A Design Framework for Developing a Reconfigurable Driving Simulator

zur Erlangung des akademischen Grades eines
DOKTORS DER INGENIEURWISSENSCHAFTEN (Dr.-Ing.)
der Fakultät Maschinenbau
der Universität Paderborn

genehmigte
DISSERTATION

von
M.Eng. Bassem Hassan
aus *El Dakahlia, Ägypten*

Tag des Kolloquiums:	13. Juni 2014
Referent:	Prof. Dr.-Ing. Jürgen Gausemeier
Korreferent:	Prof. Dr. -Ing. habil. Ansgar Trächtler

List of Published Partial Results

[HBA+13] HASSAN, B.; BERSSENBRÜGGE, J.; ALQAISI, I.; STÖCKLEIN, J.: Reconfigurable Driving Simulator for Testing and Training of Advanced Driver Assistance Systems. In: Proceedings of 2013 IEEE International Symposium on Assembly and Task Planning (ISAM), July 30 - August 2 2013, Xi'an, China, 2013, pp. 337-339 – ISBN 978-1-4799-1656-6

[HG13] HASSAN, B.; GAUSEMEIER, J.: Concept for a Task–Specific Reconfigurable Driving Simulator. In: Proceedings of SIMUL 2013, the Fifth International Conference on Advances in System Simulation (IARIA), October 27 - November 1 2013, Venice, Italy, 2013, pp. 40-46 – ISBN: 978-1-61208-308-7

[HWK+12] HASSAN B.; WAßMANN, H.; KLAAS, A.; KEßLER, J.H.: Cascaded Heterogeneous Simulations for Analysis of Mechatronic Systems in Large Scale Transportation Scenarios. In: Proceedings of the 2012 Spring Simulation Multi-Conference, March 26 - 29 2012, Orlando, Florida, USA, Spring Simulation Multiconference Books: Emerging M&S Applications in Industry & Academia Symposium (EAIA), 2012, pp. 32-39 – ISBN: 978-1-61839-787-4

[HWK+13] HASSAN, B.; KLAAS, A.; WASSMANN, H.; GRAFE, M.: Kaskadierte Simulationen und Visualisierungen für die Analyse mechatronischer Systeme in umfangreichen Transportszenarien. In: Gausemeier, Jürgen; Grafe, Michael (Hrsg.): 11. Paderborner Workshop "Augmented & Virtual Reality in der Produktentstehung", 18. - 19. April 2013, HNI-Verlagsschriftenreihe, Band 311, Paderborn, Germany, 2013, pp. 159-176 – ISBN 978-3-942647-30-4

[KGG+11a] KREFT, S.; GAUSEMEIER, J.; GRAFE, M.; HASSAN, B.: Automatisierte Trassierung virtueller Straßen auf Basis von Geo-Informationssystemen. In: Gausemeier, Jürgen; Grafe, Michael (Hrsg.): 10. Paderborner Workshop "Augmented & Virtual Reality in der Produktentstehung", 19. - 20. Mai 2011, HNI-Verlagsschriftenreihe, Band 295, Paderborn, Germany, 2011, pp. 253-266 – ISBN 978-3-942647-14-4

[KGG+11b] KREFT, S.; GAUSEMEIER, J.; GRAFE, M.; HASSAN, B.: Automated Generation of Virtual Roadways based on Geographic Information Systems. In: Proceedings of the ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, August 29 - 31 2011, Washington DC, USA, 2011, DETC2011-48141, pp. 1525-1532 – ISBN 978-0-7918-5479-2

[TH13a] TAN, Y.; HASSAN, B.: A Method for Testing Camera Based Advanced Driving Assistance Systems. In: Proceedings of 2013 IEEE International Symposium on Assembly and Task Planning (ISAM), July 30 - August 2 2013, Xi'an, China, 2013, pp. 151-154 – ISBN 978-1-4799-1656-6

[TH13b] TAN, Y.; HASSAN, B.: A Concept of Camera test-bench for testing Camera Based Advanced Driver Assistance Systems. In: Proceedings of IDETC/CIE – ASME 2013 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, August 4 - 7 2013, Portland, USA, 2013, DETC2013-12996 – ISBN: 978-0-7918-5586-7

Abstract

Driving simulators have been used successfully in various application fields for decades. They vary widely in their structure, fidelity, complexity and cost. Nowadays, driving simulators are usually custom-designed for a specific task and they typically have a fixed structure. Nevertheless, using the driving simulator in an application field, such as the development of the Advanced Driver Assistance Systems (ADAS), requires several variants of the driving simulator. Therefore, there is a need to develop a reconfigurable driving simulator which allows its operator to easily create different variants without in-depth expertise in the system structure. In order to solve this challenge, *a Design Framework for Developing a Reconfigurable Driving Simulator* has been developed. The design framework consists of a procedure model and a configuration tool. The procedure model describes the required development phases, the entire tasks of each phase and the used methods in the development. The configuration tool organizes the driving simulator's solution elements and allows its operator to create different variants of the driving simulator by selecting a combination of the solution elements, which are like building blocks. The design framework is validated by developing an ADAS reconfigurable driving simulator and by creating three variants of this driving simulator.

Zusammenfassung

Fahrsimulatoren werden seit Jahrzehnten erfolgreich in verschiedenen Anwendungsbereichen eingesetzt. Sie unterscheiden sich weitgehend in ihrer Struktur, Genauigkeit, Komplexität und in ihren Kosten. Heutzutage werden Fahrsimulatoren in der Regel individuell für eine spezielle Aufgabe entwickelt und haben typischerweise eine festgelegte Struktur. Bei der Nutzung eines Fahrsimulators in einem Anwendungsbereich wie der Entwicklung von fortgeschrittenen Fahrerassistenzsystemen (FFAS) werden jedoch mehrere Varianten des Fahrsimulators benötigt. Es besteht daher Handlungsbedarf für die Entwicklung eines rekonfigurierbaren Fahrsimulators, der es dem Betreiber des Fahrsimulators ermöglicht, ohne umfassende Fachkenntnisse problemlos verschiedene Varianten zu erstellen. Um diese Herausforderung zu bewältigen wurde eine *Entwicklungssystematik für die Entwicklung eines rekonfigurierbaren Fahrsimulators* entwickelt. Die Entwicklungssystematik besteht aus einem Vorgehensmodell und einem Konfigurationswerkzeug. Das Vorgehensmodell beschreibt die benötigten Entwicklungsphasen, die vollständigen Aufgaben jeder Phase und die in der Entwicklung eingesetzten Methoden. Das Konfigurationswerkzeug organisiert die Lösungselemente des Fahrsimulators und ermöglicht dem Betreiber des Fahrsimulators, durch Auswählen einer Kombination von Lösungselementen nach dem Baukastenprinzip verschiedene Varianten des Fahrsimulators zu erstellen. Die Entwicklungssystematik wird durch die Entwicklung eines rekonfigurierbaren FFAS-Fahrsimulators und durch die Erstellung von drei unterschiedlichen Varianten dieses Fahrsimulators validiert.

Acknowledgements

This PhD thesis was carried out during my work as research associate at the Product Engineering group, Heinz Nixdorf Institute, the University of Paderborn in Germany, from April 2009 to April 2014. This work is the result of my research activities in industrial and academic projects in the mechatronic development, virtual prototyping and driving simulators fields.

My special thanks go to Prof. Dr.-Ing. Jürgen Gausemeier, the Head of the Chair of Product Engineering at the Heinz Nixdorf Institute, the University of Paderborn, for his advice and support. He has always challenged and encouraged me. Moreover, working under his supervision has enhanced my technical and personal skills.

Likewise, I would like to thank my co-supervisor Professor Dr. -Ing. habil. Ansgar Trächtler, the Head of the Chair of Control Engineering and Mechatronics at the Heinz Nixdorf Institute, the University of Paderborn. In addition, my thanks go to Prof. Dr. XX and Prof. Dr. XX, who were members of the oral examination commission.

Further, I would like to thank all my colleagues that helped me during my time at the Paderborn University, especially my team leader Michael Grafe; team members: Jan Berssenbrügge, Kareem Abdelgawad, Jörg Stöcklein and Yin Tan; our secretaries: Sabine Illigen and Alexandra Dutschke, as well as our IT-Manager Karsten Mette. Moreover, I would like to thank Jennifer Clos for her support for making the figures look better and Mahynoor Schindel for her English proofreading.

Finally, I would like to thank my parents, my brothers and my sister for their great support during my life. Last but not least, my heartfelt thanks go to my wife Sally for her patience and support, as well as to my daughters: Jana and Lien.

Paderborn, June 2014 *Bassem Hassan*

To my family, Sally, Jana and Lien

Table of Contents

Page

Appendix

Contents

Page

1 Introduction

Heading towards autonomous driving, modern automobiles today are no longer just pure mechanical devices. Rather they are equipped with various sensors and electronic control units (ECUs), which monitor the vehicle and its environment, as well as control the vehicle behaviour [Trä05]. Most of the automotive manufacturers develop modern systems which help the vehicle driver in the complex driving task. Such systems are called Advanced Driver Assistance Systems (ADAS). ADAS support and help the driver in his driving tasks, raise traffic safety, increase efficiency of energy, and provide a comfortable drive [KCF+10].

The development and testing of the in-vehicle systems, such as ADAS, is a challenge due to their complexity and dependency on the other vehicle systems, initial conditions, and the surrounding environment. The testing of ADAS in reality leads to significant efforts and cost. Therefore, virtual prototyping and simulation are widely used instruments in the development of such complex systems [GPD+06].

Virtual prototyping is well-established in facilitating the development of new vehicle systems and components [Mey07]. It is the process of building, simulating, and analysing virtual prototypes. Virtual prototypes are the digital representations (models) of the real prototypes. It allows the verification of the properties and the functions of the product in the early development phases without having to build a real prototype. This saves time and costs [GEK01]. One of the most useful virtual prototyping tools in the automotive field are driving simulators.

Driving simulators allow the ADAS developer to investigate the interaction between the human driver, the ECU virtual prototype and the vehicle, while the human driver steers a virtual vehicle in a virtual environment. Driving Simulators rank among the most complex testing facilities used by automotive manufacturers during the development process. They are based on close collaboration of different simulation models at runtime [Neg07]. These partial models represent dedicated aspects of the different vehicle components, as well as the vehicle environment [EFG+03], [Kau03].

Driving simulators vary in their structural complexity, fidelity and their cost. They range from simple low-fidelity, low-cost driving simulators such as computer-based driving simulators to complex high-fidelity, high-cost driving simulators such as high-end driving simulators with complex motion platforms [WH09].

1.1 Problem Definition

Driving simulators are considered as one of the most complex test rigs used in the development of automotive systems. Using driving simulators in ADAS development requires different driving simulator variants. Each variant should be equipped with different levels of detail relating to all its entire models, in order to perform different test

methodologies, e.g. Software-in-the-Loop (SiL), Model-in-the-Loop (MiL), Hardware-in-the-Loop (HiL), or Driver-in-the-Loop (DiL) [GPD+06]. For example, testing ADAS in SiL focuses on testing the ADAS basic functions and control algorithms. Therefore, it does not require a motion platform or a detailed vehicle model. However, testing the same ADAS in DiL focuses on the interaction between the driver, the vehicle and the system. That is why it requires a motion platform and a detailed vehicle model.

Nowadays, existing driving simulators are usually task-specific devices which are individually custom-developed by suppliers for a specific usage during the ADAS development. These driving simulators can only be configured by a driving simulator expert. This is done by exchanging one or more of their entire components. Existing driving simulators do not allow their operator to change the system architecture or to exchange simulation models without in-depth knowledge of the driving simulator's components and structure.

The development of a driving simulator is a costly and complex task; the testing and training of ADAS often requires more than one configuration of a driving simulator. That is why there is a need for developing a reconfigurable driving simulator that allows the system operator to reconfigure it in a simple way without in-depth expertise in the system.

1.2 Objectives

The aim of this research work is to develop "*A Design Framework for Developing a Reconfigurable Driving Simulator*". The purpose of this design framework is to support developers to develop a reconfigurable driving simulator. Moreover, it should support the operators of these reconfigurable driving simulators to easily create different driving simulators variants.

Based on the requirements of the driving simulator, the design framework should describe the required development phases which are necessary for the development. Additionally, it should describe the variant creation phases in order to create the desired variant of the developed reconfigurable driving simulator. The core of this design framework is the procedure model which structures and defines the required development and variant creation phases. The procedure model should organize all the tasks that have to be carried out in each phase and should describe the methods or algorithms to be used in fulfilling each task. In addition, a configuration tool should be prototypically implemented. This configuration tool should organize and store the driving simulator's solution elements and should allow the driving simulator's operator to create different variants of the driving simulator by selecting a combination of the solution elements which are like building blocks.

The applicability of the design framework should be verified by developing ADAS reconfigurable driving simulators and by creating three variants of it.

1.3 Approach

In order to achieve these objectives, **chapter 2** describes the problem area of this research and states a detailed problem analysis. It starts with the definition of key terms and the classification of the work in the context of the research activities. In this context, driving simulators will be briefly introduced in terms of their history, application fields, classification and structure. Driving simulators are typical mechatronic systems. Therefore, the development processes of mechatronic systems and their specification techniques will also be briefly described. The focus of this work is on the usage of driving simulators in supporting design, development, testing and training of ADAS. Therefore, the ADAS development process and the usage of the driving simulators in the ADAS development will also be described. Then, the problem is described and the reconfigurable driving simulator term is defined. Finally, the requirements of the design framework are defined.

Chapter 3 analyses the state of the art. It starts with a short review of an existing method for the selection of the driving simulator in the automotive field. Then, seven driving simulators are identified as previous approaches towards developing a reconfigurable driving simulator. The seven identified driving simulators are classified into four categories: low-level, mid-level, high-level and multi-level driving simulators. The final two sections of this chapter identify the call for action and describe the main solution approach.

Chapter 4 is the core of this work. It describes "*A Design Framework for Developing a Reconfigurable Driving Simulator"*. This chapter describes the design framework and its main components. The procedure models, their phases as well as the entire tasks of each phase are described in detail. The entire tasks and results of the procedure model are presented with the help of two existing driving simulators as a case study.

Chapter 5 describes the implementation prototype of the configuration tool. Additionally the design framework is validated by a validation example. The validation example is the development of ADAS reconfigurable driving simulator and the creation of three task-specific variants of this driving simulator.

Chapter 6 contains the summary of the work and an outlook on future work.

The appendix provides additional figures and information.

2 Problem Analysis

The main aim of the problem analysis is to define the requirements of the reconfigurable driving simulator design framework. This chapter is going to clarify the problem area. Section 2.1 defines the main terms and expressions, and then it describes the classification of the work based on the defined terms. Section 2.2 describes the fundamentals of the driving simulators. Section 2.3 describes the mechatronic system, its development processes, their specification techniques and a short description of the reconfigurable mechatronic systems. Section 2.4 gives an overview of the advanced driver assistance systems, their classification, development cycle, and the useful usage of the driving simulators during the ADAS development cycle. Section 2.5 describes the problem and section 2.6 defines the reconfigurable driving simulator term. Finally, section 2.7 defines the requirements of the design framework.

2.1 Definition of Terms and Classification of the Work

This work describes "*A Design Framework for Developing a Reconfigurable Driving Simulator"*. According to DUMITRESCU, a design framework consists of a generic procedure model and its related supporting tools e.g. functions, methods, algorithms, etc. as well as the related software tools. The design framework supports the development process of technical systems [Dum11, p. 5f.], [DAG12].

The main objective of this work is to support the development of driving simulators. The term "*development*" comprises the required tasks and activities needed to solve a technical problem. This results in a technical system [Kre12, p. 9]. The main objective of the design framework is to develop a reconfigurable driving simulator. The term "*reconfigurable*" is an adjective of the verb "*reconfigure*" which means "*to arrange the elements or setting of something*" [Oxf14-ol]. The OXFORD DICTIONARY OF ENGLISH defines the term **simulator** as:

> *"A machine designed to provide a realistic imitation of the controls and operation of a vehicle, aircraft, or other complex system, used for training purposes" [Oxf14-ol].*

Driving simulators are typically designed and built for a special purpose in order to support a specific analysis task. During this work, driving simulators are used for supporting the development of **Advanced Driver Assistance System** (ADAS). Driving simulators are considered as virtual prototyping tools, which allow the investigation of mechatronic system aspects, environment, and the interaction between the driver and the vehicle systems [Kre12, p. 9], [CDF+07, p. 2], [Zee10, p. 157f.]. This work describes a design framework to develop a reconfigurable driving simulator whilst considering the ADAS testing and training requirements.

This work is a part of the TRAFFIS project (German acronym for Test and Training Environment for Advanced Driver Assistance Systems), which is funded by the European Union and the Department for Economy, Energy, Industry, Mid-Tier Business, Skilled Crafts and Trades of North Rhine-Westphalia, Germany.

The project consortium consists of the Heinz Nixdorf Institute – University of Paderborn; dSPACE GmbH, a worldwide provider of solutions for the development and testing of automotive control units; Varroc Lighting Systems GmbH, a worldwide automotive components' supplier in the vehicle lighting field; ILV UG & Co.KG, one of most modern traffic safety and training centres in Europe; and UNITY AG, a technology-oriented consulting firm for strategies, processes and systems. The objectives of the TRAFFIS project are: supporting industrial development, testing, and training of the modern ADAS with the help of a reconfigurable driving simulator, as well as the transfer of know-how and project results to research centres and industry.

2.2 Driving Simulators

Vehicle driving is one of the most common activities worldwide. Nevertheless, it is a dangerous and complex task. Driving is an interaction between the driver, the vehicle, and the vehicle environment. It requires full mental and motoric concentration as well as total attention to a multitude of possible traffic scenarios, and could be disturbed by many factors [ARC11, p. 1]. The driving simulators were initially developed to support the driver training programs [FCR+11]. Due to the rapid increase of the performance of electronics and computing power, driving simulators are used nowadays in a wide range of applications. In the next sections, driving simulators are discussed in detail: section 2.2.1 states the early history of driving simulators, section 2.2.2 describes the application fields of driving simulators, section 2.2.3 describes the classification of the driving simulators and the driving simulators' guidelines, and section 2.2.4 defines the structures and components of driving simulators.

2.2.1 Early History of Driving Simulators

The idea of using simulators in driving training issues originates from the first flight simulator which was developed in the early 1910s. The first known flight simulator used for training was developed by the French aircraft manufacturer ANTOINETTE circa 1910. The ANTOINETTE flight simulator was a mechanical device which used to train pilots on using the cockpit controllers [All09, p. 1f.]. As shown in Figure 2-1, the ANTOINETTE flight simulator was a simple cockpit equipped with a similar aircraft controller. The movement of the cockpit was produced by the instructors according to the pilot's interaction with the controller.

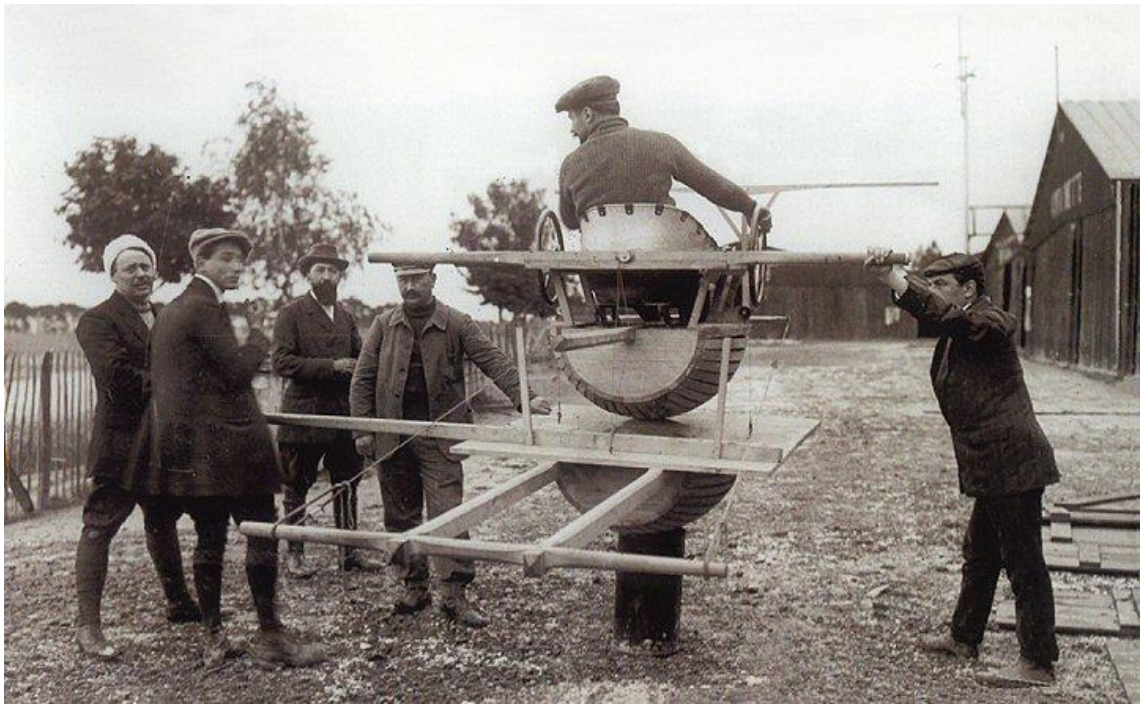


Figure 2-1: The first known flight simulator in1910: ANTOINETTE's rudimentary flight simulator [Ook14-ol].

The widespread use of flight simulators inspired researchers to apply the same concept for road vehicles.

The vehicle driver interacted continually with the surrounding environment. Therefore, in order to simulate a vehicle drive, the visual representation of the surrounding environment had to be illustrated. The first attempts to develop a driving simulator were developed in 1934 by MILES & VINCENT. It was built to provide people with an obligatory drivers' test to highlight the high rate of traffic accidents. The MILES & VINCENT device consisted of a driving cabin with usual controllers, a movable light projector driven by motors, and miniature models which represented the vehicle environment. It allowed a driver to change the vehicle speed and driving direction. According to the speed and direction, the light projector moved over the miniature models and produced a shadow of them on a screen. The driver could then see an illusion of the driving situation on a screen [MV34]. Figure 2-2 shows the MILES & VINCENT driving simulator.

The miniature models approach continued to be used for many years. In 1972, WEIR & WOJCIK had enhanced the MILES & VINCENT ideas by using a video camera. They mounted the miniature models over a rotating model belt. The video camera was mounted over a motor-driven stand. It could be translated to the right and to the left and could also be tilted along its vertical axis. The video camera moved according to the driver input and delivered the recorded scene to a screen in the front of the driver [WW71], [CH11], [HS11].

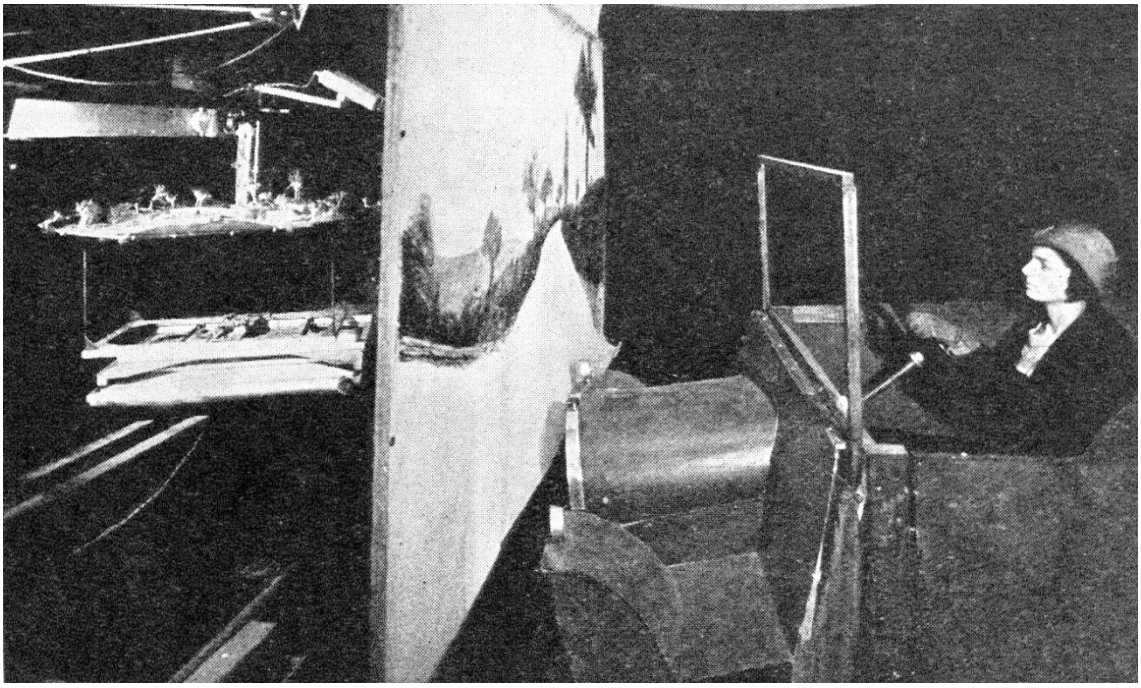


Figure 2-2: The first known driving simulator in 1934: MILES & VINCENT driving simulator [MV34, p. 254].

