French automobile manufacturer Groupe PSA agreed to acquire Opel. However, regulatory approvals are still pending.

The company designs, engineers, manufactures, and distributes Opel-branded passenger vehicles, light commercial vehicles, and vehicle parts for distribution in Africa, Asia, Europe, and South America. Opel designed and manufactured vehicles are also sold under the Buick brand in the United States, Canada, Mexico, and China, the Holden brand in Australia and New Zealand, and the Vauxhall brand in Great Britain.

The company has twelve plants and four development and test centers in eight European countries. Opel employs around 34,500 people in Europe, with more than 16,500 in Germany. Opel and Vauxhall are present in over 50 countries. [1]

1.2 Problem Description

The amount of electrical consumers in vehicles has increased dramatically in recent years, and is expected to rise even further in the future [2]. Hydraulic and mechanical actuators that were predominant in the past now are being replaced by electric actuators that, in order to function properly, need some amount of electric power as well network connections and controls. In comparison with hydraulic and mechanical actuators, electrical actuators generally exhibit a better functional efficiency and lower maintenance costs. However, with the present increase in electrical and electronic components in vehicles, a considerably larger amount of electrical power is required. Therefore, such increase has raised the complexity involved in the electrical power grid system of vehicles, which in turn has raised the costs of vehicle design.

In order to aid design engineers, it is highly desirable to have available a reliable simulation environment that allows them to evaluate the performance, identify problems in the early design stages, and make improvements in the electrical power grid system.

1.3 Motivation and Justification

The present work extends the activities and results obtained in a prior internship period at the company Adam Opel GmbH.

During the internship, analysis, optimization and new implementations were performed in the charge balancing tool (CHABAL). The purpose of the CHABAL tool is to simulate the charges and discharges in the vehicle's batteries on various conditions of driving and use of electrical energy consumers. It was noted during the internship the difficulty for adding improvements and new systems in the CHABAL tool due to the way the models were connected. In the present work, a broader and more modular tool was created with the aim of archiving a more transparent interaction between the models used as well as of including more realistic features in the simulations.

The CHABAL tool has only one simulation scenario that is to analyze the levels of State of Charge (SOC) in the battery under diverse drive cycles and loads consumptions. For the current work was proposed to design a simulation environment able to run the charge balance simulations and other simulation scenarios like the analysis of transient responses in the electrical power grid, vehicle response using electro-mechanical components such as brakes and electrical steering systems, optimization of controls, fuel consumption and pollutants emissions. In order to run each of the simulation scenarios, the user needs the models in question. The simulation environment is created in such a way that the user searches the models in a database and connects them to the simulation environment with a transparent interaction between the models used. The transparent interaction is due the fact that when the user selects a model to place in the simulation environment, the simulation environment by itself handles the model connections.

This form of connectivity between models follows the concepts of Model-Based Engineering (MBE), where the simulation environment has a common architecture and the models are connect to architecture with a "plug-n-play" interface. The simulation environment was developed in Matlab/Simulink environment. It integrates with IPG CarMaker and Functional Mockup Unit's (FMU).

In order to solve problems of heterogeneity of the modeling tools and to allow the exchange of models between different modeling tools, the tool independent standard function mock-up interface was created where it is possible to exchange and co-simulate dynamic models using a combination of XML-files and compiled C-code. The tool developed in the final project work is compatible with the functional mock-up interface (FMI) standard, allowing the connection of Functional Mockup Unit's to the simulation environment.

In addition to the simulation environment being compatible with the FMI standard, the present work has integration with the CarMaker software by IPG Automotive. The software is used in co-simulation and offers a virtual environment of simulations for passenger cars and light-duty vehicles. With CarMaker, it is possible to precisely model real-world test scenarios, including the entire surrounding environment. With the use of FMU and CarMaker it is possible to connect valid models of the mechanical parts of the vehicle to the simulation framework, allowing the Electric power System & Control integration engineers to spend more time in their prime task, the vehicle's electrical power grid.

The electrical power grid in the vehicle over the years has been increasing its importance in the stages of vehicle development. A tool capable of integrating all these components and performing simulations is important to the company, assisting in the development stages, detecting possible errors in the initial phases of the project and aiding in the search for improvements in the system.

1.4 Objectives

The main objective of this work is to develop a simulation environment for the electrical power grid in commercial vehicles by relying on model-based engineering concepts. The outcome is a tool created in MatLab/Simulink environment able to help the engineers to run simulations following the concepts of model-based engineering.

The work has the objective to make the simulation environment able to run offline simulations, as Model in the Loop (MIL) and Software in the Loop (SIL). For each kind of simulation the user has to be able to collect models from a data base and connect them in a user-friendly manner.

To help the models exchange between different modeling environments and to improve the tool simulation capabilities, the framework has integrity with FMI standard and IPG CarMaker.

1.5 Document Structure

This work is organized as follows:

- Chapter 2 summarizes the theoretical fundamentals, explaining the concepts, modeling and simulations for the vehicle systems. Additionally the chapter show fundamentals in modeling techniques and FMI standards.
- Chapter 3 describes the problem in greater technical detail and the system requirements. For the problem description is described the vehicle development process, the multi-domain simulation framework and model-based engineering. For the system requirements is described the simulation scenarios, the Functional Flow Block Diagram (FFBD) and the requirements for the project.
- Chapter 4 deals with the methodology and the system design for the simulation environment development. The chapter describes in detail the architecture of the system and the models integration. Additionally the chapter shows the user framework.
- Chapter 5 explains the implementation of the project, showing how was created the architecture for the simulation environment and the integrations with the CarMaker and FMI standard. Furthermore the chapter explains how was made the model data base and the tool used to manipulate the models.
- Chapter 6 presents the obtained results from the work, showing the simulation environment usage and results. Additionally the chapter will show some comparisons between FMU models and models causality.
- Chapter 7 presents the concluding remarks and future perspectives.

Chapter 2

Vehicle systems: Modeling, Concepts and Simulation

This chapter will deal with the main vehicle systems fundamentals and tools involved in the present work. Section 2.1 will describe some concepts for dynamic systems modeling. The description of some vehicle systems will be summarized in the Section 2.2, where in Subsection 2.2.1 will be described the Electrical power grid components, Subsection 2.2.2 the vehicle dynamics components, and Subsection 2.2.3 the electronic control unit. Section 2.3 will describe some concepts for the functional mock-up interface (FMI) standard. Furthermore, the Section 2.4 will show the final comments about the chapter.

2.1 Modeling

This section presents the main theoretical fundamentals for dynamic systems modeling.

One of the main and most challenging steps in the design and analysis of a mechatronics system is to generate a model for control analysis, diagnosis design, sensor selection/positioning, and actuator sizing. Modeling is a difficult task especially for mechatronic systems. Indeed, mechatronic systems are governed by many effects of different engineering disciplines (mechanical, electrical, pneumatic, thermal, etc.) and various technological components (sensors, controllers, actuators, transducers, etc.) [18].

One way for mechatronic system modeling is to study and analyze the exchange of power in the system, taking into consideration the energy storage, dissipation, and transformation. Power is the rate of doing work, the amount of energy consumed per unit of time. In electrical systems power can by given by the equation $P(t) = i(t)v(t)$ or in rotational systems by the equation $P(t) = \tau(t)\omega(t)$.

In the both equations above, it is possible to see that power is given by the multiplication of a through variable, $i(t)$ and $\tau(t)$, by a across variable, $v(t)$ and $\omega(t)$. A through variable can be determined when the variable value can be measured with a gauge connected in series to an element. A across variable can be determined when the variable can be measured with a gauge connected in parallel to an element [19].

Table 2.1 shows the relation of across variables, through variables and elements for some physical domains. For characteristics, E1 means that the through variable is time derivative of a conservation variable, and E2 that the product $e(t)f(t)$ results in power.

Table 2.1: *Power variables [20]*

System (Characteristics)	Variable		Elements		
	Throught $f(t)$	Across $e(t)$	Resistance R	Capacity C	Inductance L
Electric (E1,E2)	Current [f]=A	Voltage [e]=V	Ohmic Resistor [R]=Ohm=V/A	Capacitor [C]=F=As/V	Inductor [L]=H=Vs/A
Translational Mechanical (E1, E2)	Force [f]=N	Velocity [e]=m/s	Friction [R]=m/Ns	Mass [C]=kg	Spring [L]=m/N
Rotational Mechanical (E1,E2)	Torque [f]=Nm	Angular Velocity [e]=rad/s	Friction [R]=1/Nms	Moment of Inertia [C]=kg m^2	Stifness [L]=1/Nm
Hidraulic (E2,E1 conditional)	Fluid Flow [f]=m^3/s	Pressure [e]=N/m^2	Flow Resistance [R]=Ns/m^5	Volume Storage [C]=m^5/N	Inertia [L]=kg/m^4
Pneumatic (E1)	Mass Flow [f]=kg/s	Pressure [e]=N/m^2	Pneumatic Resistance [R]=1/ms	Mass Storage [C]=kgm^2/N	-
Thermic (E1)	Heat Flow [f]=Nm/s=W	Temperature [e]=K	Thermal Resistance [R]=K/W	Thermal Capacity [C]=Ws/K	-

It is possible to use computer tools to create and analyze dynamic systems modeling. Some tools can work with causal models and others with acausal models. In the sequel one discusses the main differences between causal and acausal model techniques.

2.1.1 Causality

Most of the general-purpose simulation softwares on the market such as Matlab/Simulink, assumes that a system can be decomposed into block diagram structures with causal interactions. A causal model is an abstract model that describes the causal mechanisms of a system. A causal model is composed of:

- Inputs
- Variables
- State variables
- Relations between inputs and (state) variables constraining the value of outputs and variables
- Relations between inputs and (state) variables constraining the value of the derivatives of state variables

The inputs of causal models handle the data coming from the environment.The output handle the data to be exported to the environment. State variables of a causal model are used to compute observable quantities. In causal modeling the data flow is explicit, it is possible to simulate a causal model using value propagation first and then integration.

In nature, dynamic systems are acausal. We never know whether in resistor current causes voltage or voltage causes current. Causality is artificially made because physical laws must be transformed in convenient computational descriptions [21]. There are tools that enable the possibility to create acausal models, as Sieme's Amesin or SimScape library for Simulink.

Acausal models have some advantages over causal modeling as show in table 2.2.

Table 2.2: *Causal-Acausal Modeling Diferences [22]*

Acausal	Causal
Acausal models are easy to build and modify	Causal models are difficult to build and extremely hard to modify
Acausal models require highly elaborated tools to handle them efficiently	Causal models generaly don't require elaborated tools to handle them efficiently
Acausal modeling is a convenient way to express specifications	Causal modeling is a convenient way to express explicit computations

One concludes from the table above that, in general, acausal models are more intuitive to build but they need more elaborated tools to handle.

Regardless of the tools used for modeling, it is important that engineers can exchange models between then. In order to solve this heterogeneity in modeling tools, a standard was created to solve this problem. This standard is called FMI and some topics will be described in the Section 2.3. The next section will describe some systems present in the vehicle and some ways to model the vehicle dynamics.

2.2 Vehicle systems

In this Section will be described some theoretical fundamentals for the electrical power grid and the vehicle dynamics.

The car is a mechatronic system, where the engineers during the vehicle's project have to achieve a synergistic optimization between the fields of mechanical engineering, electrical engineering and software engineering, in order to project more functions in the vehicle at lower cost, less weight and installation space, and better quality.

The next Subsections will describe the systems present in this multi-domain field of engineering.

2.2.1 Electrical power grid

The development in electronic systems and the continuous string of innovations have found their way into vehicles. Since 1970s the original equipment manufacturers (OEM) were looking forward to include new technologies for producing safer and more fuel-efficient cars. All those goals are bound to the driver benefits, letting the car owner a driving pleasure experience.

The figure 2.1 shows an example of the possible electronic systems that can be found in a car nowadays.

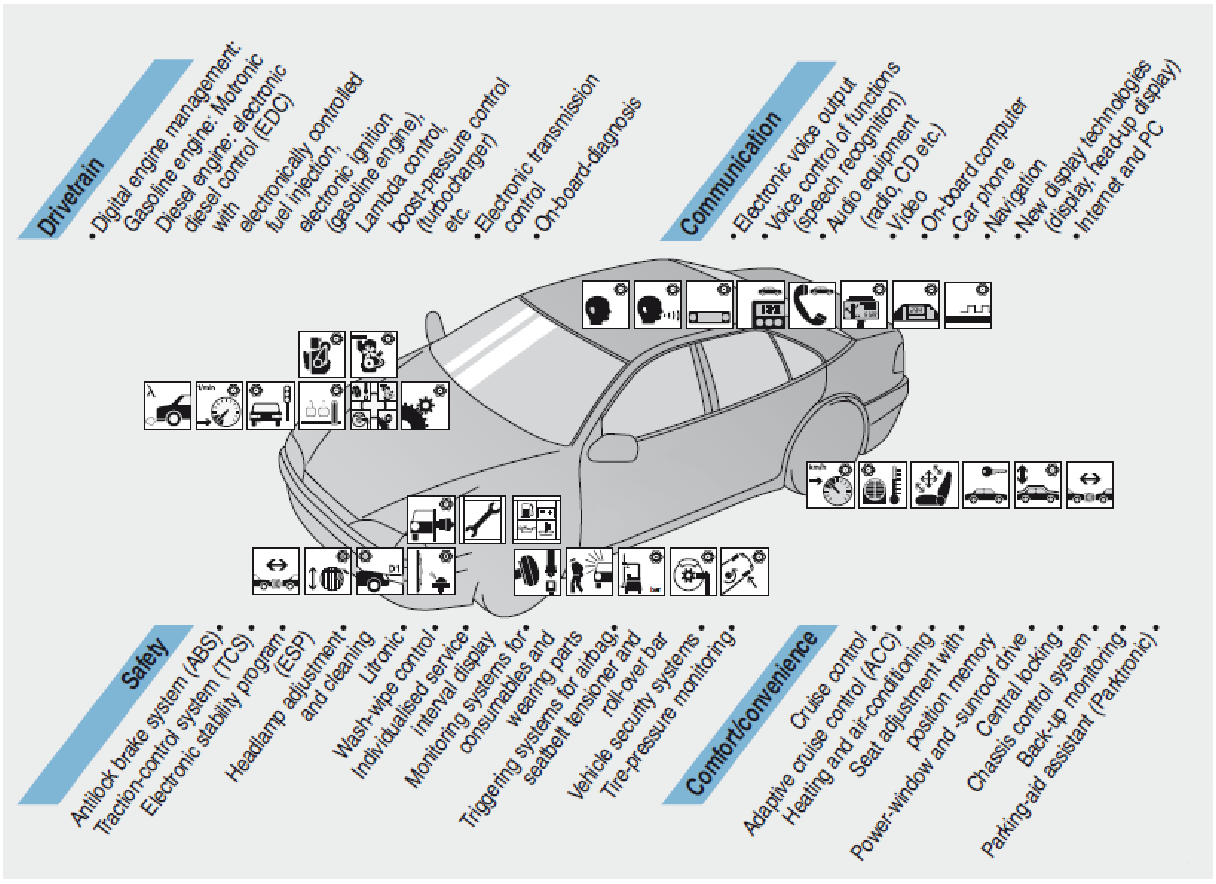

Drivetrain
•Digital engine management:
Gasoline engine: Motronic
Diesel engine: electronic diesel control (EDC)
with electronically controlled fuel injection,
electronic ignition (gasoline engine),
Lambda control,
boost-pressure control (turbocharger) etc.
•Electronic transmission control
•On-board-diagnosis
Communication
•Electronic voice output
•Voice control of functions (speech recognition)
•Audio equipment (radio, CD etc.)
•Video
•On-board computer
•Car phone
•Navigation
•New display technologies (display, head-up display)
•Internet and PC
Safety
Antilock brake system (ABS)•
Traction-control system (TCS)•
Electronic stability program (ESP)•
Headlamp adjustment and cleaning•
Litronic•
Wash-wipe control•
Individualised service interval display•
Monitoring systems for consumables and wearing parts•
Triggering systems for airbag, seatbelt tensioner and roll-over bar•
Vehicle security systems•
Tire-pressure monitoring•
Comfort/convenience
Cruise control•
Adaptive cruise control (ACC)•
Heating and air-conditioning•
Seat adjustment with position memory•
Power-window and -sunroof drive•
Central locking•
Chassis control system•
Back-up monitoring•
Parking-aid assistant (Parktronic)•

Figure 2.1: *Vehicle Electronics [2]*

In order to keep all the systems working in a desired voltage level, the electronic power sources of the cars have to be in accordance with the desired project specifications. The main power sources ,and in most cases the only ones, are the alternator and the batteries. The batteries are the electrical power storage and the alternator the electrical power generator using the engine power. Figure 2.2 illustrate the basic vehicles electrical power grid, where the Electrical Control Unit (ECU) and Body Control Module (BCM) are vehicle computers, the batteries are normally 12V Lead Acid chemistry (Pb), the Remote Voltage Control (RVC) is an internal signal in the car for generator voltage set point and loads are any kind of consumers in the car.

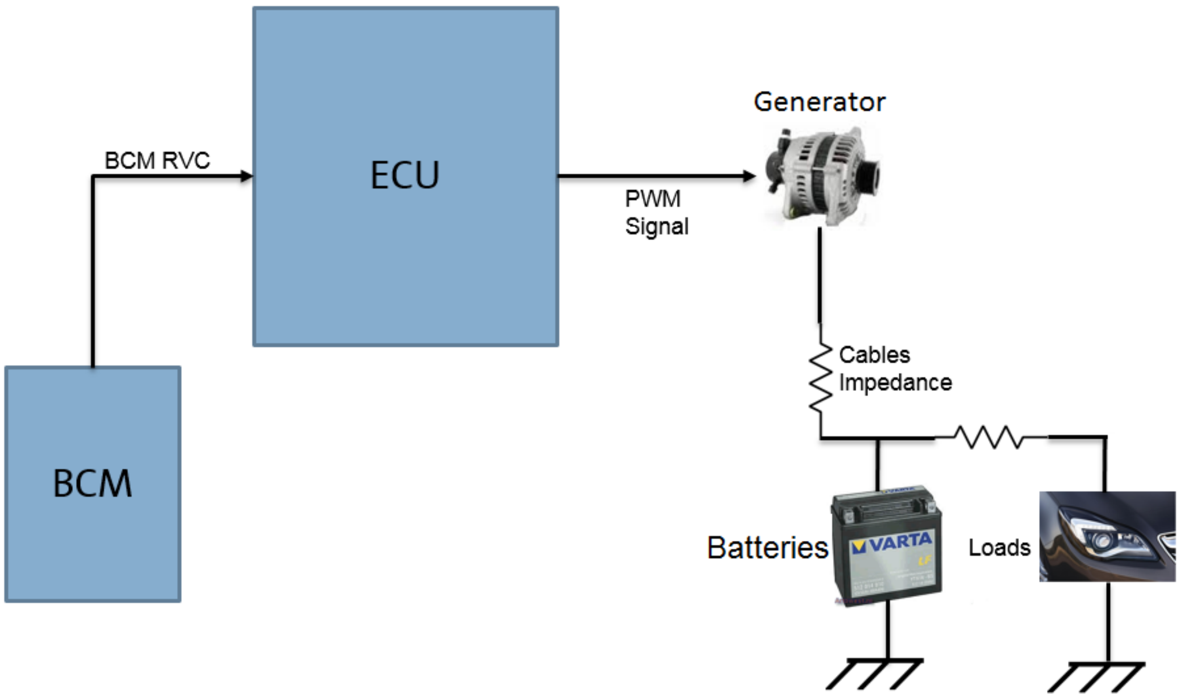


Figure 2.2: *High Level Electrical Power grid*

The fallowing Subsections will give more details about the various systems that can be found in the electrical power grid in the vehicle.

2.2.1.1 Generator

The generator is a device that converts the mechanical energy of the crankshaft into electrical energy to charge the battery and supply power to vehicle electrical loads. A conventional 12V automotive alternator is a claw pole type AC synchronous machine coupled to the crankshaft of the engine over a belt drive with a gear ratio ranging from 2.5 – 3.0. Internal electronics consists of a DC diode rectifier which converts the AC power to DC making it compatible with the vehicle electrical system. However, as the generated DC output power is uncontrolled, a voltage regulator is used to control the DC output by sensing the voltage at the generator or battery terminals and adjusting the output accordingly. The alternator has two terminals: L-terminal and F-terminal, to communicate with the engine controller (i.e. Engine control module (ECM)) with Pulse Width Modulation (PWM) signal. The ECM sends the set-point voltage to the L-terminal as a pulse-width signal based on the instantaneous electrical load in the vehicle and other relevant operating conditions. Depending on the external terminal voltage and the internal set point, the voltage regulator adjusts the DC excitation current of the rotor windings to modulate the generator output. The information about the actual loading of the generator is sent back to the ECM through F-terminal.[3]

Figure 2.3 shows the three phase generator schematics. The top graph (*a*) shows the three-phase alternating voltage, the middle graph (*b*) shows the generator voltage, formed by the envelope curves of the positive and negative half-waves, and the bottom graph (*c*) shows the rectified generator voltage.

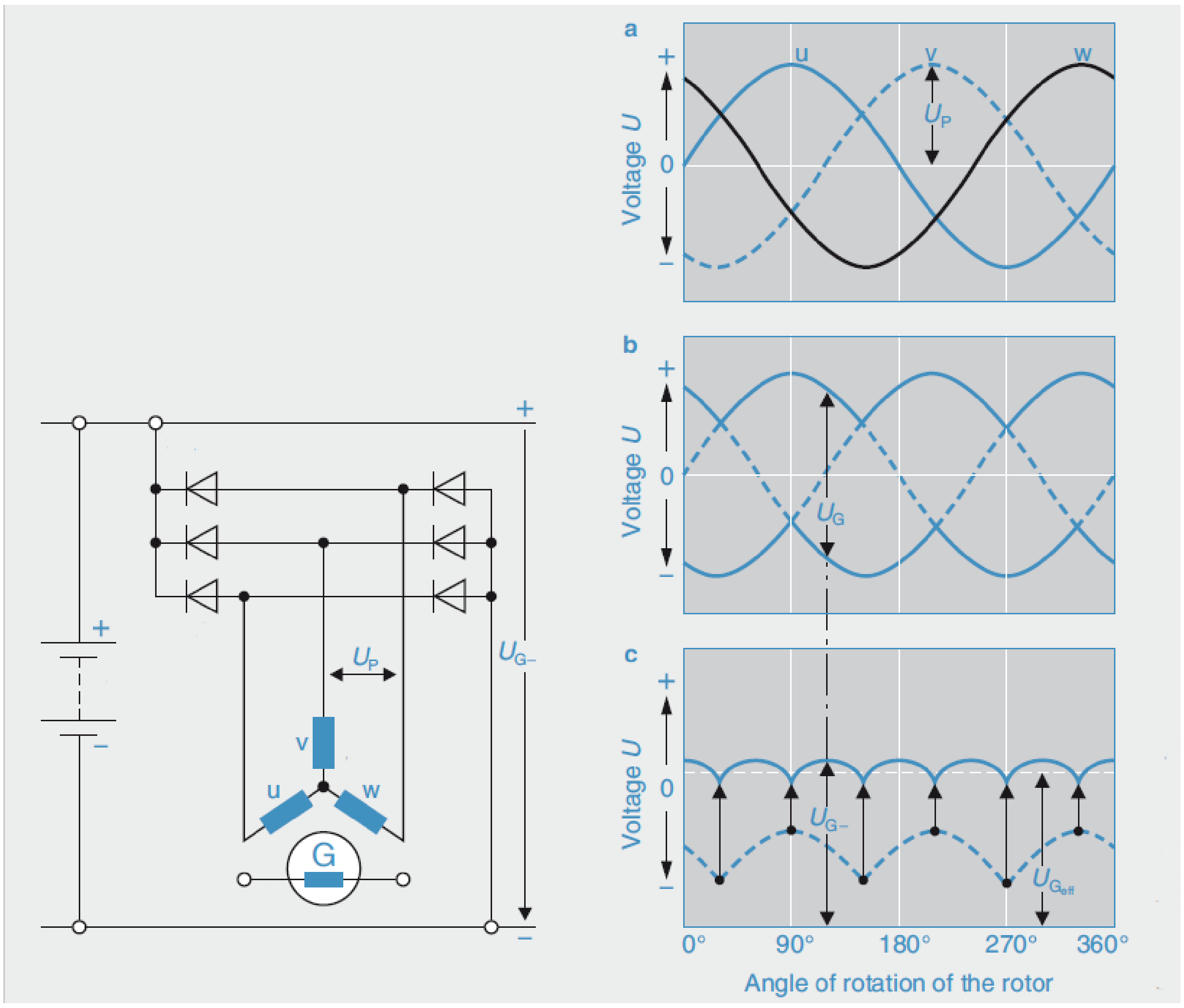


Figure 2.3: *Three-phase generator schematic [2]*

2.2.1.2 Battery

Batteries are electrochemical storages which are mainly responsible for starting internal combustion engine in the vehicle and meeting transient power demands of the consumers [3]. When the generator can not supply the required amount of power to the system, for example in cases when the engine is off, or when the engine's rotation speed is not enough to let the generator to work properly, the batteries will provide the amount of energy necessary to keep the system functional. Furthermore, the batteries can be also used to store electrical energy in events called recuperation, when some energy of the car was going to be wasted, for example when the car is braking and the car's kinetic energy becomes heat in the brake system. With this it becomes possible to save fuel, thus helping the vehicle to operate in accordance with emission regulations.

Batteries vary from each other according to the chemistry used. Most of the internal combustion cars today uses the Lead Acid batteries. They are inexpensive compared to other chemistries, have a good cold crank performance and can be recycled without causing significant harm to environment [4]. Due to their low specific energy, energy density and heavy weight compared to other battery technologies, their application is mainly restricted to conventional and hybrid vehicles [5].

The three most common types of Lead Acid batteries used today as starter batteries are the Flooded Lead Acid Battery (FLA), the Enhanced flood battery (EFB), and the Absorbent glass mat battery (AGM). As FLA has the lowest cost amongst these batteries, it is mainly used in conventional engine vehicles. Due to flowing liquid electrolyte, they require more maintenance compared to other counterparts. Moreover, they are not useful for recuperation application due to low charge acceptance property. However, the advanced lead acid batteries i.e. EFB and AGM, are widely used for recuperation as they have high charge acceptance and allows deep cycling to handle frequent start-stop situations. Moreover, the AGM battery is characterized by immobile electrolyte which requires less maintenance. Each cell has a nominal voltage of 2.0V [3]. The figure 2.4 illustrates an EFB.

With the advances on the Lithium batteries technology there are studies to change the classical Pb starter batteries with Lithium batteries or to use both in parallel in the electrical power grid. In hybrid and EV vehicles, lithium batteries are common.

The Lithium batteries technology has some advantages over the Pb, as that Lithium batteries are lighter, can store a charge longer and can withstand charge/discharge cycles better than a lead-acid battery. The ratio of the maximum safe output current to the battery's rated capacity is much higher in a lithium battery; therefore, you need less "rated amps" to accomplish the same amount of work. In other words, they can dump or absorb huge amounts of current in relation to their rating, making them better for recuperation events [6].

The downside of the Lithium batteries technology is that their output drops much faster than lead-acid batteries as the temperature goes down. If not properly charged, they're much more susceptible to individual cell failures. Other downside is the price, that they are much more expensive than lead-acid batteries. With improvements in the technology and higher demand of production, their price will drop in the future.