Driving simulators are rapidly evolving. This is due to the rapid development of digital computers, Central Processing Units "CPU" and Graphical Processing Units "GPU". The digital computers allowed more complex models and 3D visualization to be simulated and represented in real-time [ARC11]. The first driving simulator with a motion platform was developed by Volkswagen in the early 1970s. This driving simulator had a 3 degrees of freedom motion platform [CH11]. The number of driving simulators had increased worldwide by the early 1970s: 28 driving simulators were developed and used [Slo08]. During the 1980s, many automobile manufacturers, e.g. Volkswagen and Mercedes-Benz, developed driving simulators in order to investigate the interaction between the driver and vehicle systems [Str05].

2.2.2 Driving Simulators Application Fields

Driving simulators have a wide range of applications. SLOBE categorized the driving simulators' area of use in three main fields: research, training and entertainment [Slo08]. FISCHER et al. categorized the driving simulators' area of use in three main fields: engineering, medicine and psychology [And11], [FCR+11], [Kan11]. Merging both categorizations results in that driving simulators are usually used in research approaches in engineering, medicine and psychology as well as being used for training and entertainment issues. The wide range of driving simulators application fields can be categorized and stated as follows:

- **Using driving simulators in research**: Driving simulators are used in research for the following purposes:
    - **Engineering research**:
        - Evolution of interior and exterior **vehicle design**.
        - Design, test and evaluation of new **in-vehicle systems** e.g. Advanced Driver Assistance Systems (ADAS) [FCR+11].
        - Early validation of **roadway geometries** [CH11].
        - Supporting the testing of **traffic control devices [**CH11].
        - Investigating the influences of **signs and signals** on the driver activities [FCR+11].
    - **Medical research** [FCR+11]:
        - Investigation of the **patient's driving ability**.
        - Investigation of **medication side effects** on the driving ability.
    - **Physiological research** [FCR+11]:
        - Finding out the **human limitations** in certain driving situations.
        - Drivers' **rehabilitation** after accidents.
- **Using driving simulators in training** [Str05]:
    - Supports the driving training for **driving school students** who apply to get a driving license.
    - Supports **road safety training** programs.
    - Supports **emergency driving training** on public roads.
- **Using driving simulators in entertainment**:
    - Computer and video games [Slo08].
    - Edutainment.

2.2.3 Driving Simulators' Classification and Guidelines

Driving simulators vary in their structural complexity. Therefore, there is a need to classify them. A classification attempt was done by JAMSON. In this attempt, he classified driving simulators according to their utility, usability and cost. JAMSON distinguishes between them as follows:

> *"Utility (or fidelity) describes the degree to which the simulator's characteristics replicate the driving task faithfully. Its usability, on the other hand, describes how versatile the simulator is in terms of ease of reconfiguration from study to study. Research simulators would ideally have good usability, whereas training simulators may focus on strong utility. The financial outlay in the development of a driving simulator is often intrinsically linked to its utility and the ability of the simulator to excite the three main sensory modalities. This often leads to a classification based on cost." [Jam11, p. 12-3].*

That leads to classifying driving simulators into three main categories: low-level, mid-level and high-level driving simulators.

Low-Level driving simulators: They have restricted fidelity, high usability i.e. the ability of simulating different scenarios, and they are usually low-cost driving simulators. Typically, they have a single display which provides a narrow horizontal field of view and a gaming steering wheel as a Human-Machine-Interface (HMI) [Jam11, p. 12-3f.].

Mid-Level driving simulators: They have a greater fidelity than the low-level driving simulators, high usability and they are mid-cost driving simulators. Typically, they have multiple displays which provide a wide horizontal field of view, a real vehicle dashboard as an HMI and sometimes they are equipped with a simple motion platform [Jam11, p. 12-4].

High-Level driving simulators: They have great fidelity, high usability and they are high-cost driving simulators. Typically, they almost have a 360 degrees horizontal field of view and a complete real vehicle as an HMI which is mounted on a high-end motion platform with at least 6 degrees of freedom [Jam11, p. 12-4].

Driving Simulators Guidelines

Due to the wide variation of driving simulators' structural complexity, fidelity and usability, a lot of research institutes and governmental organizations have defined guidelines for using driving simulators in a specific task. These guidelines determine the essential prerequisites that have to be fulfilled by a driving simulator in order to be successfully used in a specific task.

The most known guidelines are those which define the prerequisites of using driving simulators in driving training approaches. For example, on 15 July 2003, the European Parliament and the European Council stated and published the directive 2003/59/EC regarding "the initial qualification and periodic training of drivers of certain road vehicles for the carriage of goods or passengers". This directive allows the usage of a top-of-the-range simulator in the periodic training approaches [EUP03]. Based on this directive, "the German Federal Ministry of Transport, Building and Urban Development" published a recommendation/guideline regarding the usage of a powerful driving simu-

lator in the training approaches [IHK07]. This guideline defines the prerequisites and specifications which have to be fulfilled by a driving simulator in order to be used in the training approaches. For example, it defines that a driving simulator should have a minimum of a 180 degrees horizontal field of view; the motion platform should have a minimum of one rotational degree of freedom for the pitching motion with circa +/- 10 degrees and a minimum of one translational degree of freedom with circa 5 cm range of motion [IHK07].

2.2.4 Driving Simulators' Structures and components

Driving simulators consist of various components. There are many available publications which describe driving simulators' structure and components. ALLEN et al. describe the essential driving simulator as follows:

> *"The major elements of a typical driving simulator as summarized in Figure 2.1 include: cueing systems (visual, auditory, proprioceptive, and motion), vehicle dynamics, computers and electronics, cabs and controls, measurement algorithms and data processing and storage." [ARC11, p. 2-2]*

Figure 2-3 shows ALLEN et al. illustration of the driving simulator's functional components.

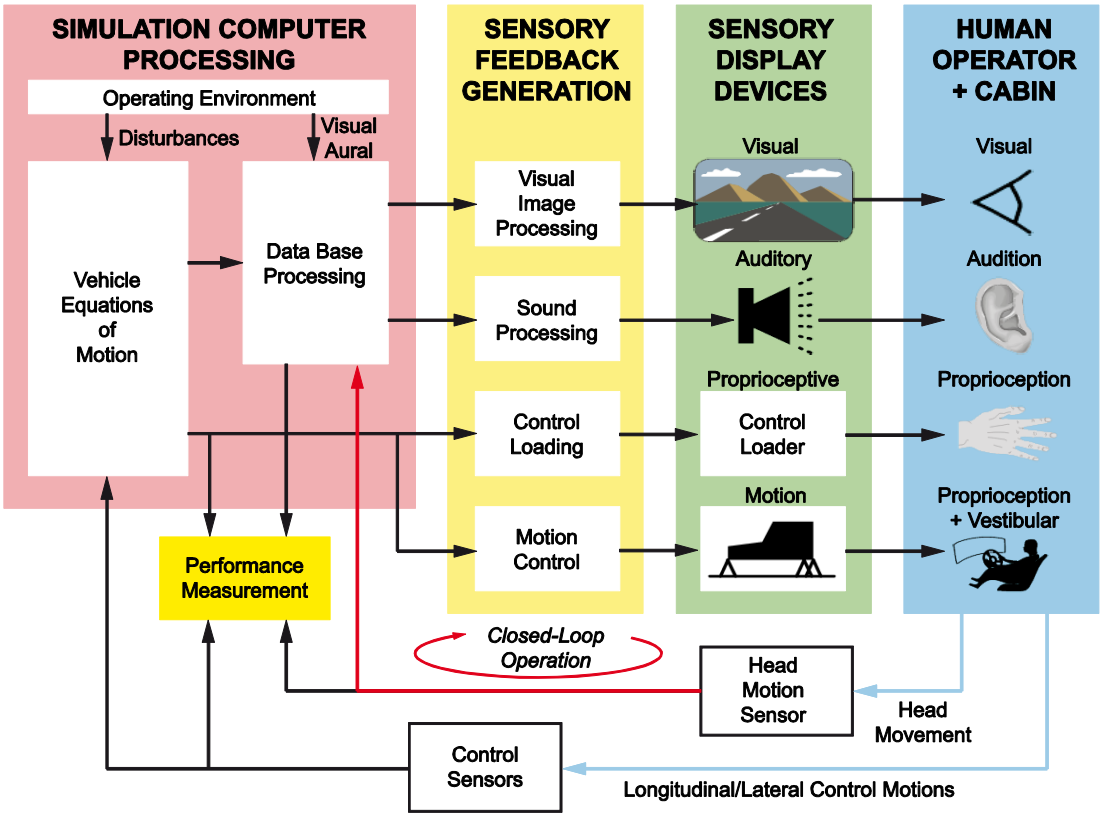


Figure 2-3: Functional components of driving simulators according to ALLEN et al. [ARC11, p. 2-2].

Based on these publications: [ARC11, p. 2-2f.], [Kre12, p. 23ff.], [Neg07, p. 22ff.], [Zee10, p. 159f.], [NDW09, p. 40f.], [Kau03, p. 11ff.], [KG11], the driving simulators' components could be classified in three categories: hardware components, software components and resources as follows:

- **Driving simulators' hardware components:** The most essential hardware components are described as follows:
    - **Input device:** It is the Human-Machine-Interface input device which provides the essential required signals for driving a virtual vehicle. The input signals are typically: acceleration pedal position, brake pedal position, steering wheel angle and gear selector position.
    - **Visualization and acoustic devices:** The visual devices display the computer-generated virtual scene to the driver. They are typically screens or projectors. The acoustic device (e.g. speakers) generates tones, according to the simulated environment sounds.
    - **Motion platform:** This is a mechatronic device which contains a mechanism, which generates motion according to the simulated vehicle movements to produce an illusive haptic feeling of being in motion**.**
- **Driving simulators' software components**
    - **Applications:** There are many software applications that support the operation of driving simulators. They usually support the modelling of the simulation models, e.g. environment creation software, as well being a part of the simulation run-time calculations e.g. visualization software. They also allow the analysis of the simulation results.
    - **Models:** They are the mathematical representation of the entire vehicle components and vehicle's environment. The models have to be calculated and executed during simulation run-time in order to simulate their represented components e.g. vehicle models, other traffic participants models, environment models, etc.
    - **Interfaces:** They interface the diverse hardware and software components of a driving simulator with each other.
- **Driving simulators' resources**
    - **Computers**: They provide the required computing power in order to execute the different applications, models, and interfaces of a driving simulator.
    - **Computer interfaces**: These are resource components which interface hardware components with the simulator computers by converting the physical signals to their respective information signals.

The mentioned components of driving simulators indicate that a driving simulator is a typical **mechatronic system**. It consists of a mechanical mechanism e.g. the motion platform, electronics components e.g. hardware interfaces, control components e.g. motion platform controller as well as information technology components e.g. the various software and simulation models.

2.3 Mechatronic Systems

The term "*Mechatronic*" was initiated by merging the two terms: "Mechanics" and "Electronics". The term "*Mechatronic*" was initially presented in Japan in 1969 by K. KIKUCHI [HTF96]. The term "*Mechatronic*" refers to the extension of mechanical system functions by using electronics components. It has been broadened due to the usage of microelectronics and information technology. Nowadays, it describes the interaction between mechanics, electrics/electronics, control and software components [Sch89], [MDR91], [Wei92].

During this work, the definition of "*Mechatronic*" is considered, according to HARASHIMA, TOMIZUKA und FUKUDA, as described in the VDI-guideline 2206 "Design methodology for mechatronic systems" [VDI2206], as follows:

> *"[Mechatronics is]... the synergetic integration of mechanical engineering with electronic and intelligent computer control in the design and manufacturing of industrial products and processes" [HTF96, p. 1].*

According to the VDI-guideline 2206 [VDI2206] and DUMITRESCU [Dum11, p. 8], a typical mechatronic system structure consists of 4 main units: a basic system, sensors, actors and information processing. These four main units of the mechatronic system form the system closed control loop. Additionally, the mechatronic system environment and the Man-Machine-Interface have to be considered. Figure 2-4 shows the basic structure of a mechatronic system.

The **basic system** is usually a mechanical, electromechanical, hydraulic or pneumatic structure or a combination of them. The **sensors** are generally responsible for determining some of the basic system variables and conditions. The variables determined by sensors are the input of the information processing unit. Regarding the sensors input, the **information processing** unit regulates the system variables, in order to follow a predefined desired manner of the basic system. The regulation of the system is done by the **actors** unit [VDI2206].

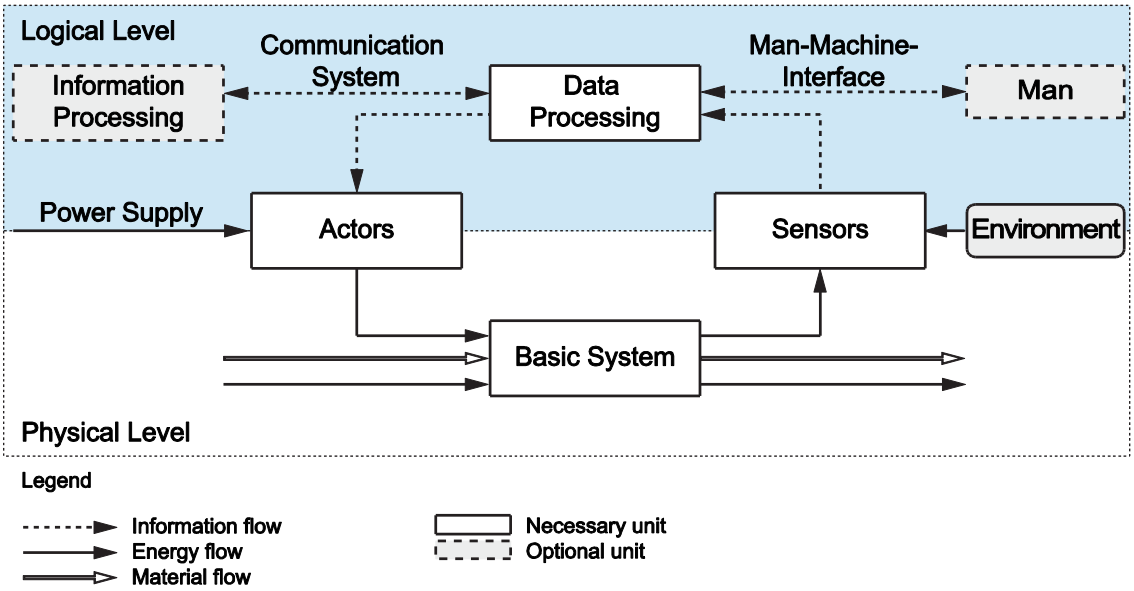


Figure 2-4: Basic structure of a mechatronic system according to [VDI2206, p. 14] and [Dum11, p. 8].

The mechatronic system units are interfaced together with the help of three types of flows as follows [VDI2206]:

- **Material flows:** Material flows describe the exchange of materials such as gases, liquids or solids.
- **Energy flows:** Energy flows describe the exchange of any form of energy such as mechanical, thermal or electrical energy.
- **Information flows:** Information flows describe the information exchange between the system units, e.g. the measured variables.

2.3.1 Development of Mechatronic Systems

The VDI-Guideline 2206 “Design methodology for mechatronic systems” describes a development procedure of mechatronic systems based on the V-model which is commonly used in the software engineering development process. The software engineering V-model had been adapted to meet the mechatronic development approaches. Figure 2-5 shows the V-model as a macro-cycle containing the modelling and model analysis [VDI2206, p. 29].

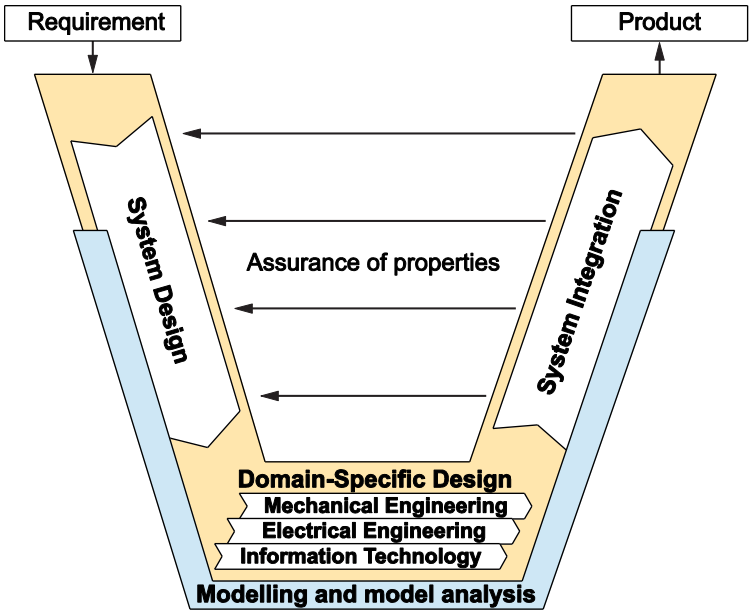


Figure 2-5: The V-model as a macro-cycle [VDI2206, p. 29].

The V-model defines the essential generic tasks which have to be carried out in order to develop a mechatronic system as follows [VDI2206, p. 29f.]:

- **Requirements**: The definition of the product requirements is the starting point of the development. The requirements define the essential tasks which have to be accomplished. Additionally, the requirements are used to evaluate the product after completing the development.
- **System design**: The main objective of this task is to define a cross-domain solution concept. Therefore, the system function has to be divided into sub-functions which can be fulfilled by solution elements or active structures.
- **Domain-specific design**: The developed cross-domain solution concept has to be further concretized. Typically, this concretization is done based on the involved domains.
- **System integration**: In this task, all individual results from various involved domains have to be integrated together.
- **Assurance of properties**: The development progress has to be continually checked against the solution concept and the defined requirements.
- **Modelling and model analysis**: The system properties have to be continually assured and analysed with the help of models and computer-aided tools.
- **Product**: The result of the micro-cycle is either the product or increasing of the product maturity.

2.3.2 Specification Techniques of Mechatronic Systems

The development procedure of mechatronic systems based on the V-model shows that the development process starts with the definition of the requirements. Based on these requirements, the cross-domain solution concept has to be carried out. There is a gap between the two tasks. Usually, the requirements describe the overall system requirements in general and they are usually hard to interpret for all the involved domains. Therefore, there is a need to close this gap by developing a domain-spanning description of the solution concept. Especially, in the early design phases, this is called "principle solution" [GAC+13, p. 8], [GFD+09, p. 201].

A suitable specification technique for the domain-spanning description of the principle solution of mechatronic systems has been developed within the Collaborative Research Centre 614 "Self-optimizing Systems and Structures in Mechanical Engineering", of the University of Paderborn. This specification technique is called CONSENS – "*Conceptual Design Specification Technique for the Engineering of Complex Systems"*.

There have been many attempts towards the establishing of such a specification technique. An overview of the previous attempts is described by GAUSEMEIER et al. [GFD+09, p. 207ff.]. The conclusion of the state of the art investigation results is:

> *"The analysis of the current state of the art shows that there are a lot of approaches on specifying mechatronic systems. One part of the approaches focuses on kinematic, dynamic and controlling behavior. Other approaches give priority to communication relations, operating procedures of the system and state transitions. All of the analysed approaches just fulfill a single part of the requirements on the addressed specification technique, stated in Sect. 3. This applies especially for the aspect of a holistic description of the principle solution. Furthermore, the analyzed approaches do not provide a widespread transition from the domain-spanning specification towards the domain-specific concretization." [GFD+09, p. 209]*

Therefore, the CONSENS specification technique is used for the driving simulator development task during this work as a mechatronic specification technique.

CONSENS – "Conceptual Design Specification Technique for the Engineering of Complex Systems"

CONSENS is used in order to describe the discipline-specific principle solution of a complex mechatronic system. It divides the principle solution description into 7 aspects which are mapped into partial models. The principle solution description consists of a coherent system of partial models as shown in Figure 2-6 [GGT13, p. 3]. The specification of the principle solution is usually done by several modelling iterative.

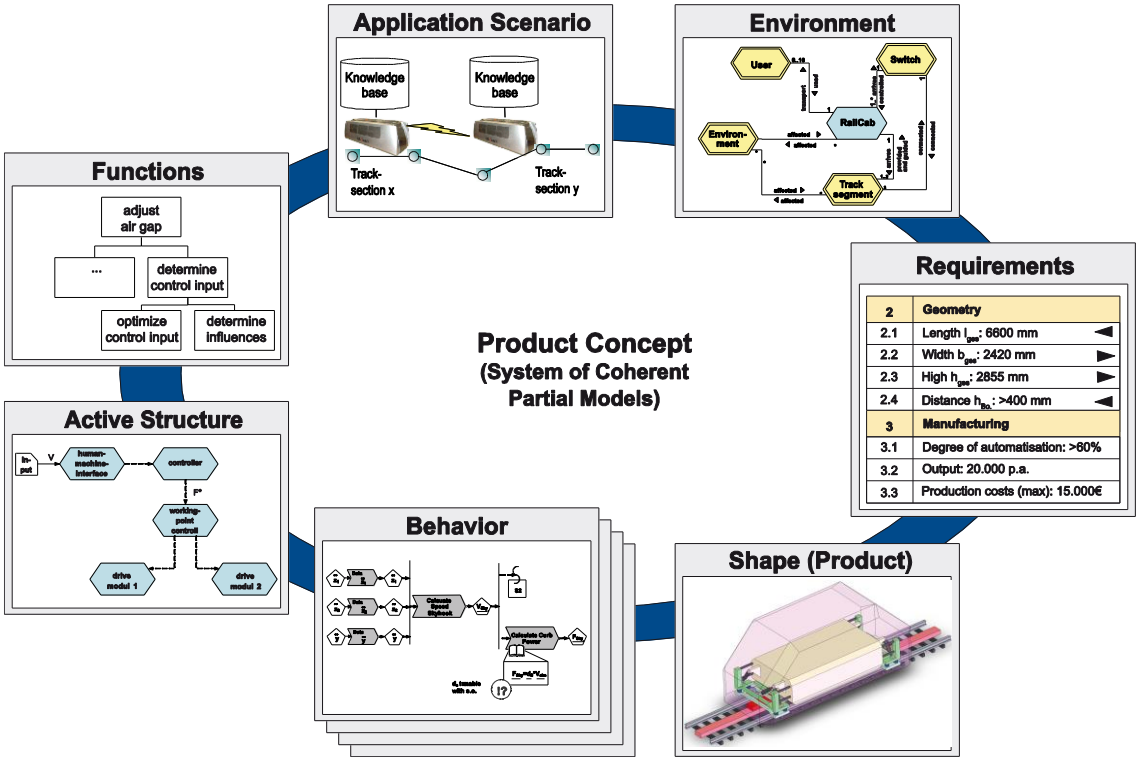


Figure 2-6: The coherent partial model of the specification technique - CONSENS [GGT13, p. 3].

The CONSENS partial models are described as follows [GGT13, p. 3f.]:

- **Environment**: The environment partial model defines the external influences which affect the system under development.
- **Application scenarios**: The application scenario describes some system operational application scenarios of the system in terms of, way of use, operation models, etc..
- **Requirements**: This partial model collects and organizes the system requirements which need to be covered and implemented during the development process.
- **Functions**: The functions partial model describes the system and its entire components' functionality in a top down hierarchy.
- **Active structure**: The active structure partial model structure describes the entire system in more detail, namely in the form of systems' components active principles.
- **Shape**: The shape partial model describes the first shape demonstration of the product e.g. a 3D-CAD model of the system under development.
- **Behaviour**: The behaviour partial model describes the states and the states transitions of the system's behaviour.

2.3.3 Reconfigurable Mechatronic Systems

The aim of this work is to define "*A Design Framework for Developing a Reconfigurable Driving Simulator"*. In the previous section, it was clear that the driving simulator is a mechatronic system. This section gives a brief introduction about the reconfigurable mechatronic systems.

The complexity of mechatronic systems has been encouraged by advancement in technologies and applications e.g. rapidly increasing CPUs computing power and the increased performance of electronics components. This opens up new opportunities to develop reconfigurable mechatronic systems.

A general definition of reconfigurable systems is stated by SIDDIQI and WECK as follows:

> *"Reconfigurable systems can attain different configurations at different times thereby altering their functional abilities. Such systems are particularly suitable for specific classes of applications in which their ability to undergo changes easily can be exploited to fulfil new demands, allow for evolution, and improve survivability." [SW08, p. 1]*

In the past few years, the usage of the term "*Reconfigurable*" has dramatically increased in published technical papers. The most widespread works related to reconfigurable systems have been done in the following fields: In the computing field through the invention of Field Programmable Gate Arrays (FPGAs), in Reconfigurable Manufacturing Systems (RMS) and in Reconfigurable Machine Tools (RMT) as well as in many other applications. [SW08, p. 1].