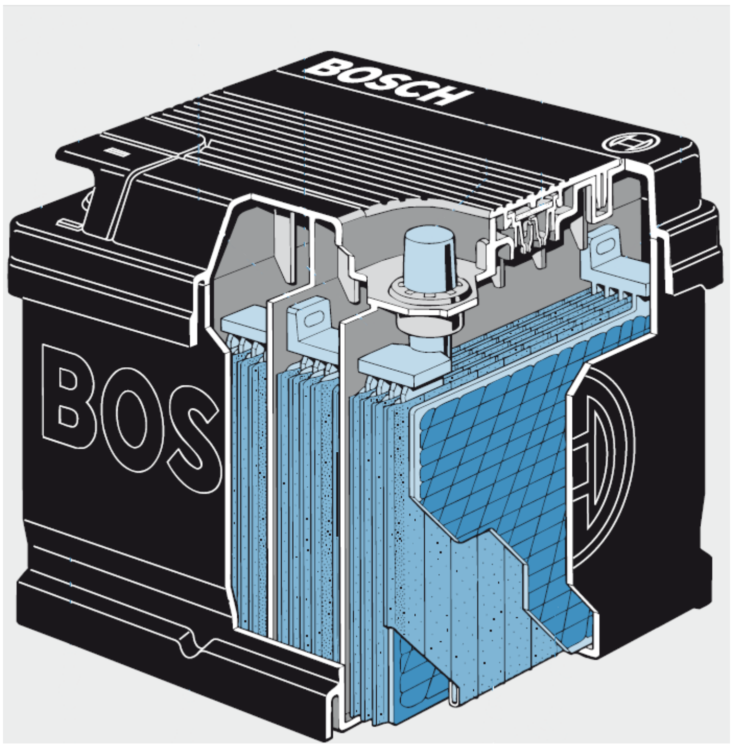


Figure 2.4: *Maintenance free battery [2]*

2.2.1.3 Starter

Internal-combustion engines must be cranked by a starter at a minimum speed before they can supply sufficient energy in sustained operation from the combustion cycles to cover the momentary requirements for the compression and gas exchange cycles. When an engine is first started, the bearing surfaces are not adequately lubricated so that high levels of friction have to be overcome when cranking the engine [2].

To start the engine it is necessary to transfer the torque from the starter motor to the internal combustion engine (ICE) in order to overcome the frictions and allow the ICE to operate on its own. The engagement between starter ring gear and the engine is made on the engine flywheel for manual transmission engines and on the torqueconverter housing for automatic transmissions. The starter normally is designed for high speeds and low torque, thats what allows the dimensions and the weight of the starter to be kept small.

The power supply for the starter motor comes from the batteries and the transient peak current during a cold crank event is normally around 800A. For the correct operation, the batteries have to be in conditions that can provide this amount of energy during the crank event.

2.2.1.4 Wiring Harness

The purpose of the wiring harness is to distribute power and signals within a motor vehicle. A wiring harness in the present day, mid-class passenger car with average equipment has approximately 750 different lines, their length totaling around 1,500 meters. In recent years the number of contact points has practically doubled due to the continuous rise in functions in the motor vehicle [2].

In the process of the vehicle development the wiring harness design is very important due the fact that voltage drops in the cables can lead to electrical systems malfunctions. Not only the voltage drops are analyzed in the project but as well the leak-tightness, electromagnetic compatibility (EMC), temperatures effects, line routing and Ventilation of the wiring harness.

The figure 2.5 illustrates an example of wiring harness in the vehicle. In the figure it is possible to see an example of how the cables are distributed along the vehicle with the corresponding connections.

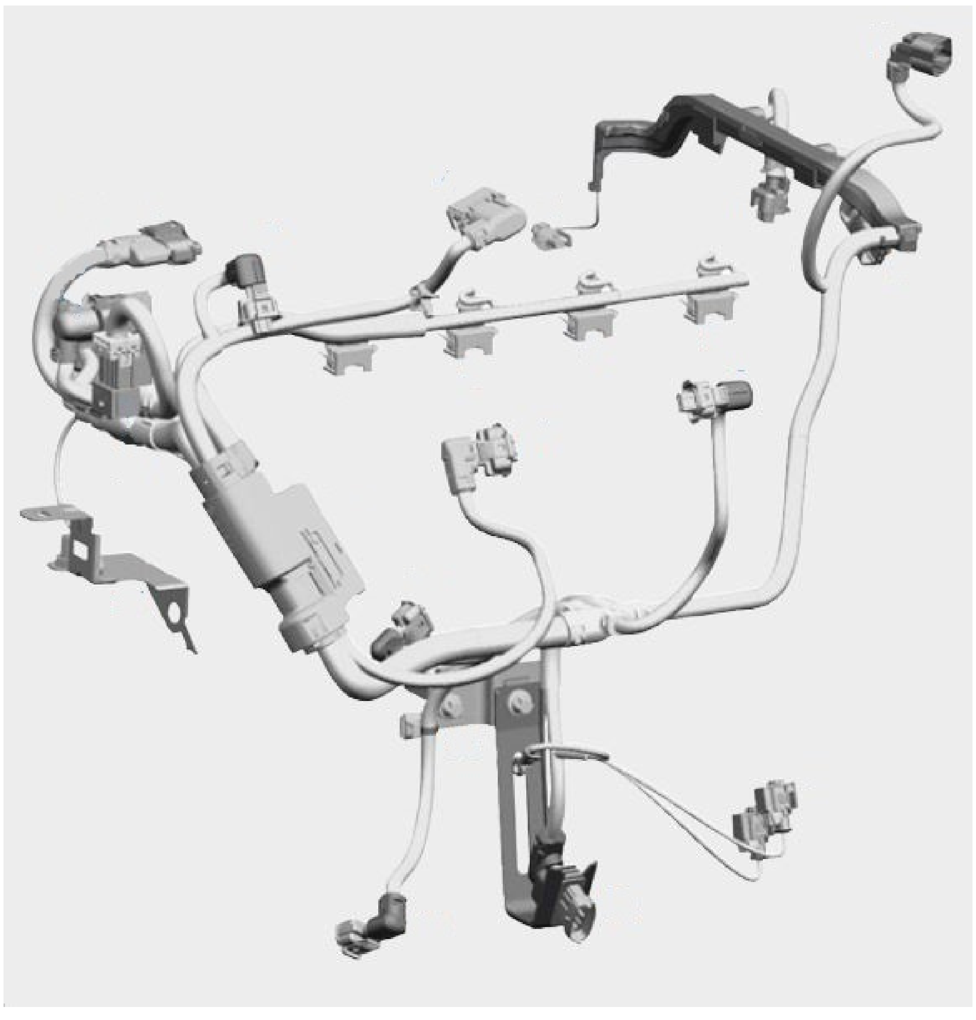


Figure 2.5: *Vehicle's wiring harness [2]*

2.2.1.5 Loads

Every component in the car that consumes electrical energy is a load in the electrical power grid. Some loads are continuous consumers, as the ignition system, fuel injection, etc., others are long time consumers as the lights, rear window heating, etc, and others are short time consumers like stop lamps, turn signals, etc. The number of loads are dependent for the period of time of the year, as for instance heaters in the winter time or coolers in the summer time. During driving conditions some loads are only activate during a certain period of time, for example some secondary-air pumps, which are only on during some minutes after the engine goes on. Moreover, during the time some consumers are on, the amount of power used is not constant. Some loads can present dynamics in the power consumption, having high peaks of power in the transient and less consumption in the permanent state.

It was common for loads to behave as only power consumers in the electrical power grid, but new systems can act as generators as well during transient responses. With the constant innovations in electro-mechanical actuator systems, some hydraulic and mechanical parts were replaced by electro-mechanical actuators. As example the iBoost brake system or the electric power steering system (EPS). These components connected to the electric grid do not act only as consumers, but at some moments as generators, giving back to the system some amount of energy.

2.2.1.6 Integrated Brake booster system

The Integrated brake booster system or iBooster is a vacuum-independent, electromechanical brake booster. The system was developed by Robert Bosch GmbH and has integrated a motor in the brake system to control the degree of brake boosting via a two-stage gear unit for situation-dependent support on demand. This dispenses with today's costly, continuous process of generating a vacuum using either the internal combustion engine directly or a vacuum pump. Not only does this save fuel in itself, it also allows more comprehensive use of fuel-saving functions that stop the engine for periods of time, such as start-stop or coasting [8].

Accordingly to Bosch, the new brake system can build up full braking pressure three times faster than the conventional brake systems. The system is capable to recovers almost all the energy lost in typical braking operations by ensuring deceleration rates of up to 0.3g [9].

2.2.1.7 Electrical Power Steering system

The electrical power steering system (EPS) system differs from the conventional mechanical steering systems or hydraulic assist systems by using an electro-mechanical actuator to assist the steering maneuvers. Compared with the conventional systems, the EPS have advantages in fuel economy, safety, disturbance rejections and start-up on low temperature. The figure 2.6 illustrates the EPS system where is show the steering wheel connected to the assist motor that helps to delivery torque to the rack that is attached to the wheels.

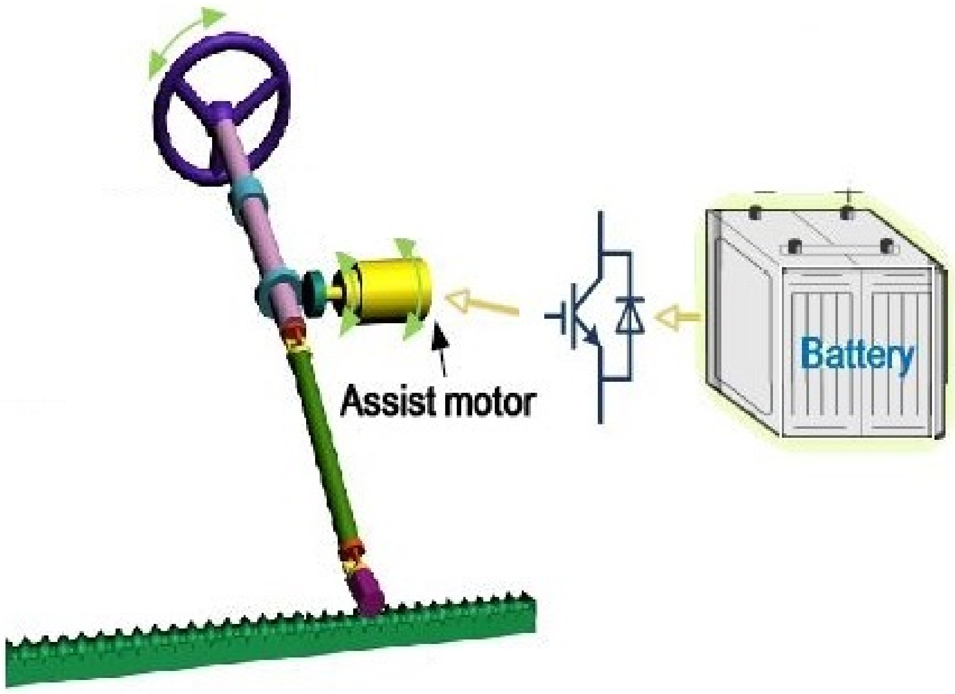


Figure 2.6: *EPS system [10]*

The control system for the EPS seeks the target motor current reference that comes from a characteristic curve based on the input of torque sensor and vehicle speed sensor. The relationship of the real-time vehicle speed is derived from vehicle speed sensor and the steering wheel torque is expressed by the following function $I_T = f(V, T_s)$, where I_T is motor current, V is vehicle speed, and T_s is motor torque. The figure 2.7 illustrates the EPS control system. In the figure it is possible to see that for the controls there are two primary inputs, the driver torque signal detected by a torque sensor on the steering wheel and the vehicle speed signal. Along with other system variables, they are fed into and electric control module which determines the reference current based on the torque map. Normally, the classical proportional–integral–derivative (PID) controller is employed [10].

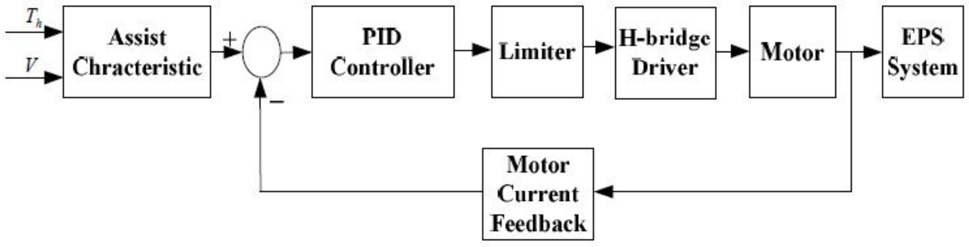


Figure 2.7: *EPS control system [10]*

2.2.2 Vehicle dynamics

Vehicle dynamics is a part of engineering primarily based on classical mechanics but it may also involve electrical engineering, chemistry, communications, psychology etc. The vehicle dynamics encompasses the interaction of the driver, the vehicle, the load and the environment [11]. This section will focus on the vehicle itself.

While the vehicle is driving there are forces acting on the vertical, longitudinal and transversal axis as show in figure 2.8. In order for the vehicle to move, the motive force of the engine (engine torque) must overcome all forces that resist motion (all longitudinal and lateral forces) such as are generated by road gradient or camber.

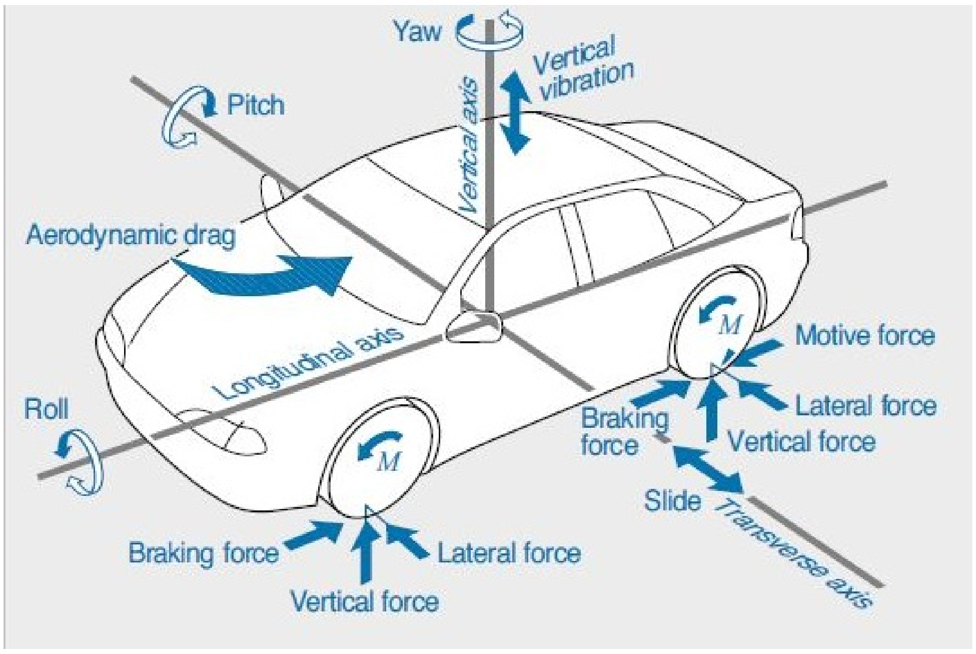


Figure 2.8: *Forces acting on a vehicle [12]*

The next subsections will give a brief explanation of the longitudinal and lateral dynamics of the vehicle.

2.2.2.1 Longitudinal

The longitudinal dynamics of the vehicle can be classified in two categories, the dynamic of the power train and the dynamic of external forces. The dynamic of power train comprises the engine, the transmission, the final drive and the wheels, the dynamic of external forces includes aerodynamic drag force, gravitational force in addition rolling resistance and traction force which arises from the physical evolution between the tire and road while rotating the wheel on the road [13].

The power train in the vehicle is responsible to transmit the torque and velocity from the engine to the wheels. The transmission of torque normally can be made by manual or automatic transmission systems, where the driver shift the gears in the manual transmission system and the gears are shifted automatically in the automatic transmission system.

With a model of the power train together with a model of the longitudinal external forces it is possible to simulate the behavior of the vehicle over the longitudinal axis.

The external forces are summarized by the rolling resistance, the gravitational force, the aerodynamic force,and the traction force. With the power train is possible to create a model for the longitudinal vehicle dynamics. Figure 2.9 show the longitudinal forces exerting on vehicle when traveling on the inclined road.

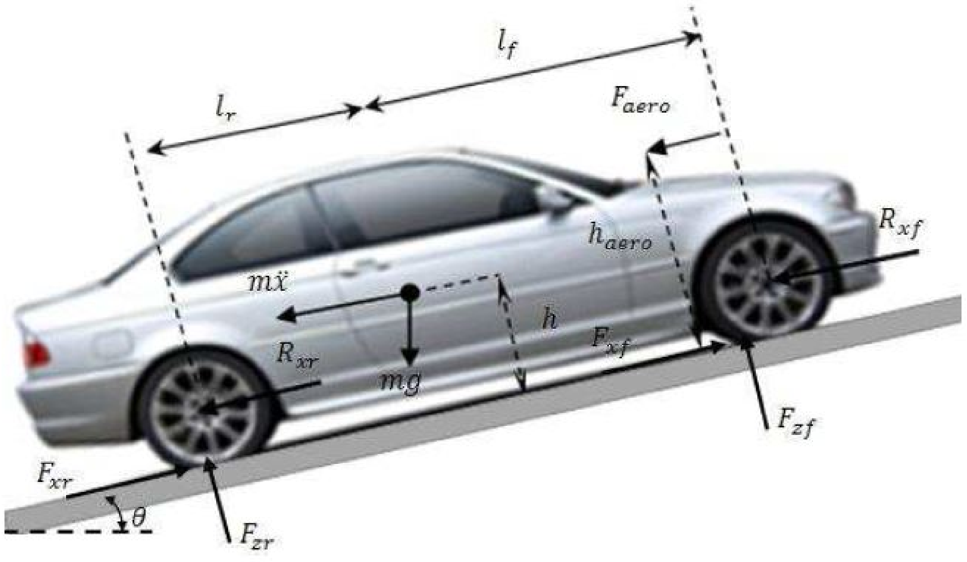


Figure 2.9: *Longitudinal forces [14]*

Applying the second Newton's law in the vehicle illustrated in figure 2.9 the acceleration $\ddot{x}$ can be obtained [13]:

$$m\ddot{x} = F_{xr} + F_{xf} - F_{aero} - R_{xf} - R_{xr} \pm mgsin(\theta) \tag{2.1}$$

where F_{xr} and F_{xf} express the longitudinal traction force for the front and rear wheel respectively, F_{aero} is the aerodynamic force, R_{xf} and R_{xr} are the rolling resistance produced on each wheel.

The next topic will show the additional forces present in the vehicle to understand the lateral vehicle dynamics.

2.2.2.2 Lateral

When a vehicle drives through a curve at low lateral acceleration, small lateral forces will be needed for course holding. Then, hardly lateral slip occurs at the wheels. To understand the behavior of this dynamic many models were created. This section will give a brief explanation of the linear single track model.

Single track models allow a physically plausible description of the driving behavior of vehicles without major modeling and parameterization effort. This model is based in a series of simplifications:

- The velocity of the vehicle's center of gravity is considered to be constant along the longitude of its trajectory.
- All lifting, rolling and pitching motion will be neglected.
- The vehicle's mass is assumed to be concentrated at the center of gravity S
- The front and the rear tires will be represented as one single tire on each axle. The imaginary tire contact points V and H, which the tire forces are to act upon, lie along the center of the axle.
- The pneumatic trail and the aligning torque resulting from the slip angle of the tire will be neglected.
- The wheel-load distribution between front and rear axle is assumed to be constant.
- The longitudinal forces on the tires, resulting from the assumption of a constant longitudinal velocity, will be neglected.

The first two assumptions lead to four constraints for the six degrees of freedom of rigid bodies in the model. As a result, the only possible motion left is the heading angle (yaw angle) ψ_v, which only occurs in the form of the yaw rate $\dot{\psi}_v$, and the side slip angle β. The side slip angle represents the direction of the deviation of the center of gravity from the vehicle's steering axis. The steering angle δ of the front axle serves as the input parameter.

The model is called single track model or bicycle model because the two wheels in the front and the two wheels in the rear are put together. The figure 2.10 shows the mathematical description of the linear single track model.

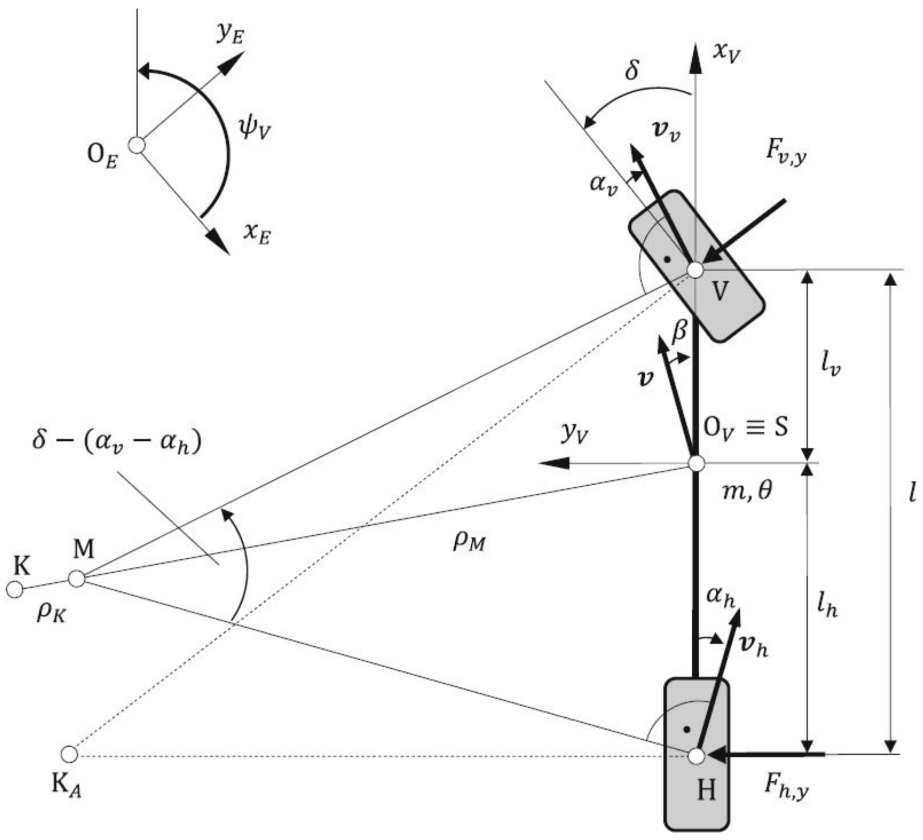


Figure 2.10: *Linear Single Track Model mathematical description [15]*

Assuming that *m* is the vehicle mass, θ is moment of inertia , *v* is the vehicle velocity, $c_{\alpha,v}$ and $c_{\alpha,h}$ are the cornering stiffnesses for the front and back wheels, respectively, and

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \dot{\psi}_v \\ \beta \end{bmatrix} \tag{2.2}$$

it is possible obtain the following linear state equation [15] :

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} -\frac{1}{v}\frac{c_{\alpha,v}l_v^2+c_{\alpha,h}l_h^2}{2} & -\frac{c_{\alpha,v}l_v-c_{\alpha,h}l_h}{\theta} \\ -1-\frac{1}{v^2}\frac{c_{\alpha,v}l_v-c_{\alpha,h}l_h}{m} & -\frac{1}{v}\frac{c_{\alpha,v}+c_{\alpha,h}}{m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} \frac{c_{\alpha,v}l_v}{\theta} \\ \frac{1}{v}\frac{c_{\alpha,v}}{m} \end{bmatrix} \begin{bmatrix} \delta \end{bmatrix} \tag{2.3}$$

Vehicle dynamics modeling is important to aid the creation of control algorithms to handle the vehicles driving stability. The control laws and other system controls are put in the electric control unit (ECU). The next section will describe the basics about the ECU's technology.

2.2.3 Electronic Control Unit

The electronic control unit is a generic term for any embedded system that controls one or more of the systems or subsystems in a transport vehicle. Types of ECU include Electronic/engine Control Module (ECM), Powertrain Control Module (PCM), Transmission Control Module (TCM), Brake Control Module (BCM or EBCM), Central Control Module (CCM), Central Timing Module (CTM), General Electronic Module (GEM), Body Control Module (BCM), Suspension Control Module (SCM), and others. Taken together, these systems are sometimes referred to as the car's computer and sometimes one assembly incorporates several of the individual control modules [16].

All the open-loop and closed-loop algorithms of the electronic system run inside the control unit. The heart of the control unit is a microcontroller with the program memory in which is stored the program code for all functions that the control unit is designed to execute. The input variables for the sequence control are derived from the signals from sensors and setpoint generators. They influence the calculations in the algorithms, and thus the triggering signals for the actuators. These convert into mechanical variables the electrical signals that are output by the microcontroller and amplified in the output stage modules. This could be mechanical energy generated by a servomotor (power-window unit), for example, or thermal energy generated by a sheathed-element glow plug [12].

Figure 2.11 shows the system blocks of a control unit.

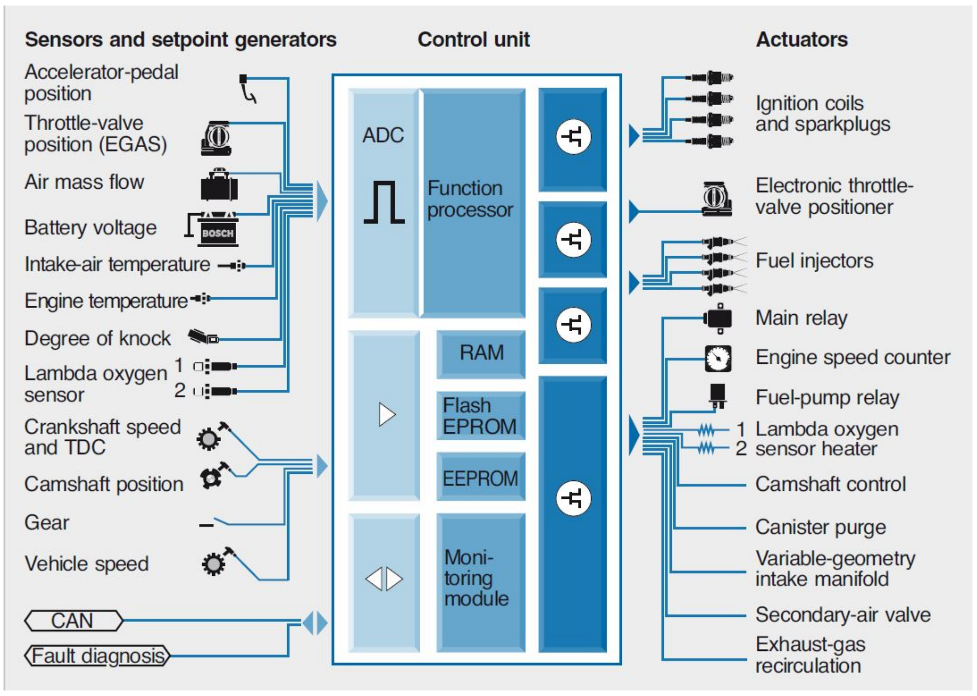


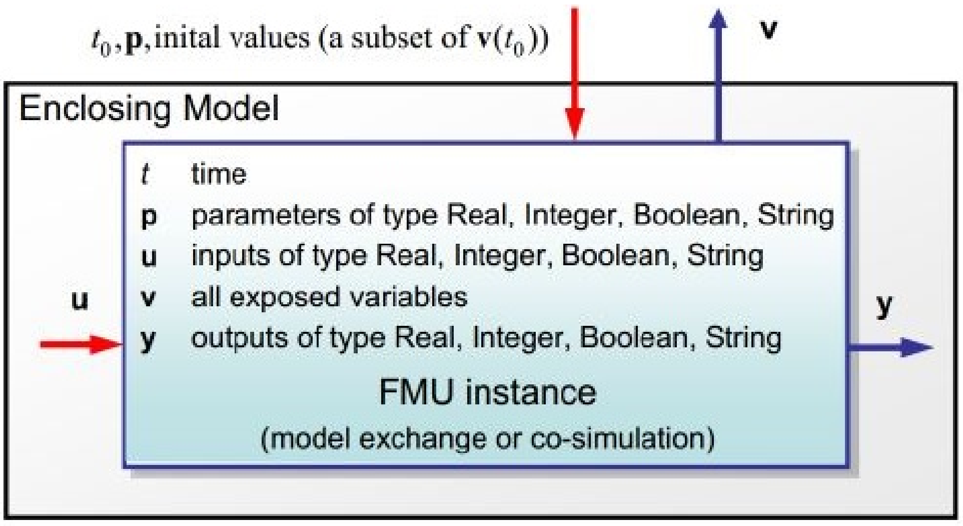
Figure 2.11: *Function modules of an electronic system [12]*

To aid the engineers to build the controls to put inside the ECU it is important to understand the physical behavior of the dynamic system of interest by creating models. In Section 2.1 was demonstrated some model concepts. To help the engineers to run simulations with models tool independent was created the function mock-up interface, the next section will explain more about the standard.

2.3 Function Mock-up Interface

The Functional Mockup Interface is an open standard and a tool independent interface which allows exchange of dynamic models between different modeling tool environments. The intention to develop such an interface is that if a system is made of many subsystems, it should be able to create a virtual product by combination of the subsystem models, thereby enabling a better understanding of the system in early product development phases. There are two functional mock-up interface versions existing today, FMI 1.0 and FMI 2.0.

Moreover, FMI allows modeling of different components in their respective state-of-the-art tooling environments. The executable model package which implement this interface is called a Functional Mockup Unit (FMU). Figure 2.12 shows the schematic diagram of the FMU where red arrows shows input to the FMU and blue arrows represent FMU output.



t_0, **p**, inital values (a subset of **v**(t_0))
v
Enclosing Model
t time
p parameters of type Real, Integer, Boolean, String
u inputs of type Real, Integer, Boolean, String
v all exposed variables
y outputs of type Real, Integer, Boolean, String
u
y
FMU instance
(model exchange or co-simulation)

Figure 2.12: *FMU schematic [17]*

There are two main design ideas to implement FMU instances for simulation with other models:

- **FMI for Model Exchange:** The FMU requires the solver of the simulation environment to perform continuous-time or discrete time numerical integration.

- **FMI for Co-Simulation:** The FMU has a local solver which allows simulation of large heterogeneous systems and multi-rate numerical integration.

The FMU package is a zipped file with an extension *.fmu. It contains several files:

- XML Model Description file: It provides information about the description of interface data of the model. This means that all the information that is not required during model integration like signal names, units, initial values, is included in this file. The simulating tool importing the FMU will parse the model description file to initialize the values in the tooling environment.
- Compiled C file: It consist of differential, algebraic, discrete equations of the model converted into causal form. These files are implemented in source code or binary form depending on the execution requirements of the real time target machines.
- Additional files for documentations, maps, icons in their FMU specific file format.

The FMI standard has some benefits as listed below, an due these benefits the standard is being adopted for both industrial and research purposes. Currently, a total of 91 modelling and simulation tools support or are planned to support the FMI standard [3].

- The components developed by the supplier in different modelling environments are integrated by an Original Equipment Manufacturer (OEM) to evaluate the component behavior through system level testing and validation.
- It enables reuse of supplier models with the protection of supplier model intellectual property (IP) rights.
- Extensive testing with different scenarios (including worst-case) which may not be otherwise reproducible.
- It enables early component design validation and model based optimization of controller and plant models.
- Resources and costs saving with increased product quality and efficiency.

2.4 Final Comments

This chapter introduced a couple of concepts and basics for many vehicle systems, modeling and FMI standards. All this information was important for the current work in order to create a framework capable to simulate the electrical power grid. As the vehicle is a complex mechatronic system, it is necessary to understand the major dynamics in the vehicle and the way the models are created. Models exchange are becoming a trend along the OEM's with FMI standard, making important to know how to integrate FMU's in the simulation environment. The next chapter will describe the problem treated in the present work and present the design requirements for the electrical power grid simulation environment.

Chapter 3

Problem Description and System Requirements

This chapter describes the problem treated in the present work based on the technical concepts and tools exhibited in the previous chapter. Furthermore, it will also present and justify the design requirements for the electrical power grid simulation environment. Section 3.2 will show some topics about the multi-domain simulation framework. Section 3.3 will describe the main ideas for model-based engineering. Sections 3.4, 3.5 and 3.6 will present the Simulation Scenarios, functional flow block diagram and system requirements respectively. Furthermore, Section 3.7 will show the final comments about the chapter.

3.1 Vehicle Development Process

To support the development of commercial vehicles, many OEM's created vehicle a development processes (VDP) to aid the product development. With these processes it is possible to build more competitive cars in the market, saving money from development and production.

During the stages of the VDP, vehicle simulations are very important to detect possible problems in early stages of the development program and to help to optimize the many vehicle systems. This section will give the basics concepts for the VDP.

Today, commercial vehicles are highly complex consumer products that have to meet a high number of requirements, as safety, emissions, recyclability and fuel economy. Commercial mid-size vehicles could have more then 2000 parts during assembly, and can go to more then 3000 when build with all the additions. Every part has a development process, where have to be designed, developed, integrated, validated, released, sourced, tooled, and shipped to a plant for assembly. Additionally every part has to work since the first time the user turn on the car, and every time regardless the ambient conditions.

In order to ensure this, many car makers have documented and timed process for product development. The process are used to coordinate standardized work across functions and product subsystems in a tightly timed and orchestrated effort to bring new vehicles to market on time, within budget, and high quality levels. The idea to standardized such work is to make the process repeatable and capable to ensure quality.

The common activities during the development process of interest are the engineering, manufacturing, finances, planning, marketing, sales, and others. Figure 3.1 shows an example of a VDP overview where, PI is Program Initiation, PC is Program Concept, PA is Program Approval, SR is Styling Release, PV is Prototype Validation, PPV is Pre-Production Vehicle, and SOP is Start of Production.

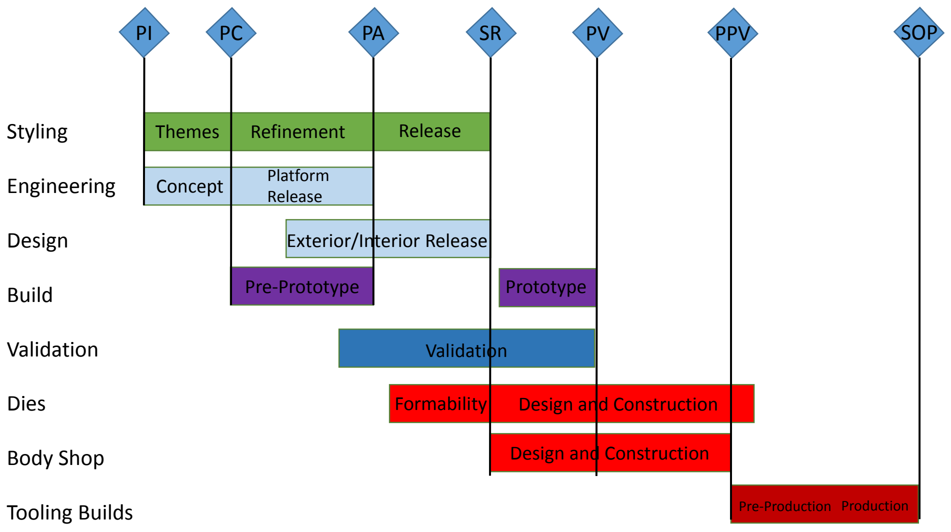


Figure 3.1: *Vehicle Development Process Overview [23]*

Engineering completes computer aided analysis (called virtual assessments) throughout the VDP to guide the design, identify engineering and manufacturing problems early, and eliminate the need for some physical builds and physical test. The next section will present the problem description to create a multi-domain simulation framework.

3.2 Multi-Domain Simulation Framework

In chapter 2 some systems present in a vehicle were illustrated. As a mechatronic system, the vehicle is a multidisciplinary system that is composed by the interaction of many subsystems from different engineering disciplines, as mechanical, electrical, software and control engineering. The interactions between different subsystems of a process are often very complex.

The engineers during the development and validation phases in the VDP have to deal with the difficulties of the integration of different subsystems. It is common that the development process of single parts are made as islands, where different parts are developed by different departments and there are no interactions between then. When an engineer wants to run a simulation that depends on models from other departments, the task to integrate all the models can be an arduous task.

Many effort is being carried by the OEM's to help the models exchange, trying to solve this problem. As described in chapter 2, FMI standard is a possible solution. Other companies are looking the problem and proposing solutions, as example the software CarMaker by IPG automotive, that helps the engineers to integrate many systems in a single simulation environment.

To help to carry this multi-domain simulation scenario, the software and systems development paradigm Model-Based Engineering is a possible solution. The next Section will give some explanations about MBE concepts.

3.3 Model-Based Engineering

Model-Based Engineering is a software and system development paradigm that emphasizes the application of visual modeling principles and best practices throughout the system development life cycle [24]. This section will describe the concepts of MBE.

Commercial mid-size vehicles comes with a high number of variants. According to a Siemens presentation about architecture driven design [25], one vehicle deals with many types of components, resulting in 2160 models to be evaluated. This vehicle could be presented through 1 architecture and 48 component models, capable to represent the 2160 configurations. This solution can be called as architecture driven design.

The idea is that the engineers have a common simulation architecture and they search the models from a valid model pool. This avoid unstructured simulation process where there are no control, security and traceability about the model quality. Figure 3.2 illustrated an example of a MBE workflow where in the number *1* represents the modeling engineers populating the model's data base, the number *2* the simulation engineer looking for the correct models and the number *3* the framework tool to run the simulation.

For every block in the architecture is defined a template, setting the interface contract. The contract is formed by the input/output signals and variables present in the subsystem. The models that fallows the template are called instrumented models. The instrumented models can be connect to the architecture as "plug-n-play" and have the

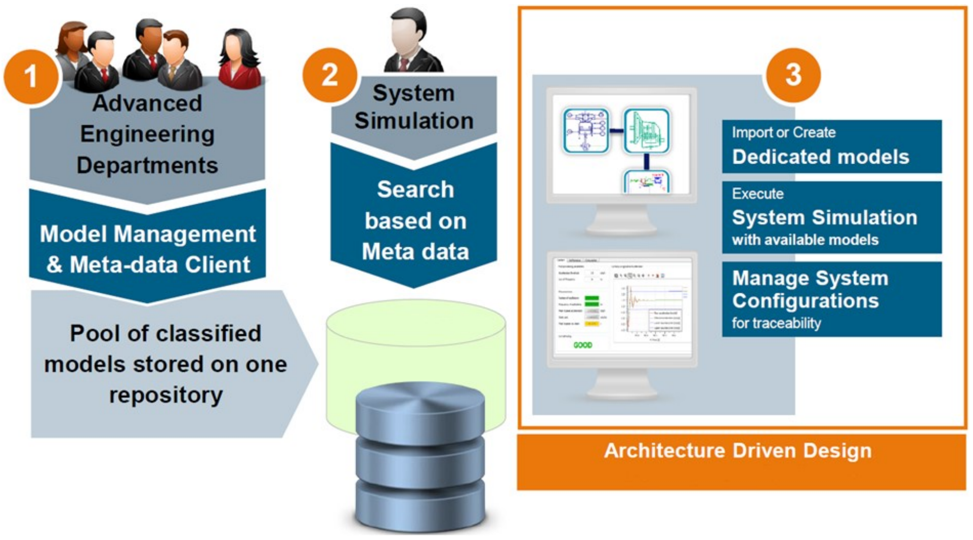

1
Advanced Engineering Departments
Model Management & Meta-data Client
Pool of classified models stored on one repository
2
System Simulation
Search based on Meta data
3
Import or Create
Dedicated models
Execute
System Simulation
with available models
Manage System Configurations
for traceability
Architecture Driven Design

Figure 3.2: *MBE Workflow [25]*

re-usability increased as they are connect with a single connector to the system, avoiding adapters.

With the FMI standard it is possible to solve the modeling tools heterogeneities, thus making the models exchange easier. If the FMU fallows the interface contract they can be connected to the architecture, growing the model's pool size.

With the architecture and the instrumented models, the simulation engineer can run several simulation scenarios, and analyze the attributes changes in the system due the subsystem variants. Figure 3.3 shows an example of variants evaluation during several simulation scenarios.

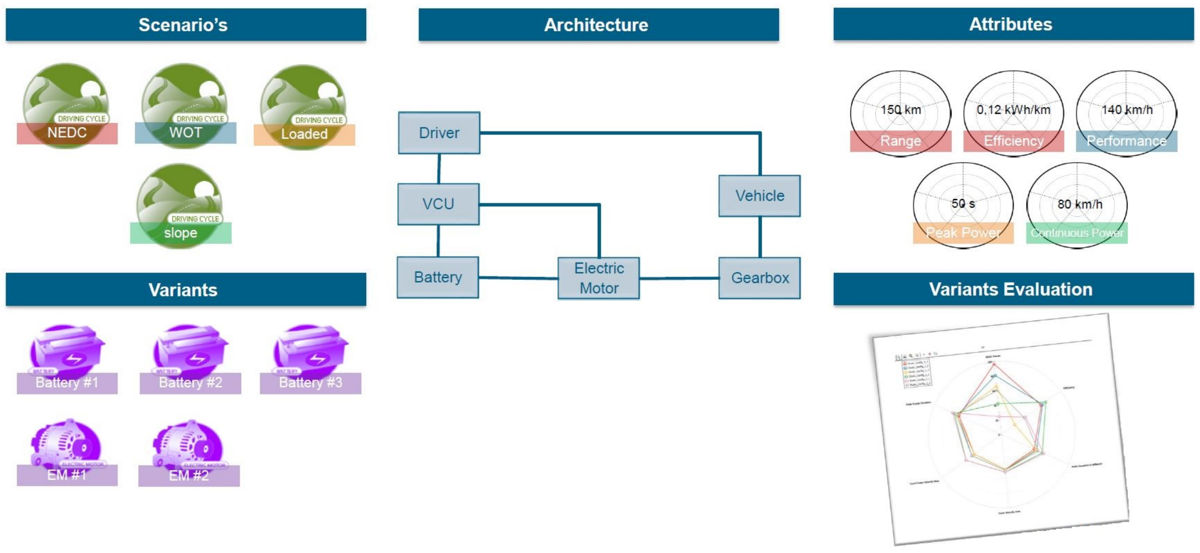


Figure 3.3: *Simulation with architecture overview [25]*

Using MBE, it is possible to increase the control and collaboration for models exchange, reduce the rework by reusable assets, save time and focus on engineering and give a better tool for analyze the system under diverse configurations and simulation scenarios. With this concepts the system requirement for the simulation environment were defined. The next sections will summarize the simulation scenarios, functional flow block diagram (FFBD) and requirements for the current work.

3.4 Simulation Scenario

To create the simulation scenarios for the simulation environment, it was asked to the engineers in the Electrical Power Systems & Controls integrations department, what kind of simulations they would like to have with such simulation tool. After an analyses of all the simulation scenarios proposed by the team, a list of scenarios were created. Table 3.1 summarizes the simulation scenarios.

Table 3.1: *Simulation scenarios*

Simulation Scenario - Name	Simulation Scenario - Description

Electrical Power Grid evaluation	Simple Vehicle dynamics;
	Complex Vehicle dynamics;
	Design Support

Charge Balance evaluation	SOC over time; Power Consumption
Dual Storage System evaluation	eRegen Comparison; CO2 emissions
Power Quality evaluation	Components electrical dynamics; Noise evaluation
StopStart evaluation	Performance analysis; SOF analysis
Autonomous Drive evaluation	Critical maneuvers dynamics
Controls Strategy evaluation	Optimize controls strategy

3.5 Functional Flow Block Diagram

To create the FFBD, the difficulties of the simulation tool CHABAL were taken into account in order to create a better simulation environment. The ideas of the MBE was taken into consideration as well to make the tool more capable and easy for the engineers in the team.

Figure 3.4 show the first level of the FFBD and figure 3.5 the second level of the FFBD.

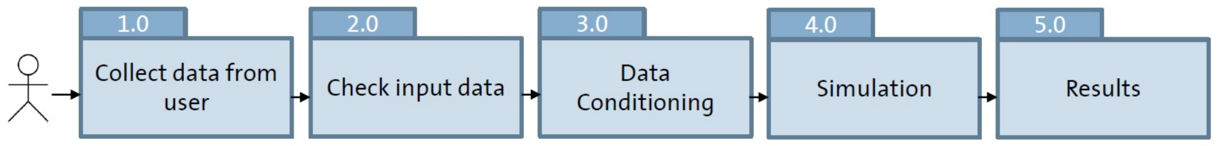


Figure 3.4: *Functional flow block diagram. First level.*

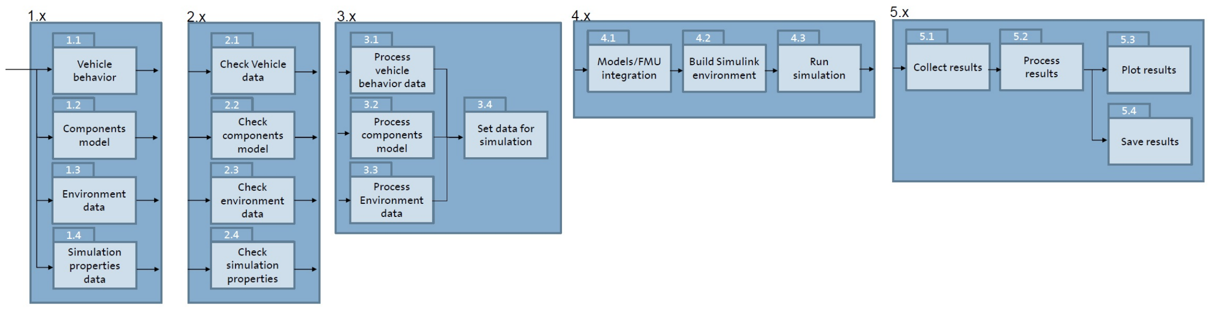


Figure 3.5: *Functional flow block diagram. Second level*

3.6 Requirements

To define the requirements, the simulation scenarios and the FFBD were taken into consideration. The concepts of MBE were taken into account as well, to create a tool more capable to aid the engineers work. The list above show the project requirements.

- 1) The simulation environment shall be capable to run offline simulations

  1.1) The simulation environment shall be capable to run MIL offline simulations

  1.2) The simulation environment shall be capable to run SIL offline simulations

- 2) The simulation environment shall be capable to run in Matlab/Simulink environment

  2.1) The simulation environment shall be capable to use simscape library

  2.2) The simulation environment shall be capable to run in the R2014b version

  2.3) The simulation environment should be capable to run in the R2017a version

- 3) The simulation shall be capable to run the simulation scenarios

  3.1) The simulation environment shall be capable to measure the charge balance in the electrical power grid

  3.1.1) The simulation environment shall be capable to emulates the energy flow in the electrical power grid under different drive scenarios

  3.1.2) The simulation environment shall be capable to emulates the battery SOC during charge and discharge events

  3.2) The simulation environment shall be capable to use dual storage system topology in the simulation

  3.2.1) The simulation environment shall be capable to use eRegen logics in the controls

  3.2.2) The simulation environment shall be capable to evaluate CO2 consumption

  3.2.3) The simulation environment shall be capable to simulate two batteries in parallel in the electrical power grid

  3.3) The simulation environment shall be capable to simulate the component dynamics in the electrical power grid

  3.3.1) The simulation environment shall be capable to use power profiles for the loads in the grid

  3.4) The simulation environment shall be capable to simulate StopStart events

  3.4.1) The simulation environment shall be capable to simulate StopStart events over different ambient temperature

  3.4.2) The simulation environment shall be capable to evaluate SOF

  3.5) The simulation shall be capable to simulate critical maneuvers in the vehicle

  3.5.1) The simulation environment shall be capable to simulate vehicle lateral dynamics

  3.5.2) The simulation environment shall be capable to simulate longitudinal dynamics

- 4) The simulation environment shall be capable to use Instrumented Models

  4.1) The user shall be able to switch component models using instrumented models in architecture

4.2) The simulation environment shall be capable to use FMU as instrumented models

- 5) The user shall be able to set the simulation environment by the GUI
- 6) The simulation environment shall be capable to show the simulation results to the user
- 7) The simulation environment shall be capable to save the results in data

3.7 Final Comments

This chapter presented the problem description and the system requirements of the current work. It was introduced the VDP to understand the role of the simulation environment in the vehicle development process and the difficulties to create a multi-domain simulation framework. The basic ideas of MBE were introduced in order to understand a possible solution for the current problem. Furthermore the system requirements for the Development of an Electrical Power Grid Simulation Environment for Commercial Vehicles was summarized. The next chapter will discuss the methodology used in the work and the system design.

Chapter 4

Methodology and System Design

This chapter will deal with the methodology of the present work and the system design. The Section 4.1 will show the methodology. The architecture design of the simulation environment will be described in Section 4.2. Section 4.3 will deal with the instrumented models design. The design of the user framework will be described in Section 4.5. Section 4.7 will show the final comments about the chapter.

4.1 Methodology

The present work used the V-Model methodology for the development process. The V-Model is an unique, linear development methodology used during a project development life cycle. The V-Model focuses on a fairly typical waterfall-esque method that follows strict, step-by-step stages. While initial stages are broad design stages, progress proceeds down through more and more granular stages, leading into implementation, and finally back through all testing stages prior to completion of the project.

Much like the traditional waterfall model, the V-Model specifies a series of linear stages that should occur across the life cycle, one at a time, until the project is complete. For this reason V-Model is not considered an agile development method, and due to the sheer volume of stages and their integration, understanding the model in detail can be challenging for everyone on the team, let alone clients or users [26].

The figure 4.1 shows the project development flow through the V-Model.

The format of the V-Model show the steps to be done during the various stages of the development process. The process starts at the top-left stage, and over time goes toward the top-right. After the verification and validation the development go back to the left-stage for project adjustments and iterate. The iterations goes until the project is complete.

For the present work, all the steps of the V-Model were followed. The first step, concept of operations, is described in chapter 3 by the simulation scenarios and FFBD. The second stage, requirements and architecture, are described in chapter 3 and section 4.2. The next stage, detailed design, is described in section 4.3 and 4.4. The implementation section is show in chapter 5. After the implementation, integration and

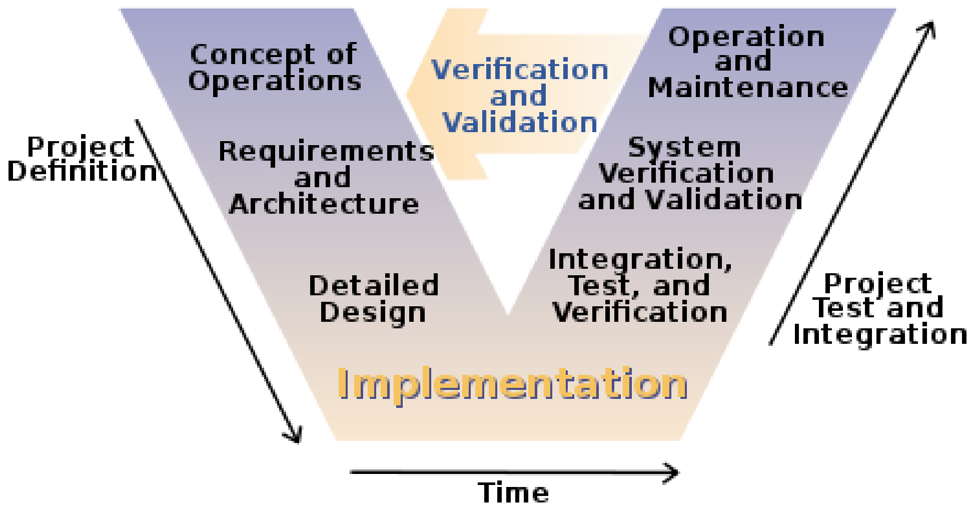

Concept of Operations
Verification and Validation
Operation and Maintenance
Project Definition
Requirements and Architecture
System Verification and Validation
Detailed Design
Integration, Test, and Verification
Project Test and Integration
Implementation
Time

Figure 4.1: *The V-model development flow [26]*

tests were made, as show in chapter 6. With the development cycle complete, was made the verification and validation, looking for possible problems to be fixed in the next iteration.

The next sections will provide the system design, describing the architecture, instrumented models and user framework design.

4.2 Architecture

As illustrated in section 3.2, the vehicle is a mechatronic system with complex iterations between different subsystems. The simulation framework has to be capable to simulate such complex system in a friendly way for the simulation engineers. As show in section 3.3,it is possible to create a simulation environment following the concepts of MBE with a common architecture facilitating the creation of the simulation scenarios.

To design the system architecture, first is necessary to understand what is the basic architecture of a mechatronic system. A typical mechatronic system takes up signals, processes them and outputs signals into forces and movements [27]. The figure 4.2 shows a basic architecture of a mechatronic system.

In the figure 4.2 it is possible to see that the system is broken in 4 major blocks: Plant, sensor, information processing and actuator. In classic control theory such approach is very common, where the plant is the process we want to control. The sensor is responsible to measure the variables in the process used by the control law. The information processing is where the control law will run and the actuator is responsible to act in the process to change the internal states.