SIDDIQI and WECK define three main requirements in order to develop a reconfigurable system. The requirements are described as follows [SW08]:

1. **Multiability**: The system should have the ability to perform different functions in different times.
2. **Evolvability**: The system has to be easily changeable over time by removing, adding and/or exchanging new functions or elements
3. **Survivability**: The system has to be functional with minimum predetermined failure.

During this work, only reconfigurable driving simulators are considered. Driving simulators usually consist of a large number of hardware and software components which are combined together to build a driving simulator variant which fulfils a specific task. The reconfigurable mechatronic system concept could be applied to driving simulators. This allows reconfiguring a driving simulator structure in an easy way to fulfil a specific task.

2.4 Advanced Driver Assistance Systems

As mentioned before, driving simulators are used in different fields of application. This work is focussing on the usage of driving simulators in supporting design, development, testing and training of Advanced Driver Assistance Systems (ADAS). The following section describes the ADAS system.

Driving is one of the most popular daily activities people do. Nevertheless, it is a complex and dangerous activity. The driver has to concentrate on many tasks at the same time. The driver's tasks can be classified in three categories according to their priority: primary, secondary and tertiary. The primary driving tasks consist of the following: vehicle navigation by selecting the desired route to move from a place to the next place, vehicle guidance in both longitudinal and lateral directions and vehicle stabilization and controlling of the vehicle position and velocity. The secondary driving tasks consist of the following: switching direction indicators, lights, windshield washer system, etc. The tertiary driving tasks consist of the following: controlling the audio system, air conditioning, infotainment devices, etc. [ARC11, p. 1], [Neg07, p. 6ff.].

Therefore, the automotive manufacturers are developing ADAS with the aim of helping the vehicle driver in the complex driving task. ADAS support and help the driver in his driving tasks, raise road traffic safety, increase efficiency of energy, and grant a comfortable drive [KCF+10].

Using ADAS in cars and trucks has great benefits regarding accident prevention. HUMMEL et al. had analysed thousands of accidents' insurance claims in Germany in order to investigate the safety benefits of ADAS. They found that using one ADAS such as the "Emergency Brake Assist System" can prevent up to 45% of a specific type of accident [HKB+11].

The definitions of the Driver Assistance System (DAS) and the Advanced Driver Assistance System (ADAS) are specified during the project Response 3[1] as follows:

> *"Driver Assistance Systems are supporting the driver in their primary driving task. They inform and warn the driver, provide feedback on driver actions, increase comfort and reduce the workload by actively stabilising or manoeuvring the car. They assist the driver and do not take over the driving task completely, thus the responsibility always remains with the driver. ADAS are a subset of the driver assistance systems." [Res09, p. 4]*

ADAS generally interact between the driver, the vehicle and the vehicle environment. The vehicle environment varies rapidly based on traffic flows and driving situations.

[1] RESPONSE 3 is a subproject of the integrated project PReVENT, which is a European automotive industry activity co-funded by the European Commission. The objective of the projects is to contribute to road safety by developing and demonstrating preventive safety applications and technologies.

Therefore, modern vehicles are equipped with various types of sensors which recognise and analyse the environment. Additionally, the sensory data which is detected by each sensor could be integrated together to assure its accuracy. This is called “Sensor Fusion” [AWH10]. Figure 2-7 shows the different types of ADAS sensors, their positions and their related functions [Bed10-ol].

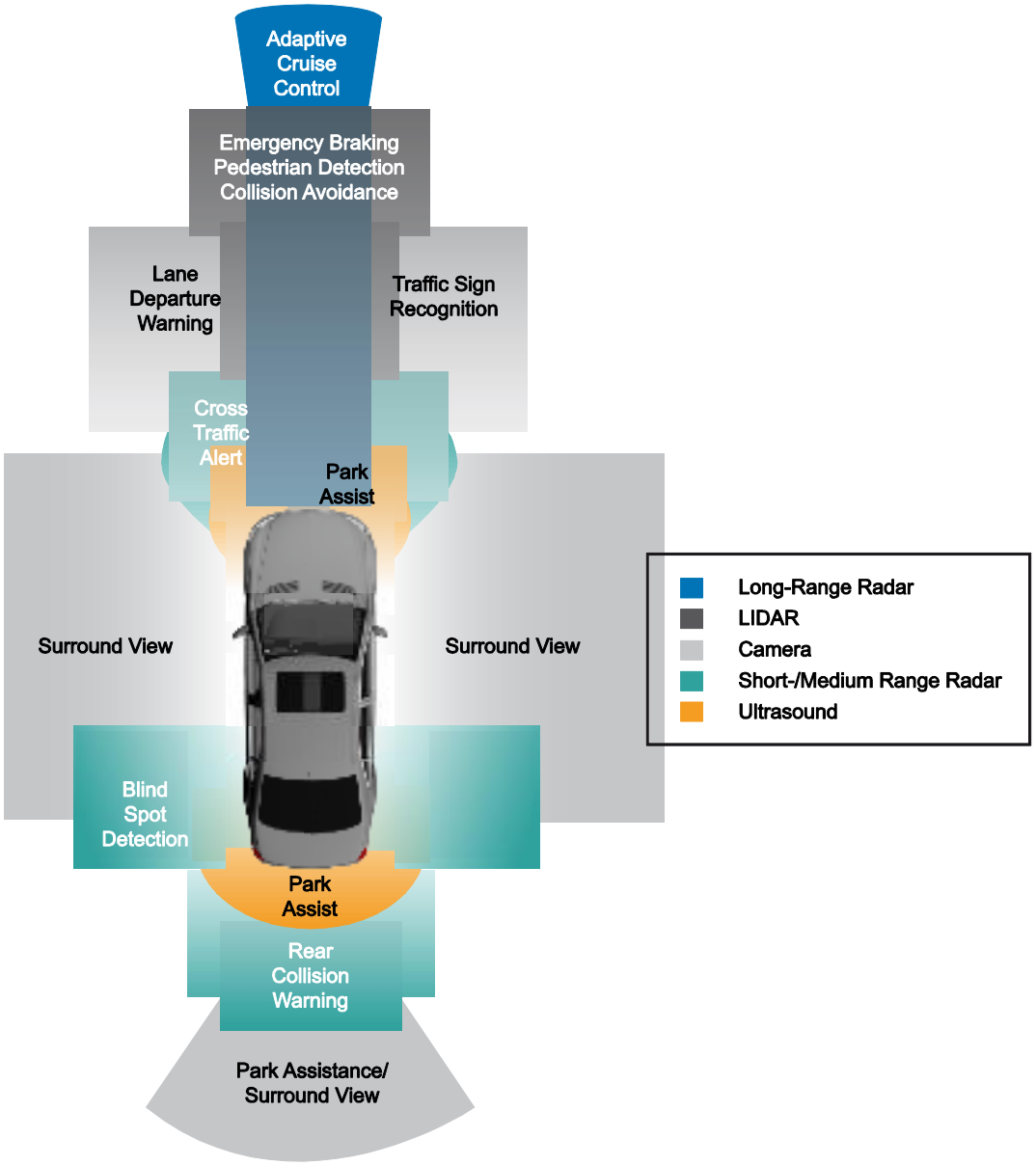


Figure 2-7: ADAS sensor types, placement, and their main functionalities, according to [Bed10-ol].

ADAS Classification

GOLIAS et al. classified the ADAS according to their functionality as follows [GYA02]:

- **Driver support systems**
    - **Driver information systems** such as: navigation devices, Traffic Message Channel (TMC), etc.
    - **Driver precipitation systems** such as: night vision systems, diverse parking systems, etc.
    - **Driver comfort systems** such as: hands-free, infotainment systems, etc.
    - **Driver monitoring systems** such as: attention assistance system, driver drowsiness detection, etc.
- **Vehicle support systems**
    - **General vehicle control systems** such as platooning[2], stop and go assistance systems
    - **Longitudinal and lateral control** such as: Adaptive Cruise Control (ACC), Lane Keeping Assistance (LKA), Lane Change Assistance (LCA), etc.
    - **Collision avoidance systems** such as: intersection collision warning, pre-crash assistance systems, etc.
    - **Vehicle monitoring systems** such as: On-Board Diagnostic systems (OBD), tachographs, etc.

The last few years have witnessed a phenomenal growth in wireless communication and its applications, such as cellular phone networks (mobile networks) and wireless fidelity networks (WiFi). Based on the development in the wireless communication field, many automobile manufacturers are developing a new generation of ADAS nowadays which is called **Car2X** (also known as vehicle-to-vehicle and vehicle-to-infrastructure communications). The main objectives of the Car2X ADAS are increasing traffic safety and traffic flow efficiency [RF09].

The main idea of the Car2X systems is to equip vehicles and infrastructure with communication equipment such as Dedicated Short Range Communication (DSRC), as defined by the IEEE802.11p. This communication equipment allows the communication and the exchange of information between a vehicle and another vehicle, as well as between vehicles and the infrastructure. Figure 2-8 shows the main components of the

[2] Platooning is an innovative method which maximizes a highway throughput. Vehicle platooning divides vehicles in a roadway into groups and controls their velocities simultaneously according to the traffic situation.

Car2X systems, which are vehicle stations and roadside stations, as well as types of communications. Vehicle-to-Vehicle communications are illustrated in green and Vehicle-to-Infrastructure communications are illustrated in blue [MSK+11].

There are many applications of Car2X, such as emergency vehicle drive warning and intersection collision warning.

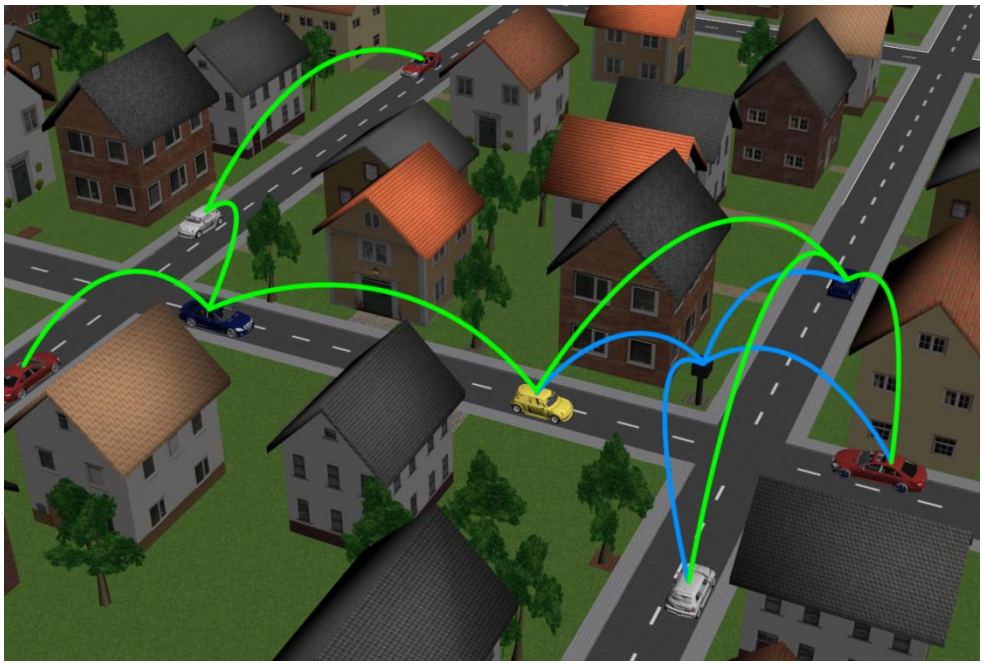


Figure 2-8: Car2X system components and types of communication.

2.4.1 ADAS Development Process

The previous section showed the future trends of ADAS and its great benefits in accident prevention. The ADAS active systems are systems which actively intervene in the vehicle movement by accelerating, braking or steering. In the case of an active ADAS giving a fail alarm or controlling the vehicle wrongly, it could be more harmful for the driver than if the system did not exist [CM11]. Therefore, such a safety-critical system must be tested extensively during its development process.

The development of safety systems in the automotive field follows the V-model which was previously described in section 2.3.1. There are a lot of approaches to adapt the generic V-model for the development of ADAS, e.g. by MAURER [Mau09, p. 45f.], KLEIN et al. [KOM+09, p. 4] GIETLINK et al [GPD+06]. The testing of ADAS during the different design and validation phases is based mainly on the "Virtual Prototyping".

Virtual prototyping is defined by GAUSEMEIER et al. and translated from the original German text as follows:

> *"A virtual prototype or a digital mock-up is an internal computer representation of a real prototype, […]. The virtual prototype is an exten-*

sion of the digital mock-up, because in addition to the shape, there are other aspects taken into account such as kinematics, dynamics, strength, etc." [GEK01, p. 384 f.]

GIETLINK et al. have described the sequential design and validation phases in the development of automotive safety-critical systems. Moreover, they defined the test methodologies which have to be used in each phase [GPD+06]. Figure 2-9 shows the ADAS design and development cycle based on the V-model. It represents the sequential design and validation phases in the development of automotive safety-critical systems [GPD+06, p. 4].

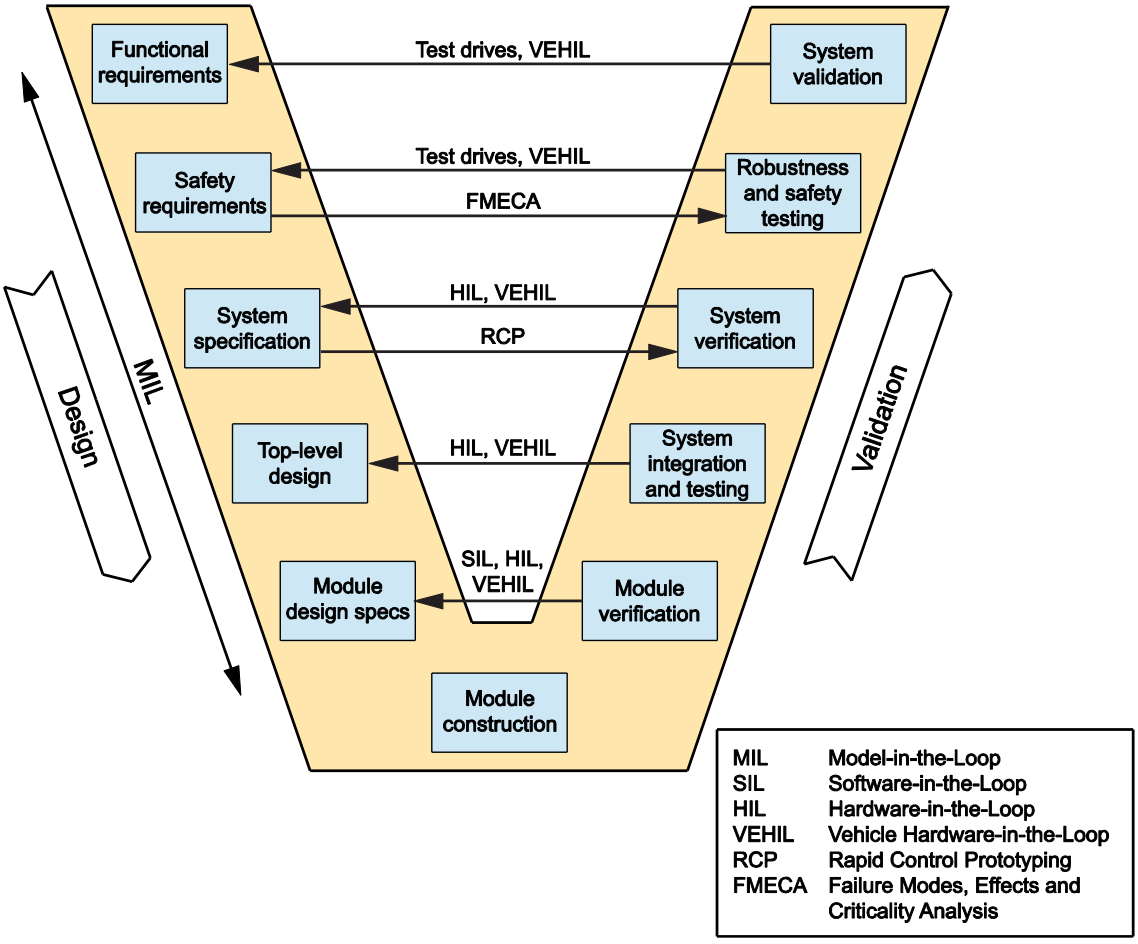


Figure 2-9: The V-model represents the sequential design and validation phases in the development of automotive safety critical systems [GPD+06].

The various used test methodologies are defined as follows [GPD+06]:

- **Model-in-the-Loop (MiL)**: MiL supports the early design of ADAS functionality by modelling the ADAS controller functionally. The model of the ADAS controller is simulated in a closed loop together with vehicle components and environment models.

- **Software-in-the-Loop (SiL)**: After developing the ADAS functionality successfully in a MiL environment, the real control unit code can be generated, integrated and simulated in a closed loop together with vehicle components and environment models under real time conditions. This is called SiL.
- **Hardware-in-the-Loop (HiL)**: Based on the SiL environment, in HiL a model component of the ADAS system (typically, the control unit) could be replaced by its representative real hardware component. The test HiL environment consists of a combination of real and simulated components.
- **Vehicle Hardware-in-the-Loop (VHiL)**: Based on the HiL environment, in VHiL the simulated vehicle model is replaced by a real vehicle. But the vehicle remains operating in an indoor laboratory.
- **Rapid Control Prototyping (RCP)**: RCP is a test method which allows the ADAS developer to test and iterate their simulated ADAS functionality in a real vehicle with the help of RCP tools.
- **Failure Modes, Effects and Critically Analysis (FMECA)**: FMECA is a method which is used to investigate the problems that may have happened from a single failure of the ADAS system e.g. a shortcut in the circuit between two connection pins.

GIETLINK et al. have described the different test methodologies of ADAS control units in combination with the vehicle components, but they did not consider the testing of ADAS systems in combination with vehicle components and the driver together in a driving simulator.

2.4.2 Using Driving Simulators in ADAS Development

Driving simulators are virtual prototyping tools which allow the design, testing and validation of ADAS in a closed loop together with vehicle components, environment and driver [Eng08]. ADAS control units and vehicle components could be real, virtual or a combination of real and virtual components.

In fact, they are named as standard tools for design, development and test by most of the automobile manufacturers. Some examples are cited as follows: Daimler Benz [Zee10], Volkswagen [Nor94], BMW [HN07], Toyota [Cha08], Mazda [YS08], Ford [GCB+06] etc.

The benefits of using virtual prototyping and driving simulators during ADAS development

Using driving simulators in the ADAS design, development, and testing has the following benefits:

- **Time, cost and effort reduction** by developing virtual prototypes instead of developing real prototypes.
- **Dangerous experiments** which are not safe enough to be tested in reality or closed tracks could be executed **safely** in driving simulators. For example, emergency braking assistance, pre-crash assistance systems, etc.
- Realising a **better understanding of the ADAS** by building its mathematical representation (modelling).
- Driving simulators allow the developers to investigate the **interaction between ADAS, vehicle and driver**.
- Driving simulator experiments are **reproducible** with the same parameters and conditions.
- The experiments of driving simulators are independent of environmental conditions such as weather, day/night, etc.

2.5 Problem Description

Driving simulators are complex mechatronic systems which consist of mechanical, electronic, control and software components. They vary in their cost, structural complexity and validity from low-level to high-level driving simulators. These simulators are successfully used in different fields of applications.

Driving simulators are powerful virtual prototyping tools, which allow the design, testing and validation of in-vehicle systems such as ADAS in a closed loop together with vehicle components, environment and driver. The ADAS development process needs different test environments e.g. SiL, MiL, HiL etc. Each of these test environments requires different driving simulator structures and different levels of detail of the driving simulator components.

Despite the fact that the development of driving simulators is costly and complex, the available driving simulators in the market nowadays are usually special purpose facilities. They are individually developed by suppliers for a specific task. These driving simulators could not be reconfigured or in the best case, they have some exchangeable components. Only a driving simulator expert can modify the system architecture or exchange one or more of its entire components. The existing driving simulators do not allow the system operator to change the system architecture or to exchange simulation models without in-depth know-how of the driving simulator system and its architecture.

Therefore, there is a need to develop *A Design Framework for Developing a Reconfigurable Driving Simulator*. This design framework defines the main development steps towards developing a reconfigurable driving simulator. Moreover, it allows driving simulator operators without in-depth expertise in the system to reconfigure a driving

simulator easily according to their individual preferences. The design framework should consist of the following essential components:

- **Procedure model**: It should define the required tasks systematically in order to develop a reconfigurable driving simulator and the variant creation phases. Moreover, it should consider the different mechatronic disciplines and has to simplify the system complexity. The procedure model should consider the different ADAS development environments.
- **Supporting tools**: In order to develop a reconfigurable driving simulator, there are a lot of methods and algorithms that should be used. The used methods and algorithms contain existing approaches. Also, new approaches have to be developed. The methods and algorithms have to be organized within the procedure model.
- **Software tool**: The design framework should be supported by an easy-to-use software tool, which allows the operator to reconfigure a driving simulator without in-depth knowledge of the system.

2.6 Reconfigurable driving simulator definition

There are a large number of existing driving simulators, which vary from high-level facilities to low-level driving simulators e.g. computer games. In most of their descriptions or brochures, they are defined as a "reconfigurable driving simulator". Therefore, the term "reconfigurable driving simulator" has to be clearly-defined with the help of two questions: "Which driving simulator components could be reconfigured?" and "Who can reconfigure the driving simulator?". Based on the answers of the questions, the term "Reconfigurable Driving Simulator" will then be defined.

Which driving simulator components could be reconfigured?

The term "**reconfigurable driving simulator**" is sometimes misused instead of using the term "**driving simulator with exchangeable components**" or the term "**driving simulator with parameterized models**". As mentioned in section 2.2.4, driving simulators consist of various components. These components are classified into three categories: hardware, software and resources. There are many driving simulators which have **exchangeable hardware components** e.g. vehicle mock-up, motion platform, visualization system. Other driving simulators have **exchangeable software components** e.g. vehicle model, traffic model, etc. Most driving simulators have **parameterized simulation models** e.g. a parameterized vehicle model to simulate different vehicle types, parameterized traffic models to simulate different traffic scenarios, etc.

Who can reconfigure the driving simulator?

The term "reconfigurable driving simulator" is sometimes misused instead of using the term "**modular driving simulator**" or "**configurable driving simulator**". Many driv-

ing simulators could be customised individually **by their manufacturer** according to the customer requirements. These are "**modular driving simulators**". Some driving simulator components could be exchangeable or some components could be added or removed. These are **configurable driving simulators** which can be reconfigured or upgraded only by their manufacturer or developer.

Reconfigurable driving simulator definition

A driving simulator is reconfigurable when different configurations can be used optimally in different tasks at different times. The reconfiguration should be feasible by the operator without in-depth expertise in the system structure. The operator can create different configurations by changing the system structure (adding or removing some of its entire components) and by exchanging the entire system components with other suitable components.

2.7 Requirements of the Design Framework

Based on the problem analysis, the essential requirements of the *Design Framework for Developing a Reconfigurable Driving Simulator* have been formed. In the next section, the essential requirements of the design framework procedure model, the reconfigurable driving simulator and the configuration tool are clarified.

2.7.1 Requirements of the procedure model

The procedure model is the design framework core. It defines the required phases in a hierarchy in order to develop and create variants of a reconfigurable driving simulator. Each phase contains entire tasks. These tasks have to be carried out in order to achieve the phase objectives. The following requirements of the procedure model have to be fulfilled:

R1 – Systematic procedure: The development and variant creation phases should be systematically described with the help of a procedure model. The procedure model should support driving simulator developers in order to develop a reconfigurable driving simulator. Additionally, it should support a reconfigurable driving simulator's operator in the creation of task-specific driving simulator variants.

R2 – Complexity reduction: Driving simulators are complex mechatronic systems which use several technologies varying from computer graphics to controlling of motion platforms. Therefore, the procedure model should reduce the complexity of the system for the developers of the driving simulators. Additionally, the procedure model should take in consideration that the operators of the driving simulators do not have in-depth expertise in driving simulator technologies. Nevertheless, they have to be able to create task-specific driving simulator variants.

R3 – Domain-Spanning: Driving simulators are mechatronic systems. They consist of mechanical components, electronics components, control components, and software components. Therefore, the procedure model should consider dealing with these different mechatronic domains. Moreover, the developers of a driving simulator, which are typically an interdisciplinary design team, should be able to understand and use the procedure model.

R4 – High potential for automation: Some tasks of the procedure model use several function and algorithms. These functions and algorithms should have a high potential to be performed automatically with the help of a computer-aided tool.

2.7.2 Requirements of the reconfigurable driving simulator

The main objective of the design framework is to develop a reconfigurable driving simulator. It should fulfil the following requirements:

R5 – Driving simulator reconfigurability: The design framework must allow a driving simulator operator to reconfigure a driving simulator without in-depth knowledge of the system structure. The operator can create different task-specific configurations by changing the entire system structure, by adding or removing entire components or by exchanging the entire used solution elements with other ones. The reconfiguration process should be done without the help of the developer or the manufacturer of the driving simulator.