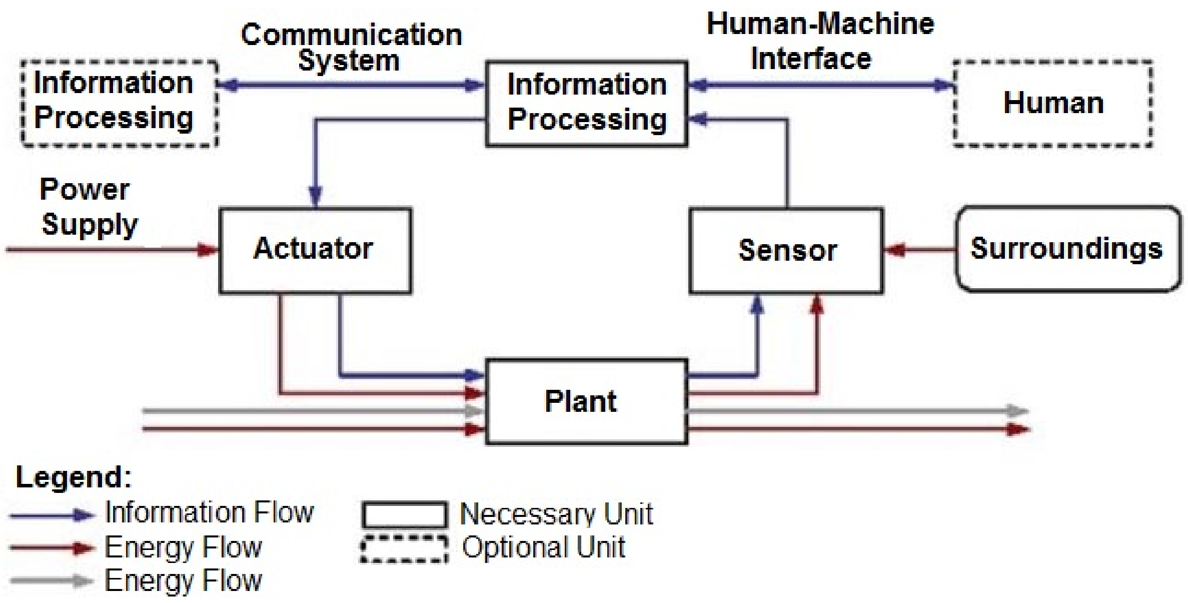


Figure 4.2: *Basic architecture of a mechatronic system [27]*

Following the basic architecture of a mechatronic system, was created the architecture design of the present work. In figure 4.3 is shown the upper level of the framework architecure.

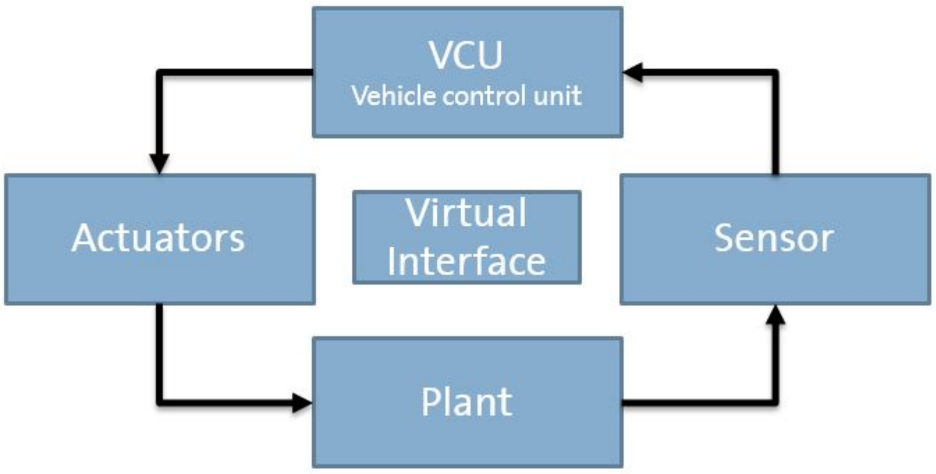


Figure 4.3: *Simulation Environment Architecture: Upper level*

Inside each major block, the components subsystems can be integrated in the simulation environment. For the plant is understood the power train, vehicle dynamics, electrical power storages and temperature dynamic models. The sensor block is where the sensor subsystems are placed, like CAN communication delays or others. The vehicle control unit (VCU) is where the ECU subsystems are placed. In the actuators block is placed the subsystems that acts in the plant, sometimes receiving signals from the

ECU. The virtual interface block is where occurs the interface between the user and the simulation environment. Figure 4.4 illustrates the architecture design with more details.

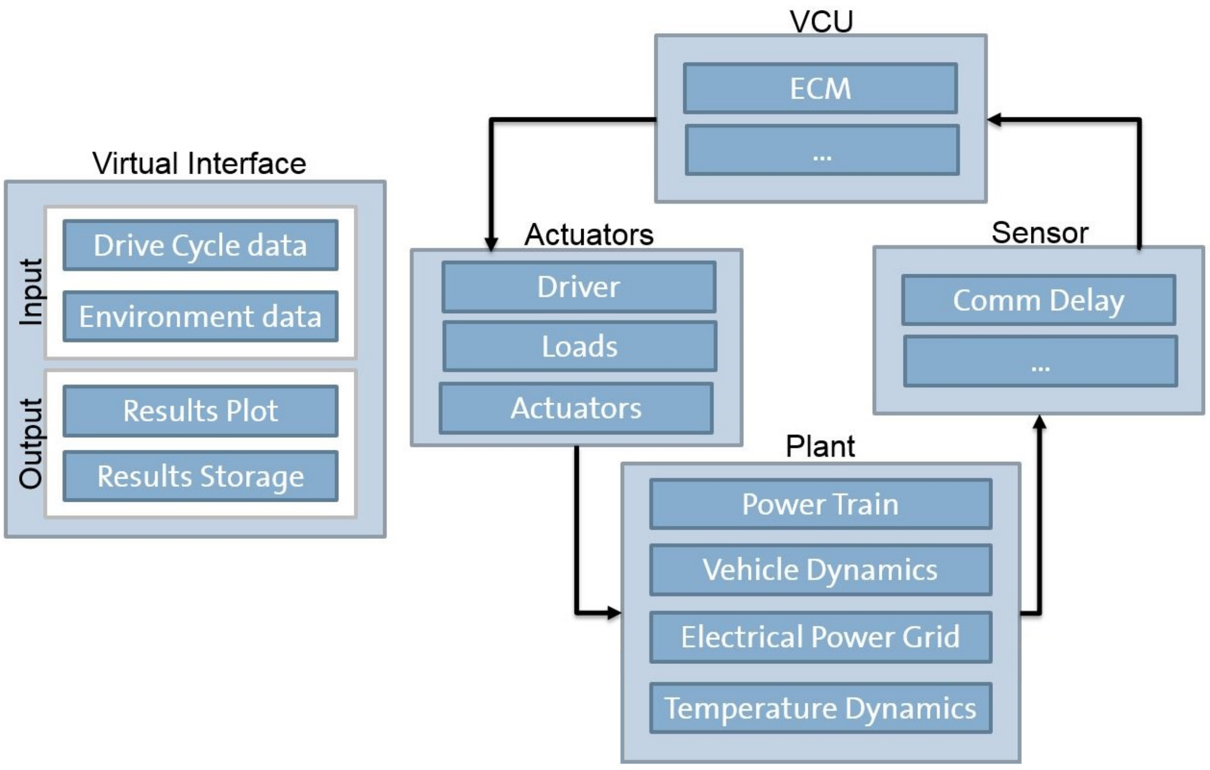


Figure 4.4: *Simulation Environment Architecture: Subsystems*

The idea of the framework is to provide a simulation environment to help the simulation engineers to run the simulation scenarios. The way the engineer connects the components do not change the architecture structure.

In order to make possible to use "plug-n-play" models it is necessary to define templates for the components. The next section will describe the models template.

4.3 Instrumented Models

In Chapter 3 it was demonstrated the concepts of MBE using architecture driven design. With models template it is possible to make the connections "plug-n-play" with single connectors. In order to create the templates for the models used in the simulation framework it is necessary to map all the subsystems that the simulation scenarios will need.

The creation of this list is not an easy task, and for the same component it is possible to have many different I/O configurations. To solve this problem, was created the concept of model wrapper/core for instrumented models. The next topics will describe in more details the wrapper and core models design.

4.3.1 Wrapper

The wrapper is the way the component is connect to the simulation framework. The idea is to have a single connector from the subsystem to the environment. Inside the wrapper is made the proper connections from the environment to the subsystem model, called model core. With the use of wrappers, it is possible to connect any model core to the simulation environment using a single connector.

Figure 4.5 illustrates the wrapper design.

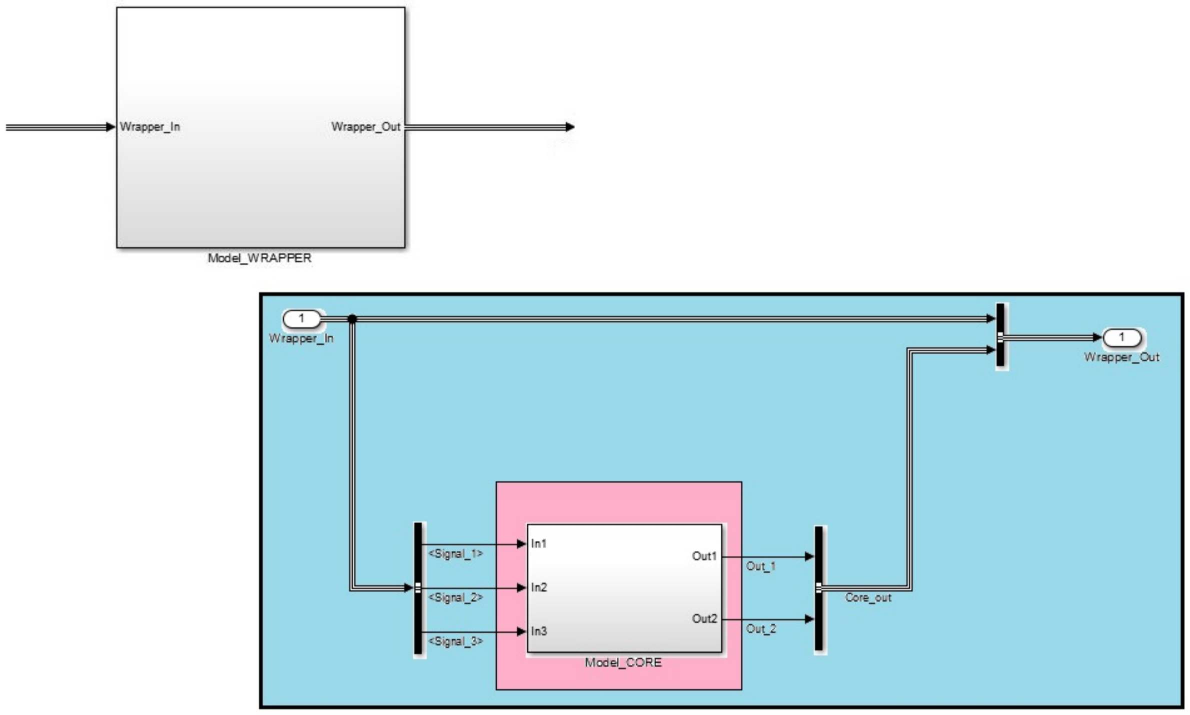


Figure 4.5: *Instrumented model. Wrapper/Core design*

4.3.2 Core

The core is the model itself. The number of inputs and outputs can vary depending the way the model was created. With the help of the wrapper, it is possible to connect the model core to the simulation environment with a single connector. To merge many signals in one, is created a bus to rout signals through the simulation environment.

The figure 4.5 illustrate how the model core is connected to the wrapper and the signal routing.

To aid the models manipulation and scenarios configuration is necessary an interface between the user and the simulation framework. The next section will describe the user framework design.

4.4 User Framework

This section will describe the interface between the user and the simulation framework in order to help the engineer to set the simulation scenario and the models manipulation.

To aid the engineer to use the tool is necessary do design a graphical user interface (GUI) that can be able to integrate the models from the data base to the simulation environment and the simulation scenarios. The figure 4.6 illustrates the user framework design.

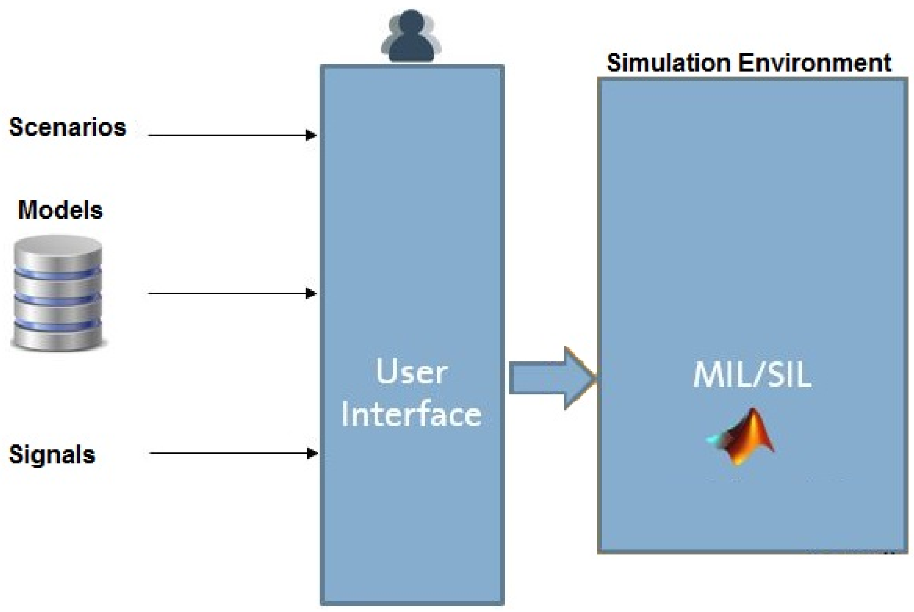


Figure 4.6: *User framework design*

As show in figure 4.6, the user interface has to be able to integrate the models, scenarios and signals to the simulation environment. For the creation of the scenarios was used the software CarMaker in co-simulation with the simulation tool. For the models integration is used the Object-Oriented Simulation Model Tree (OSIMOT) tool created by the power train department in Adam Opel Gmbh. For the signals manipulation were created the Transient Profile Creator (TPC) tool to help the signals creation. The next chapter will demonstrate how the tools were integrated to the simulation environment.

4.5 Final Comments

This chapter presented the methodology and the system design of the present work. It was introduced the architecture, instrumented models and user framework design.

Moreover was demonstrated the v-Model methodology used in the project. The next chapter will show the implementations of the simulation environment.

Chapter 5

Implementation

This chapter will deal with the implementations of the simulation environment. The Section 5.1 will describe the architecture implementation. The CarMaker and OSIMOT integration will be shown in Section 5.2 and 5.3. The model database using the OS-IMOT tool will be demonstrated in section 5.4. The Section 5.5 will present the FMU integration. The transient load profile tool integration will be described in Section 5.6. Furthermore the final comments about the implementations will be shown in Section 5.7

5.1 Architecture

For the architecture implementation was followed the architecture design illustrated in section 4.2. To adapt the vehicle system to the design was analyzed the major systems in the vehicle and how they exchange information and energy. Figure 5.1 illustrates the upper level of the systems to be fit in the architecture design.

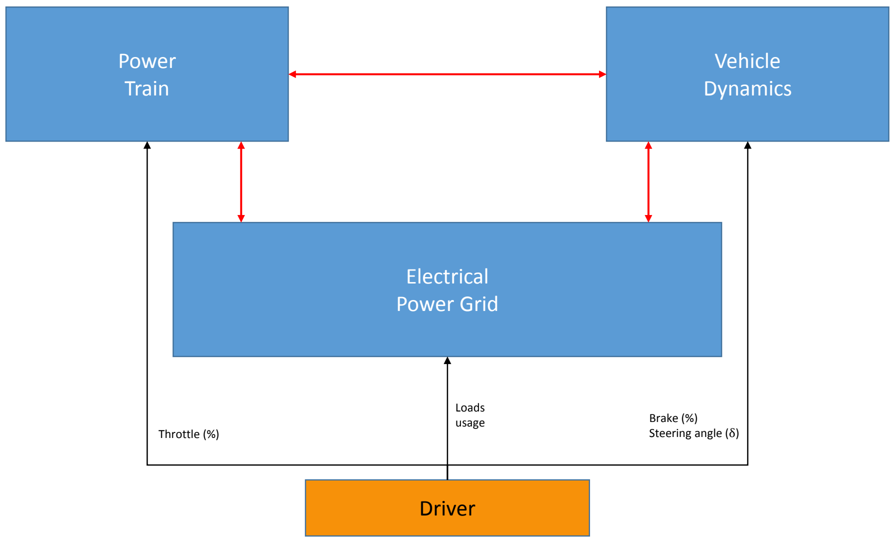


Figure 5.1: *Vehicle major subsystems: Upper level*

As shown in figure 5.1, it is possible to see that the driver acts in both systems, sending informations, like acceleration pedal position, steering angle, gear number (for manual transmission), brake pedal position and loads usage. The other systems are exchange energy between then, as seen by the red arrows.

To adapt this major systems to the architecture design was made a further analysis to collect the major subsystems in each system and to see how they interact. The figure 5.2 shows the energy and information transfers between the systems in more detail.

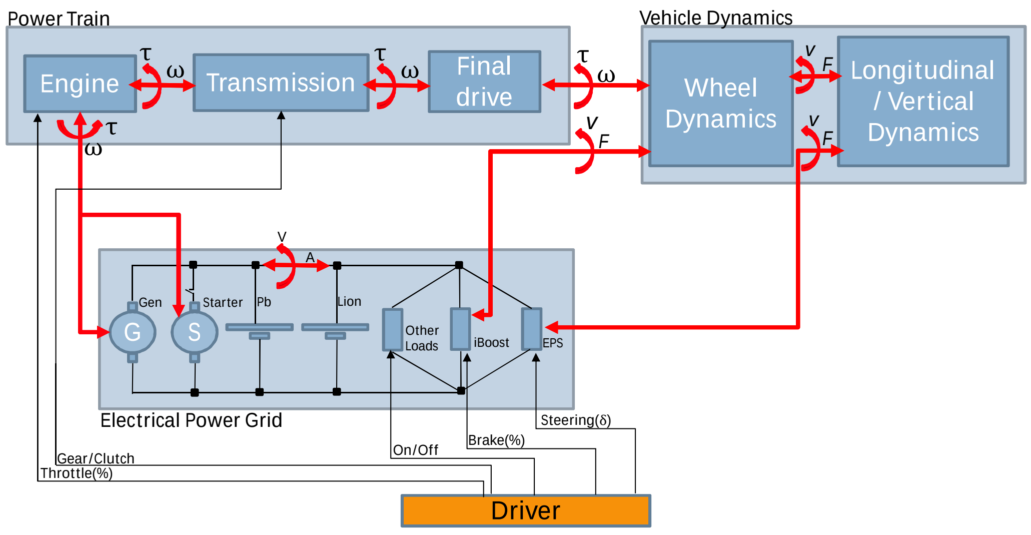


Figure 5.2: *Vehicle major subsystems: Detailed*

To adapt the vehicle to the simulation environment was necessary to brake the subsystems showed in the figure 5.2 to the architecture design. The figure 5.3 shows the implementation of the vehicles systems to the simulation environment.

The implementation was made in Matlab/Simulink environment. To fallow the instrumented models design, the connections between the models were made with a single connector. To merge all the signals in one bus it was used the bus creator from simulink for signal routing. It is possible to see in the figure 5.3 that between the actuator and plant systems there are 3 connections. The arrow is the uni-directional information between the causal models. The two other connections are the positive and negative electric physical ports for the acausal connections between the acausal

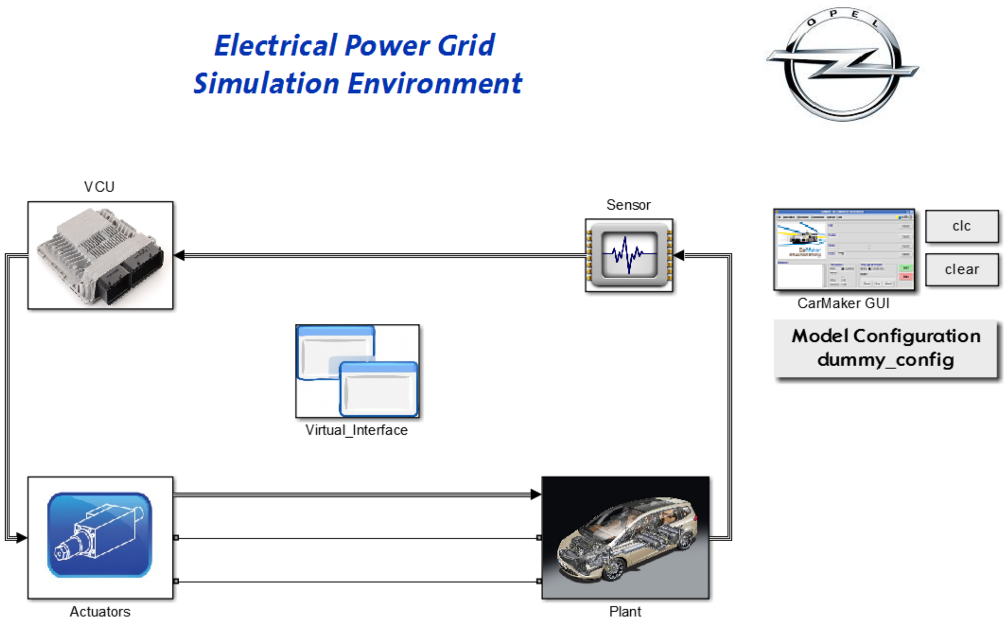


Figure 5.3: *Simulation environment architecture: Upper level*

models. Simulink has the simscape library for acausal modeling.

All the systems have communication with the virtual interface system. To create the connections it was used the blocks *goto* and *from* from Simulink. The figure 5.4 illustrates an example of how were made the connections.

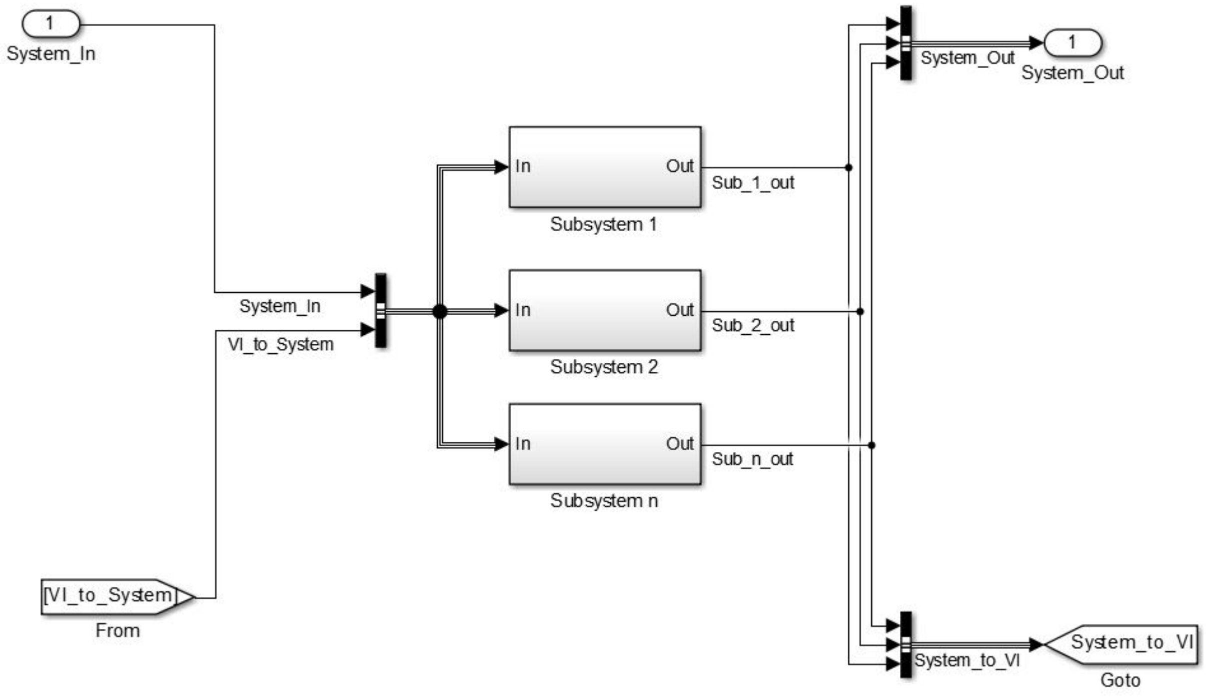


Figure 5.4: *Simulation environment architecture: Generic connections*

The next subsections will give more details about the implementation of the plant, actuators, VCU, sensor and virtual interface systems.

5.1.1 Plant

In the plant system is placed the power train, vehicle dynamics, temperature dynamics and electrical power grid models. Figure 5.5 illustrates the models used in the plant system.

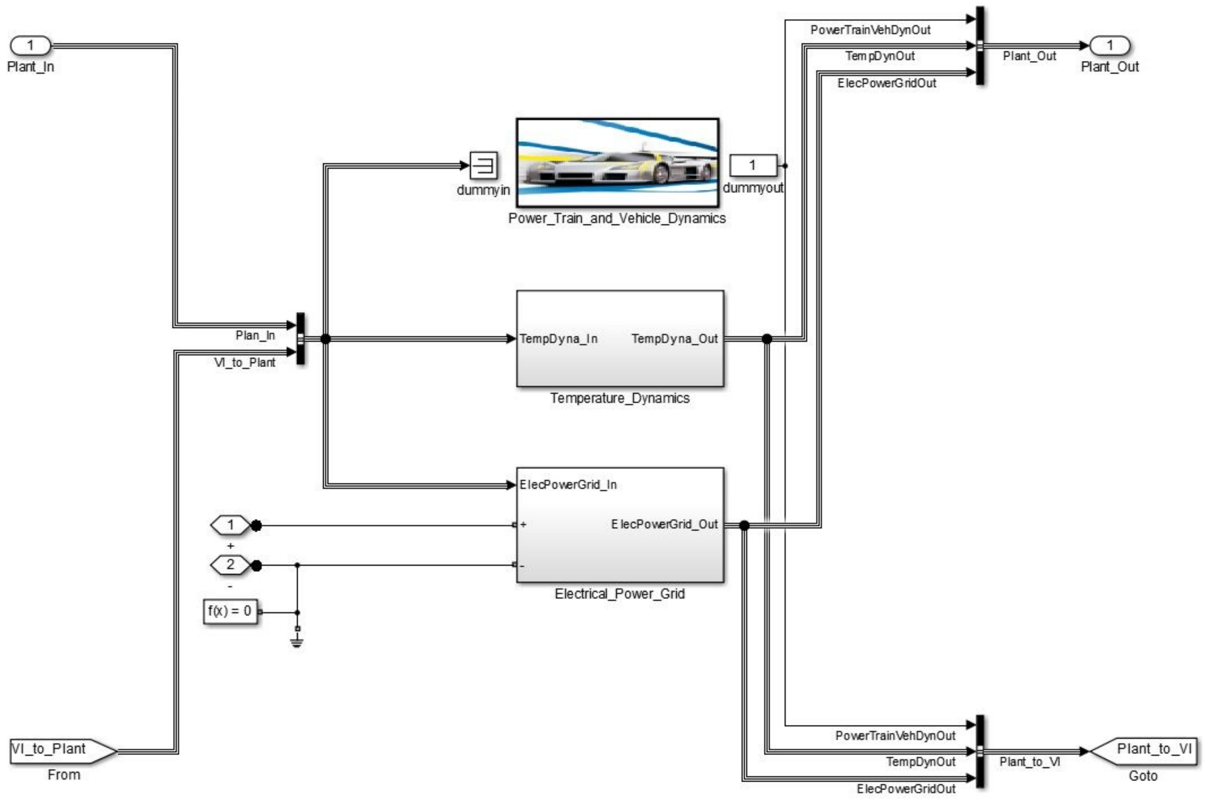


__Figure 5.5:__ Simulation environment architecture: Plant

Inside each subsystem it is possible to have the models of single components to be used in the simulation. The connection template showed in the figure 5.4 is used to keep the models using single connectors. For the models configuration was implemented the OSIMOT tool. More details about the tool will be given in the section 5.3.

The power train and vehicle dynamics model have no connection to the signal bus. This is because these models are connected to the CarMaker software that runs in co-simulation with the Simulink. In section 5.2 will be given more details about the interface between Simulink and CarMaker.

5.1.2 Actuators

In the actuators system is placed the components that act in the plant, receiving signals from the VCU. The figure 5.6 shows an example of models placed in the actuators system.

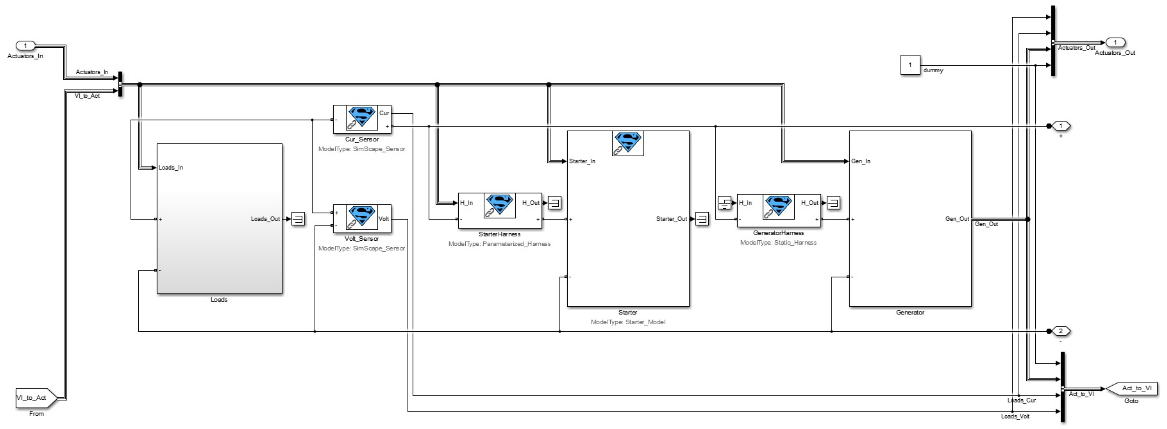


Figure 5.6: Simulation environment architecture: Actuators

It is possible to see in figure 5.6 that there are models for the loads, cable harness, starter and generator. The models have connections to the signal bus and to the electrical physical ports.

5.1.3 VCU

The VCU represents the ECU's from the vehicle. The figure 5.7 shows an example of the implementation of the ECM, BCM and signal bypass.

To complete the signal loop, it was necessary to create the signal bypass model to forward signals from models that are not connected direct. For example a signal from the plant system that needs to be an input in the actuators system.

5.1.4 Sensor

In the sensor system is placed the dynamics of the sensors that are collecting information from the plant and sending to the VCU. Sometimes there have no dynamics to be considered, so a bypass model can be used to pass the signal forward. The figure 5.8 shows an example of the implementation of a Pb battery sensor and a bypass sensor.

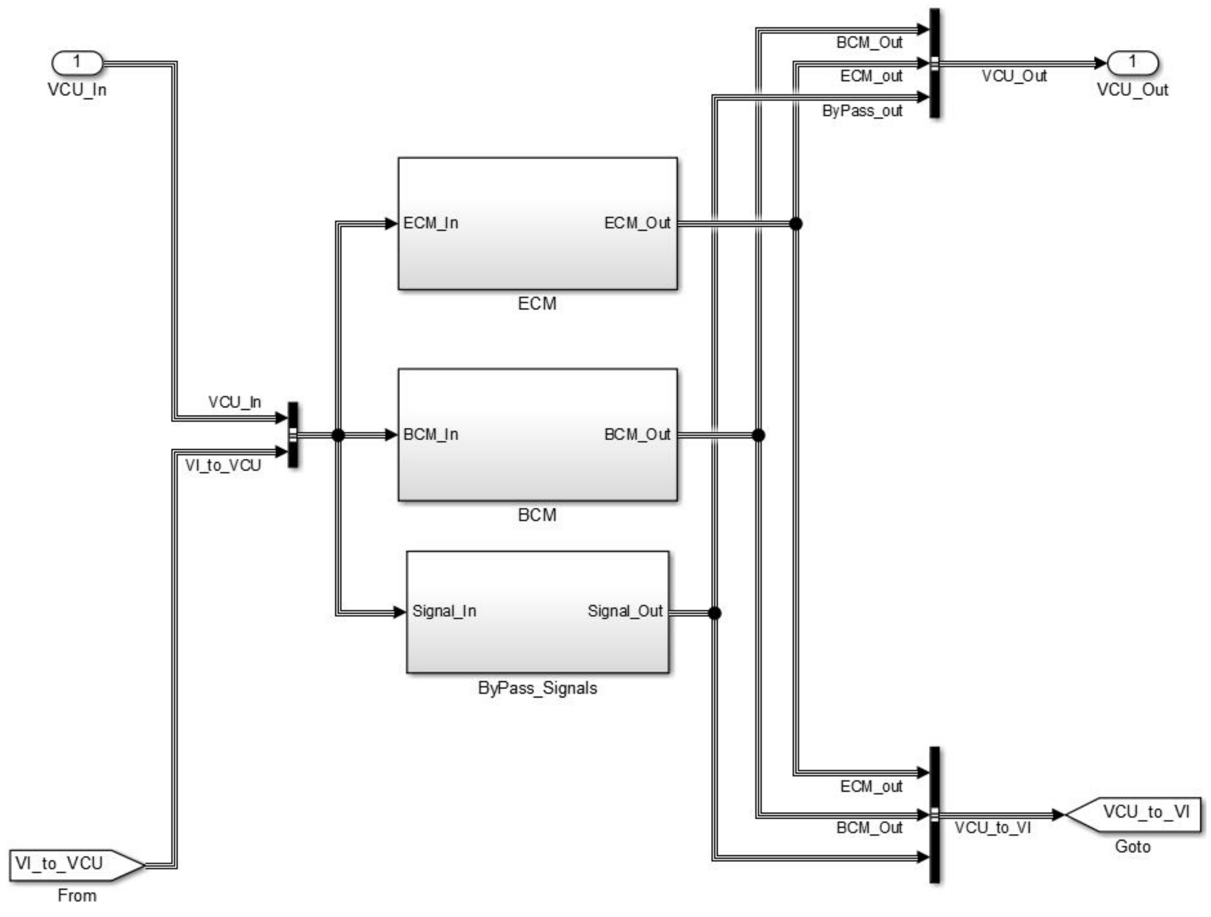


Figure 5.7: *Simulation environment architecture: VCU*

5.1.5 Virtual Interface

To have an interface between the user and the signals in the simulation framework was implemented the virtual interface system. This system has interface with all the other systems in the simulation environment using *goto* and *from* Simulink blocks. The virtual interface system is used also to make the interface between some CarMaker signals and the Simulink models. The figure 5.9 shows the virtual interface system and the subsystems implemented.

In the virtual interface system it is possible for the user to observe the signals from the simulation and to insert signals for the models. The figure 5.10 shows an example of implementation for the actuators interface. In the *Inter_In* port it is possible to see the signals going to *scope* blocks to display the simulation results. In the *Inter_Out* it is possible to see the signals going to the other models, like some signals to the generator in the *GenInputs* bus or user created signals to simulate the loads usage in the *DriverInputs* bus. It is possible to note the orange block *PT.Engine.rotv* that is used for the CarMaker interface. This topic will be explained in the next section.

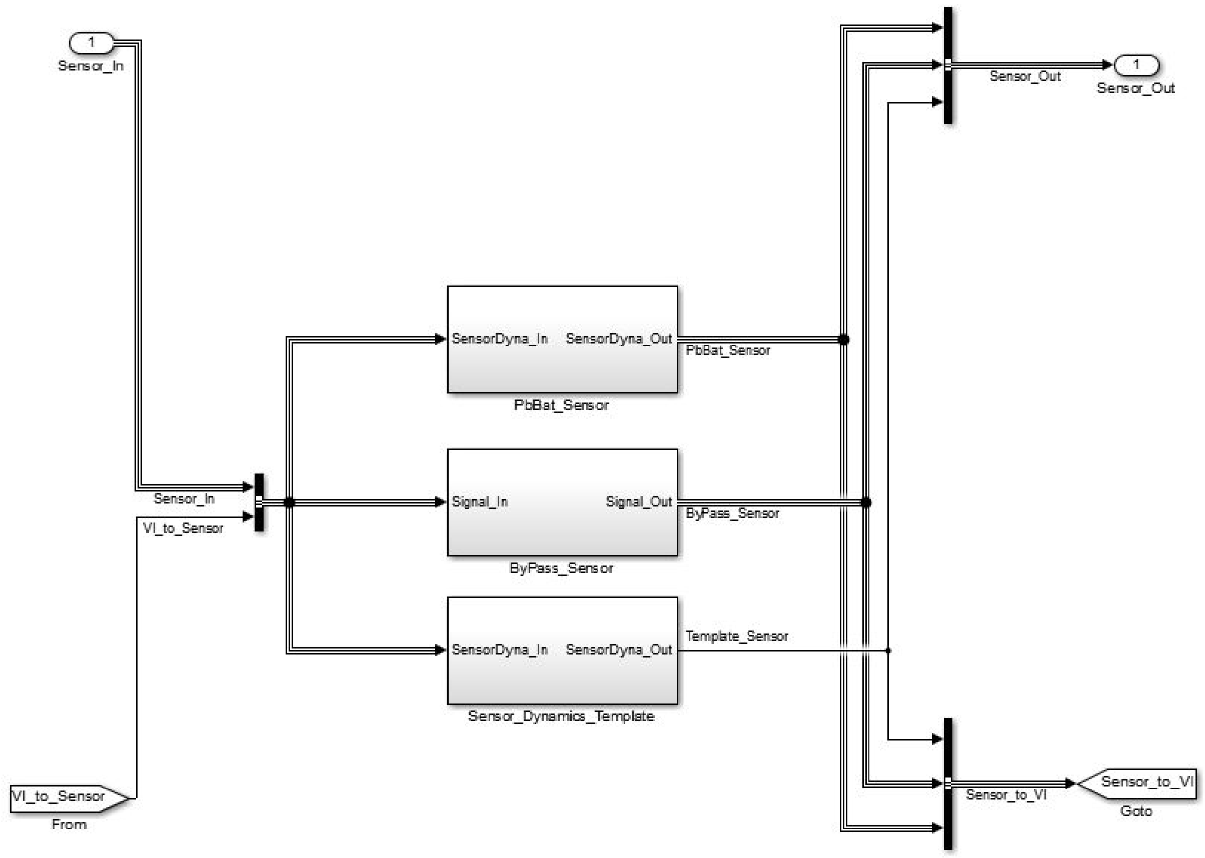


Figure 5.8: *Simulation environment architecture: Sensor*

5.2 CarMaker Integration

To add more applicabilities to the simulation environment and aid the engineers to create the simulation scenarios it was made the integration between the CarMaker and the Matlab/Simulink. This section will explain in more details how the implementation was made.

CarMaker for Simulink is a complete integration of IPG's vehicle dynamics simulation software, CarMaker, into The MathWorks' modeling and simulation environment, Matlab/Simulink. The highly optimized and robust features of CarMaker were added to the Simulink environment using an S-Function implementation and the application programming interface (API) functions that are provided by Matlab/Simulink.

Access to CarMaker simulation results is possible using the *cmread* utility that can be called from within Matlab. This utility loads the data of any CarMaker simulation results file into the Matlab workspace. After that, the data can be manipulated and viewed, e.g. for post processing purposes, using any of the available Matlab tools [28].

When is made a CarMaker project the folder *src_cm4sl* is created. This is the de-

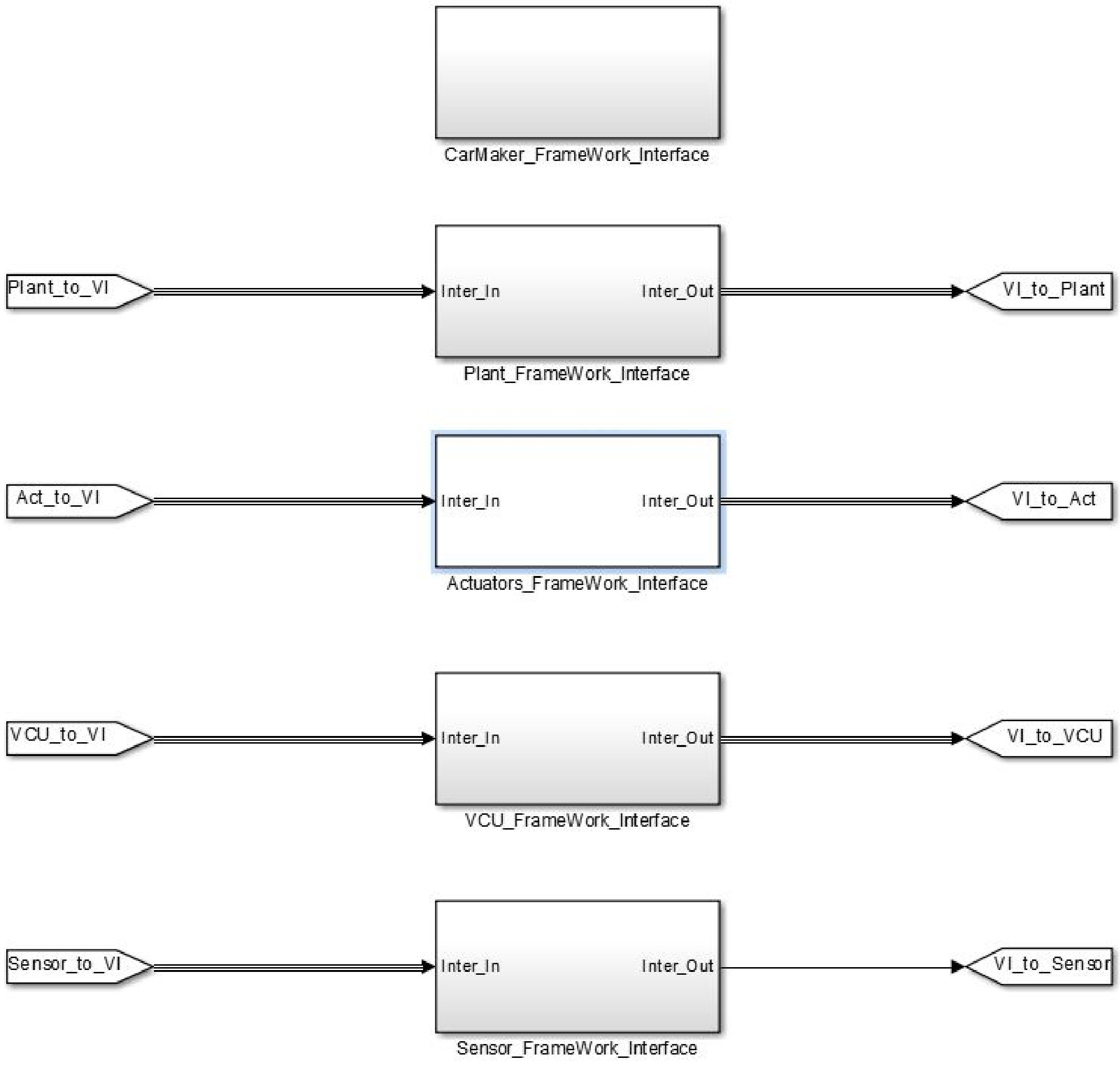


Figure 5.9: *Simulation environment architecture: Virtual Interface*

fault place where the simulink project will be added to have integration with CarMaker. Inside the folder there is a *generic.mdl* file that have the CarMaker simulink blocks to be added to the Simulink project. For the integration it is just necessary to copy and paste the *CarMaker* and *Open CarMaker GUI* blocks to the Simulink project. Other important file is the *cmenv.m* that set the paths for the Matlab to integrate the CarMaker functions. In the figure 5.3 it is possible to see the CarMaker GUI block and in the figure 5.5 it is possible to see the CarMaker block in the vehicle dynamics and power train system.

If the user clicks two times in the CarMaker GUI block, the block has a callback function that will open the CarMaker software. The figure 5.11 illustrates the basic view of the software showing the screen where you can set the drive scenarios, the *IPGMovie* to see the vehicle dynamics during the simulation and the *Instruments* to

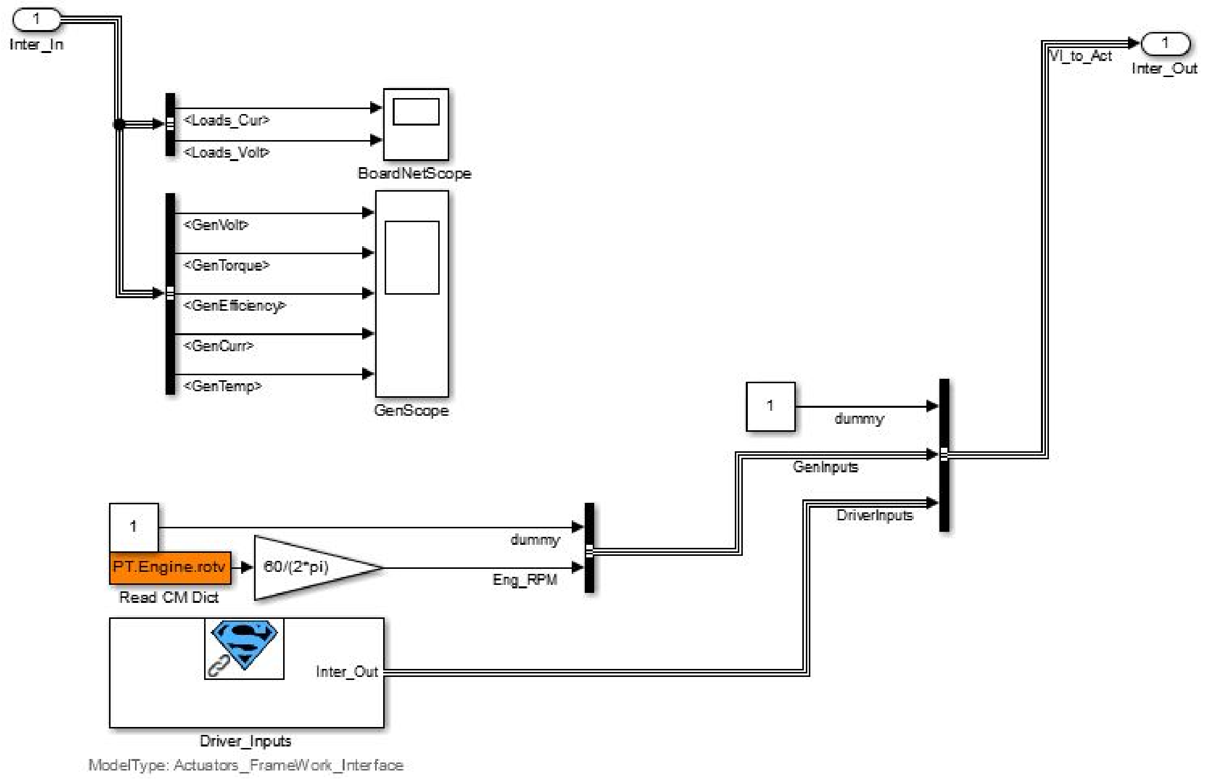


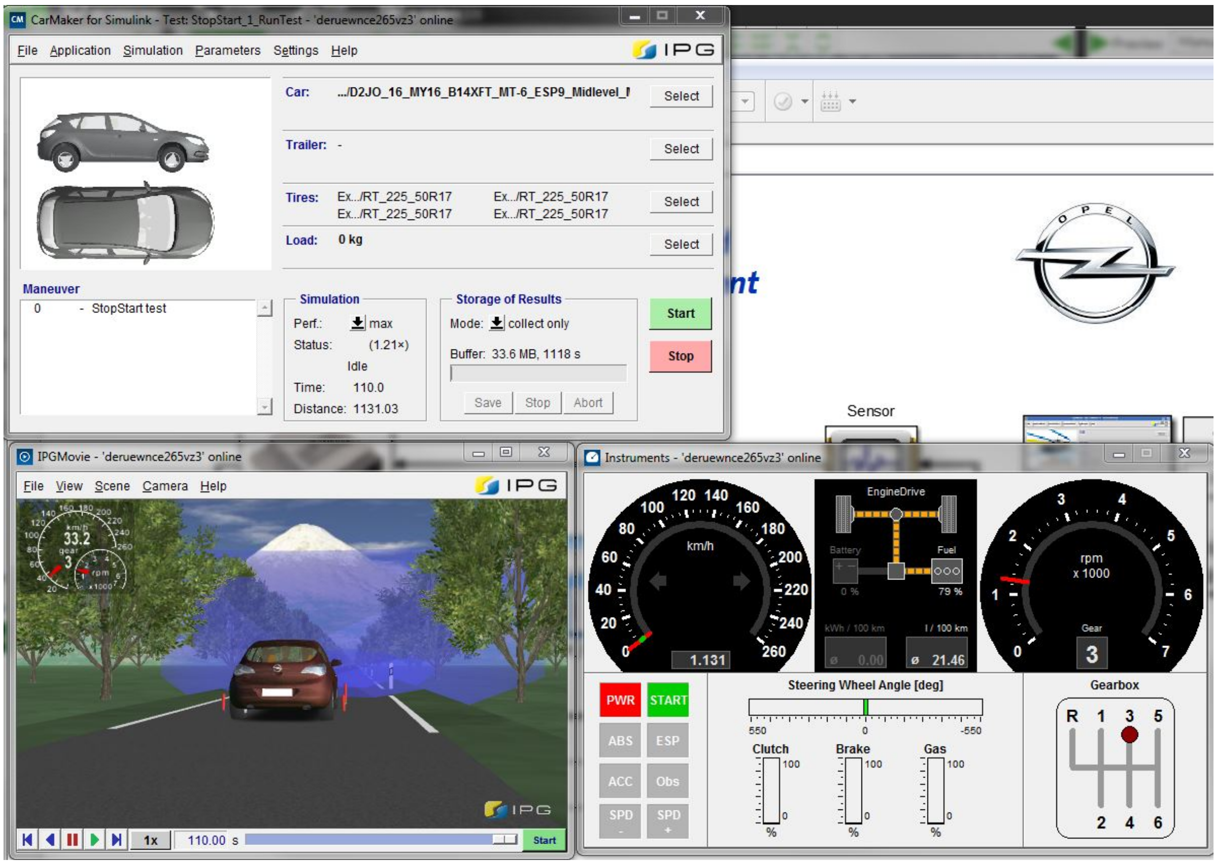
Figure 5.10: *Simulation environment architecture: Virtual Interface, Actuators interface*

see the vehicle's cluster, pedals positions, fuel consumption and gear selection.

To integrate the signals from the CarMaker model to the Simulink it is possible to use the CarMaker library added to the Matlab environment. Figure 5.12 illustrates the Simulink library browser showing the possible blocks to be integrated to the Simulink project.

With the block *Read CM Dict* and *Write CM Dict* it is possible to manipulate the CarMaker simulation signals. In the figure 5.10 in the orange color the block *Read CM Dict* is used to get the values from the engine RPM. When manipulating signals it is important to take a look in the CarMaker documentation to check the signal unit. The list off all the signals that can be used in the Simulink and the units can be found in the CarMaker reference manual in the user accessible quantities chapter.

With the CarMaker integration, it is possible to give more capabilities to the simulation framework. To aid the models manipulation the OSIMOT tool was integrated. The next section will explain how this integration was made.


CarMaker for Simulink - Test: StopStart_1_RunTest - 'deruewnce265vz3' online

Car: .../D2JO_16_MY16_B14XFT_MT-6_ESP9_Midlevel_I

Trailer: -

Tires: Ex.../RT_225_50R17 Ex.../RT_225_50R17 Ex.../RT_225_50R17 Ex.../RT_225_50R17

Load: 0 kg

Maneuver: 0 - StopStart test

Simulation: Perf.: max; Status: (1.21×) Idle; Time: 110.0; Distance: 1131.03

Storage of Results: Mode: collect only; Buffer: 33.6 MB, 1118 s

IPGMovie - 'deruewnce265vz3' online

Instruments - 'deruewnce265vz3' online

Figure 5.11: *CarMaker GUI*

5.3 Object-Oriented Simulation Model Tree (OSIMOT) integration

The Object-Oriented Simulation Model Tree was created by the power train department in Adam Opel GmbH to aid the models configuration in Simulink projects. The OSIMOT tool works similar to the normal Simulink library but with some more functionality.

To integrate the tool to the simulation framework it is necessary to add to the Matlab path all the folders used by the OSIMOT. When the path is set, it is possible to add the model configuration block to the Simulink project and use the OSIMOT features. In figure 5.1 it is possible to see the model configuration block using the *dummy_config* configuration. The block has a call back that when clicked two times opens the model configurator framework as show in figure 5.13.