R6 – Reengineering of existing driving simulators: The design framework is mainly established in order to support the development of new reconfigurable driving simulators. Driving simulators are typically designed and built for a special purpose in order to support a specific task. Each driving simulator consists of a set of software and hardware solution elements which are developed over a long time and almost all of them are costly. Therefore, the design framework should have the ability to reengineer the existing driving simulators into reconfigurable ones. Moreover, the existing solution elements have to be used within the reengineered driving simulator.

R7 – Supporting the development of ADAS: The ADAS development process needs different test environments e.g. SiL, MiL, HiL etc. Each of these test environments requires different driving simulator structures and different levels of detail of the driving simulator components. The reconfigurable driving simulator should cover the different required test environments for the development of ADAS.

2.7.3 Requirements of the configuration software tool

In order to allow the driving simulator operator to create a driving simulator variant or to reconfigure an existing variant, there is a need for a software tool which has to fulfil the following essential functional requirements.

R8 – Separation of concerns: The configuration tool has to prevent the operator from dealing with complex procedures such as mathematical functions or algorithms. The user has to deal with an easy-to-use graphical user interface. The configuration software tool has to separate the concerns between the user interface and the internal complex operations.

R9 – Modular and extendable system structure: The configuration tool must be implemented in a modular way. This has a great benefit as it reduces the coupling degree of the entire software modules. Additionally, the configuration tool must have an extendable structure by adding new functions or algorithms in the future.

3 State of the Art

There are thousands of driving simulators spread all around the globe. They are complex mechatronic systems and include different technologies, which widely range from computer graphics to controlling a complex motion platform. The publications about driving simulators usually take one technology into consideration or just a partial aspect of developing a specific driving simulator. The state of the art in this chapter will only consider the publications which are related to the development methods of driving simulators and the previous approaches towards developing a reconfigurable driving simulator.

This chapter surveys an existing driving simulator selection method and previous approaches towards developing a reconfigurable driving simulator. Section 3.1 describes the driving simulator selection method according to NEGELE. As mentioned previously in section 2.2.3, driving simulators are usually classified according to JAMSON into three categories: low-level, mid-level and high-level driving simulators. In sections 3.2, 3.3 and 3.4, two driving simulators of each category will be described. Section 3.5 describes a previous approach of multi-level driving simulators. In section 3.6, the need for action is derived from the state of the art analysis. Finally, section 3.7 defines the solution approach.

3.1 The Driving Simulators Selection Method according to NEGELE

NEGELE developed a method called the "Application Oriented Conception of Driving Simulators for the Automotive Development". He considered driving simulators as one of the most complex test rigs used in the automotive development. The development of a driving simulator requires a wide expertise in different technologies and disciplines, which widely range from the visualization techniques to platform motion control. This essential know-how is not in the core competence of the automotive manufacturer. Therefore, driving simulators which are used as automotive test rigs are usually developed by driving simulator suppliers. Nevertheless, it is tough for automotive engineers, who do not have a basic knowledge of driving simulator technologies to select and specify a driving simulator which fits with a specific-task [Neg07].

Therefore, NEGELE developed a method which allows automotive engineers to formulate the requirements and specifications of a driving simulator for a specific application. The main objective of the method is to define the relationships between the automotive applications and driving simulators' specification [Neg07].

Automotive engineers could select a **driving simulator type** based on **two main criteria**: a **driving task category** and a **driver stimulus-response mechanism,** according to the application of the required driving simulator.

The **driving tasks** are categorized into **primary tasks**, **secondary tasks** and **tertiary tasks**. The primary tasks consist of **vehicle navigation**, **vehicle guidance** and **vehicle stabilization**. The **driver stimulus-response mechanisms** are categorised into the following: **skills-based responses** which are senso-motoric responses (e.g. acceleration or steering), **rule-based responses** (e.g. driving slower in a curve) and **knowledge-based responses** (e.g. route planning with the help of paper maps) [Neg07].

The driving simulator application should be defined by means of the following: a **driving task category** (Which driving tasks should be investigated?) and a **driver stimulus-response mechanism** (Which driver stimulus-response mechanism is relevant?). For example, if the driving simulator application is the testing of vehicle dynamics, then the application is focussing on a primary driving task (**vehicle stabilization**) and investigating a **skills-based response** of the vehicle driver [Neg07].

Figure 3-1 shows the intersections matrix between the five driving tasks categories: (vehicle stabilization, vehicle guidance, vehicle navigation, secondary tasks and tertiary tasks) and the three driver stimulus-response mechanisms: (skills-based responses, rule-based responses and knowledge-based responses). These result in 15 types of driving simulators which are marked from 1a to 5c [Neg07, p. 94].

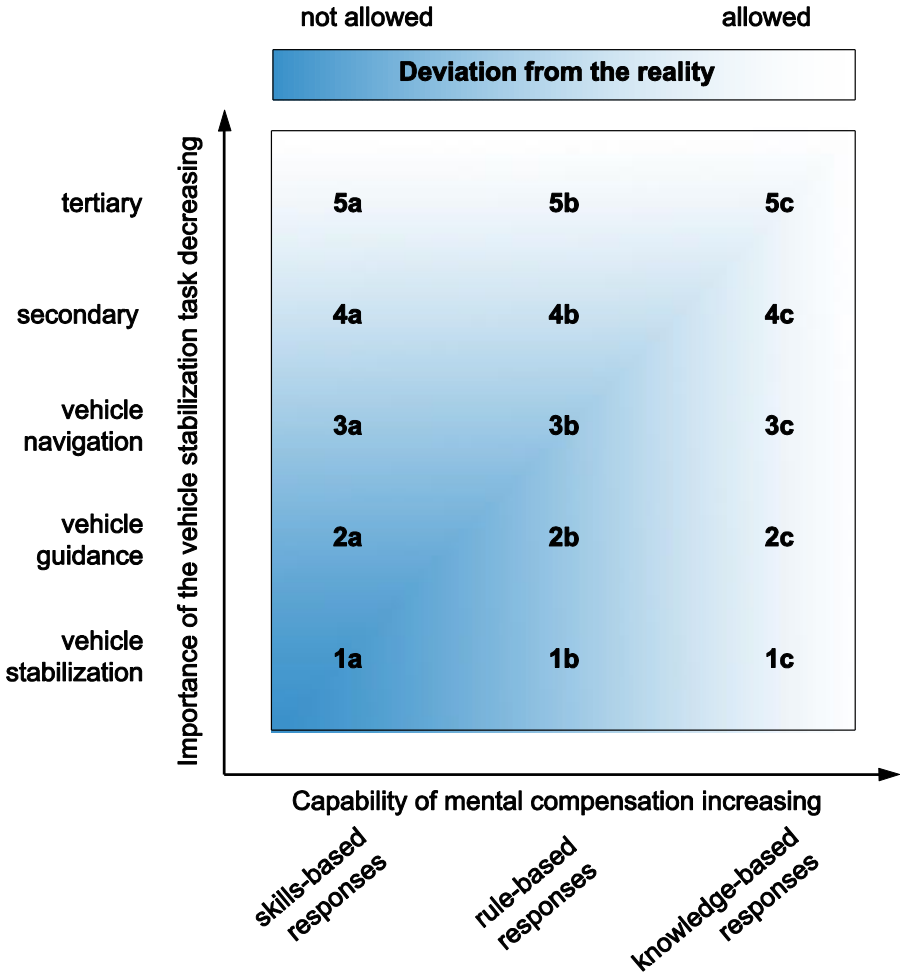


Figure 3-1: Scheme for classifying driving simulator applications [Neg07, p. 94].

Each driving simulator type is described by a profile table. The profile table specifies the entire components of the driving simulator variant. NEGELE divided the simulator into 26 components grouped into 6 groups. Figure 3-2 shows an example of one of the profile tables according to the simulator type 1a [Neg07].

The simulator type 1a fits with the application which focusses on the vehicle stabilization tasks and on investigating the driver's skills-based responses e.g. testing a new vehicle dynamics component. This simulator type should have the following characteristics [Neg07, p. 102]:

Visualization system

- The distance between the driver's eyes and the visualization device should be type A1 (i.e. < 0.8 m).
- The horizontal field of view should be type B2 (i.e. from 120 degrees to 140 degrees).
- It should not have a stereo visualization system or head tracking system.
- The visualization device for the vehicle rear mirrors should be type E2 (i.e. flat displays instead of the original rear mirrors).
- The visualization type in front of the windshield should be type F3 (i.e. Edge-Blinding visualization).
- The visualization resolution should be type G2 (i.e. 2 to 3 arc minute/pixel).
- The visualization frame rate should be type H1 (i.e. $\geq$ 60 Hz).
- The projector types should be type J2 (i.e. the projector reaction time < 8 ms).

Motion System

- The motion platform should be type K1 (that means that a motion platform is a hexapod on a carriage with 1 transitional degree of freedom (DOF) with a displacement of 20–50 mm) .
- The motion platform could be type M1, M2 or M3 (M1 means 7 DOF, M1 means 8 DOF and M3 means 9 DOF) .
- The vehicle dynamics model should be type N3 (i.e. the model is built based on multi-body simulation).
- The tire model should be type O4 (i.e. 3D finite-element tire model).

The simulator profile table also describes the acoustic simulation, the environment model, the traffic simulation and the vehicle mock-up.

Skills-based responses and vehicle stabilization tasks				
Visualization system				
Viewing distance: **A1**	Field of view: **B2**	Stereo visualization: **--**	Head tracking: **--**	Rear mirrors: **E2**
Front display continuity: **F3**	Resolution: **G2**	Frame rate: **H1**	Projector types: **J2**	
Motion system				
Motion platform < 6 DOF: **K1**	Standard motion platform: **--**	Motion platform > 6 DOF: **M1/M2/M3**	Vehicle dynamics: **N3**	Tire: **O4**
Acoustic simulation				
Primary sound: **P1 (P2)**		Secondary and tertiary sound: **Q1 (Q2)**		Sound system: **R1, R3**
Driving cabin and Man-Machine-Interface				
Mock-Up: **S3**	MMI: **T2**	Steering: **U3**	Pedals: **V2**	
Environment model and traffic				
Environment models concept: **W1**	Road type: **Y1**	Traffic simulation: **(Z1)**	Manually controlled traffic vehicle: **--**	

Figure 3-2: Driving simulator profile for skills-based responses and vehicle stabilization tasks [Neg07, p. 102].

Evaluation

The method of NEGELE allows automotive engineers to formulate the requirements and the specifications of a task-specific driving simulator. The focus was on how to specify the requirements of a driving simulator to fit with a specific task. He did not consider the reconfigurability of driving simulators and he did not mention a driving simulator's development method.

Nevertheless, the method is useful as a preliminary work for driving simulator operators. They can use NEGELE's method to specify the preferred driving simulator's requirements and its entire components, then they can use the design framework described in this work in order to create a specific driving simulator variant.

3.2 Existing Low-Level Driving Simulators

Low-level driving simulators have restricted fidelity, high usability and they are usually low-cost driving simulators. Typically, they have a single display which provides a narrow horizontal field of view and a gaming steering wheel as a Human-Machine-Interface (HMI) [Jam11, p. 12-3f.].

The following sections describe two previous approaches towards developing low-level reconfigurable driving simulators.

3.2.1 A Modular Architecture based on the FDMU Approach

FILIPPO et al. had developed "a *modular architecture for a driving simulator based on the FDMU approach*". This approach describes a modular and easily configurable

simulation platform for ground vehicles based on the **Functional Digital Mock-Up approach** (**FDMU**). FDMU is a framework developed by the Fraunhofer Institute. The framework consists of a central component called "*Master Simulator"*, which connects different components through an application called "*Wrapper*". Each module communicates with the master simulator through its own wrapper application and a standardized Functional Building Block (FBB) interface. Figure 3-3 shows the basic scheme of the FDMU architecture [FSS+13].

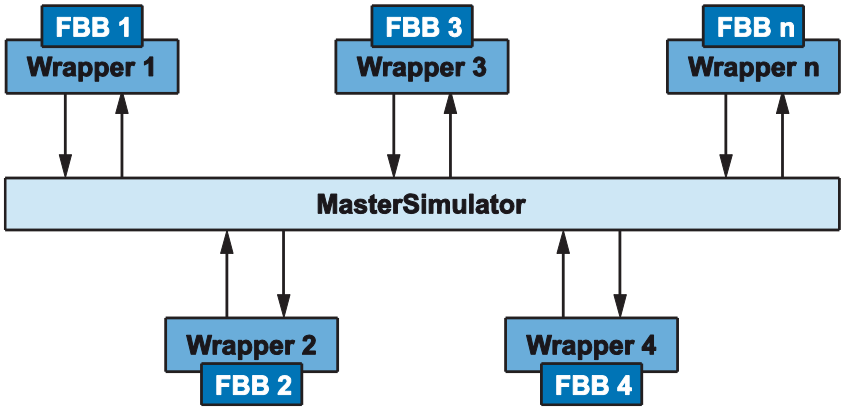


Figure 3-3: Basic scheme of FDMU architecture [FSS+13, p. 4].

FILIPPO et al. had developed a driving simulator based on the FMDU architecture. This driving simulator consists of two hardware components and two software components. The hardware components are a motion platform, which is an off-the-shelf Steward platform, and an input device, which is an off-the-shelf USB steering wheel and pedals. The software components are the master simulator simulation core and a simple vehicle model implemented with the help of Open Modelica [FSS+13].

Evaluation

The developed approach: "*A Modular Architecture for a driving simulator based on the FDMU Approach*" focusses on the interfacing of the different components of the driving simulator with the help of an FMDU modular structure. The problem with this approach is that in order to add or exchange any component, a wrapper application has to be reprogrammed or adjusted for the new component. The approach does not describe how to add, remove or exchange any of the four pre-programmed components. Indeed, the approach is promising for simulation core components, which interface the driving simulator components with each other. But it could not be used in a reconfigurable driving simulator without some enhancements e.g. the master simulation has to be dynamically adjustable depending on the connected modules without being pre-programmed by the user.

3.2.2 QuadDS and HexDS Driving Simulators

QuadDS and HexDS driving simulators are commercial turnkey driving simulators developed by the Mechanical Simulation Corporation, which is a supplier of vehicle behaviour simulation software such as CarSim and TruckSim. The software packages of the Mechanical Simulation Corporation are used in over 140 driving simulators around the world [Car14-ol].

The QuadDS and HexDS driving simulators are developed for engineering applications which require an accurate vehicle dynamics model. They could be used in different application areas such as the design and testing of the Electronic Stability Program (ESP) controllers, the design and testing of ADAS, etc. [Car14-ol].

The QuadDS is equipped with 3 DOF and 1 vibration DOF motion platform, which is actuated by four linear actuators. The QuadDS visualization devices consist of three 60” LCD displays and a smaller LCD to visualize the instrument cluster. It is also equipped with a 5.1 surround audio system. The driver input controllers of the QuadDS are a force feedback steering wheel, pedals and an automatic shift lever. It could be configured as a car or it could have a truck/bus seating configuration. The QuadDS software is based on the CarSim or the TruckSim software packages [Car14-ol]. Figure 3-4 shows the QuadDS driving simulator.

Figure 3-4: A QuadDS driving simulator running TruckSim [Car14-ol].

The other variant of the CarSim driving simulator is the HexDS. It is equipped with a 6 DOF motion platform (hexapod), which is actuated by six linear actuators. The HexDS visualization devices consist of three 40” LCDs. The driver input controllers, the audio system and the software packages are identical to the QuadDS [Car14-ol]. Figure 3-5 shows the HexDS driving simulator.

Figure 3-5: A HexDS driving simulator running TruckSim [Car14-ol].

Evaluation

The QuadDS and HexDS driving simulators are **modular** driving simulators and both variants are operated by using the same software packages (CarSim or TruckSim). They **could be configured** according to the customer requirements by means of the following: two variants of motion platforms (3 DOF or 6 DOF), the vehicle model (truck model or passenger car model) and the visualization devices (three 40" LCD displays or three 60" LCD displays). The QuadDS and HexDS driving simulators are not **reconfigurable driving simulators** because as well-developed as they are, the user cannot exchange the entire components or add a new component to the system without the help of the manufacturer.

3.3 Existing Mid-Level Driving Simulators

Mid-level driving simulators have a greater fidelity than the low-level driving simulators as well as high usability. Typically, they have multi-displays which provide a wide horizontal field of view, a real vehicle dashboard as an HMI, and they are sometimes equipped with a simple motion platform [Jam11, p. 12-4].

The following sections describe two previous approaches towards developing reconfigurable mid-level driving simulators.

3.3.1 The Heinz Nixdorf Institute – ATMOS Driving Simulator

The Atlas Motion System (ATMOS) driving simulator[3] of the Heinz Nixdorf Institute was developed by "Rheinmetall Defence Electronics GmbH". The driving simulator was first developed for the German Army in 1997 with the aim of performing safety training for the military truck drivers. The Heinz Nixdorf Institute of the University of Paderborn built the ATMOS driving simulator in 2009 in cooperation with Rheinmetall Defence Electronics GmbH (RDE). Figure 3-6 shows the Heinz Nixdorf Institute – ATMOS driving simulator.

Figure 3-6: ATMOS driving simulator at the Heinz Nixdorf Institute.

The ATMOS driving simulator is equipped with a motion platform that consists of two dynamical parts with 5 degrees of freedom (DOF). These two parts are independent of each other and the system is fully electrically actuated. The first dynamical part is the moving base. It has 2 DOF and is used to simulate the lateral and longitudinal accelerations of the simulated vehicle. It can move in the lateral plane and at the same time, it has the ability to tilt around the lateral axis with a maximum angle of 13.5 degrees and around the longitudinal axis with a maximum angle of 10 degrees. Four linear actuators are used to control the movements in both directions. The second dynamical part is the shaker system, which has 3 DOF to simulate the roll and pitch angular movements and the heave translation of the simulated vehicle. The shaker is driven by a three drive crank mechanism and by three electrical motors.

[3] This section describes the ATMOS driving simulator at the Heinz Nixdorf Institute in its original delivered status by its manufacturer.

The ATMOS driving simulator has an eight-channel cylindrical projection system (powered by 8 LCD-projectors) which covers a 240 degrees horizontal field of view and three displays in order to visualize the simulated rear mirror views.

The motion platform is equipped with an innovative fixation system, which allows the usage of several driving cabins e.g. truck cabin or passenger vehicle cabin.

The ATMOS driving simulator is operated by off-the-shelf software developed by RDE. The software consists of the simulation core, an operator council GUI, a training scenario editing tool, visualization software, vehicle model, traffic model and audio generation software.

Evaluation

The Heinz Nixdorf Institute – ATMOS driving simulator (in its delivered status) has the ability to use several driving cabins. The delivered software does not allow any configurability or parameterizing of the models such as the vehicle model. Therefore, during this work the software components have to be replaced by software components developed by the Heinz Nixdorf Institute.

3.3.2 The University of Central Florida Driving Simulator

The University of Central Florida (UCF) driving simulator is operated in the Centre of Advanced Transportation Systems Simulations (CATSS). It has evolved since the late 1990's into a mid-level driving simulator with the aim of conducting research in transportation, human factors and real-time simulation. The UCF driving simulator is equipped with a hexapod motion platform with 6 DOF. It has a passenger vehicle cabin as an input device. The vehicle cabin is mounted over the motion platform. The UCF has a visualization system which consists of 5 displays: one for the front view, two for side views and two for the left and middle rear mirrors. The simulator is also equipped with an audio system, force feedback steering wheel and the main operator console [AYR+07], [GKR03]. Figure 3-7 shows two variants of the UCF driving simulator: a passenger vehicle cabin and a commercial truck cabin.

The simulator was designed with an exchangeable vehicle cabin. The user can choose from a commercial truck cabin and a passenger vehicle cabin according to the test requirements. The vehicle model could also be changed according to the used vehicle cabin [GKR03].

Figure 3-7: The two variants of the UCF driving simulator: a passenger vehicle cabin (left) and a commercial truck cabin (right) [GKR03].

Evaluation

The UCF driving simulator has **exchangeable driving cabins** and exchangeable vehicle models. It **could be configured** according to the customer requirements by choosing from the passenger car cabin with its respective vehicle model or the commercial truck cabin with its respective vehicle model. The UCF driving simulator is not a **reconfigurable driving simulator** because only the driving cabin and vehicle model are exchangeable. Moreover, the driving simulator user cannot exchange the entire components or add a new component to the system without the help of the manufacturer.

3.4 Existing High-Level Driving Simulators

High-Level driving simulators have great fidelity, high usability and they are high-cost driving simulators. Typically, they almost have a 360 degrees horizontal field of view and a complete real vehicle as an HMI, which is mounted on a high-end motion platform with at least 6 degrees of freedom [Jam11, p. 12-4].

Toyota Research Driving Simulator (TRDS)

The world's largest, most advanced and most expensive driving simulator is the Toyota Research Driving Simulator (TRDS). It was inaugurated in 2007 and it is located at the Toyota Motors Technical Centre in Higashifuji, Japan. While its development costs have not been made public, most estimates exceed $100 million [Jam11, p. 12-2]. Figure 3-8 shows the Toyota Research Driving Simulator (TRDS).

The TRDS has the world's most advanced driving simulator motion platform. The motion platform is actuated by hydraulic and electrical actuators. It has a 9 DOF motion platform as well as additionally having 3 vibrations DOF. The TRDS dome has a diameter of 7.1 m which can be moved as follows: ±17.5 m in X-direction, ±10 m in Y-direction, and ±0.6 m in Z-direction, ±25 degrees roll-rotation, ±25 degrees pitch-rotation and ±330 degrees yaw-rotation [GB11, p. 7-10].

Figure 3-8: Toyota Research Driving Simulator (TRDS) [TTV14-ol].

The TRDS will be excluded from the evaluation in this section because of the lack of published information about its specifications, its structure and its reconfigurability. The following sections describe two previous approaches towards developing high-level reconfigurable driving simulators.

3.4.1 VTI Sim IV Driving Simulator

The Swedish National Road and Transport Research Institute (VTI) inaugurated the Sim IV driving simulator in May 2011 at the VTI Centre in Gothenburg, Sweden. The VTI Sim IV is used in research projects in different application areas such as in-vehicle system development, HiL of ADAS, road design, driver behaviour investigation, etc. The Sim IV driving simulator is equipped with an 8 DOF motion platform which consists of the two following parts: the XY-motion base which provides linear motion in X-Y directions and a hexapod which provides 6 DOF. The VTI Sim IV dome can be moved as follows: -4 to +3 m in X-direction, ±3.1 m in Y-direction, and -2.6 to +2.4 m in Z-direction, ±16.5 degrees roll-rotation, -15.5 degrees to +16 degrees pitch-rotation and ±20.5 degrees yaw-rotation. The VTI Sim IV has a cylindrical visualization system powered by 9 projectors and gives a 190 degree horizontal field of view and three rear mirrors displays. It has an exchangeable driving cabin [FSA+11]. The VTI Sim IV is operated by VTI's simulator software which is based on Open Source and an in-house developed code. The software is modular and could be integrated with other external software components. Figure 3-9 shows the VTI Sim IV driving simulator.

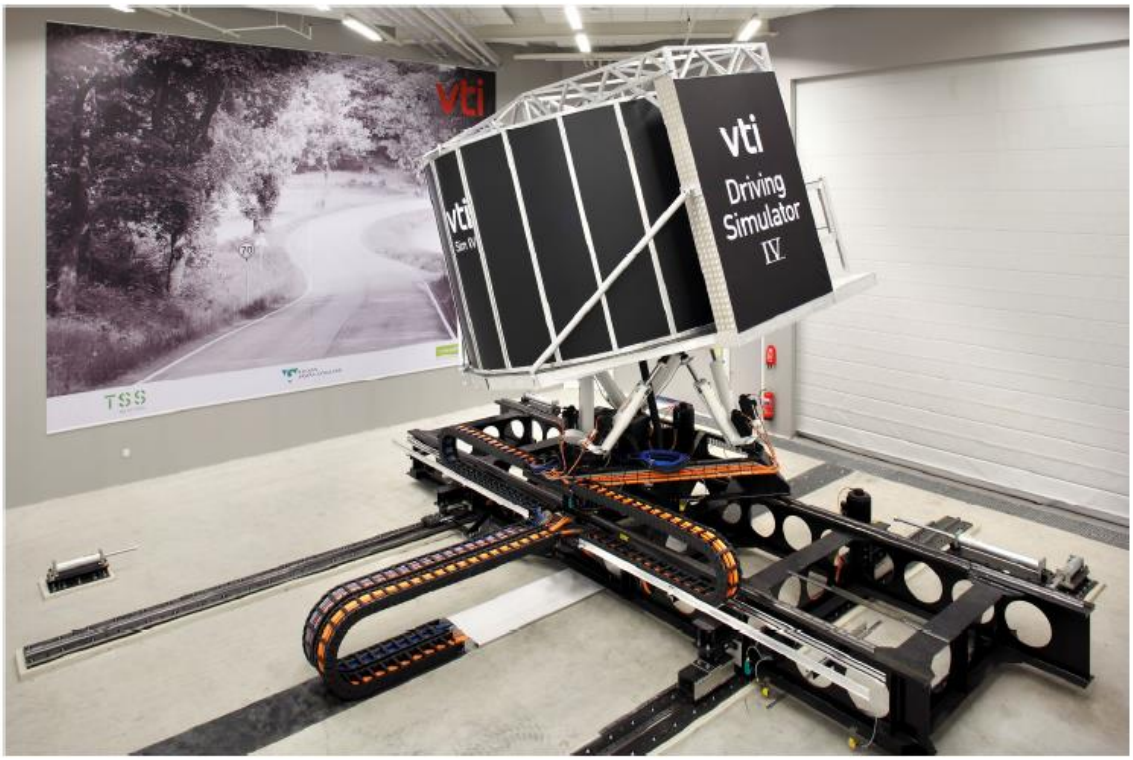


Figure 3-9: VTI Sim IV driving simulator [FSA+11, p. 5].