In the figure 5.13 it is illustrated the models associated to the *dummy_config* configuration. The user can change and create any configuration for the models used in the Simulink project. If the user with the mouse right click in any subsystem in the

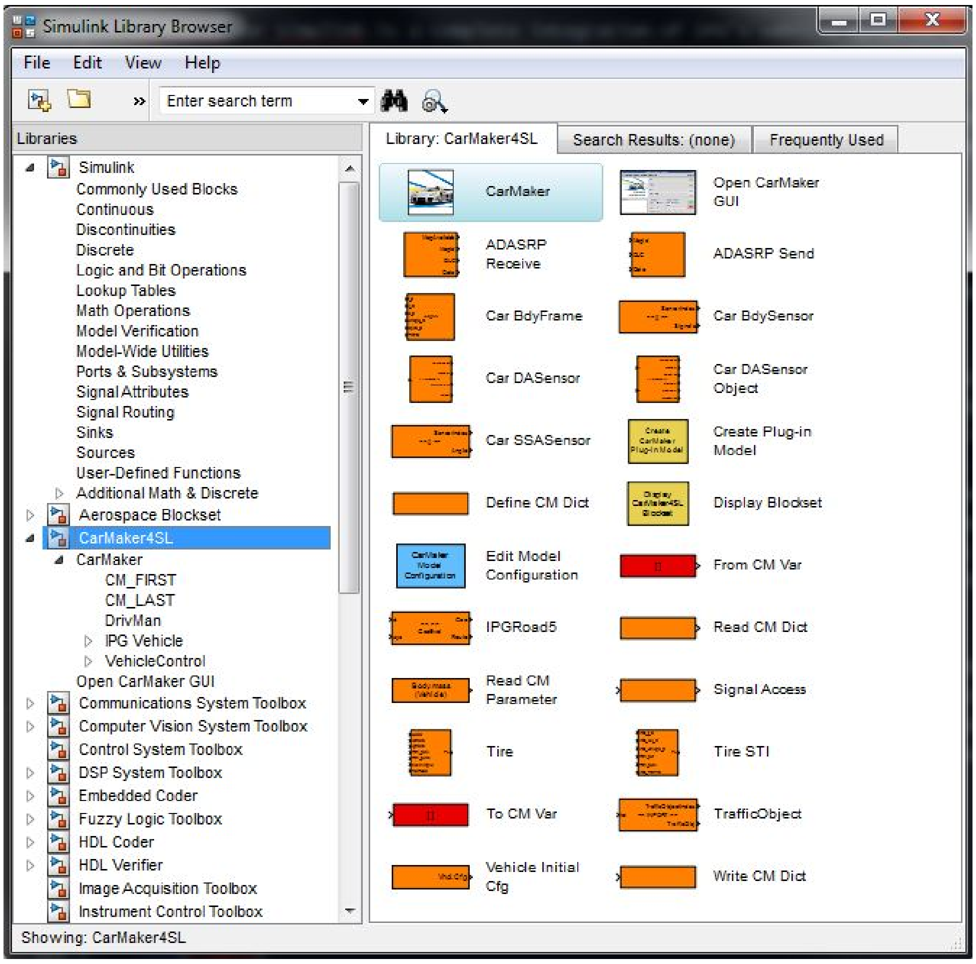


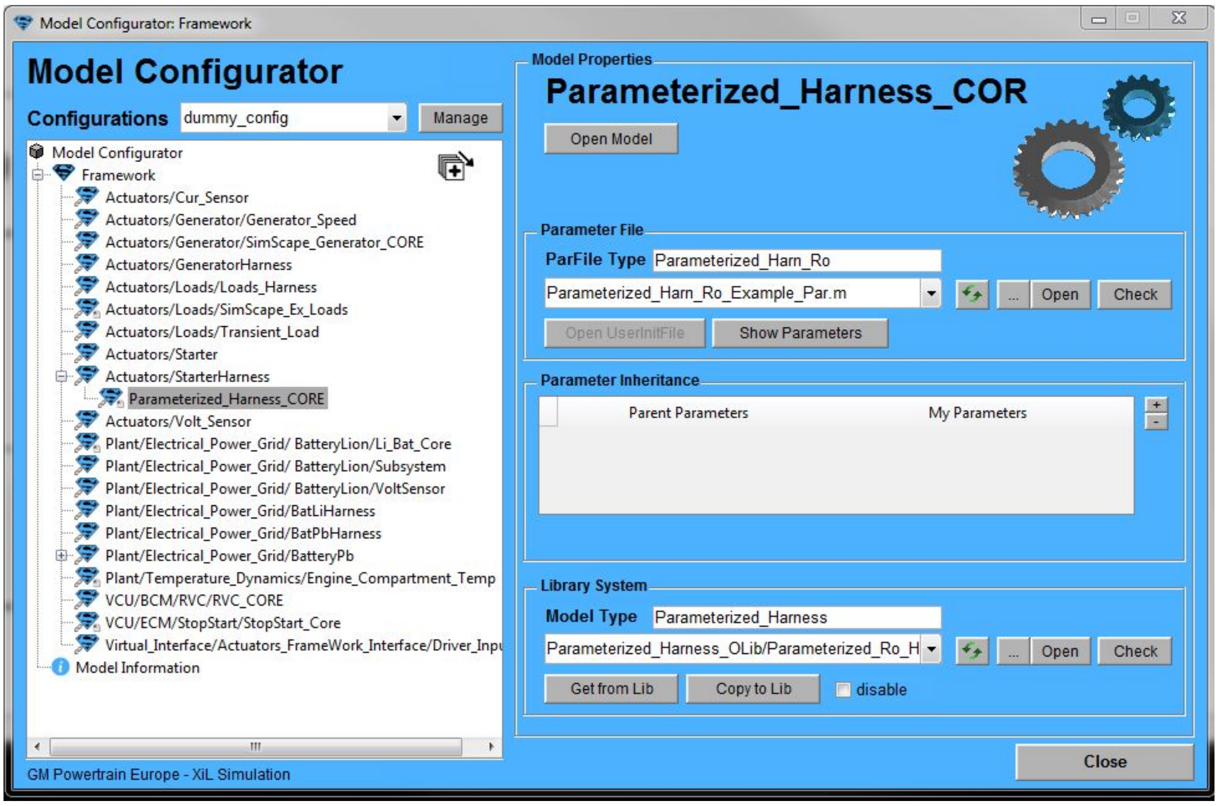
Figure 5.12: *CarMaker simulink library*

Simulink, a new *model configuration* item will be shown in the list. In the *model configuration* option it is possible to open the OSIMOT GUI as showed in the figure 5.13 by pressing the *open model configuration* option or making the subsystem as a OSIMOT block, integrating all the OSIMOT capabilities to the subsystem.

The model configuration framework shows all OSIMOT blocks in the Simulink project associated to the configuration and the subsystem hierarchy when you have OSIMOT blocks inside OSIMOT blocks. With the model configuration framework it is possible to set parameter files to associate with the OSIMOT model blocks or to exchange models from the library system.

The parameter files are created following a template and each file is associate to a single model. The user can open the files, that are Matlab scripts, and change the parameters values according the model specifications. It is possible to create parameters inheritance if the user want to change some specif parameter from a model without having to change the parameters files from all the models.

In the library system the user can exchange models from a model database created


Model Configurator: Framework

Model Configurator

Configurations: dummy_config

- Model Configurator
  - Framework
    - Actuators/Cur_Sensor
    - Actuators/Generator/Generator_Speed
    - Actuators/Generator/SimScape_Generator_CORE
    - Actuators/GeneratorHarness
    - Actuators/Loads/Loads_Harness
    - Actuators/Loads/SimScape_Ex_Loads
    - Actuators/Loads/Transient_Load
    - Actuators/Starter
    - Actuators/StarterHarness
      - Parameterized_Harness_CORE
    - Actuators/Volt_Sensor
    - Plant/Electrical_Power_Grid/ BatteryLion/Li_Bat_Core
    - Plant/Electrical_Power_Grid/ BatteryLion/Subsystem
    - Plant/Electrical_Power_Grid/ BatteryLion/VoltSensor
    - Plant/Electrical_Power_Grid/BatLiHarness
    - Plant/Electrical_Power_Grid/BatPbHarness
    - Plant/Electrical_Power_Grid/BatteryPb
    - Plant/Temperature_Dynamics/Engine_Compartment_Temp
    - VCU/BCM/RVC/RVC_CORE
    - VCU/ECM/StopStart/StopStart_Core
    - Virtual_Interface/Actuators_FrameWork_Interface/Driver_Inpu
  - Model Information

Model Properties: Parameterized_Harness_COR

Parameter File — ParFile Type: Parameterized_Harn_Ro; Parameterized_Harn_Ro_Example_Par.m

Parameter Inheritance — Parent Parameters | My Parameters

Library System — Model Type: Parameterized_Harness; Parameterized_Harness_OLib/Parameterized_Ro_H

GM Powertrain Europe - XiL Simulation

Figure 5.13: *OSIMOT model configurator framework*

using the OSIMOT tool. The model exchange is transparent to the user if the models have the same number of I/O ports. If no, the user needs to handle the signals connections. The next section will give more details about the model database using the OSIMOT library system.

5.4 Model Database

With the OSIMOT tool it is possible to configure the subsystems in the project from a model database. This section will demonstrated how the model database integration was made.

To integrate a model database to the simulation environment, it was used the OSIMOT functionality for library system. The user has to create a *.mdl* file where he wants to place the models to be used in the database. The model database is a set of *.mdl* files with the models inside.

To associate the models from the database to the simulation environment, they have to be OSIMOT blocks, as described in section 5.3. The figure 5.14 shows an example of models library with some subsystems inside the file.

When using the OSIMOT GUI in the library system option, the user can select the

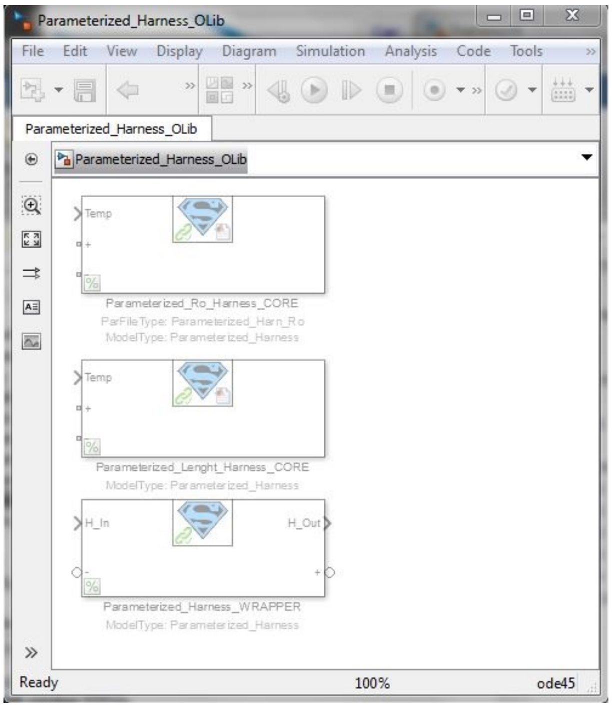

Parameterized_Harness_OLib
File Edit View Display Diagram Simulation Analysis Code Tools
Parameterized_Harness_OLib
Parameterized_Harness_OLib
Temp
Parameterized_Ro_Harness_CORE
ParFileType: Parameterized_Harn_Ro
ModelType: Parameterized_Harness
Temp
Parameterized_Lenght_Harness_CORE
ModelType: Parameterized_Harness
H_In
H_Out
Parameterized_Harness_WRAPPER
ModelType: Parameterized_Harness
Ready
100%
ode45

Figure 5.14: *OSIMOT model library system*

mdl file with the models database and select the subsystem he wants to integrate in the simulation environment.

One of the OSIMOT features is to create a link between the OSIMOT subsystem used from a library and the original subsystem in the database file. If the model somehow is different from the original one inside the library system file, the tool warns the user of the differences.

It is possible to save the files used by the library system in a common server, so the users of the simulation environment can have remote access to the models and can exchange models between the departments. With the usage of FMI standard it is possible to exchange models more easily. The next section will explain how was made the FMI integration to the simulation environment.

5.5 FMI integration

In section 2.3 was demonstrated the concepts of the FMI standard. This section will demonstrated how the FMI integration was realized.

5.5.1 Model-Exchange and Co-Simulation

With the FMI standard it is possible to simulate with model-exchange or co-simulation models. The difference between them is that for the model-exchange the models are using the same solver of the simulation environment and for the co-simulation the solver is build in the FMU model.

Using the same solver of the simulation environment, the FMU model is running at the same sample time of the simulation. When is used the co-simulation variant, the sample time of the model is defined by the build in solver. With this, it is possible to make the simulation to run faster, using less computation power.

There are many tools to create and integrate FMU models. As example the FMI Toolbox for MATLAB/Simulink from Modelon. For the simulation environment was used a Pilot Support Package (PSP) with FMI support.

The PSP has a Simulink library for FMU integration with the possibilities to use model-exchange and co-simulation models from the FMI 1.0 and 2.0 standard.

When using the library, the user can choose the FMU co-simulation block or FMU model exchange block in the FMU import inside the Simulink library browser. The block has the option to associate the FMU files to the subsystem. As a causal model, the subsystem has inputs, outputs and parameters associated to the model as defined by the FMU files.

5.5.2 Causal and Acausal modeling

When modeling a system it is possible to create causal model or acausal models, as described in section 2.2. For the models used in the PSP for the FMI integration, it is expected the causal relationship.

In the case that the model was made with a acausal relation, using physical modeling tools, like Modelica or the Simscape library for Simulink, it is possible to create a wrapper around the model to create the causal relationship [29]. When the wrapper is created, the causality is lost, and can lead to simulation errors if not made in a proper way.

5.6 Transient Load Profile integration

To create or to manipulate signals which should be used used in the simulation environment, it was created the transient load profile creator (TLPC)tool. The TLPC is a Matlab script that get signals from *.mat* files and manipulates then, merging signals, changing the shape, the sample rate, and length.

The signals can be used to simulate load transients in the simulation environment. The figure 5.15 illustrates the main GUI of the tool, where is shown an example of a starter current consumption during a cold crank event. In the figure it is possible to see in the left side the input signals, and in the right side the output signals, that are the merge of the input signals.

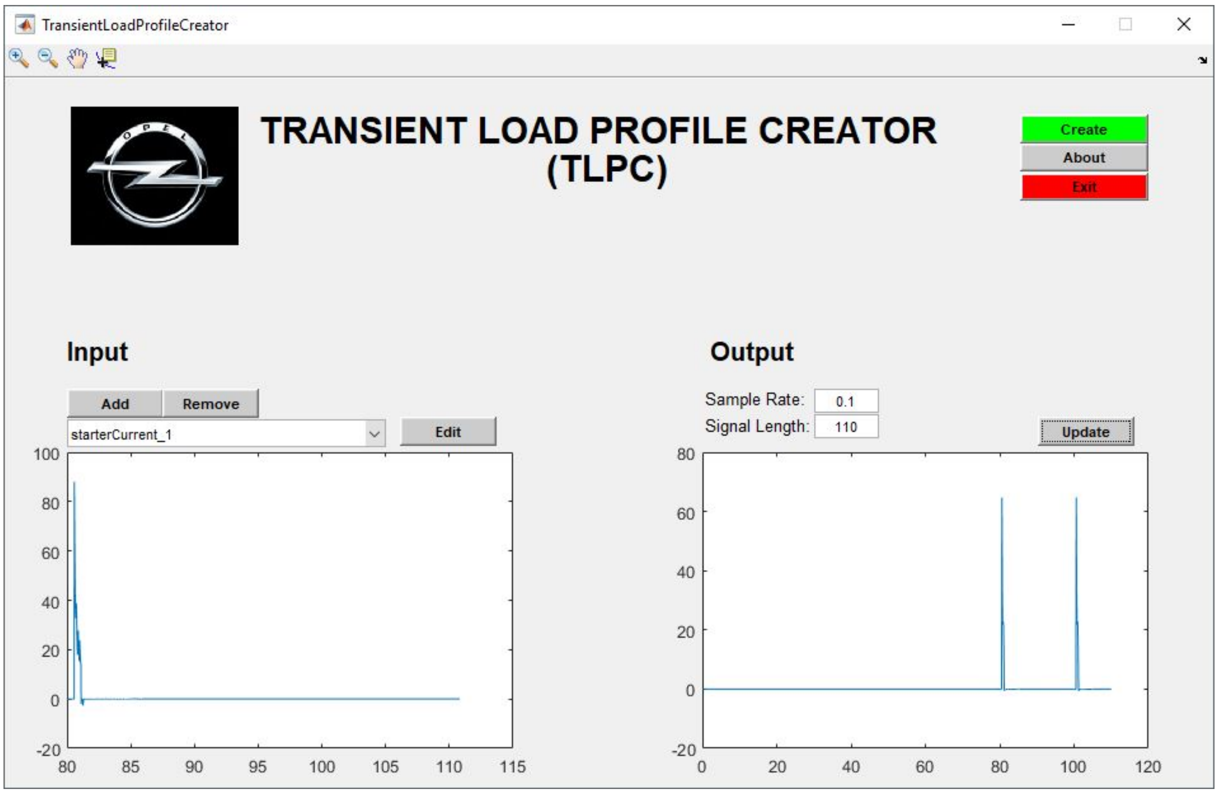
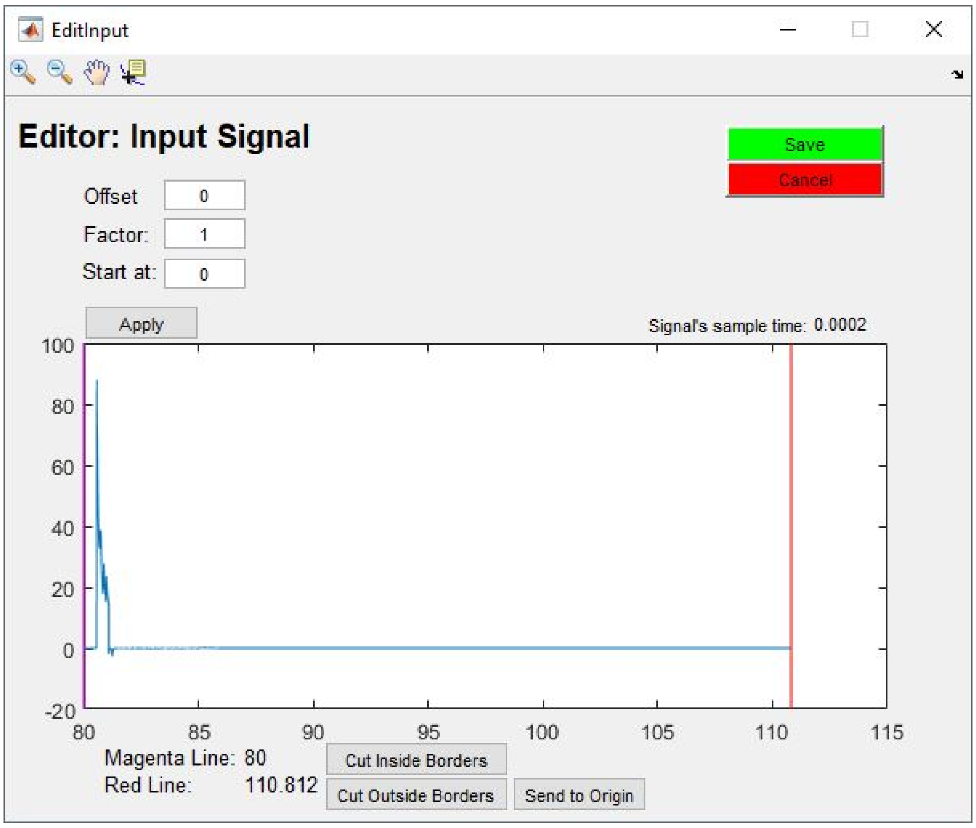


Figure 5.15: *TLPC GUI: Main screen*

Using the tool, all the signals in the input are merged and the sample rate and signal length is changed according the values in the text field. When the user clicks the green button *create* a *.mat* file is created with the signal values. With the tool it is possible to manipulate each of the input signals, by choosing the signal name in the drop down menu in the input and clicking the *Edit* button. The figure 5.16 shows the tool capabilities to manipulate the signal when the *Edit* button is pressed. As it can be seen in the figure, the user can change signals offset, rescale the signal multiplying by a factor and change the signal position in the time by changing the start value. The two vertical lines are used to crop the signal. The user just have to click in the bars and drag then to the desired position.

5.7 Final Comments

This chapter presented the implementations of the simulation environment. It was demonstrated how it was created the architecture and the integration with the CarMaker software and the OSIMOT tool. Additionally it was showed how was done the FMI,



Offset: 0
Factor: 1
Start at: 0

Signal's sample time: 0.0002

Magenta Line: 80
Red Line: 110.812

Figure 5.16: *TLPC GUI: Input signal editor*

model database and the TLPC tool integration. The next chapter will show the results of the current work and some simulation capabilities examples.

Chapter 6

Results

This chapter evaluates the simulation environment developed and analyses various simulation test results. In the Section 6.1 will be shown the simulation environment usage, demonstrating how to apply the OSIMOT tool to manipulate models, the CarMaker software to select the use case scenarios and the TLPC tool to integrate signals. In Section 6.2 will be demonstrated the simulation results from a driving scenario example. The Section 6.3 will show some results using FMU models. The Section 6.4 will demonstrate some comparisons between acausal and causal models. Furthermore, the Section 6.5 will give the final comments about the chapter.

6.1 Simulation environment Usage

This section will demonstrate how to change one model in the simulation environment using the OSIMOT tool, how to select the simulation scenario using the CarMaker software, and how to use the signals created by the TLPC tool in the simulation environment.

6.1.1 Models manipulation using OSIMOT

To show the application of the OSIMOT tool for model exchange, a model for the starter system harness is used as example.

The figure 6.1 shows the model wrapper as explained in section 4.3. The wrapper has one input for the signal bus and two connections for the physical ports used by the acausal models created in Simscape. As the output is not used, the port is terminated.

In the figure 6.2 it is illustrated the model core, with the engine compartment temperature as input coming from the signal bus and two physical ports. Using the OSIMOT tool, if the user opens the OSIMOT GUI and select the starter harness model, he can click the 3 points button to select the new model he wants to use, as show in the figure 6.4. Clicking apply the user selects the new model. In the library system text fields, the new model name will appear, and he can click the *Get from Lib* button to replace the model in the simulation tool. If the model contains the same number of inputs and

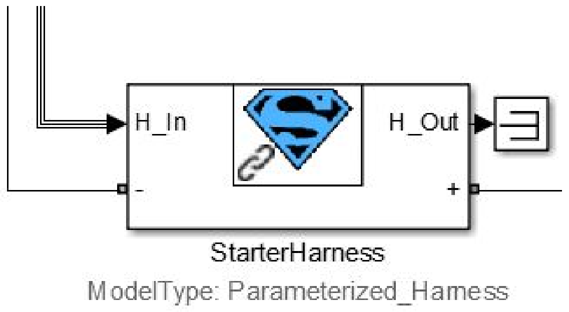


Figure 6.1: *Starter harness model: Wrapper*

outputs, the connections are done automatically. If no, the user has to connect the ports manually.

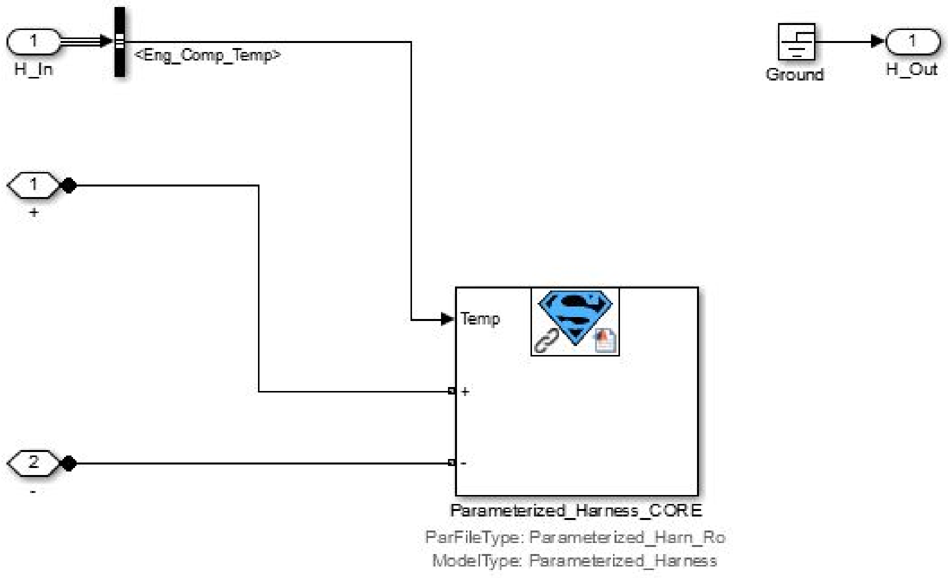
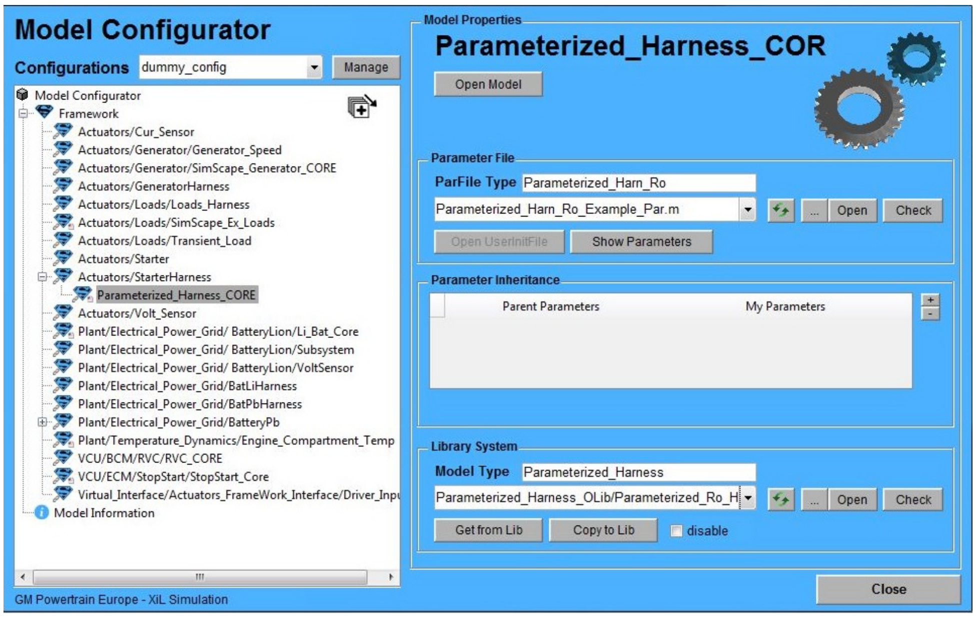

1
H_In
<Eng_Comp_Temp>
Ground
1
H_Out
1
+
Temp
+
-
2
-
Parameterized_Harness_CORE
ParFileType: Parameterized_Harn_Ro
ModelType: Parameterized_Harness

Figure 6.2: *Starter harness model: Core*

6.1.2 Drive scenarios using Carmaker

Using the CarMaker software it is possible to select drive scenarios to be used in the simulation environment. The drive scenarios are CarMaker projects with the data from the vehicle, road and maneuvers. In section 5.2 was illustrated the CarMaker GUI. In the GUI, the user can click *file* and *open* to selected a CarMaker project as illustrated in figure 6.5. If the user wants to change something in the scenario, he can click in the *parameters* button. The figure 6.6 shows the maneuver option getting the values of car speed, and gear from the *StopStart_110s_Drivecycle.dat* file.



Configurations: dummy_config

- Model Configurator
  - Framework
    - Actuators/Cur_Sensor
    - Actuators/Generator/Generator_Speed
    - Actuators/Generator/SimScape_Generator_CORE
    - Actuators/GeneratorHarness
    - Actuators/Loads/Loads_Harness
    - Actuators/Loads/SimScape_Ex_Loads
    - Actuators/Loads/Transient_Load
    - Actuators/Starter
    - Actuators/StarterHarness
      - Parameterized_Harness_CORE
    - Actuators/Volt_Sensor
    - Plant/Electrical_Power_Grid/ BatteryLion/Li_Bat_Core
    - Plant/Electrical_Power_Grid/ BatteryLion/Subsystem
    - Plant/Electrical_Power_Grid/ BatteryLion/VoltSensor
    - Plant/Electrical_Power_Grid/BatLiHarness
    - Plant/Electrical_Power_Grid/BatPbHarness
    - Plant/Electrical_Power_Grid/BatteryPb
    - Plant/Temperature_Dynamics/Engine_Compartment_Temp
    - VCU/BCM/RVC/RVC_CORE
    - VCU/ECM/StopStart/StopStart_Core
    - Virtual_Interface/Actuators_FrameWork_Interface/Driver_Inpu
  - Model Information

Parameterized_Harness_COR

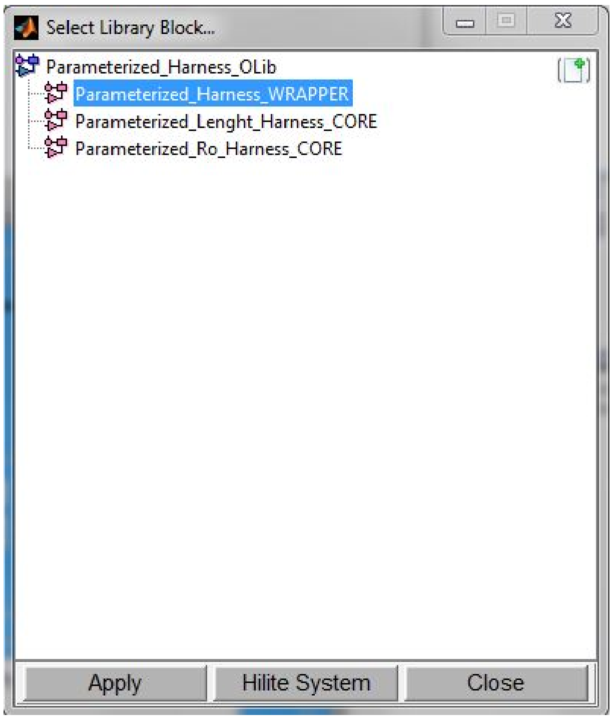
ParFile Type: Parameterized_Harn_Ro

Parameterized_Harn_Ro_Example_Par.m

Model Type: Parameterized_Harness

Parameterized_Harness_OLib/Parameterized_Ro_H

Figure 6.3: *OSIMOT GUI: Principal screen*

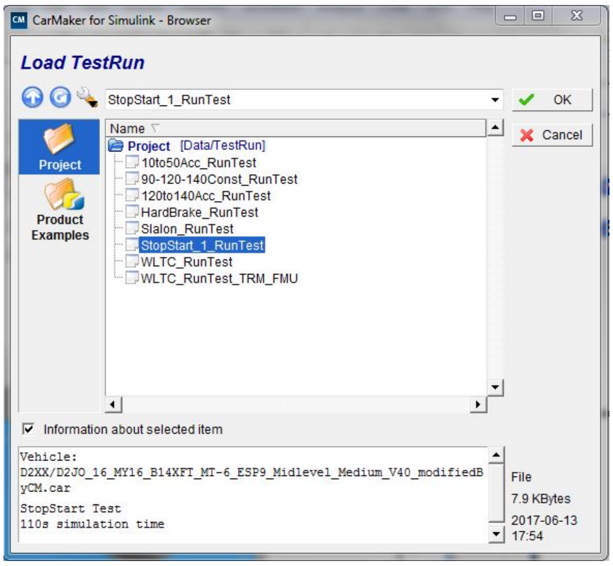


- Parameterized_Harness_OLib
  - Parameterized_Harness_WRAPPER
  - Parameterized_Lenght_Harness_CORE
  - Parameterized_Ro_Harness_CORE

Figure 6.4: *OSIMOT GUI: Library selection*

6.1.3 Transient signals using TLPC

As described in section 2.1.1, some components in the electrical power grid work as consumers and generators, as example the EPS system or the iBoost system. The figure 6.7 illustrates some signals from the EPS system during a drive cycle. It is possible to see in the red color the current consumption of the EPS system. The current



StopStart_1_RunTest

Name

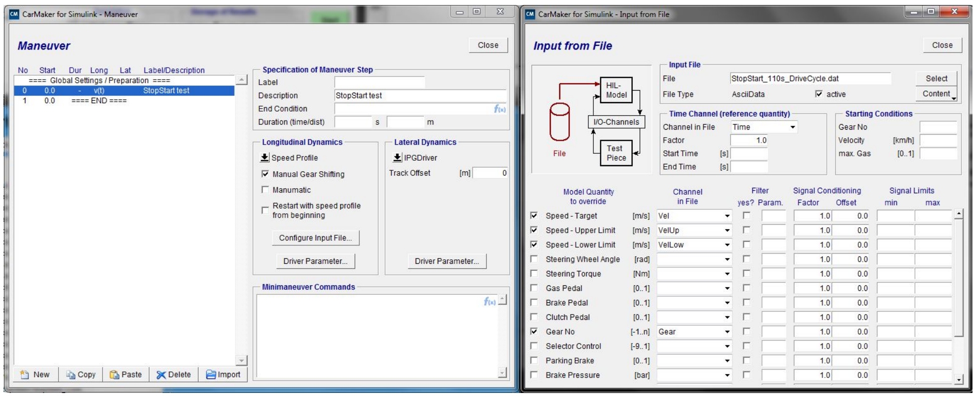
- Project [Data/TestRun]
  - 10to50Acc_RunTest
  - 90-120-140Const_RunTest
  - 120to140Acc_RunTest
  - HardBrake_RunTest
  - Slalon_RunTest
  - StopStart_1_RunTest
  - WLTC_RunTest
  - WLTC_RunTest_TRM_FMU

☑ Information about selected item

```
Vehicle:
D2XX/D2JO_16_MY16_B14XFT_MT-6_ESP9_Midlevel_Medium_V40_modifiedB
yCM.car
StopStart Test
110s simulation time
```

File
7.9 KBytes
2017-06-13
17:54

Figure 6.5: *CarMaker GUI: Project selection*



No	Start	Dur	Long	Lat	Label/Description
==== Global Settings / Preparation ====					
0	0.0	-	v(t)		StopStart test
1	0.0	==== END ====			

Specification of Maneuver Step
Label
Description: StopStart test
End Condition
Duration (time/dist) s m

Longitudinal Dynamics
Speed Profile
☑ Manual Gear Shifting
☐ Manumatic
☐ Restart with speed profile from beginning
Configure Input File...
Driver Parameter...

Lateral Dynamics
IPGDriver
Track Offset [m] 0
Driver Parameter...

Minimaneuver Commands

Input File
File: StopStart_110s_DriveCycle.dat
File Type: AsciiData ☑ active

Time Channel (reference quantity)
Channel in File: Time
Factor: 1.0
Start Time [s]
End Time [s]

Starting Conditions
Gear No
Velocity [km/h]
max. Gas [0..1]

	Model Quantity to override		Channel in File	Filter yes?	Filter Param.	Signal Conditioning Factor	Signal Conditioning Offset	Signal Limits min	Signal Limits max
☑	Speed - Target	[m/s]	Vel	☐		1.0	0.0		
☑	Speed - Upper Limit	[m/s]	VelUp	☐		1.0	0.0		
☑	Speed - Lower Limit	[m/s]	VelLow	☐		1.0	0.0		
☐	Steering Wheel Angle	[rad]		☐		1.0	0.0		
☐	Steering Torque	[Nm]		☐		1.0	0.0		
☐	Gas Pedal	[0..1]		☐		1.0	0.0		
☐	Brake Pedal	[0..1]		☐		1.0	0.0		
☐	Clutch Pedal	[0..1]		☐		1.0	0.0		
☑	Gear No	[-1..n]	Gear	☐		1.0	0.0		
☐	Selector Control	[-9..1]		☐		1.0	0.0		
☐	Parking Brake	[0..1]		☐		1.0	0.0		
☐	Brake Pressure	[bar]		☐		1.0	0.0		

Figure 6.6: *CarMaker GUI: Maneuver*

changes from positive to negative during certain times, showing current consumption for the positive values and current generation for the negative values.

To aid the engineers to analyze this transient behaviors in the electrical power grid, it is possible to use a controlled current source as illustrated in figure 6.8 to feed the current profile in the system.

Using the TLPC tool it is possible to create the signal to be put in the controlled current source as illustrated in figure 6.9. The signal used in the transient profile was originally real data from the car. Using the TLPC the sample time and the position of the signal in the time were changed.

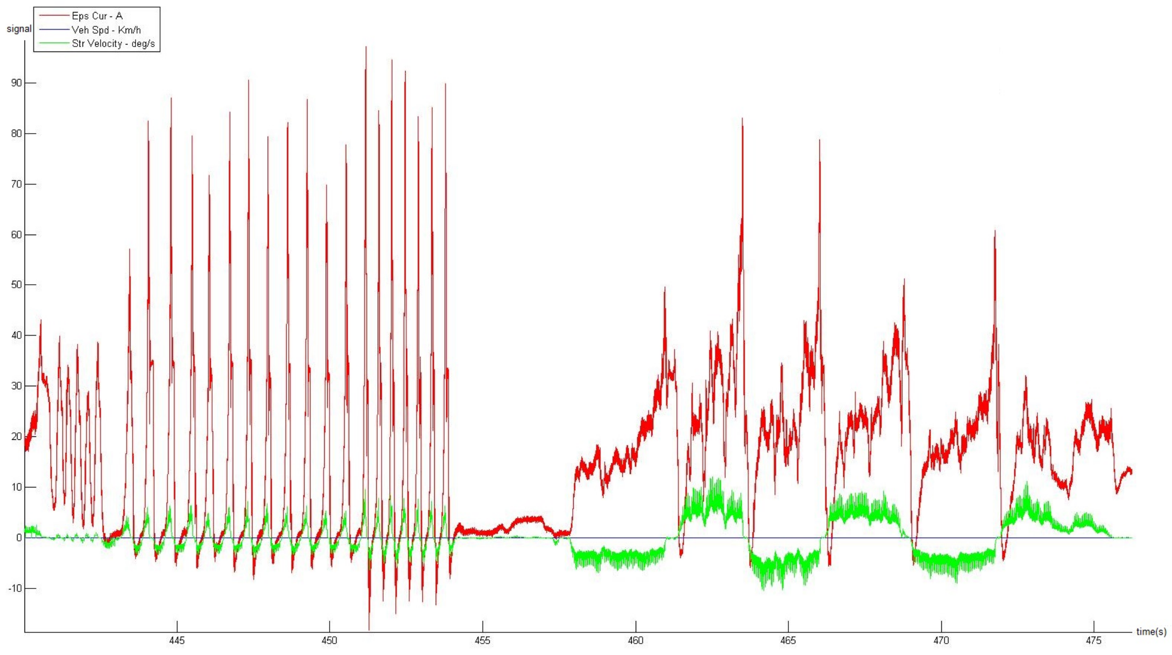


Figure 6.7: *EPS signals*

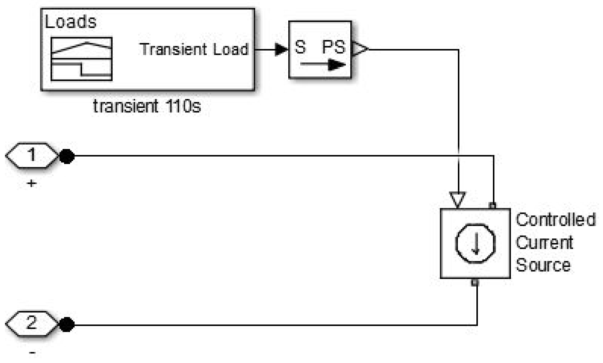


Figure 6.8: *Simulink model for current profile*

6.2 Simulation environment results

This section will show some components signals after a simulation. The simulation scenario was a straight line drive profile.

In figure 6.10 it is possible to see vehicle speed and engine rotation signals from the simulation. In the figure it is possible to see as well the fuel consumption over time.

For the simulation it was used inside the VCU system one model for the StopStart system. The StopStart system, under certain conditions, turn off the engine to reduce the pollutants emissions. In the figure 6.11 it is possible to see the vehicle speed and the StopStart signal. The StopStart signal goes from 0 to 1 when the StopStart conditions are triggered. This bit is an input for the generator model, to simulate the

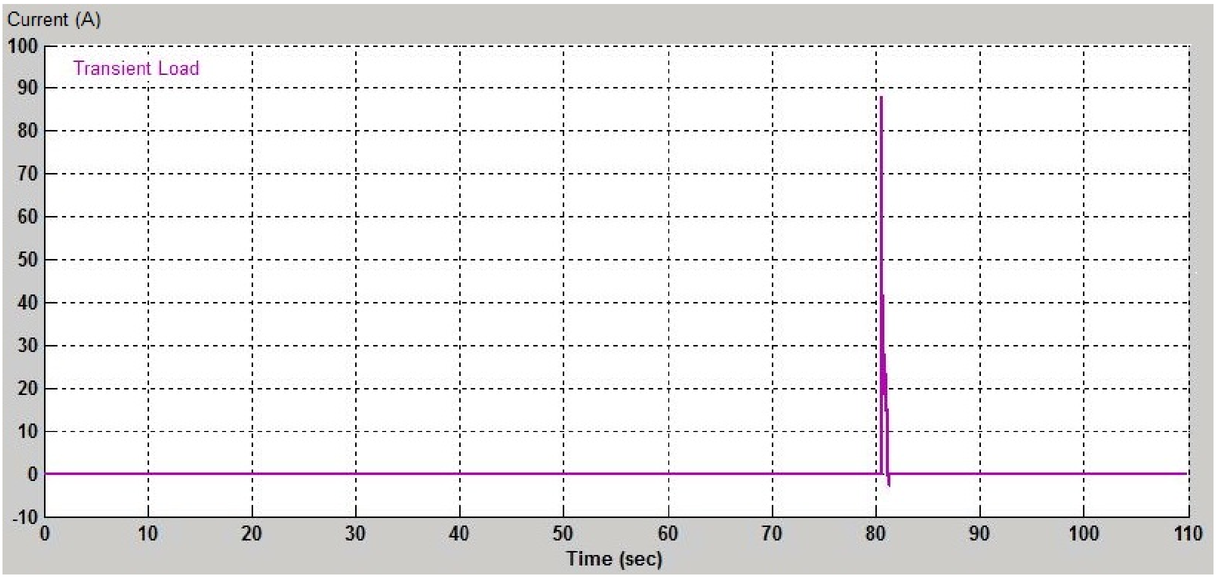


Figure 6.9: *Transient current signal*

StopStart effect, due the fact that when the engine is off, the generator can not supply energy to the electrical power grid.

In the figure 6.12 it is possible to see the generator signals in the simulation. It is possible to see that the current generated by the generator goes to 0 when the car is in a StopStart condition. The CarMaker software is running in co-simulation with the Simulink, so it is possible to get CarMaker signals and manipulate then to feed the Simulink models. In this case, the engine RPM input of the generator model is manipulate to be equals to the CarMaker values during no StopStart situations, and equals to 0 when occurs StopStart situations. Furthermore, it is possible to see the effects of the transient load in the generator torque and current during the time 80.

The figure 6.13 illustrates the loads current consumption. The current value is changing over time according the driver loads usage. The loads used over the time are configured in the virtual interface. In the time 80 it is possible to see a high peak of current to simulate a transient current profile.

The figure 6.14 shows the Lion battery signals over the time. It is possible to see that during the StopStart event the amount of current provided by the battery is bigger due the fact that the generator can not supply energy. It is possible to see a drop in the battery voltage during the high peak of current at the time 80. This simulates the battery voltage drop when delivering high amounts of current.

6.3 Simulation with FMI

This section will compare some FMU models from the original one. Was used the Modelon toolbox to convert a Matlab *S-Funciton* to a FMU.

The figure 6.15 shows the Simulink model to run the comparison. It is possible to see in green the original *S-Function* and in the other blocks the FMU models from the PSP to simulate the model-exchange and co-simulation FMUs for FMI 1.0 and 2.0.

The figure 6.16 shows the signals output from the models to compare. It is possible to see that there are no difference between the 1.0 and 2.0. The model-exchange models have a behavior similar to the original one. This small difference in the transient is due the model conversion. The co-simulation models have a bigger difference due the fact that the solver sample time is bigger than the original. It is possible to see the jumps in the values due the sample time. Despite the differences in the sample time, the values converge to the same ones from the original model.

6.4 Acausal and Causal models results

In section 5.5.2 it was demonstrated how to convert acausal models to causal ones. As described in the section, it is necessary to be careful when creating the wrapper to change the causality. When the wrapper is created, it is possible to lost some causality relationship and the model will provide wrong values.

The figure 6.17 shows a Simulink model to analyze the differences. The model uses a first order filter to compare an acausal model created in Simscape and a causal model created in Simulink. The input for the causal model is the voltage in the voltage power supply and the output is the voltage at the capacitor. At the bottom of the figure it is possible to see two first order filters in series. The filters have different time constants. In the first one, was used only acausal models. In the second one, one of the filters was changed to a causal one using the wrapper to make the causality conversion.

As it is possible to see in the figure 6.18, when the wrapper is created, the causality is lost and the simulation results are different. The blue line is the correct one, coming from the acausal models in series. The red one is the wrong, coming from the wrapper. This occurs because the causal model is only looking for voltage and the relationship between voltage and current in the filters are lost. To avoid this error it is necessary to create a better causal model that takes in consideration the relationship between voltage and current between the two filters. With this the causality relationship is not lost and the simulation will lead to correct values.

6.5 Final Comments

This chapter presented the results of the current work. It was demonstrated how to use the tools to create the simulation scenarios and some results from a simulation example. Furthermore it was showed some FMU models comparisons and model's causality transformation using wrappers. The next section will give the final considerations about the current work.

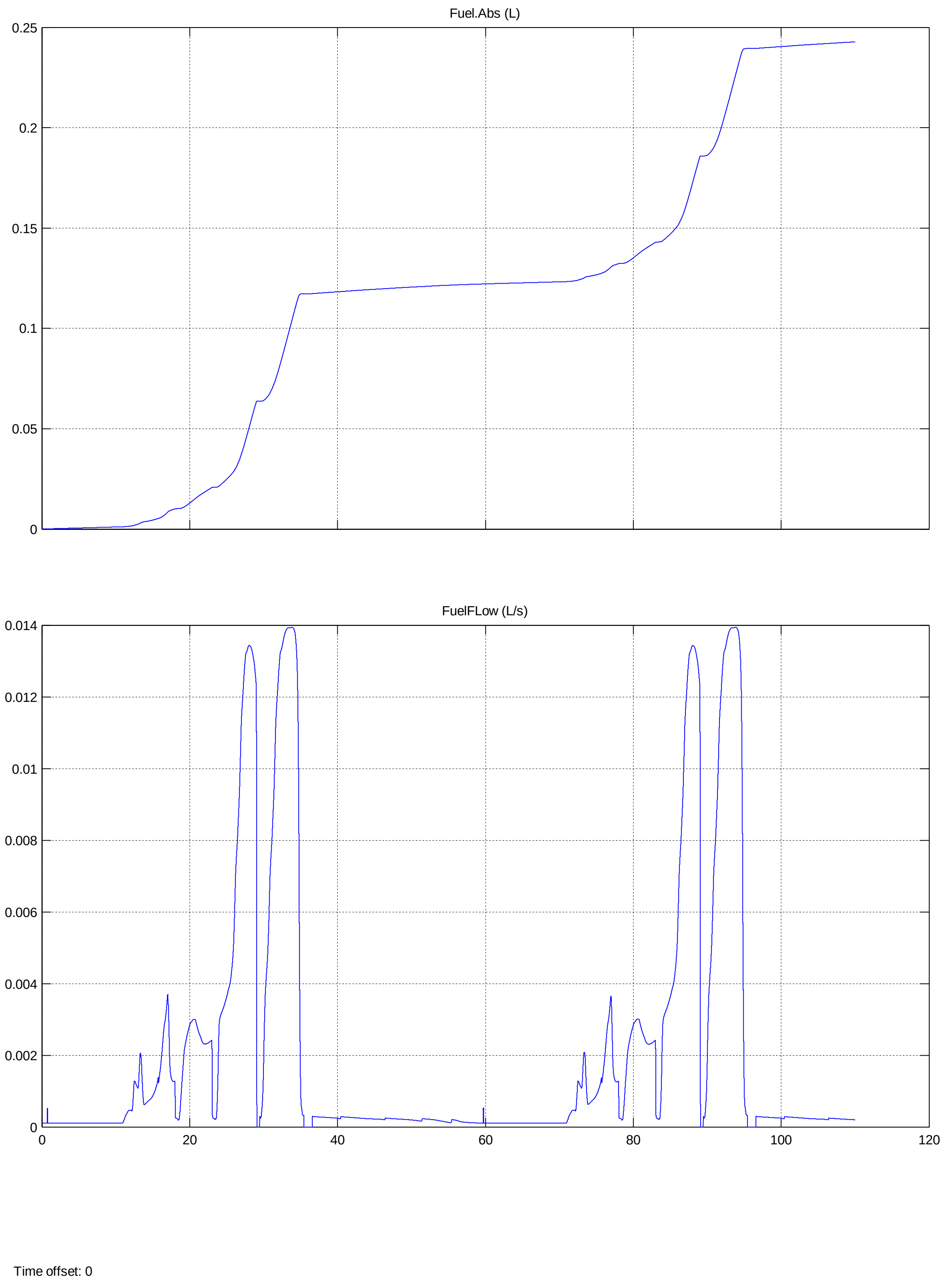


Figure 6.10: *Simulation results: Vehicle speed, engine rotation and fuel consumption*