Evaluation

The VTI Sim IV driving simulator has **exchangeable driving cabins** and a parameterized vehicle model. It **could be configured** according to the test experiment requirements by choosing from different driving cabins and their respective vehicle model parameter set. The VTI Sim IV driving simulator is not a **reconfigurable driving simulator** because only the driving cabin and vehicle model are exchangeable. The driving simulator user cannot exchange the entire components or add a new component to the system without the help of the manufacturer.

3.4.2 Daimler Full-Scale Driving Simulator

Daimler AG inaugurated the Daimler full-scale driving simulator in October 2010 in Sindelfingen, Germany. The Daimler full-scale driving simulator is used mainly in developing new ADAS and the evaluation of different vehicle dynamics concepts. It is equipped with a 7 DOF motion platform which consists of the following two parts: the lateral 12 m long rail system which provides linear motion in Y-direction and a hexapod which provides 6 DOF. The dome of Daimler full-scale driving simulator has a diameter of 7.5 m which can be moved by a rail system for 12 m (in X or Y directions) and by the hexapod as follows: +1.4 to -1.3 m in X-direction, ±1.3 m in Y-direction, and ±1 m in Z-direction, ±20 degrees roll-rotation, -19 degrees to +24 degrees pitch-rotation and ±38 degrees yaw-rotation. Figure 3-10 shows the Daimler full-scale driving simulator.

The Daimler full-scale driving simulator has a cylindrical visualization system powered by 8 projectors and gives 360 degrees horizontal field of view and three rear mirrors displays. It has several exchangeable driving cabins e.g. S-Class, A-Class, Actros-Truck etc. It is operated by a Daimler in-house developed software. The used software can also operate Daimler internal fixed-base driving simulator variants [Zee10].

Figure 3-10: The Daimler full-scale driving simulator [TTV14-ol].

Evaluation

The Daimler full-scale driving simulator has **exchangeable driving cabins** and a parameterized vehicle model. It **could be configured** according to the test experiment requirements by choosing from different driving cabins and their respective vehicle model parameter set. The Daimler full-scale driving simulator is not a **reconfigurable driving simulator** because the driving simulator components are only compatible with Daimler internal components. The driving simulator user cannot exchange the entire components or add a new component to the system without the help of the manufacturer.

3.5 The National Advanced Multi-Level Driving Simulators

The multi-level driving simulators are different variants of a driving simulator as they have different levels of fidelity, usability and cost. But they are developed based on the same structure using the same software, hardware and resources components. An example of the multi-level driving simulator is the NADS driving simulator which is described in this section.

The National Advanced Driving Simulator (NADS) is a driving simulator centre located at the University of Iowa. The NADS centre has three driving simulators: the high-level driving simulator "NADS-1", the mid-level driving simulator "NADS-2" and the low-level driving simulator "NADS miniSim". The NADS driving simulators are based on the same system architecture, software and resources [NAD10].

The NADS software consists of the following components [He06, p. 2f.]:

- Real-time core: This component is the main communication mechanism between the different components through shared memory or TCP/IP protocol.
- Simulation control front-end GUI: This component is the operator council GUI which allows the operator to select, start, stop and replay test drives.
- Driving control feel and instrumentation: This component is responsible for reading the driver's control input signals via steering wheel and pedals, and send them to the vehicle model. It also forwards the vehicle data such as speed and engine RPM to the instruments.
- Vehicle dynamics: This component is a parameterized, physical-based passenger car model.
- Scenario control: This component is responsible for the other traffic participants' behaviour in order to simulate different traffic scenarios.
- Visual rendering: This component is responsible for rendering the virtual scene to the driving simulator displays.
- Audio engine: This component is responsible for providing audio cues of the virtual experiment.
- Driving data collection and analysis: This component is responsible for collecting the simulation data during simulation run-time and stores it in a file for the analysis.

The NADS driving simulators vary in their motion platform, driving cabins, audio systems and visualization devices [NAD10]. The following sections describe the NADS-1 and NADS miniSim in order to illustrate the difference between their motion platforms, driving cabins, audio systems and visualization devices.

The NADS-1

The NADS-1 driving simulator is one of the most advanced high-level driving simulators in the world. It has a 13 DOF motion platform. The dome of the NADS-1 driving simulator has a diameter of 7.3 m which can be moved as follows: ±9.75 m in X-direction, ±9.75 m in Y-direction, and ±0.6 m in Z-direction, ±25 degrees roll-rotation, ±25 degrees pitch-rotation and ±330 degrees yaw-rotation [NAD10], [GB11, p. 7-10].

The NADS-1 driving simulator has a cylindrical visualization system powered by 8 projectors and gives a 360 degrees horizontal field of view and three rear mirrors displays. It has three exchangeable driving cabins as follows: entire car, sport utility vehicle and truck cabin. It is operated by NADS in-house developed software. The used software can also operate the NADS-2 and NADS miniSim driving simulators. The NADS-1 is used for research and development as well as clinical and training applications, which need a high fidelity driving simulator [NAD10]. Figure 3-11 shows the NADS-1 driving simulator.

Figure 3-11: The NADS-1 driving simulator [Nad14-ol].

The NADS miniSim

The NADS miniSim is a low cost PC-based portable driving simulator. The NADS miniSim can be customised to meet the client's specific needs. It uses the same software built into the larger NADS simulators. The NADS miniSim is used for research and development as well as clinical and training applications, which do not need a high fidelity driving simulator [He06], [NAD10].

The NADS miniSim is built in a modular way and can be configured for a specific task by means of the following component varieties:

- Display devices: These could be 1, 3, or 5 displays.
- Input Device: This could be an off-the-shelf USB steering wheel and pedals, force feed-back steering wheel and pedals, quarter vehicle or part of a vehicle cabin. (See Figure 3-12).
- Vehicle model: This could be a personal car model or truck model.
- Motion System: The NADS miniSim could be equipped with a small motion platform that provides motion to the driver cabin.

Figure 3-12 shows two different variants of the NADS miniSim. The left variant is equipped with part of a vehicle cabin as an input/instrumentation device and the right one is equipped with a quarter-vehicle as an input/instrumentation device.

Figure 3-12: Two variants of NADS miniSim [NAD14-ol].

Evaluation

The NADS-1 and NADS miniSim driving simulators are **modular** driving simulators which have been developed based on the same software components. They could be configured for different applications according to the customer specifications. The NADS minSim is a low-level **configurable** driving simulator. It is a **promising approach towards developing a reconfigurable driving simulator**. However, **it is not a reconfigurable driving simulator**, because as well-developed as it is, the user cannot exchange the entire components or add a new component to the system without the help of the manufacturer.

3.6 Call for Action

The analysis of the existing methods and approaches towards a reconfigurable driving simulator has shown that there is no method, approach or developed driving simulator to date which covers all the previously defined requirements in section 2.7. Figure 3-13 shows the evaluation overview of the state of the art individual approach according to the previously defined requirements. The evaluations are briefly described as follows:

R1 – Systematic Approach:

So far, there has been no approach which has described a development systematics or a method in order to develop a reconfigurable driving simulator. Nevertheless, NEGELE's method is useful as a preliminary step for the reconfiguration of a driving simulator. Driving simulator developers and operators can use NEGELE's method to specify the task-specific variants of the driving simulator, the variants' structure and its desired solution elements. Then, they can use the design framework described in this work in order to develop them and to create the desired variant.

R2 – Complexity Reduction:

All of the previously investigated approaches have built the driving simulator in a modular way and the complexity is partially reduced from the developer's point of view. But they have hardly described the system's internal architecture from the operator's point of view.

R3 – Domain-Spanning:

Most of the previously investigated developed driving simulators have partially considered the different mechatronic disciplines. In particular, the Daimler and NADS driving simulators have considered all the mechatronic disciplines during the development.

R4 – High Potential for Automation:

Most of the previous approaches do not have a high potential for automation in order to reconfigure a driving simulator. The exchanging of the different available solution elements is done manually. However, the Daimler and NADS driving simulators partially have the potential to exchange the available solution elements automatically.

R5 – Driving Simulator Reconfigurability:

None of the investigated methods and approaches allows the driving simulator operator to change the entire structure by adding or removing new components. However, the "modular architecture for a driving simulator based on the FDMU approach" method, as well as the Daimler and the NADS driving simulators, have promising approaches and structures towards becoming reconfigurable driving simulators.

R6 – Reengineering of Existing Driving Simulators:

Most of the previous approaches have some exchangeable components e.g. driving cabin, vehicle model, motion platform, etc. However, none of them has the ability to exchange the entire components without adapting and integrating the new components manually. Only the modular architecture for a driving simulator based on the FDMU approach as well as the Daimler driving simulator have considered the reengineering of the existing driving simulator.

R7 – Supporting the Development of ADAS:

All of the investigated driving simulators support the development of ADAS in one or more phases of the development. However, the QuadDS, HexDS, Daimler and NADS driving simulators support the development of ADAS during the whole development cycle.

R8 – Separation of Concerns:

None of the investigated methods and approaches has a configuration tool which allows the user to reconfigure the driving simulator. Nevertheless, the operator council soft-

ware of most of the existing driving simulators is based on easy-to-use graphical user interfaces.

R9 – Modular and Extendable System Structure:

None of the investigated methods and approaches has a configuration tool which allows the user to reconfigure the driving simulator. Nevertheless, most of the existing driving simulator software components are modular and extendable.

Evaluation of the investigated approaches **Evaluation scale:** ○ = Not fulfilled ◐ = Partially fulfilled ● = Totally fulfilled		**Requirements**								
		Systematic Procedure	Complexity Reduction	Domain-Spanning	High Potential for Automation	Driving Simulator Reconfigurability	Reengineering of Existing Driving Simulators	Supporting the Development of ADAS	Separation of Concerns	Modular and Extendable System Structure
		R1	R2	R3	R4	R5	R6	R7	R8	R9
Driving Simulators Selection Method	Driving Simulators Selection Method according to NEGELE	◐	◐	◐	○	○	○	◐	○	○
Low-Level Driving Simulators	A Modular Architecture based on the FDMU Approach	○	◐	◐	○	◐	●	◐	◐	◐
	QuadDS and HexDS Driving Simulators	○	◐	◐	○	○	◐	●	◐	◐
Mid-Level Driving Simulators	ATMOS Driving Simulator	○	◐	◐	○	○	◐	◐	○	○
	UCF Driving Simulator	○	◐	◐	○	○	◐	◐	◐	◐
High-Level Driving Simulators	VTI Sim IV Driving Simulator	○	◐	◐	○	○	◐	◐	◐	◐
	Daimler Full-Scale Driving Simulator	○	◐	●	◐	◐	◐	●	◐	◐
Multi-Level Driving Simulators	NADS Driving Simulator	○	◐	●	◐	◐	●	●	◐	◐

Figure 3-13: Evaluation overview of the state of the art individual approaches according to the previously defined requirements.

Conclusion:

None of the investigated methods and approaches in the state of the art meets all of the requirements, which have been previously defined in section 2.7. Most of the investigated approaches describe a modular driving simulator or a driving simulator with few exchangeable solution elements. None of them describes any systematics or approaches for the development of a reconfigurable driving simulator and none of them allows the operator of the driving simulator to reconfigure the system without in-depth expertise in the system structure.

3.7 The Solution Approach

The main aim of this work is to simplify a driving simulator structure during the development. This simple structure allows the operator to create different task-specific variants by selecting the desired solution elements of the driving simulator.

The development of reconfigurable mechatronic systems which consist almost of standardized modular components can follow the "Building Blocks Concept". The benefits of using the building blocks concept are speeding up the learning curve of the system structure based on the many years of experiences in the development of their entire components [Grä04, p. 59ff.]. Therefore, the solution approach of this work is based on the "Building Blocks Concept".

The typical virtual prototyping cycle consists of three phases: modelling, simulation and analysis. The modelling process is the developing of simplified formal models of the system under development. The system models represent the system properties. The simulation process represents the calculations of the system models with the help of numerical algorithms in order to simulate the system behaviour. The analysis process represents the interpretation of the simulation results that are usually done by extracting, preparing and visualizing the relevant information [GEK01, p. 419ff.], [Kre12, p. 19].

In order to reconfigure a driving simulator, there is a need to add a phase between the modelling and simulation phases. The new phase is the configuration phase shown in Figure 3-14. In the configuration phase the driving simulator operator can select the desired solution elements to create a task-specific variant of the driving simulator. The models which have been developed during the modelling phase will be available for the selection in addition to other existing components. The operator selects a solution element for each component. These selected solution elements, acting as building blocks, build together a driving simulator variant. Figure 3-14 shows a simplified example of the configuration process; the selected solution elements and the created variant are marked with a blue frame. As soon as a variant has been created, the driving simulator will be ready for the simulation and the analysis phases.

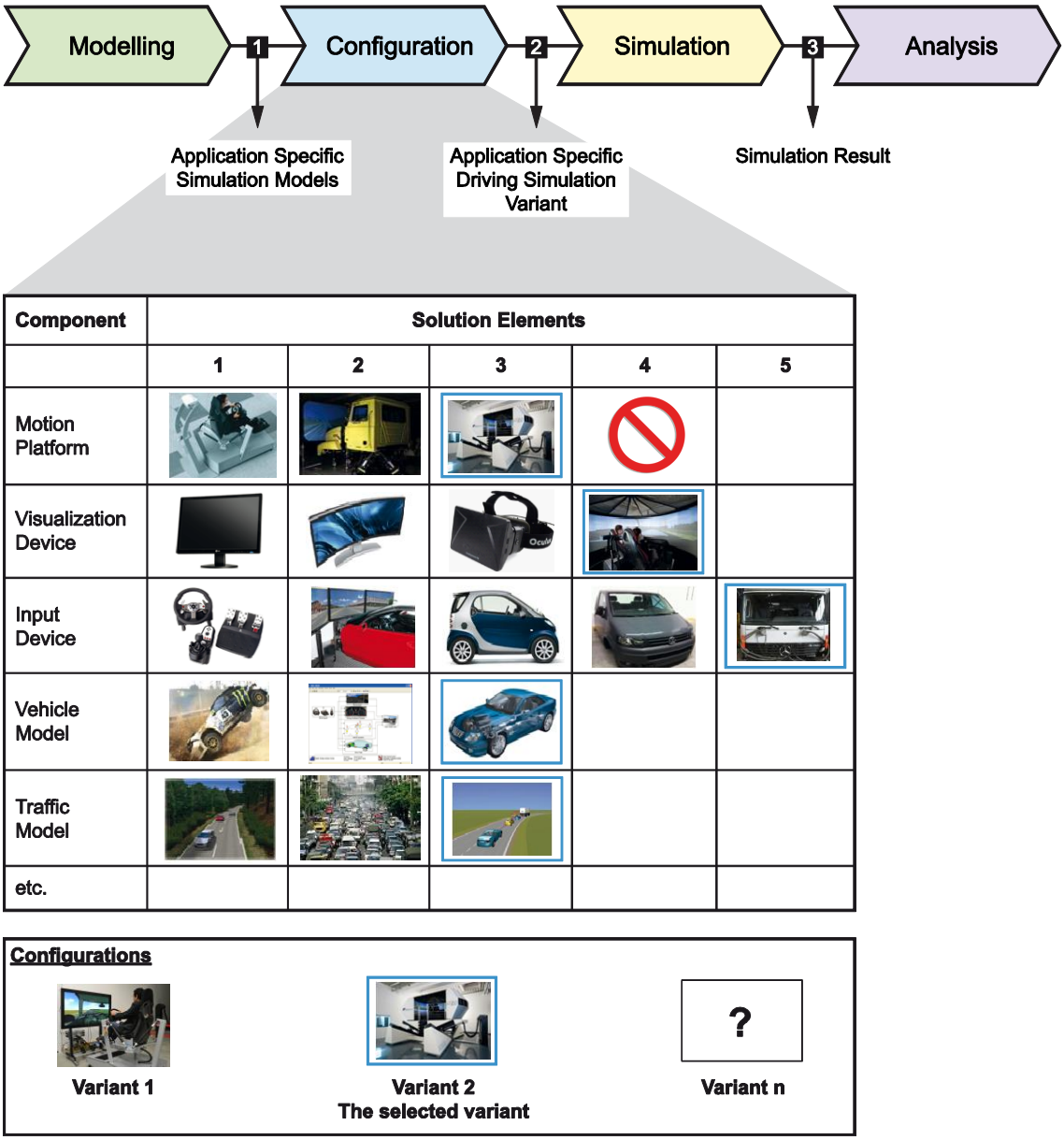


Figure 3-14: The solution approach of the reconfigurable driving simulator, according to the building blocks concept.

4 A Design Framework for Developing a Reconfigurable Driving Simulator

This chapter is the core of the present work. It describes *A Design Framework for Developing a Reconfigurable Driving Simulator*. This design framework supports driving simulator developers and operators to develop and operate a reconfigurable driving simulator. The design framework has to meet the requirements, which are derived from the problem analysis in section 2.7, and it has to satisfy the call for the actions defined within the state of the art in section 3.6. As mentioned previously in section 2.2.3, driving simulators vary a lot in their structure, fidelity and area of use. Therefore, a general design framework for developing a reconfigurable driving simulator is needed.

The design framework consists mainly of the procedure model and the configuration tool[4]. They are specifically described as follows:

- The **procedure model** defines the required phases in a hierarchy, in order to develop a reconfigurable driving simulator. Each phase contains entire tasks; these tasks have to be carried out in order to achieve the phase objectives. The procedure model organizes the required tasks in each phase and describes which method or algorithm should be used to fulfil each task. The used methods and algorithms contain existing approaches as well as new approaches, which were developed during this work. Moreover, the procedure model defines the result of each phase. This is needed as an input for the following phases.
- The **configuration tool** supports the driving simulator operators in creating a driving simulator variant or in reconfiguring an existing variant. The configuration tool organizes the existing driving simulator software and hardware components and their corresponding solution elements in a solution elements database. As soon as the solution elements database is filled, the software guides the driving simulator operator in order to create the desired driving simulator variant. The variant creation will be done by selecting a combination of solution elements, which are available in the database. Moreover, the configuration tool can deal with guidelines for testing and/or for training approaches. They can be added to the tool, and the configuration tool can check whether the created variant guideline conforms or not.

Figure 4-1 demonstrates an overview of the design framework, constituent components as well as its correlation to the various chapters of this work.

[4] The configuration tool is a software program, which is prototypically implemented during this work.

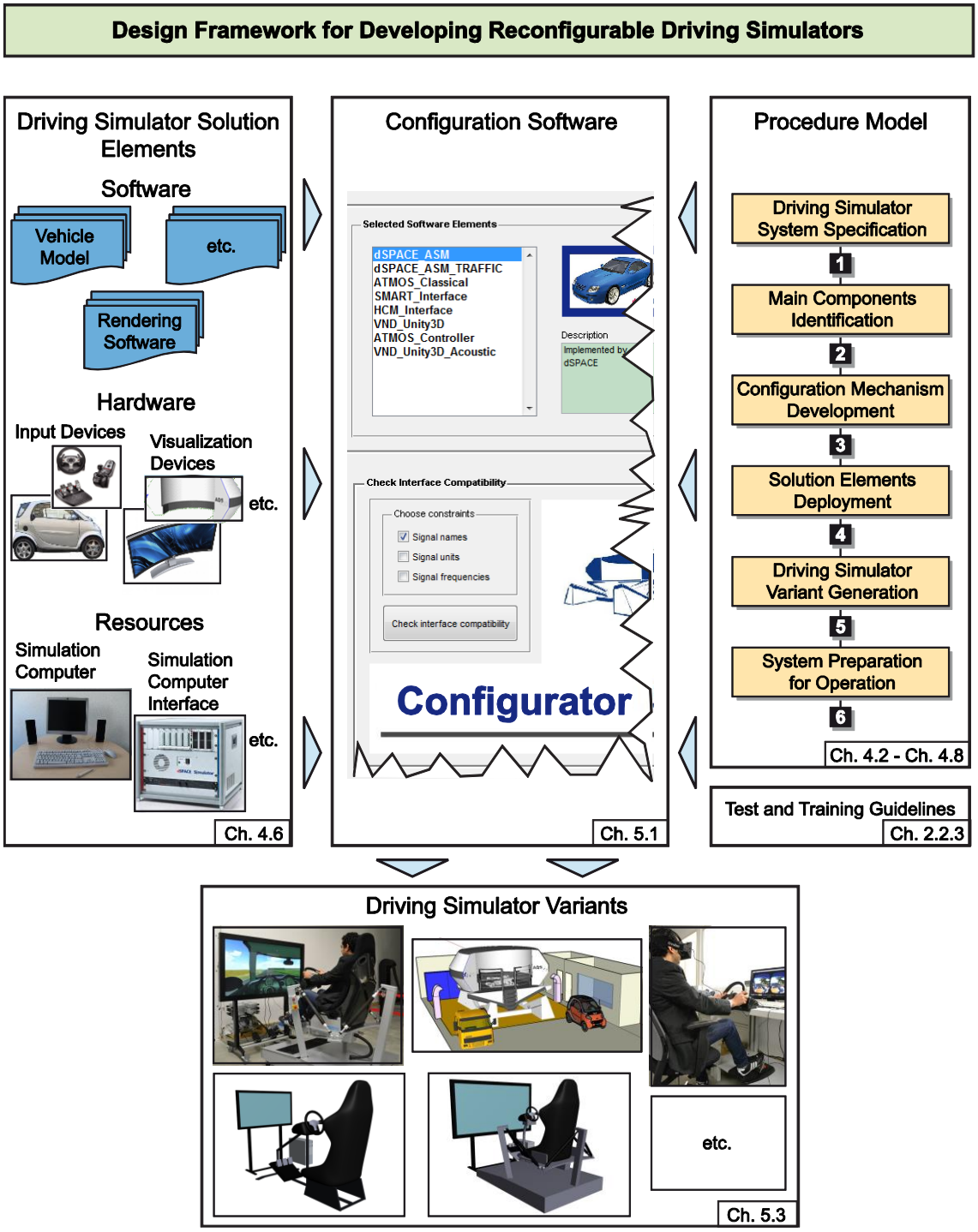


Figure 4-1: A Design framework for developing a reconfigurable driving simulator structure and components.

The following sections describe the case study and the procedure model phases. Section 4.1 describes the case study, which is the running example during this chapter. Section 4.2 gives a short overview of the procedure model as well as its main phases and milestones. The individual tasks within each phase are then presented in sections 4.3 to 4.8; each of which includes a detailed description of the entire tasks, the used utilities, meth-

ods, and/or techniques as well as the respective results. In chapter 5, the implementation prototype of the configuration tool is presented.

During this chapter, the procedure model phases, the entire tasks and results of the phases are described methodically. In order to make the procedure model more understandable, the entire tasks and results of the phases will be presented with the help of a case study example. Furthermore, in chapter 5, the design framework will be validated with the help of an ADAS task-specific driving simulator.

4.1 Case Study – HNI Existing Driving Simulators

In order to make the procedure model understandable, there is a need to illustrate the described phases, the entire tasks and results with the help of a practical example. This work's main objective is to build an ADAS reconfigurable driving simulator with a complex structure. It will have up to 27 system components and up to 67 solution elements. Therefore, two of our existing driving simulators will be used during this chapter as a running example instead of ADAS task-specific driving simulators.

The case study variants have a **simple structure** which makes the design framework more understandable. Moreover, it shows that the usage of the procedure model is **independent of the area of use**.

In the following section, the two case study driving simulators and the case study objectives are briefly described.

4.1.1 Case Study Variants

The case study is presented through two existing variants: the *HNI Airmotion ride driving simulator* and the *HNI PC-based driving simulator*. Both variants were developed individually in our laboratory at the Heinz Nixdorf Institute with the objective of advanced levelling light system evaluation [BKG08]. Although both driving simulators were developed for research purposes, they will be considered in this chapter as entertainment driving simulators which makes their structure simpler. Additionally, the modelling and analysis tools will be neglected. This allows keeping the driving simulator structure as simple as possible. The following section briefly describes both variants.

Variant 1 – The HNI Airmotion Ride Driving Simulator: This is an interactive driving simulator with a motion platform which is called Airmotion ride. The driving simulator is used for the interactive driving of a virtual vehicle in a virtual environment without other traffic participants. Airmotion ride is a commercial motion platform produced by the company FESTO [Fes14-ol]. It was integrated[5] with our in-house devel-

[5] The integration between the Airmotion ride and the VND was done by myself during my research activities.

oped visualization software: Virtual Night Drive "VND" [BKG08]. Figure 4-2 shows the HNI Airmotion ride driving simulator.

Figure 4-2: The HNI Airmotion ride driving simulator.

The HNI Airmotion ride driving simulator consists of the following hardware components, software components and resources:

Table 4-1: The HNI Airmotion ride driving simulator components

Hardware	Software	Resources
Motion Platform: Airmotion ride; a pneumatic actuated inverted hexapod	**Motion Platform controller:** A simple motion controller based on virtual vehicle position and orientation	**Simulation Computer:** A single Windows PC with a commercial processor and a commercial graphics card
Input Device: USB steering wheel and pedals	**Vehicle Model:** A simple vehicle model based on Matlab/Simulink	**Simulation Computer Interface:** USB interface
Visualization Device: 75" LED monitor	**Rendering Software:** Virtual Night Drive "VND"	
Acoustic Device: Dolby Speakers	**Acoustic Software:** Virtual Night Drive "VND"	

The HNI PC-based Driving Simulator (Variant 2): This variant is also an existing driving simulator in our laboratory. The driving simulator is used for interactive driving in a virtual environment without other traffic participants. It is a static driving simulator without motion platform and it has a Virtual Reality head-mounted display as a visualization device. Figure 4-3 shows the HNI PC-based driving simulator.

Figure 4-3: The HNI PC-based Simulator.

The HNI PC-based driving simulator consists of the following hardware components, software components and resources:

Table 4-2: The HNI PC-based driving simulator components

Hardware	Software	Resources
Input Device: USB steering wheel and pedals	**Vehicle Model:** A simple vehicle model based on game engine library	**Simulation Computer:** A single Windows PC with a commercial processor and a commercial graphics card
Visualization Device: Oculus Rift[6]	**Rendering Software:** Virtual Night Drive "VND"	**Simulation Computer Interface:** USB interface

[6] The Oculus Rift is a commercial Virtual Reality head-mounted display [Ocu14-ol].

Acoustic Device: Dolby Speakers	**Acoustic Software:** Virtual Night Drive	

4.1.2 Case Study Objectives

Both case study variants were developed individually; each one of them has its fixed structure, certain software and hardware components. Furthermore, the interfaces between the different components were done manually. Applying the procedure model on the case study variants has two benefits. The first benefit is to make the procedure model understandable by illustrating it with the help of practical examples. The second benefit is to develop both driving simulators only once; by applying the procedure model, this results in a reconfigurable driving simulator. Based on this reconfigurable driving simulator, the operator will have the ability to create both variants easily. In order to convert both existing driving simulators into one reconfigurable driving simulator, there are three objectives that have to be achieved:

- **Reengineering of two existing driving simulators:** The first objective is to reengineer the two existing variants to have the **same simulation core and to interface their entire components automatically**.
- **Change Driving Simulator Structure:** The second objective is to make the driving simulators reconfigurable; i.e. by **adding or removing one or more of their entire components** in a simple way without in-depth expertise in the system and without changing the interfaces manually. The case study illustrates that the first variant has a motion platform, while the second variant does not have a motion platform.
- **Exchange Driving Simulator Component:** The third objective is to make the driving simulators reconfigurable; in terms of **exchanging one or more of the driving simulator solution elements** in a simple way without in depth expertise in the system and without changing the interfaces manually. The case study illustrates that the first variant has a physical vehicle model developed under Matlab/Simulink, while the second variant has a simple game engine-based vehicle model.

4.2 Procedure Model Overview

The procedure model is the most essential part of the *Design Framework for Developing a Reconfigurable Driving Simulator*; it describes the theoretical fundamentals of the design framework. The procedure model supports driving simulator developers in the development of a reconfigurable driving simulator. The procedure model is kept general and could be used for different driving simulator areas of use, as well as other mechatronic systems. It consists of six consequent phases divided into two stages. Figure 4-4

shows the procedure model in the form of a phases/milestones diagram which shows each phase. It also shows the tasks that have to be carried out, as well as the results from each phase.

The six phases of the procedure model are generally divided into two stages: **The system development stage** and **the variants creation stage.** Each stage consists of three phases. The first three development phases have to be performed once by the driving simulator developer. As soon as the developer finishes the development phases, the driving simulator operator should carry out the variant creation phases each time he/she creates a driving simulator variant.

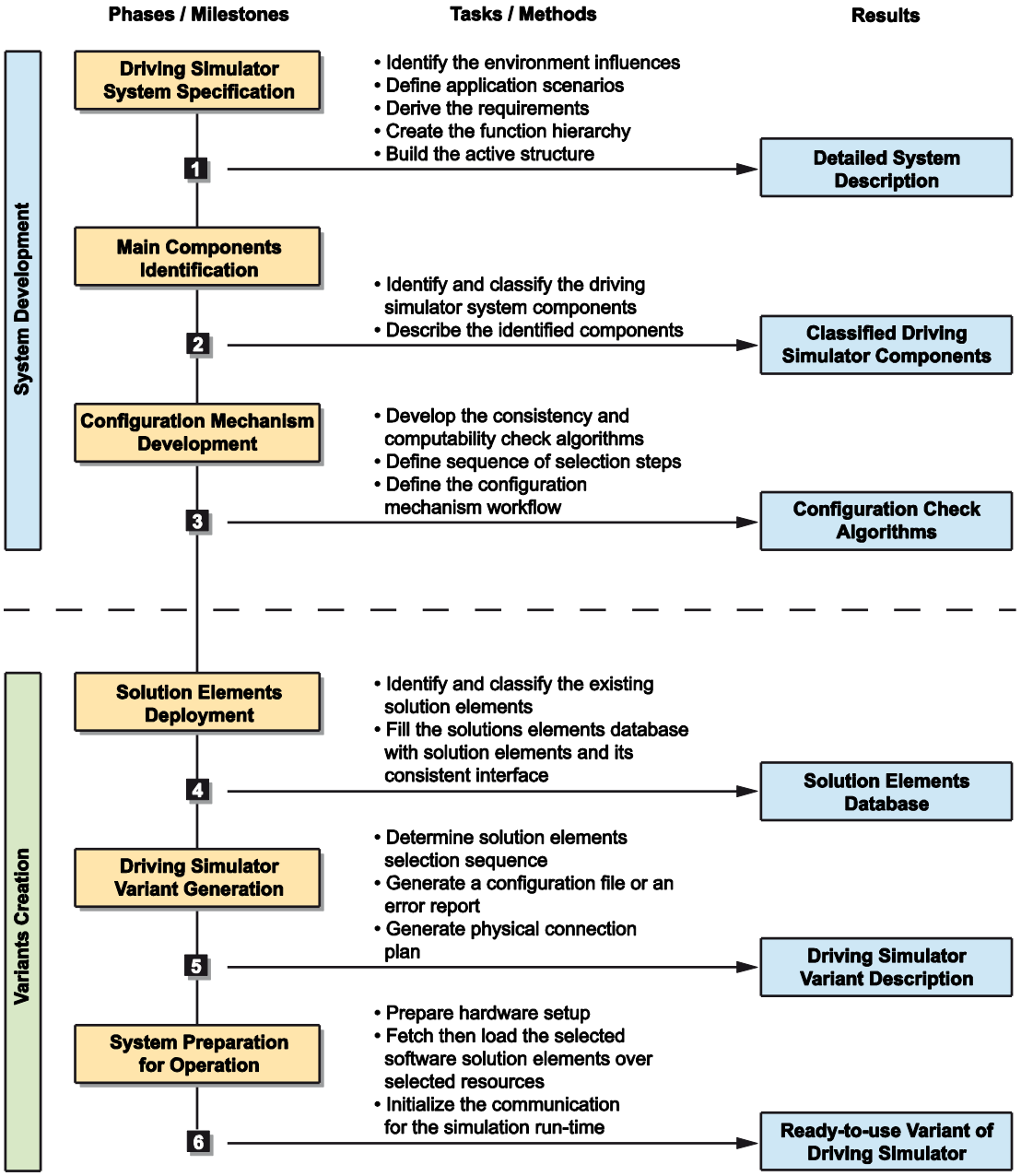


Figure 4-4: Procedure model for developing a reconfigurable driving simulator.

The first phase is the **driving simulator system specification;** in this phase, the driving simulator is considered as an advanced mechatronic system. Therefore, a powerful specification technique for mechatronic systems is needed. During this phase, the specification of the driving simulator is carried out in the form of partial models. The first phase results in detailed driving simulator specifications, which are the input of the second phase. The second phase is the **main components identification;** in this phase the main components of the driving simulator are identified and classified into software components, hardware components, as well as resources. Furthermore, the identified components have to be described. The second phase results in the classified driving simulator components and a description for each of them. The third phase is the **configuration mechanism development;** in this phase, the logical relationships between the diverse components have to be investigated and a configuration mechanism is developed. This mechanism is responsible for checking whether the combination of the selected solution elements is consistent and compatible or not. The third phase results in the consistency and compatibility check algorithms. The system development stage results in the reconfigurable driving simulator outlines, which will be used by the driving simulator operator in the variants creation stage.

As soon as the reconfigurable driving simulator outlines are developed, the operator can configure his own system with the help of the next three phases. The fourth phase is the **solution elements deployment**, in which the driving simulator operator has to register the existing solution elements in a solution elements database. The fourth phase results in the solution elements database, which contains all the solution elements organized in the form of morphological boxes. The fifth phase is **a driving simulator variant generation** that is done by selecting a combination of the available solution elements. After the selection process is completed, the configuration mechanism checks the constancy and the compatibility of the selected combination. If the selected combination is consistent and its entire solution elements are compatible with each other, a variant description file and a physical connections plan will be generated; they are the fifth phase results. Based on the variant description file and the physical connections plan, the system has to be prepared for operation in the sixth phase: the **system preparation for operation.** The preparation for the hardware components is done by connecting the hardware solution elements based on the physical connections plan. However, the preparation for the software solution elements will be done automatically based on the generated variant description file. This is done by fetching and loading the selected software solution elements on the selected resources and by initializing the communication between them. After the completion of the sixth phase, the second stage is also completed and the result is a driving simulator variant.

In the following sections, a detailed description of all needed tasks and operations during each phase, as well as the results of each phase, will be presented with the help of the case study variants.

4.3 Phase 1 – Driving Simulator System Specification

The objective of the first phase is to specify a reconfigurable driving simulator, which is a complex multidisciplinary mechatronics system. Therefore, there is a need to specify the system under a multidisciplinary development with the help of a specification technique.

As described previously in the state of the art section 2.3.2, the CONSENS – "*Conceptual Design Specification Technique for the Engineering of Complex Systems"* will be used during this work. CONSENS is developed in order to specify complex mechatronic systems. The specifications are multidisciplinary and they simplify the complexity of the developed mechatronic system by describing it using a coherent system of partial models. CONSENS is generally used in the conceptual design of a new product (mechatronic system) in early development phases [GFD+09].

Driving simulators have been designed, developed and have been in use since 1934 as stated previously in section 2.2.1 [MV34]. This means they are not new systems but have been used for decades. The development and the continued enhancement of driving simulators allow building a wide expertise regarding the system, as well as its components and its architecture. Since CONSENS is used regularly in the conceptual design of a new product, some enhancements are needed in order to be used for the development of driving simulators.

The usage of CONSENS in specifying a well-known system such as driving simulator has to be validated with the help of a usability study. The usability study[7] is carried out by specifying an existing driving simulator (HNI ATMOS driving simulator, which is described in section 3.3.1) in a retrospective way. By using this reverse engineering approach in the usability study, not only is CONSENS validated for the usage in the development of reconfigurable driving simulators, but also the needed CONSENS enhancements are carried out. Additionally, the relevant partial models of CONSENS were selected and organized in a work flow.

In the following section, the CONSENS work flow, especially for the specification of a reconfigurable driving simulator, is described.

4.3.1 CONSENS Work Flow for a Reconfigurable Driving Simulator

The specification technique "CONSENS" divides the principle solution specification into coherent partial models. The CONSENS partial models are: requirements, environment, application scenarios, functions, active structure, shape and behaviour. Each partial model specifies a precise aspect of the system under development [GFD+09].

[7] This validation study is based on a bachelor thesis supervised by myself: "Reverse Engineering eines komplexen Fahrsimulationssystems mit dem Ziel der fachdisziplinübergreifenden Systemkonzeption" by B.Sc. Alexander Birkle.

The usability study showed that the partial models' weights of importance are not equal within the development of reconfigurable driving simulators. During this work, the focus will be on five of seven CONSENS partial models. The relevant partial models are environment, application scenarios, requirements, functions and active structure. The shape and behaviour partial models will be neglected within the scope of this work.

The CONSENS work flow is divided into three steps: firstly, the environment, the application scenarios and the requirements have to be specified simultaneously. Secondly, based on the result of the first step, the function hierarchy has to be derived. The third step is to build up the active structure based on the result of the previous steps. Figure 4-5 below shows the CONSENS work flow towards specifying a reconfigurable driving simulator.

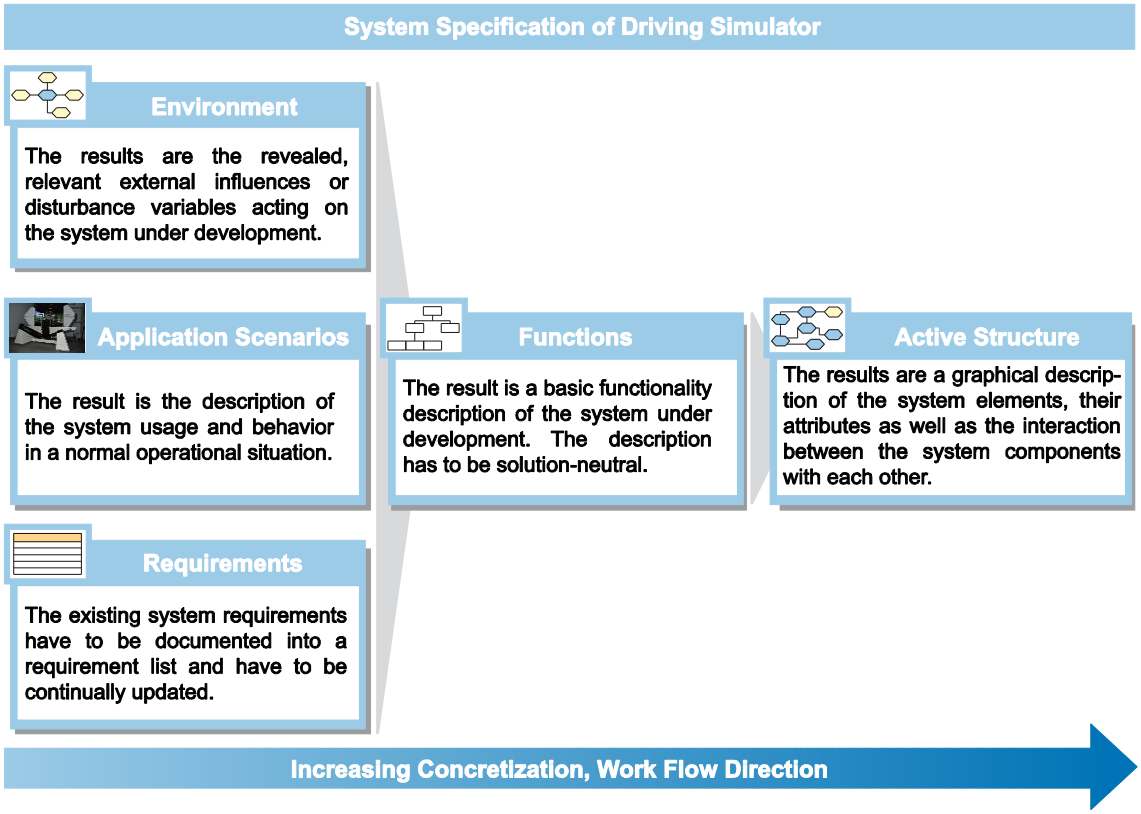


Figure 4-5: CONSENS work flow for reconfigurable driving simulator according to Gausemeier [VG12, p. 2].

The specification of the system is typically carried out in the context of expert workshops with the help of a workshop cards set. The workshops' participants are usually experts in several disciplines such as mechanical engineering, software engineering, control engineering and electrical engineering. The result of each partial model is presented in the next sections.

Important evidence: During the driving simulators specification, the entire components of the driving simulator have to be considered solution-neutral. Dealing with the different components from a specified solution point of view helps to develop a recon-

figurable driving simulator. The system has to be specified as solution-neutral during the development stage, and then during the variant creation, a solution element has to be selected to fulfil each system component function.

4.3.2 Environment

The environment partial model defines the external influences, which affect the system under development. The driving simulator has to be considered as a black box which means that the investigation is not of the system itself, but of the relevant external influences. These external influences are environment elements or disturbance variables [GFD+09].

Environment – specification results of the case study

The environment influences specifications of the driving simulator case study result in the identification of five essential environment elements as well as three disturbing influences. **The five identified essential environment elements are**:

- **Driver:** The most essential environment element is the driver. The driver uses the input devices to drive a virtual vehicle in a virtual environment. The input signals are typically as follows: acceleration pedal position, brake pedal position, gear selector position and steering wheel angle. The driver receives feedback from the simulator in the form of motion as well as visual and acoustic information.
- **Energy Source:** To power up the system, an energy source is needed.
- **Ground:** If the driving simulator is equipped with a motion platform which produces dynamic forces, then bidirectional forces interactions occur between the system and its ground.
- **Driving simulator operator:** The driving simulator operator is the person who is responsible for operating the driving simulator technically.
- **Environment:** The environment affects the driving simulator through disturbing influences such as humidity, dirt, light and temperature. The system also affects the surrounding environment by producing heat and operational noise. For the modelling of influences and in particular, the influence of disturbances, catalogues such as [VDI4005] can be used.

Figure 4-6 shows the system under development illustrated as a blue hexagon in the centre of the figure. The five environment elements illustrated as yellow hexagons, and all the interaction flows between the system and its environment are illustrated as arrows. These interaction flows and the disturbance influence are restricted between energy, information, and material flows.

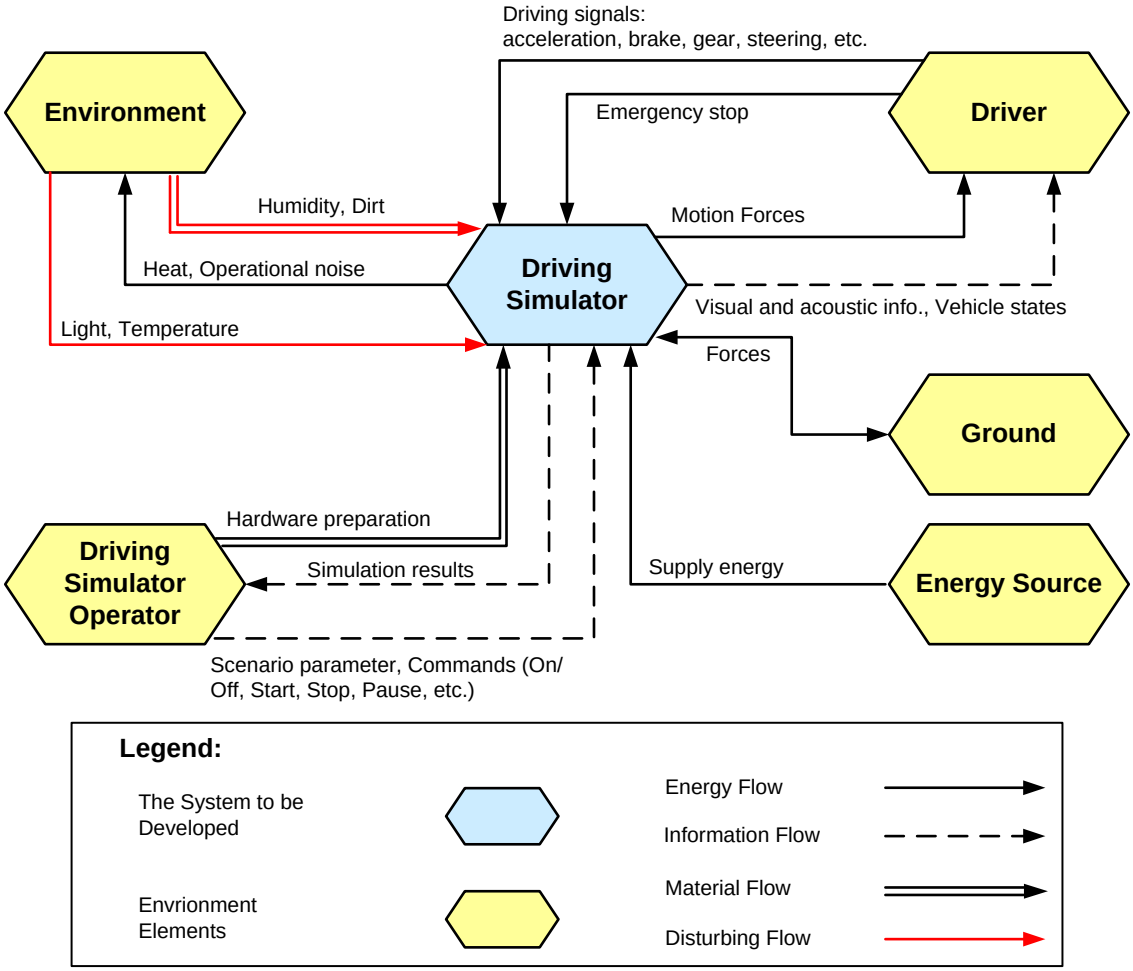


Figure 4-6: Environment model of the case study's variant 1.

The environment's specification result of variant 2 of the case study is illustrated in appendix, Figure A-1. The difference between the environment models for variant 1 and variant 2 is minimal. The difference is that the ground element and the motion energy flow have to be neglected for variant 2. That is because variant 2 does not have a motion platform.

4.3.3 Application Scenarios

The application scenarios partial model is an essential partial model of the system specification. In this specification step, some operational application scenarios are defined. Each application scenario describes the system under development in terms of way of use, operation modes, system manner and main components. By using CONSENS, each application scenario will be described in a profile page, which contains the scenario title, scenario numbering, the scenario description and a simple sketch [GFD+09].

Application scenarios – specification results of the case study

The specification of the case study results in the definition of two application scenarios: "*virtual drive with a motion platform"* and "*virtual drive on a PC"*.

Figure 4-7 shows the first application scenario virtual drive with a motion platform regarding the case study's variant 1. The application scenario is illustrated in a profile page[8], which is adapted to the reconfigurable driving simulator approaches. It contains a short description of the system's normal operation, the desired setup in the form of solution-neutral hardware and software components, as well as a simple sketch.

Status: 1.12.2013	**Application Scenario:** Virtual Drive with Motion Platform	A1	Page: 1

Description:

This is a virtual test drive in a driving simulator. The driver sits in the motion platform. The motion platform has to be equipped with an input device, which has a steering wheel and three pedals (acceleration, brake and clutch pedals). This input device allows the driver to drive and control a virtual vehicle in a virtual environment. As soon as the simulation starts, based on the driver inputs through the input device, the vehicle model calculates the vehicle movements in the virtual environment. During each sampling cycle, the vehicle model updates the position and orientation of the vehicle chassis and calculates the engine speed. Based on the new position and orientation calculated by the vehicle model, the motion platform controller calculates the new set-points for the motion platform. The rendering software visualizes the virtual environment based on the new vehicle position and orientation perspective and displays the rendered frame on a display device. The acoustic software calculates the engine sound based on the engine speed and generates tone by the acoustic device.

Setup description:

Hardware	Software
Motion Platform	Motion Platform controller
Input Device	Vehicle Model
Visualization Device	Visualization Rendering Software
Acoustic System	Acoustic Software

Sketch:

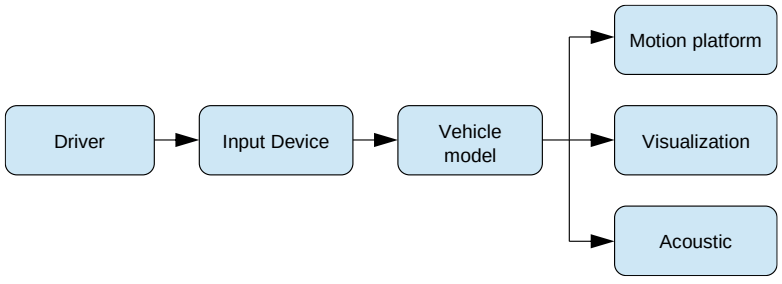


Figure 4-7: Application scenario example for the case study's variant 1.

[8] The profile page of the application scenario used during this work was enhanced and varies from the standard CONSENS profile page in order to fit with the reconfigurable driving simulator specification.

The application scenarios' specification result of variant 2 of the case study is illustrated in Appendix, Figure A-2.