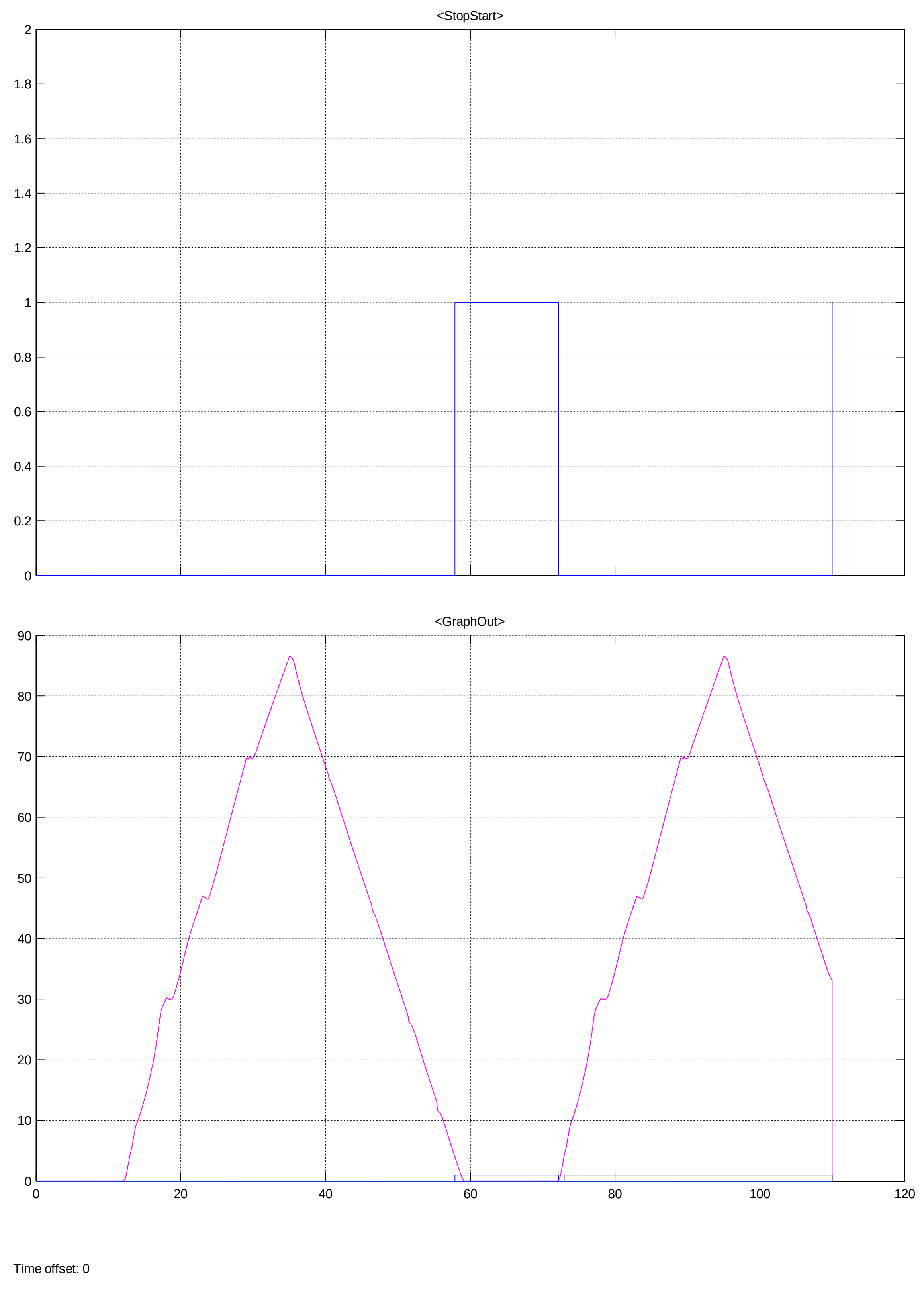


Figure 6.11: *Simulation results: StopStart bit and vehicle speed*

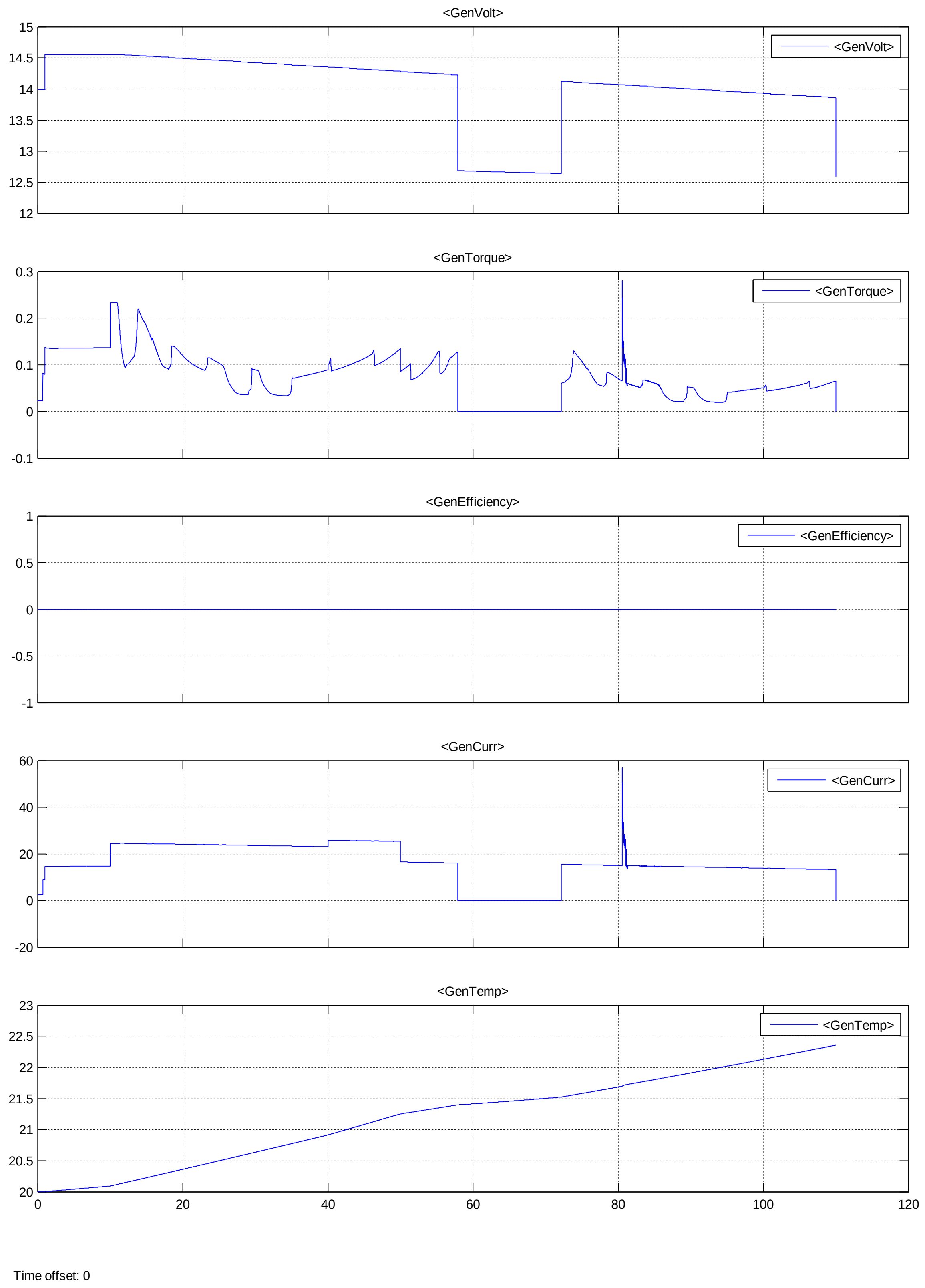


Figure 6.12: *Simulation results: Generator*

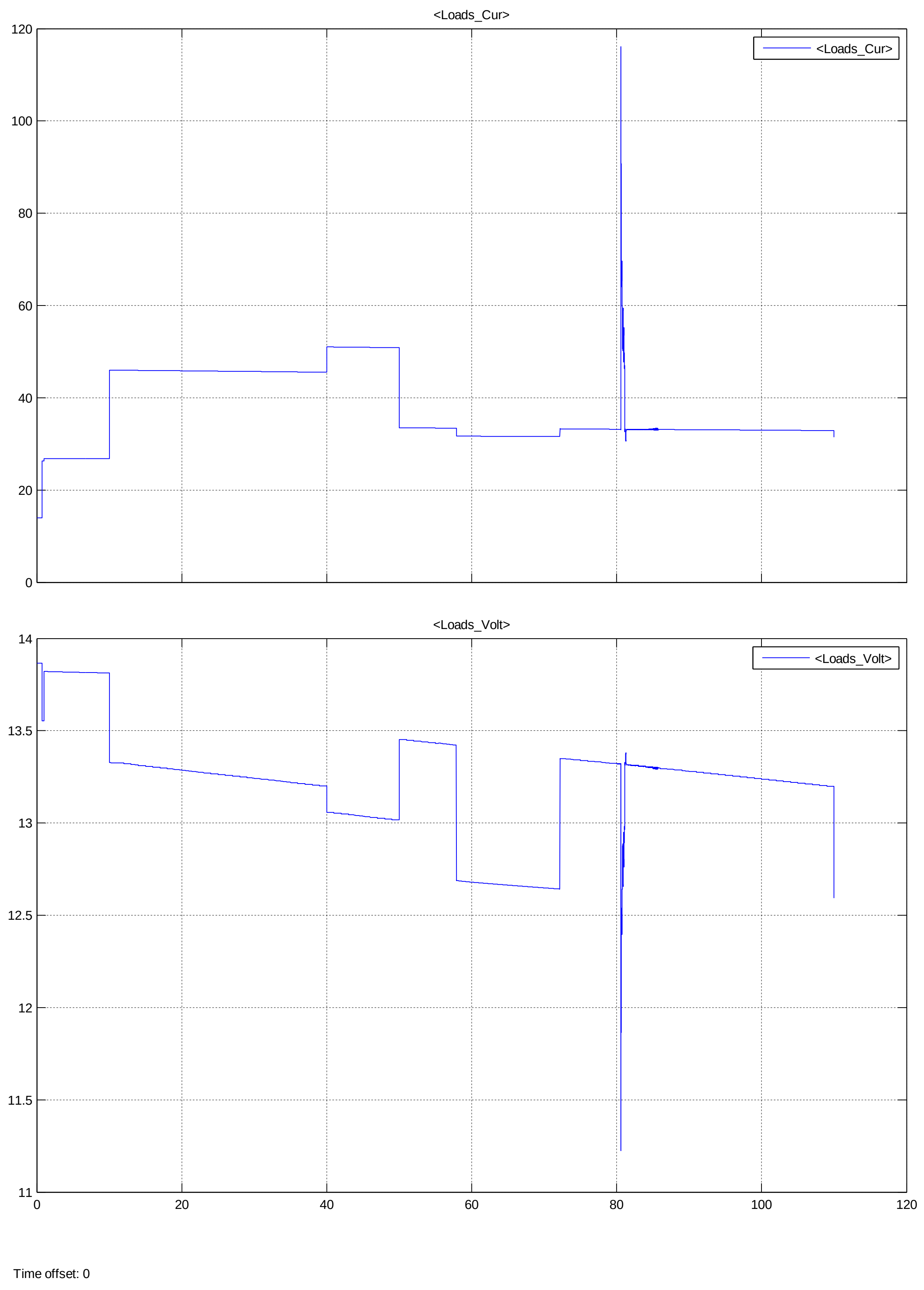


Figure 6.13: *Simulation results: Loads*

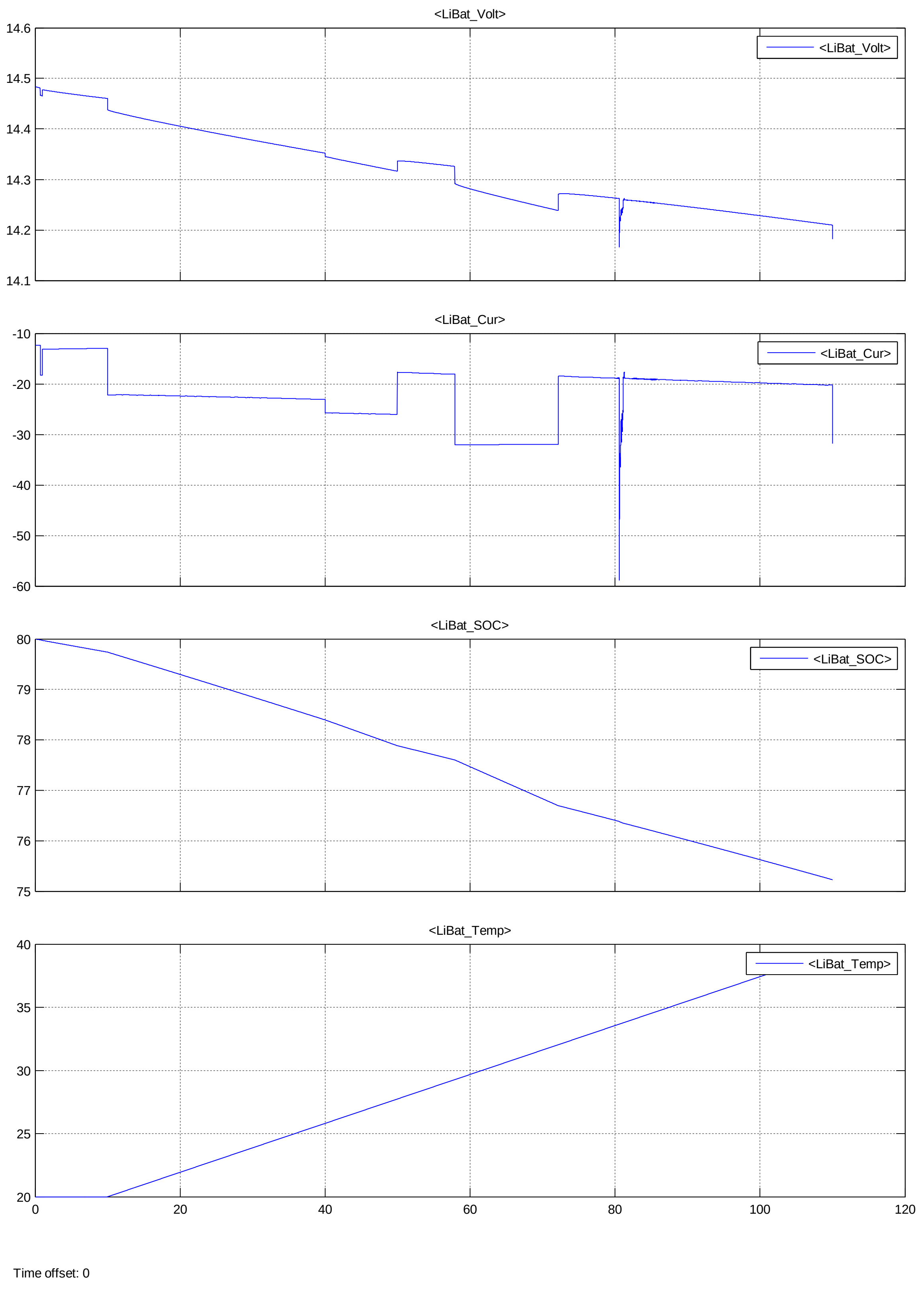


Figure 6.14: *Simulation results: LiBat*

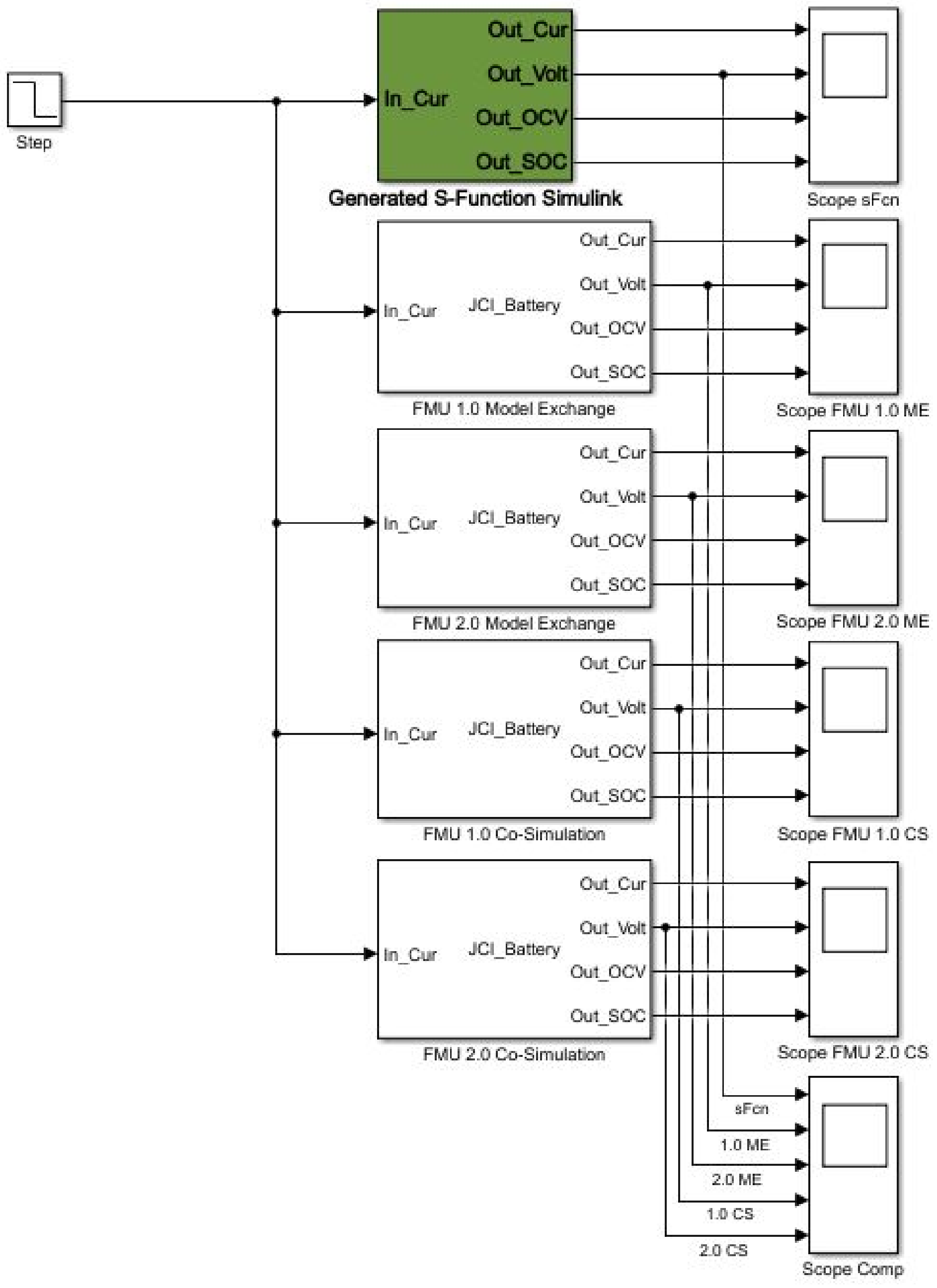


Figure 6.15: *Simulink model for FMU comparison*

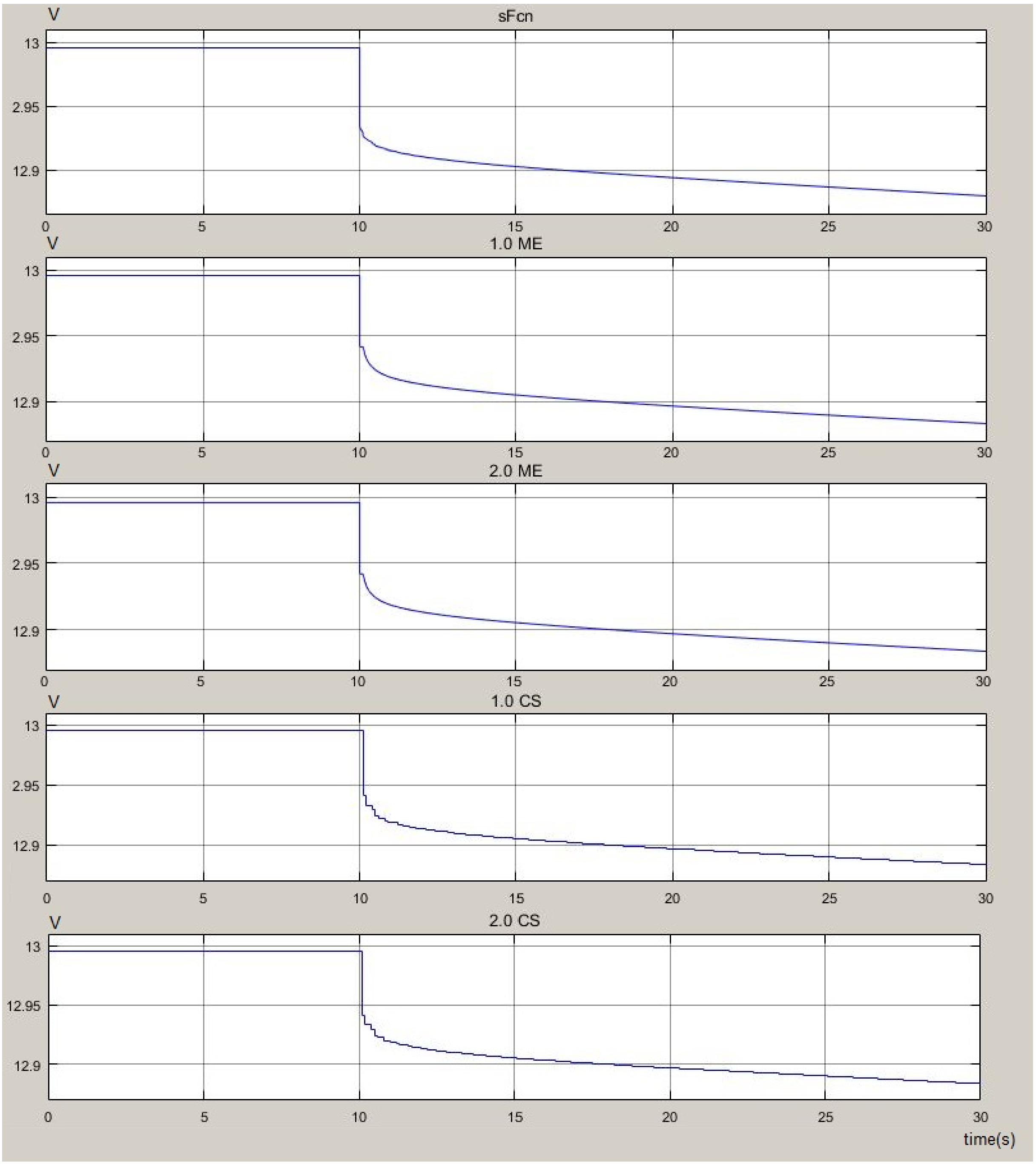


Figure 6.16: *FMU comparison results*

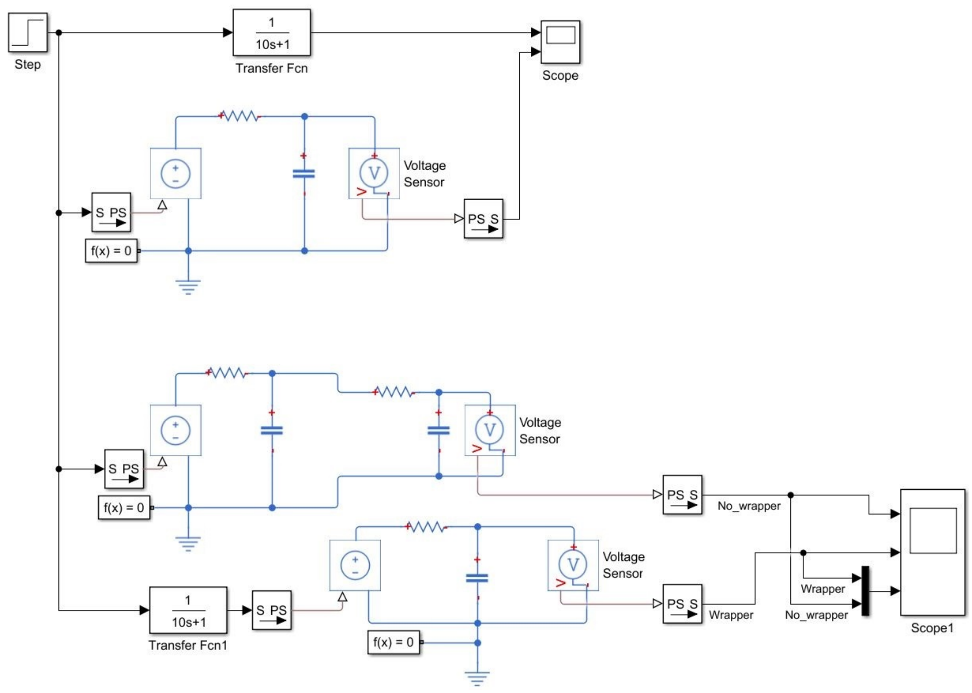


Figure 6.17: *Simulink model for causality comparison*

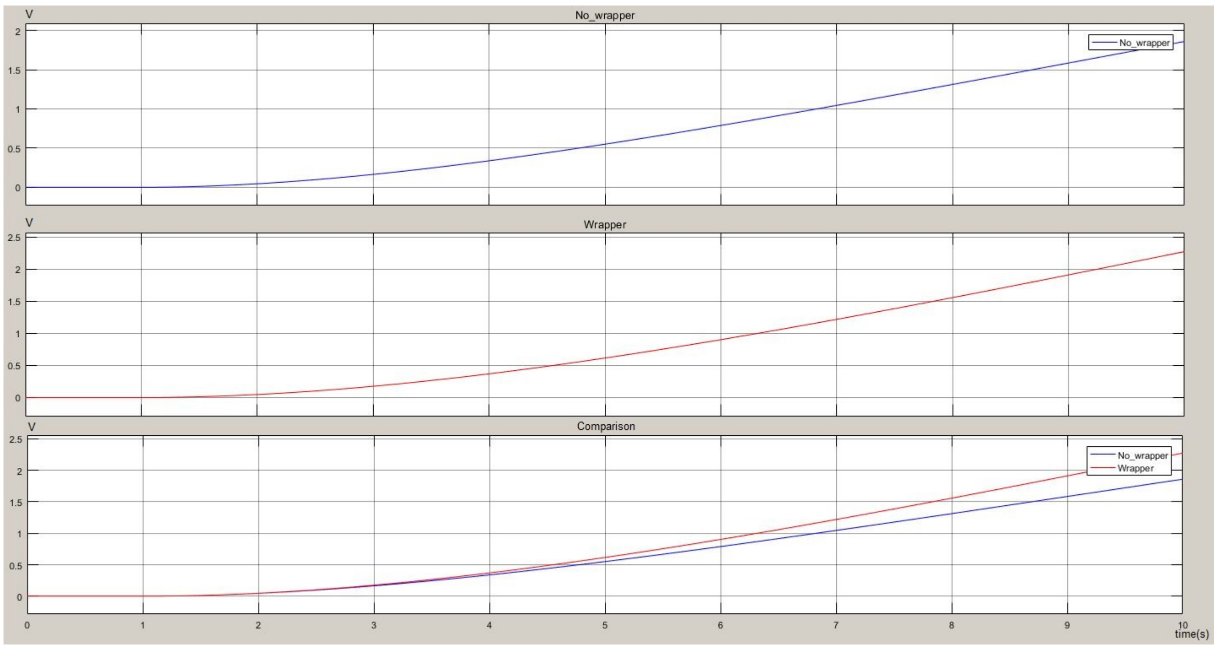


Figure 6.18: *Causality comparison results*

Chapter 7

Concluding Remarks

This chapter will deal with the final considerations in the present work. Section 7.1 will summarize the project conclusions and Section 7.2 will give the future perspectives.

7.1 Conclusions

In this work was developed a simulation environment for the electrical power grid for commercial vehicles following the MBE concepts. The creation of such simulation environment is important to the company because the environment can aid the engineers to analyze components in early stages of the DVP and to improve vehicle systems.

The simulation environment follows the MBE concepts where the user connects the models in a common architecture. To create the vehicle architecture was necessary to understand the major systems in the vehicle and how they exchange informations. The simulation environment was created in Matlab/Simulink environment and has integration with the IPG's CarMaker, FMI standard and OSIMOT tool.

With the IPG's CarMaker integration it is possible to create complex drive test scenarios to analyze the vehicle systems. The FMI integration allows the users to connect FMU's in the simulation environment. The OSIMOT tool integration allows the users to connect models from a model's database to the simulation environment in a "plug-n-play" manner.

The simulation environment has capabilities to run offline simulations as MIL or SIL, where it is possible to create the simulation scenarios proposed by the team and to analyze the systems response according the models used. With the virtual interface system it is possible to the user to see the simulation results and to store the results. The table 7.1 shows the summary of the requirements fulfilled by the present work.

The results of the present work were as expected. With the present work, the engineers have more capacity to run simulations, being able to do a better evaluation of the electrical power grid in the vehicle under diverse drive conditions.

Table 7.1: *Requirements fulfilled*

Requirement	Accomplished
The simulation environment shall be capable to run offline simulations	yes
The simulation environment shall be capable to run in Matlab/Simulink environment	yes
The simulation shall be capable to run the simulation scenarios	yes
The simulation environment shall be capable to use Instrumented Models	yes
The user shall be able to set the simulation environment by the GUI	yes
The simulation environment shall be capable to show the simulation results to the user	yes
The simulation environment shall be capable to save the results in data	yes

7.2 Future Perspectives

In this work, the model's databased created have just a few number of components in order to show the simulation environment capabilities. A bigger and more complex model's database is necessary to let the engineers to run more complex simulation scenarios.

In the present work, the models created for the library system were placed in the local machine. For further work, a distributed revision control system as *GIT* is desired to be used to manage the model's database.

The present work is an offline simulation environment capable to run SIL and MIL simulations. For future work, online simulations like hardware in the loop are desired. With the HIL capabilities new simulation scenarios can be added to the simulation environment helping the engineers to run simulations with real components and the virtual environment.

Bibliography

[1] Opel. Key facts. Available at: http://www.opel.com/company/facts.html. Access date: 19/06/2017.

[2] Bosch Automotive Electrics and Automotive Electronics, Systems and Components, Networking and Hybrid Drive, 5th Edition, Springer Vieweg, 2007

[3] K. Yash (2017). "Development of a Modular Power Grid Test Bench Concept and Partial Physical Builds" (Masterthesis)

[4] T. Lu; J. Du; X. Huang. “Research on the Life Cycle Analysis of the Reverse Supply Chain of the Lead Acid Batteries”. In: 2010 4th International Conference on Bioinformatics and Biomedical Engineering. June 2010, pp. 1–6.

[5] H. B.-Meiwes; J. Drillkens; B. Lunz; J. Muennix; S. Rothgang; J. Kowal; D. U. Sauer. “A review of current automotive battery technology and future prospects”. In: Proceedings of the Institution of Mechanical Engineers Part D Journal of Automobile Engineering 227.5 (May 2013), pp. 761–776.

[6] Lithium Batteries - Not Just for Hybrids. Available at: http://www.roadandtrack.com/new-cars/videos/a18350/lithium-batteries-not-just-for-hybrids/. Access date: 19/06/2017.

[7] Pre-engaged starter: https://www.howacarworks.com/illustrations/pre-engaged-starter. Access date: 19/06/2017.

[8] Intelligent control boosts braking power: http://www.bosch.co.jp/en/press/group-1306-13.asp. Access date: 19/06/2017.

[9] Driving-safety-systems/iBoost: http://www.bosch-mobility-solutions.com/en/products-and-services/passenger-cars-and-light-commercial-vehicles/driving-safety-systems/brake-booster/ibooster/. Access date: 19/06/2017.

[10] H. Zhang, Y. Zhang, J. Liu, J. Ren, Y. Gao; "Modeling and Characteristic Curves of Electric Power Steering System".

[11] Vehicle Dynamics. Short Course, Prof.Dr. Georg Rill, August 2007.

[12] Bosch Professional Automotive Information, Fundamentals of automotive and engine technology, Konrad Reif Ed, Springer Vieweg, 2014

[13] P. Shakouri, A. Ordys, M. Askari, Ds S. Laila,"Longitudinal vehicle dynamics using Simulink/Matlab"

[14] Types of Transmissions and How They Work: https://www.transmissionrepaircostguide.com/types-of-transmissions/. Access date: 19/06/2017.

[15] Schramm D., Hiller M., Bardini R., "Vehicle Dynamics Modeling and Simulation" 2014, XIX, 405p.327, Hardcover;http://www.springer.com/978-3-540-36044-5

[16] National Instruments White Paper on Electronic Control Units: http://www.ni.com/white-paper/3064/de/. Access date: 19/06/2017.

[17] Modelica Association Project. Functional Mock-up Interface for Model Exchange and Co-Simulation. Ver. 2.0, July 2014.

[18] R. Merzouki, A. K. Samantaray, P. M. Pathak, B. O. Bouamama,"Intelligent Mechatronic Systems - Modeling, Control and Diagnosis", Springer-Verlag London 2013

[19] Basic principles of modeling physical networks. htmlhttps://de.mathworks.com/help/physmod/simscape/ug/basic-principles-of-modeling-physical-networks.html. Access date: 19/06/2017.

[20] Prof. Dr-Ing U.Konigorski, "Modelbildung und Simulation", Sommersemester 2014, TU Darmstadt.

[21] B. Zupancic, R. Karba, A. Kunc, J. Music, Continuous System Modelling Education-Causal or Acausal approach?.

[22] S. Furic, Journées INSA Lyon.R&D Engineer "Hybrid acausal modeling using Modelica. Presentation of Modelica"

[23] Lincoln H. McGhee, "An Overview of the Vehicle Development Process A Key Product Development Enable.r An Overview of the Vehicle Development Process. A Key Product Development Enabler." March 30, 2011

[24] What is Model-Based Engineering (MBE)?: http://modelbasedengineering.com/faq/. Access date: 19/06/2017.

[25] N. Vansina, "Architecture Driven Design with LMS System Synthesis", Seminar model management, 16/03/2017, Stuttgart.

[26] V-Model: What Is It And How Do You Use It?: https://airbrake.io/blog/sdlc/v-model. Access date: 19/06/2017.

[27] ENTWICKLUNGSMETHODIK MECHATRONISCHER SYSTEME: http://www.rst.e-technik.tu-dortmund.de. Access date: 19/06/2017.

[28] CarMaker Programmer's Guide. Version 5.1.3. 1999 - 2016 IPG Automotive GmbH – www.ipg-automotive.com

[29] M. Winter, J. Taube, J. Froeschly, H. Herzog, "From Simulation to Testbench using the FMI-standard"