4.3.4 Requirements

This partial model collects and organizes the system requirements of the system under development which need to be covered and implemented during the development process. The requirement list contains functional and non-functional requirements [GFD+09]. Additionally the organized requirements distinguish between demands and wishes (D/W) [PBF +07].

Requirements – specification results of the case study

Figure 4-8 shows a part of the requirement list of the case study specification result.

		Requirements List „Driving Simulator Variant 1“	10.12.2013 Page:1
No.	**D/W**	**Requirements**	**Changes**
1		**Ergonomics and Security**	
1.1	D	The system has to be safe for humans and the environment (in accordance with DIN EN ISO 12100)	
1.2	D	Maximum Surface temperature of accessible parts 50 ° C	
1.3	D	An accessible emergency stop switch for the driver	
1.4	D	No system damage in the case of system failure	
1.5	W	Easy startup procedure	
2		**Visualization System**	
2.1	D	The rendering frequency has to be 60 Hz or more	
2.2	D	Visualization of vehicle, road and road environment objects	
2.3			

Figure 4-8: Part of the requirements list of the case study.

4.3.5 Functions

The functions partial model is built based on the previous partial models: environment, application scenarios and requirements. It describes the system and its entire components' functionality in a top-down hierarchy [GFD+09]. Each block describes a sub-function of the system. Function catalogues, according to BIRKHOFER [Bir80] or LANGLOTZ [Lan00], support the creation of the functional hierarchy.

Due to the variation of the main function, structure and required components of the stated application scenarios, the functions specification also varies in its complexity and number of its entire sub functions. Therefore, there is a need to merge the identified functions of the stated application scenarios.

Functions – specification results of the case study

The main function of the case study – variant 1 driving simulator is to *perform* a test drive. In order to achieve this function, the driving simulator has to *simulate* motion, *visualize* virtual scenes, *simulate* sound, *simulate* a virtual vehicle and *drive the virtual vehicle* through a virtual environment. Figure 4-9 shows the functions hierarchy of the first variant.

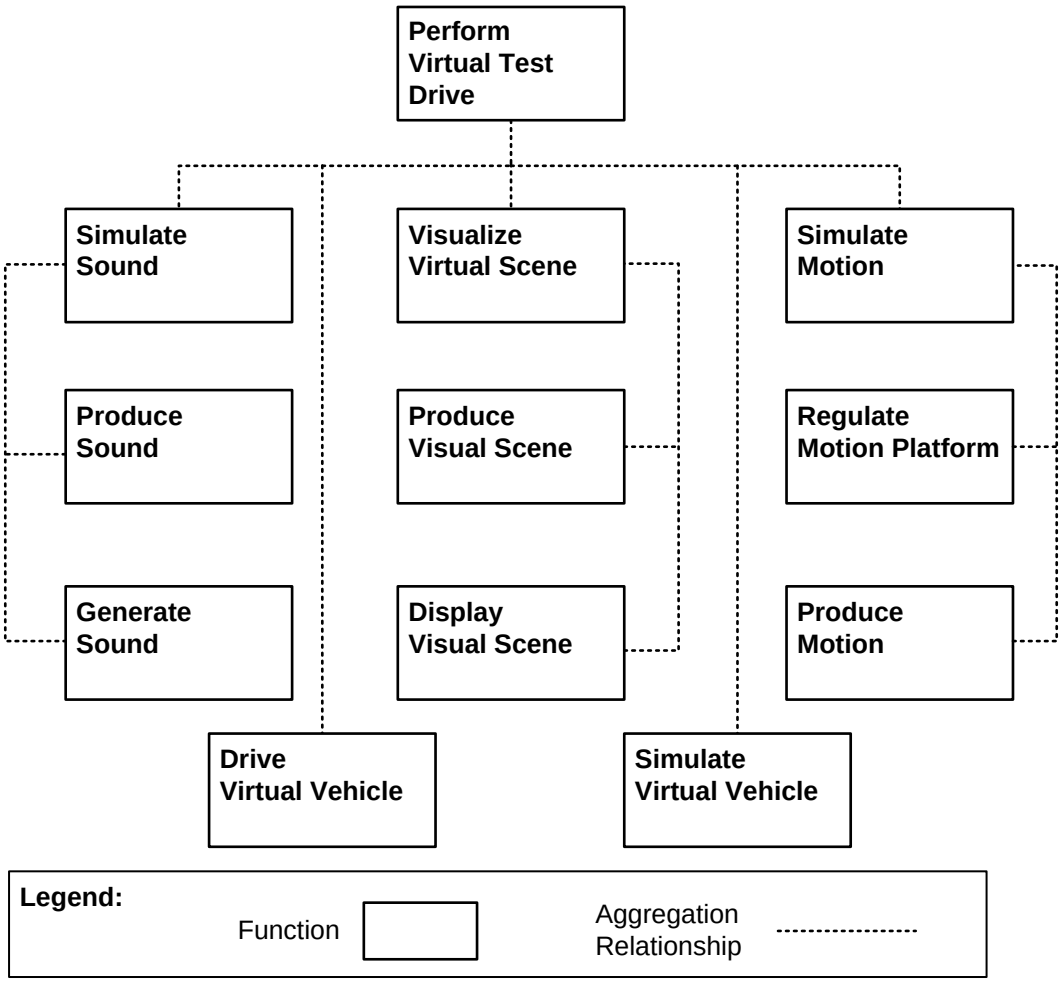


Figure 4-9: Functions model of the case study variant 1.

The functions' specification result of variant 2 of the case study is illustrated in Appendix, Figure A-3. The difference between the function models of variants 1 and 2 is minimal. The difference is that the "simulate motion" function and its sub-functions have to be neglected for variant 2. This is because variant 2 does not have a motion platform.

4.3.6 Active Structure

The active structure partial model is built based on the previous partial models results, specifically the functions partial model. The active structure describes the entire system in more details in the form of system component active principles. It describes the sys-

tem components, their attributes, the entire interfaces and how the components interact with each other. Depending on the modelling level of details, each system element could be described abstractly as an active principle or a software pattern. Additionally, material, energy and information flows, as well as logical relationships, describe the interactions between the system elements [GFD+09].

Active Structure – specification results of the case study

Figure 4-10 shows the active structure specification results for the case study variant 1. The active structure consists of eight system elements (components): five of them are software components labelled with (SW), and three hardware components labelled with (HW). Moreover, one of the environment elements (Driver) illustrates an example of the interaction between the entire components of the system and an environment element. Six of the eight components could be grouped into 4 groups e.g., rendering software and the visualization device (hardware) compose the visualization system group.

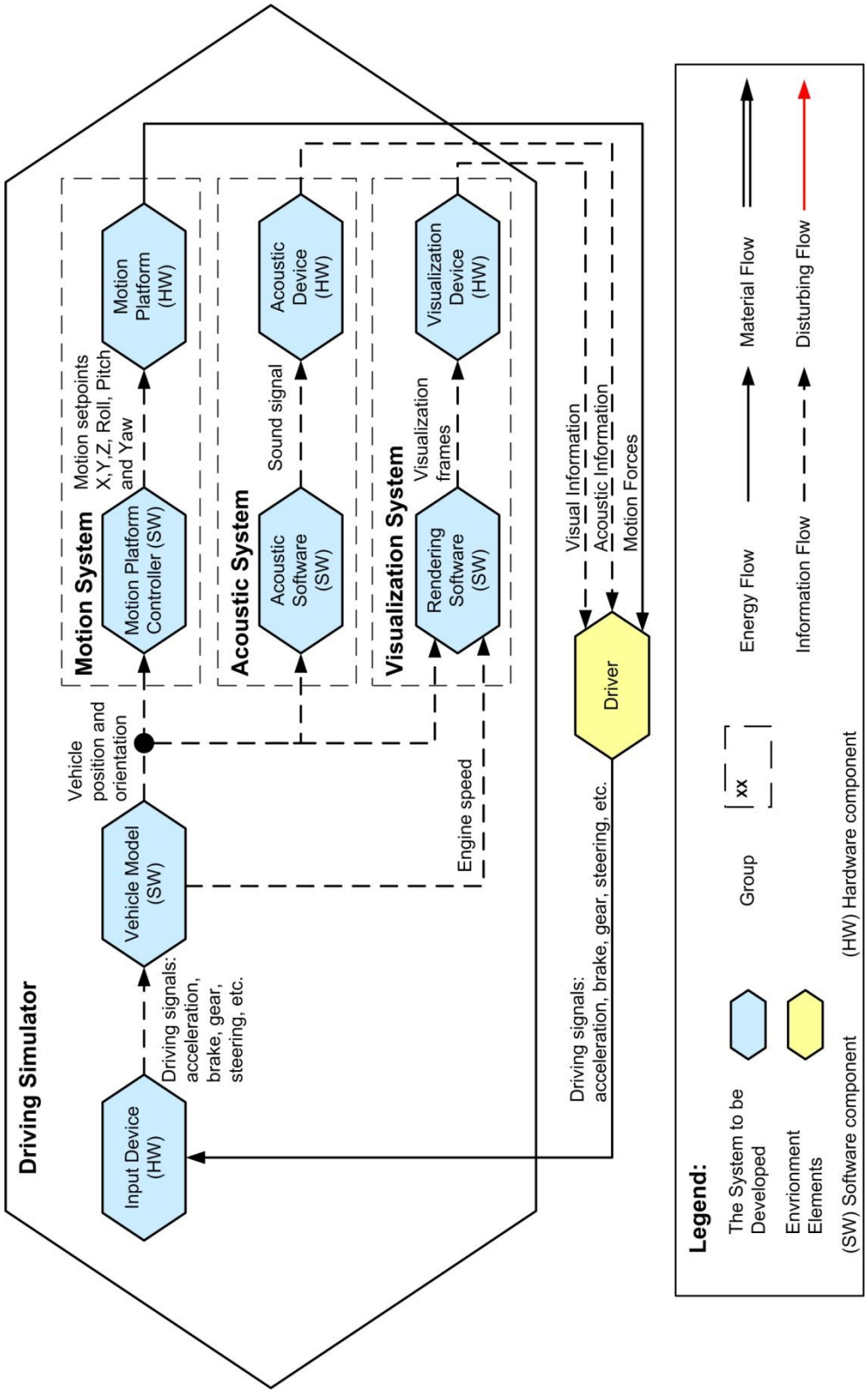


Figure 4-10: Active Structure model of the case study's variant 1.

The active structure's specification result of variant 2 of the case study is illustrated in Appendix – Figure A-4.

4.3.7 CONSENS Enhancement for the Design Framework

During the validation study and the driving simulator system specification of the case study variants, some enhancement of the CONSENS was carried out to fit the design framework for reconfigurable driving simulators. The following rules and enhancements are suggestions which have to be considered for the specification of reconfigurable systems in general.

- **Functions decomposition principle**: This enhancement refers to the CONSENS functions partial model. The functions partial model is carried out based on "the functional decomposition principle" according to SYSTEM ENGINEERING FUNDAMENTAL [DDS01].

  In order to simplify a complex system, it has to be decomposed into a set of subsystems. The common question during the system decomposition is "To which level should the system be decomposed concerning the reconfigurability?" The answer is: The system has to be decomposed as little as possible and as much as necessary. It is a compromise between the system structure complexity and the system reconfigurability. Figure 4-11 shows two examples for decomposing the driving simulator functions with the focus on the vehicle model. The first case on the left is a one-level decomposition, which keeps the system structure and the interface's topology simple, but restricts the system reconfigurability. In this case, only the whole vehicle model would be exchangeable and not any of its entire components. The second case on the right is a two-level decomposition which the vehicle model could be decomposed into (engine model, drive train model, vehicle dynamics model, etc.). This decomposition makes the system structure and the interface's topology more complex, but extends the system reconfigurability. In this case, all the entire models (engine model, drive train model, vehicle dynamics model, etc.) would be exchangeable.

  In fact, each driving simulator developer has to decide the level of decomposition depending on the area of use. For example, if the driving simulator is used for the testing of a new light assistance system, the vehicle model should not be decomposed as in the first case. On the other hand, if the driving simulator is used for testing a new vehicle dynamics system, the vehicle model has to be decomposed as the second case.

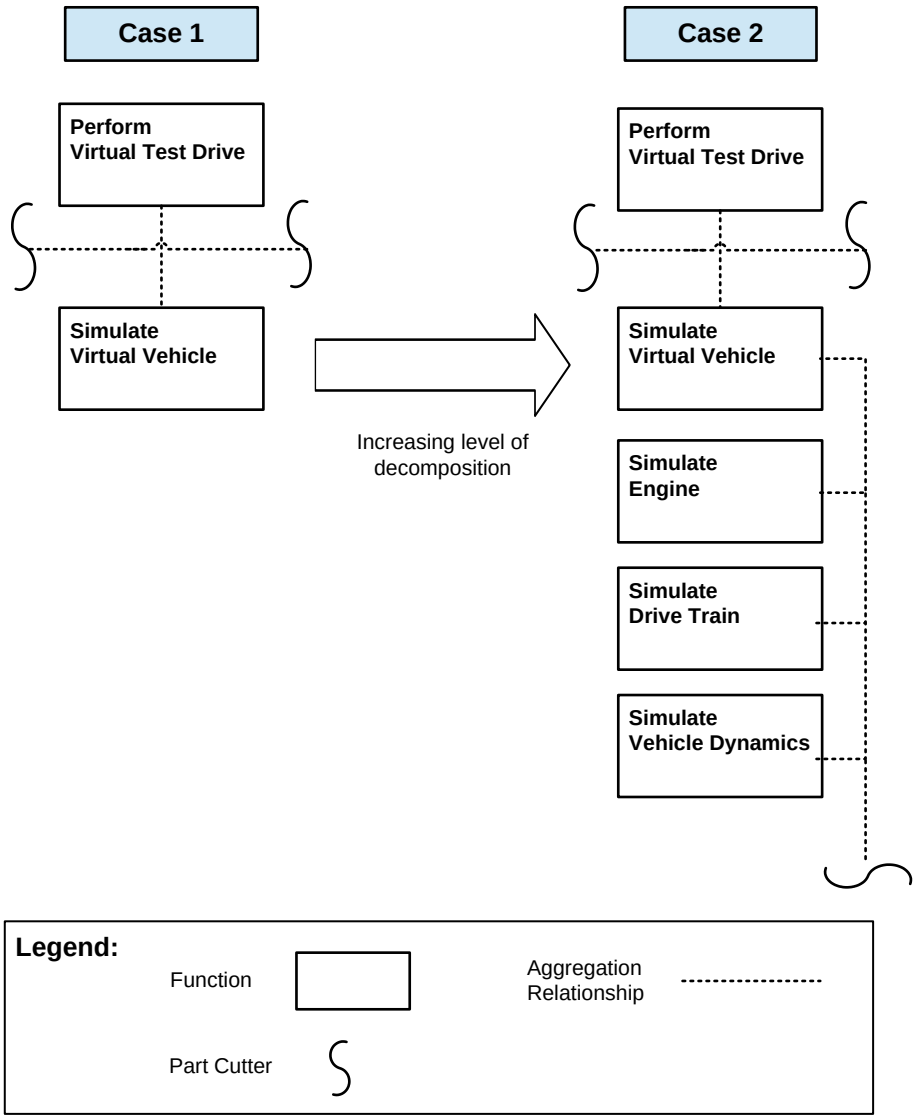


Figure 4-11: Different cases of function decomposition.

- **Intelligent Interfacing Module (IIM)**: This enhancement refers to the CONSENS active structure partial model. The main objective of this work is to develop a reconfigurable driving simulator by selecting the desired structure (system's components) and the desired hardware and software solution elements, and to avoid the implementation of the system interface topology manually. Therefore, there is a need for an interfacing component, which is named here as the Intelligent Interfacing Module (IIM). This module is able to read each generated variant configuration's description, and based on this description, it will interface all the system software components together during simulation run-time. The IIM is the interfacing heart of the system. Therefore, it must be considered as one additional system component, and it must be modelled in each variant's active structure. Figure 4-12 shows the active structure results for the case study variant 1, which was previously shown in Figure 4-10, but with the IIM as a new system component. The comparison between the active structure without the

IIM and the active structure with the IIM shows an additional advantage of using the IIM, which is making the system interfaces topology more understandable. The interface of each software component will only be with the IIM. It would be very helpful during the modelling of more complex driving simulators.

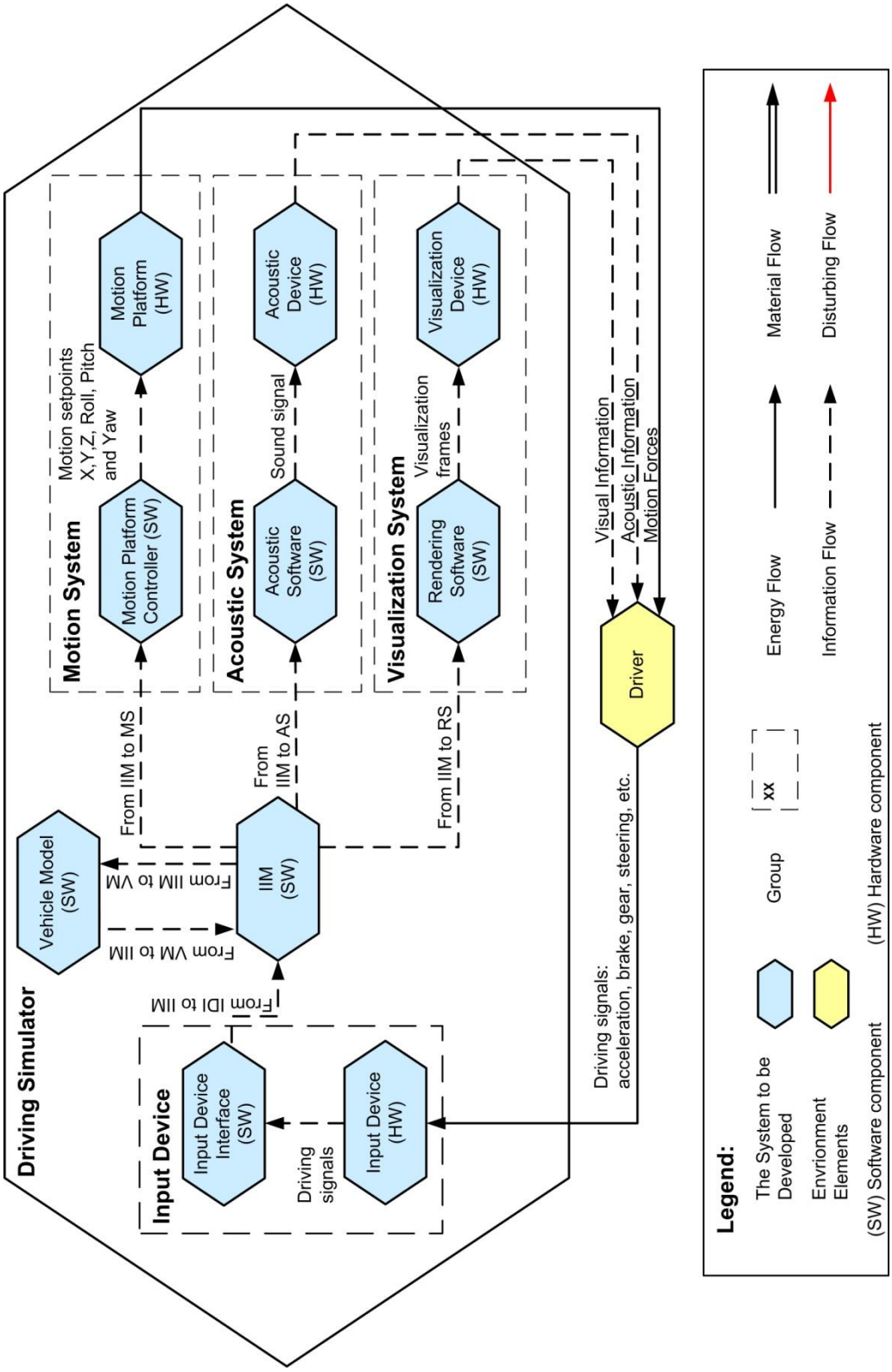
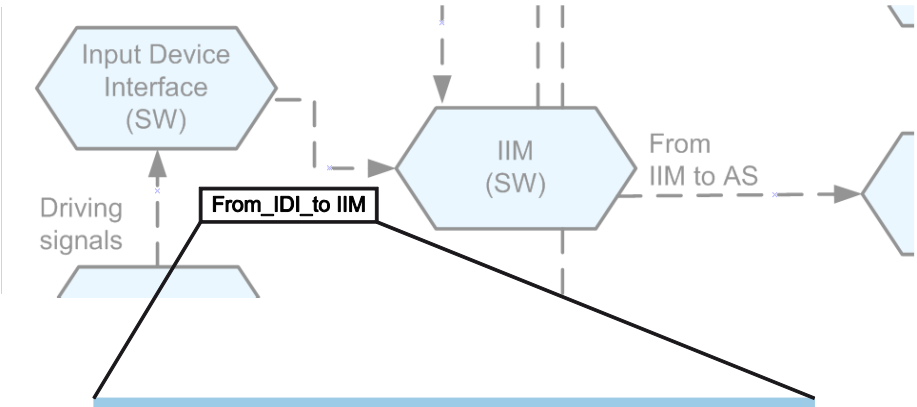


Figure 4-12: Active Structure model of the case study's variant1 with using IIM.

- **Hardware Interfacing**: This enhancement refers to the CONSENS active structure partial model. Based on the IIM concept, this interfaces all software components with each other. Hardware components could not be connected to the IIM directly. Therefore, each hardware component has to have its own software interface; a physical connection will only be available between the hardware component and the resource interface where its software interface component executes on. The information exchange between the hardware and the IIM will be done through this interface component. The comparison between the active structure without the IIM in Figure 4-10 and the active structure with the IIM in Figure 4-12 shows an additional software component, which is the input device interface. The input device interface converts the physical signals coming from the input device hardware to an information signal and forwards it to IIM. In this way, the input device hardware and IIM are connected.

- **Buses of information flows**: This enhancement refers to the CONSENS active structure partial model. In order to detail the information flows between the diverse system components, signal buses will be used. The signal bus contains one or more signals. Each signal has the following attributes: signal name, unit and a description. Therefore, each bus of information signal should be described by a signal table. Figure 4-13 shows an example of the information signal bus table.



From_IDI_to IIM
from (Input Device Interface) to (Intelligent Interfacing Module)

Signal Name	Unit	Description
Acceleration_Pedal	%	The accelerator pedal raw data.
Brake_Pedal	%	The brake pedal raw data.
Clutch_Pedal	%	The clutch pedal raw data.
SteeringWheel_Angle	deg	The steering wheel angle raw data.

Figure 4-13: Example of an information signal bus table.

The first phase results, which the driving simulator system specification describes in the form of five partial models, are: environment, application scenarios, requirements, functions and active structure. This result is the input for the second phase.

4.4 Phase 2 – System Components Identification

The second phase objectives are the identification, classification and definition of the driving simulator components based on the results of the first phase. Towards the identification of the driving simulator system components, a distinction between optional components, key components and solution elements must be defined.

Distinction between system optional components, system key components and solution elements

Each driving simulator system consists of some entire subsystems, which are hereafter called an **optional component** in this work. It describes an entire subsystem task in a solution-neutral way by means of its function. The existence of some system components is obligatory in order to build a usable driving simulator. These obligatory components are hereafter called **system key component**. In order to create an applicable driving simulator variant, each system component has to be replaced with a preferred **solution element**, which is a specific solution that could fulfil the component function.

The case study variants show examples of optional components, system key components and solution elements. A system optional component is e.g. a vehicle model, while a solution element for this component could be a simple vehicle model or a vehicle model from a certain provider. Likewise, with the classes/objects concept of object-oriented programming, here a component corresponds to a class and a solution element corresponds to an object. Each component could be replaced by one solution element in order to fulfil its functionality.

As the driving simulator structure could also be changed during the reconfiguration process, the key components have to be identified. The key components are the obligatory system components that always have to exist in the simulator structure. For example, each driving simulator has to have a visualization rendering software but a motion platform is an optional component and not a key component, because a driving simulator does not need to have a motion platform.

4.4.1 Identification of Driving Simulator Components

Based on the active structure partial model, the system components as well as the system key components can be identified with the help of the following three operations:

1. **Identify all components**: The reconfigurable driving simulator components are the union of the different variants components as follows:

   $$Sim\_Comp = \text{var}\_1\_comp \cup \text{var}\_2\_comp \cup ... \text{var}\_n\_comp$$

   Equation 1: Reconfigurable driving simulator components

Where:

Sim_Comp: Reconfigurable driving simulator components

Var_1_comp: Variant 1 components

Var_2_comp: Variant 2 components

n: Number of modelled variants

For example in the case study, if variant 1 components are {A,B,C} and variant 2 components are {A,B,D,E}, the reconfigurable driving simulator components will be {A,B,C,D,E}.

2. **Identify common components**: The common components of the reconfigurable driving simulator are defined based on the intersection between the different variants components as follows:

$$Sim\_Comp = \text{var}\_1\_comp \cap \text{var}\_2\_comp \cap ... \text{var}\_n\_comp$$

Equation 2: Driving simulator common components

For example in the case study, if variant 1 components are {A,B,C} and variant 2 components are {A,B,D,E}, the common system components will be {A,B}.

3. **Identify key components**: In order to identify the system's key components, the selection will be done based on the common components set. Each component has to be investigated individually in a logical way by eliminating the component from the set. If the driving simulator can be operated without this component, this means that it is an optional component. But if the driving simulator cannot be operated, then this means that it is a key component.

Figure 4-14 shows the identified components of the case study variants based on their active structure partial model.

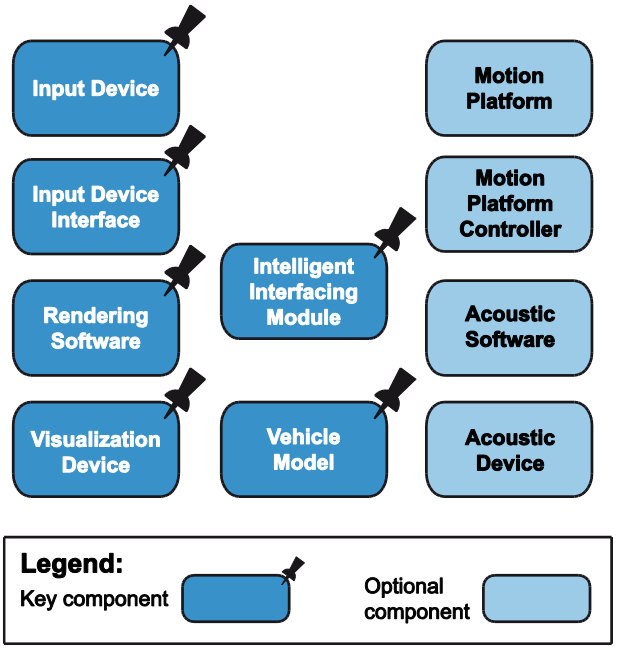


Figure 4-14: Identified optional and key components.

4.4.2 Classification of the Identified Components

In addition to the modelled software and hardware components, the reconfigurable driving simulator resources have to be taken into consideration. Each software or model needs a computing unit (e.g. a computer) to be executed on. Moreover, each hardware component needs a physical interface to communicate with its corresponding software interface.

In order to organize the identified components easily, these have to be classified under the following three categories: **hardware**, **software** and **resources**. The software category contains two subcategories: the **applications/models** and the **hardware interfaces** (previously described in section 4.3.7). The resources category contains two subcategories: the **computing units** and the **signal processing interfaces**. Figure 4-15 shows the classification of the identified components in the case study.

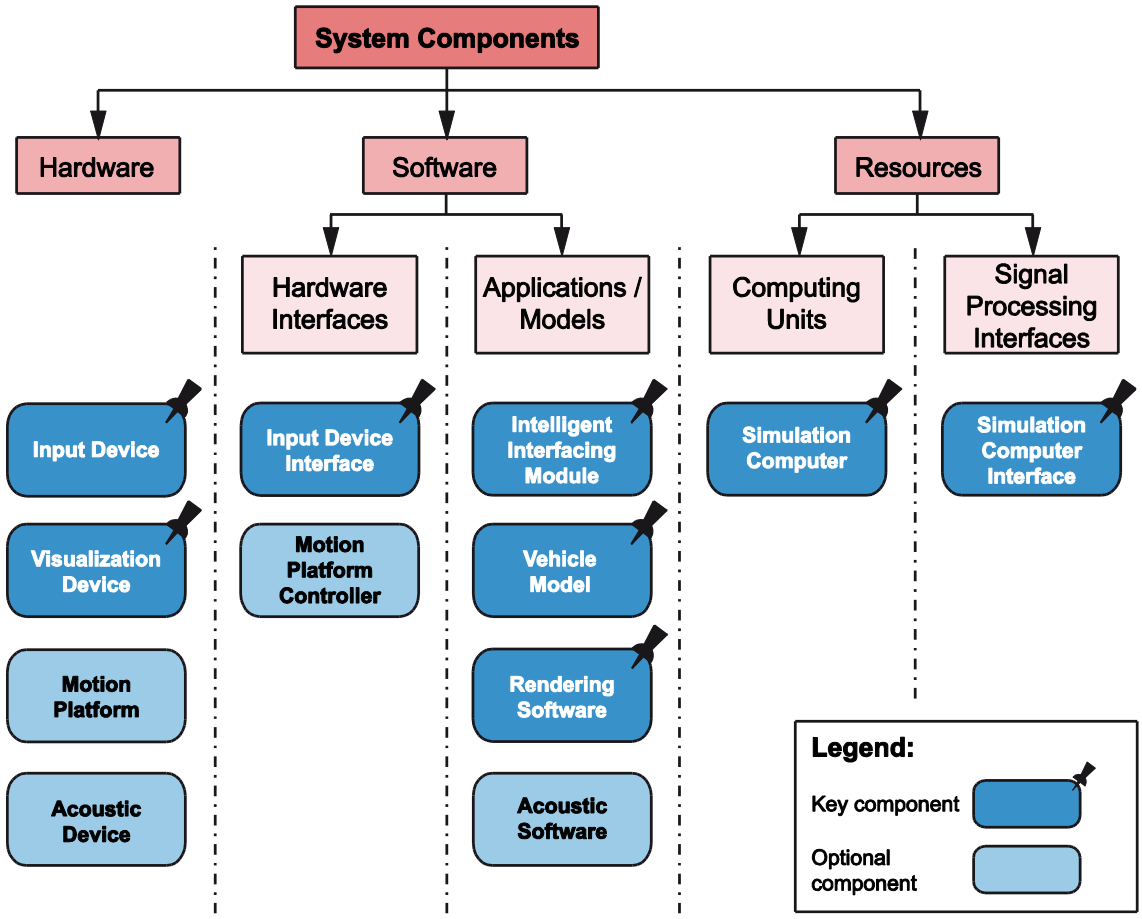


Figure 4-15: Classification of the identified components.

4.4.3 Description of the Identified Components

In order to understand the function of each component, each component has to be defined from a solution-neutral point of view. The identified components of the used case study variants are described as follows:

Input Device: This is a hardware MMI (Man-Machine Interface) between the driver and the driving simulator. It provides driving signals, e.g. acceleration pedal position, brake pedal position, etc. The input device provides the driving simulator with these signals in energy flow form.

Input Device Interface: This is a software component, which converts the energy flows (physical signals) of the input device to its computer representative information flows (digital signals).

Intelligent Interfacing Module (IIM): The Intelligent Interfacing Module (IIM) is a software component which is able to read each generated configuration description, and based on this description, it interfaces all system software components during simulation run-time.

Vehicle Model: This is a software component, which represents a vehicle and its entire components by sets of mathematical equations in order to simulate the vehicle driving behaviour.

Rendering Software: This is a software component, which generates a computer graphics representation of the virtual scene. The scene is represented typically from the driver's point of view.

Visualization Device: This is a hardware device, which displays the virtual scene.

Motion Platform: This is a hardware component based on an active mechanism which gives illusive haptic feelings of being in motion.

Motion Platform Controller: This is a software component, which synthesizes a controller for actuators of the motion platform.

Acoustic Software: This is a software component, which generates a computer generated sound of the virtual scene.

Acoustic Device: This is a hardware device, which generates the virtual simulated sounds.

Simulation Computer: This is a resource component, which processes the software components of the system.

Simulation Computer Interface: This is a resource component, which interfaces hardware components with the simulation computer by converting the energy flows (physical signals) to their respective information flows (digital signals).

The second phase results in identifying and classifying the driving simulator system components, as well as describing each component.

4.5 Phase 3 – Configuration Mechanism Development

This is the third and last phase of the development stage. The objective of the third phase is to develop a configuration mechanism, which ensures that the selected solution elements could operate together. This check is done after selecting the preferred structure and the desired solution elements. The configuration mechanism has to ensure the consistency and the compatibility of the selected structure and its entire solution elements. After the configuration mechanism ensures the selected solution element consistency and compatibility of the solution elements, it generates a configuration file. The configuration file contains a list of the selected solution elements, the interfaces' topology and the selected resources.

The configuration mechanism checks the selected solution elements. However, the solution elements will be deployed in the next phase, but it is the preferred order of the procedure. Developing the configuration mechanism before deploying the solution ele-

ments allows the mechanism to also deal with unknown solution elements, which can be added in the future.

There are two types of relationships between the selected solution elements and each other. These relationships have to be checked and confirmed by the configuration mechanism. The first relationship is the logic consistency between the selected solution elements with each other. The second relationship is the compatibility between the interfaces of the selected solution elements.

4.5.1 Consistency Check Algorithm

The consistency relationship can be determined by two levels. The first level is the **logic dependency** between components, which determines if there is a logic correlation between two components or not. The second level is the **logic consistency** between two solution elements.

Logic dependency between two components

It is a logic relationship between two components, which describes if they depend on each other logically or not. For example, the motion platform and the input device are a dependent pair of components. They depend on each other, i.e. an input device has to be mounted on a motion platform. Therefore, the motion platform dimensions and payload have to match with the selected input device.

Dependency matrix

The dependency matrix is a two-dimensional matrix which describes the logic dependency between the identified components. The components are stated in both the first row and the first column; the matrix is mirrored along its diagonal. Therefore, only the lower half of the matrix has to be filled with 0 or 1 by the **driving simulator developer**.

- **0:** means the components pair is logically independent of each other, thus the inherited solution elements belonging to these components will also be logically independent of each other.
- **1:** means the components pair is logically dependent on each other, thus the inherited solution elements belonging to these components will also be logically dependent on each other.

Table 4-3 shows the dependency matrix based on the case study’s identified components.

Table 4-3: Dependency matrix of the identified case study components.

Dependency Matrix 0 = Independent pair 1 = Dependent pair		Hardware Components				Software Components					Resources	
		Input Device	Visualization Device	Motion Platform	Acoustic Device	Vehicle Model	Rendering Software	Acoustic Software	Input Device Interface	Motion Platform Controller	Simulation Computer	Simulation Computer Interface
		A	B	C	D	E	F	G	H	I	J	K
Hardware	A. Input Device											
	B. Visualization Device	0										
	C. Motion Platform	1	1									
	D. Acoustic Device	0	0	0								
Software	E. Vehicle Model	0	0	0	0							
	F. Rendering Software	0	1	0	0	0						
	G. Acoustic Software	0	0	0	1	0	0					
	H. Input Device Interface	1	0	0	0	0	0	0				
	I. Motion Platform Controller	0	0	1	0	0	0	0	0			
Reso-urces	J. Simulation Computer	0	0	0	0	1	1	1	1	1		
	K. Simulation Computer Interface	0	0	0	0	0	1	1	1	1	1	

Logic consistency between two solution elements

It is a logic relationship between two solution elements, which describes if they are logically consistent with each other or not. The first relationship depends on whether the solution elements' parent components are independent. This means that the two solution elements inherited the independence and there is no need to check their consistency. Otherwise, if the solution elements' parent components are dependent, this means that the two solution elements inherited the dependency and have to be checked if they are consistent or not.

Consistency matrix

The Consistency matrix is a two-dimensional matrix which describes the logic consistency between the available solution elements. The solution elements are stated in both the first row and the first column. The matrix is mirrored along its diagonal. Therefore, only the lower half of the matrix has to be filled with 0, 1 or 2 by the reconfigurable **driving simulator operator**.

- **0:** means the solution elements pair is logically inconsistent with each other. This means that they could not be selected together in a driving simulator variant.
- **1:** means the parent components pair was originally logically independent of each other, thus the inherited solution elements under those components will also be logically independent of each other. This means that the solution elements do not have to be checked for consistency.

- **2**: means the solution elements pair is logically consistent with each other. This means that they could be selected together in a driving simulator variant.

Table 4-4 shows a part of a consistency matrix based on the result of the case study with the assumption that each component has two solution elements. Dealing with the solution elements in this section will be illustrated in an abstract form e.g. the solution elements will be called (A1, A2, B1, etc.); where A and B are components and A1 is the first solution element for the component A, etc. The whole consistency matrix is documented in Appendix – Table A-1.

Table 4-4: Part of the consistency matrix – example of case study solution elements.

Consistency matrix 0 = Logically Inconsistent 1 = Logically Neutral 2 = Logically Consistent			Hardware Components									
			A. Input Device		B. Visualization Device		C. Motion Platform		D. Acoustic Device		E. Vehicle Model	
			A1	A2	B1	B2	C1	C2	D1	D2	E1	E2
Hardware	A. Input Device	A1										
		A2										
	B. Visualization Device	B1	1	1								
		B2	1	1								
	C. Motion Platform	C1	2	0	2	0						
		C2	0	2	0	2						
	D. Acoustic Device	D1	1	1	1	1	1	1				
		D2	1	1	1	1	1	1				
	E. Vehicle Model	E1	1	1	1	1	1	1	1	1		

The consistency matrix is filled out based on the dependency matrix. If a pair of components is independent (0 value in the dependency matrix), e.g. A and B, their solution elements will inherit this relation (1 value in the consistency matrix). Otherwise, if a pair of components is dependent (1 value in the dependency matrix), e.g. A and C, their solution elements will inherit the dependency relationship and they are either consistent or not (respectively 2 or 0 value in the consistency matrix).

Consistency check sequence

Considering the consistency relationship which is determined by two levels matrices, the consistency check will also be performed by two level checks. Figure 4-16 shows a flowchart of the consistency check. For example, the consistency between solution elements A1 and B2 has to be checked. The first check will be based on the dependency matrix between the two parent components A and B. The second level will be based on the consistency matrix between the solution elements A1 and B2.

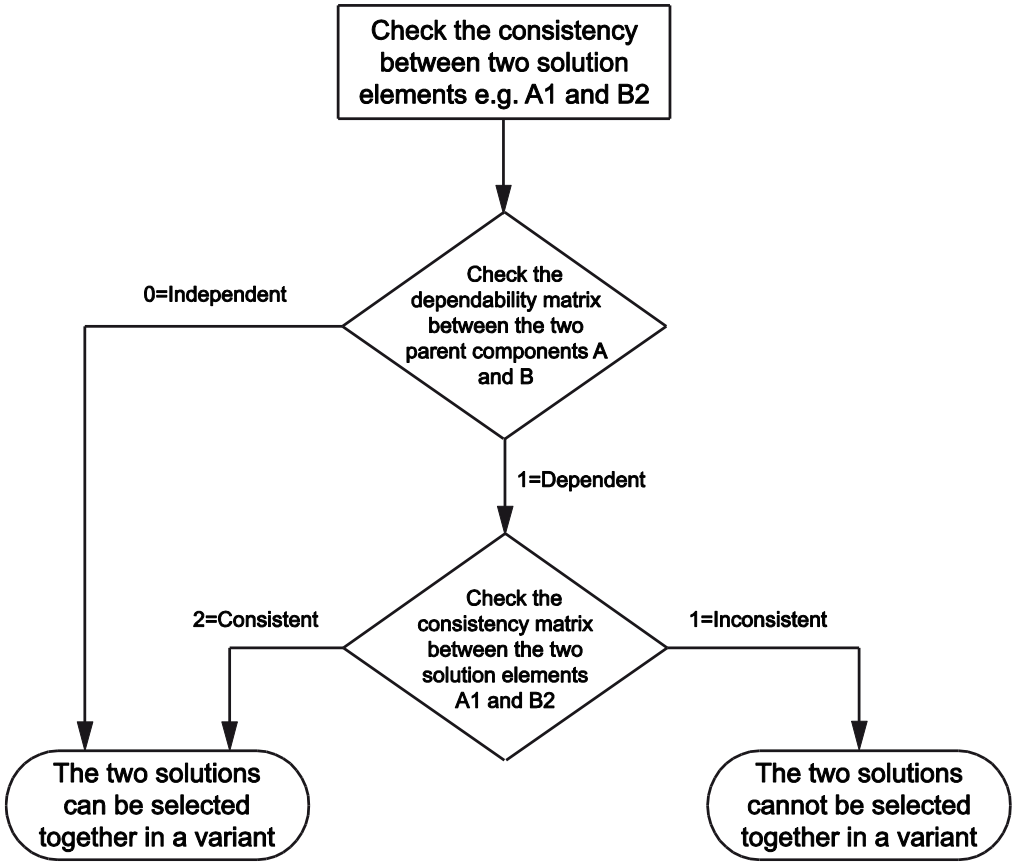


Figure 4-16: Consistency check flowchart.

4.5.2 Compatibility Check Algorithm

One of the main approaches to building a reconfigurable driving simulator is the ability of adding, removing or exchanging one or more solution elements. In order to build such a reconfigurable system, the applications/models interfaces have to be carried out automatically. Therefore, there is a need for an algorithm to check if all selected solution elements are compatible with each other or not. The compatibility here means whether the interfaces of the selected solution elements match together or not. Hence, each software component has its programming language and naming system of the input and output signals. Additionally, there is a need to extend the reconfigurable system continuously by adding new unknown solution elements. Therefore, a generic solution elements' interface concept has been developed to manage and check different existing solution elements as well as unknown solution elements which could be added in the future.

Generic solution elements' interface concept

In order to interface the entire solution elements, each solution element has to be considered as a black box. Mainly, only the input and output interfaces have to be considered. To keep the configuration process flexible and extendable, any solution element can be added as soon as its input and output interfaces are defined. The only required

task for integrating any solution element is to map its inputs and outputs to the reconfigurable driving simulator's unique signal names there, this task is called signal multiplexing.

Figure 4-17 shows an example of the signal multiplexing. A vehicle model has to be integrated as a solution element. The model will be considered as a black box, but all its input and output signals have to be mapped to the reconfigurable driving simulator's unique signal names. The output signal called "Otutput_ID563[m/s]" is the vehicle under test velocity in m/s, but this signal's unique name and unit predefined in the reconfigurable driving simulator has the name "Chassis_Velocity" and its unit is km/h.

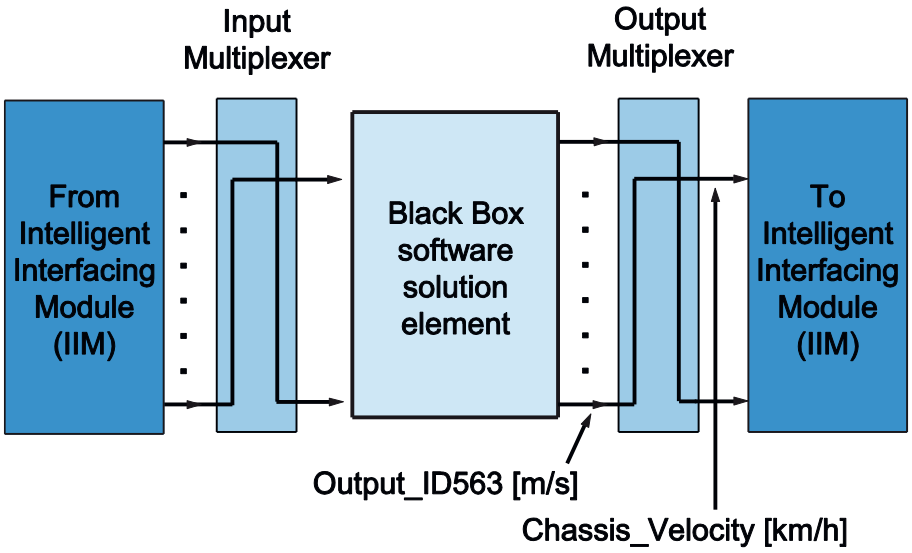


Figure 4-17: Generic solution elements interface concept.

In order to integrate this vehicle model, the user has to connect all the input and output signals with different names and units to the unique names and the units of the parent reconfigurable system. The input and output signals multiplexers should be programmed before registering the solution elements in the solution element database.

Compatibility check algorithm

After selecting the preferred solution elements, the compatibility check algorithm proofs the solution elements one by one to ensure that the input signals could be satisfied from the outputs from other solution elements. The compatibility check algorithm does not only check the signals' name but also other signal attributes such as frequency and unit to ensure the compatibility. Figure 4-18 shows a flowchart of the compatibility check. The compatibility check algorithm checks for each signal the compatibility with the help of the following steps

1. The algorithm checks each input signal of each selected solution element.
2. Each input signal has a unique name and must be delivered as an output from another selected solution element output. Therefore, the algorithm

searches by the signal unique name in all output signals of the other selected solution element.

3. If the search engine finds the input signal as an output signal of the other selected solution elements that means this input signal could be satisfied.
4. Additionally, the search algorithm can check the compatibility of the signal unit and frequency. The output signal must have a greater frequency than the input signal.
5. Then, the algorithm confirms the compatibility of this signal or stores an error in the error log.

These five steps have to be repeated for each input signal of each selected solution element.

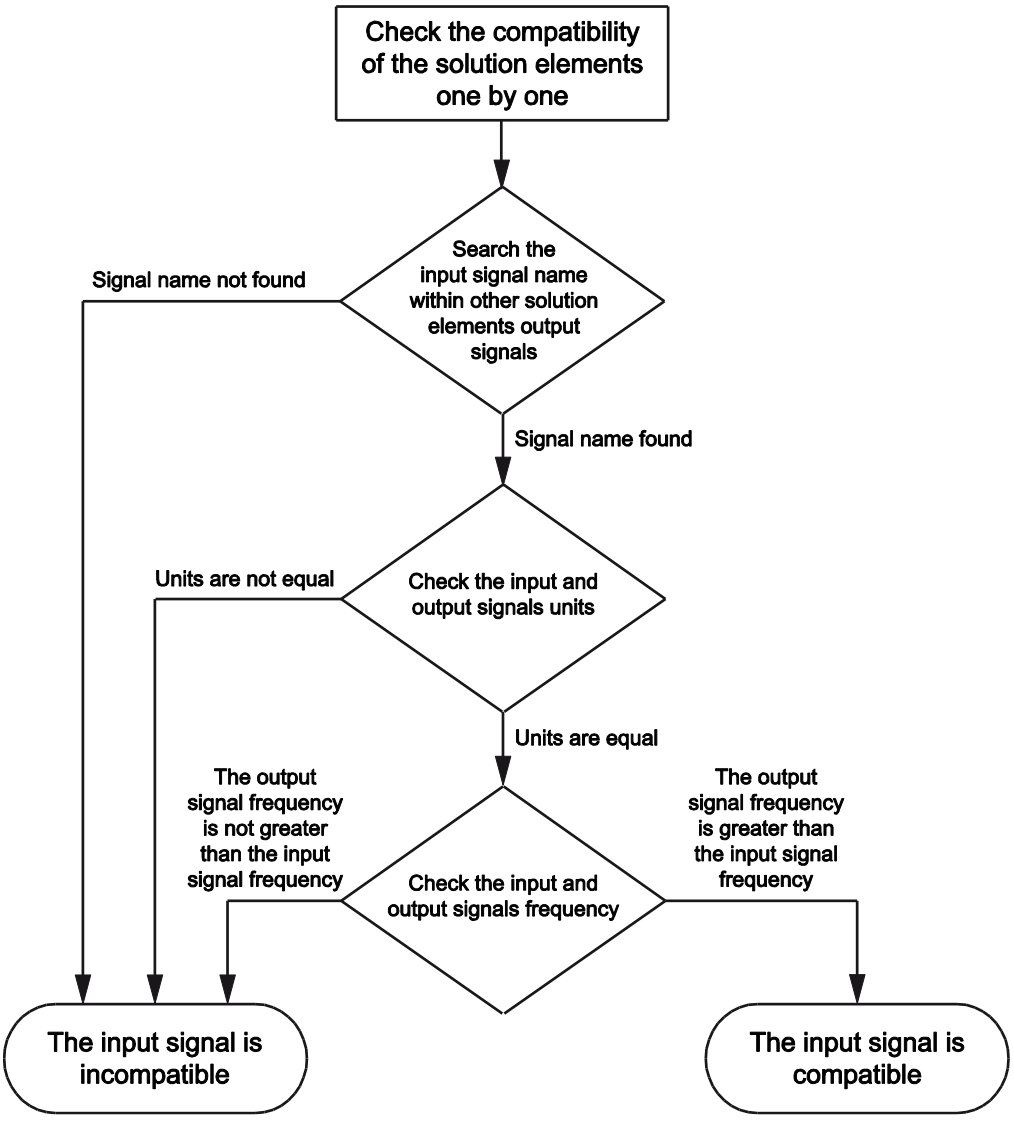


Figure 4-18: Compatibility check flowchart.

4.5.3 Configuration mechanism sequence

Figure 4-19 shows the configuration mechanism flowchart. It shows that the check of the selected solution elements is done with the help of the previously described check algorithms: the consistency check algorithm (left in the flowchart) and the compatibility check algorithm (right in the flowchart). The consistency check algorithm checks all the selected solution elements in pairs. However, the compatibility check algorithm checks them one by one. Both algorithms loop over all selected solution elements. As soon as one of the checks detects an inconsistency or incompatibility, it generates an error register. These error registers are merged at the end of the process and an **error file** is generated. If the selected solution elements are consistent and compatible, the configuration mechanism confirms this and then **a configuration file** is generated. The error file and the configuration file structure are described in phase 5.

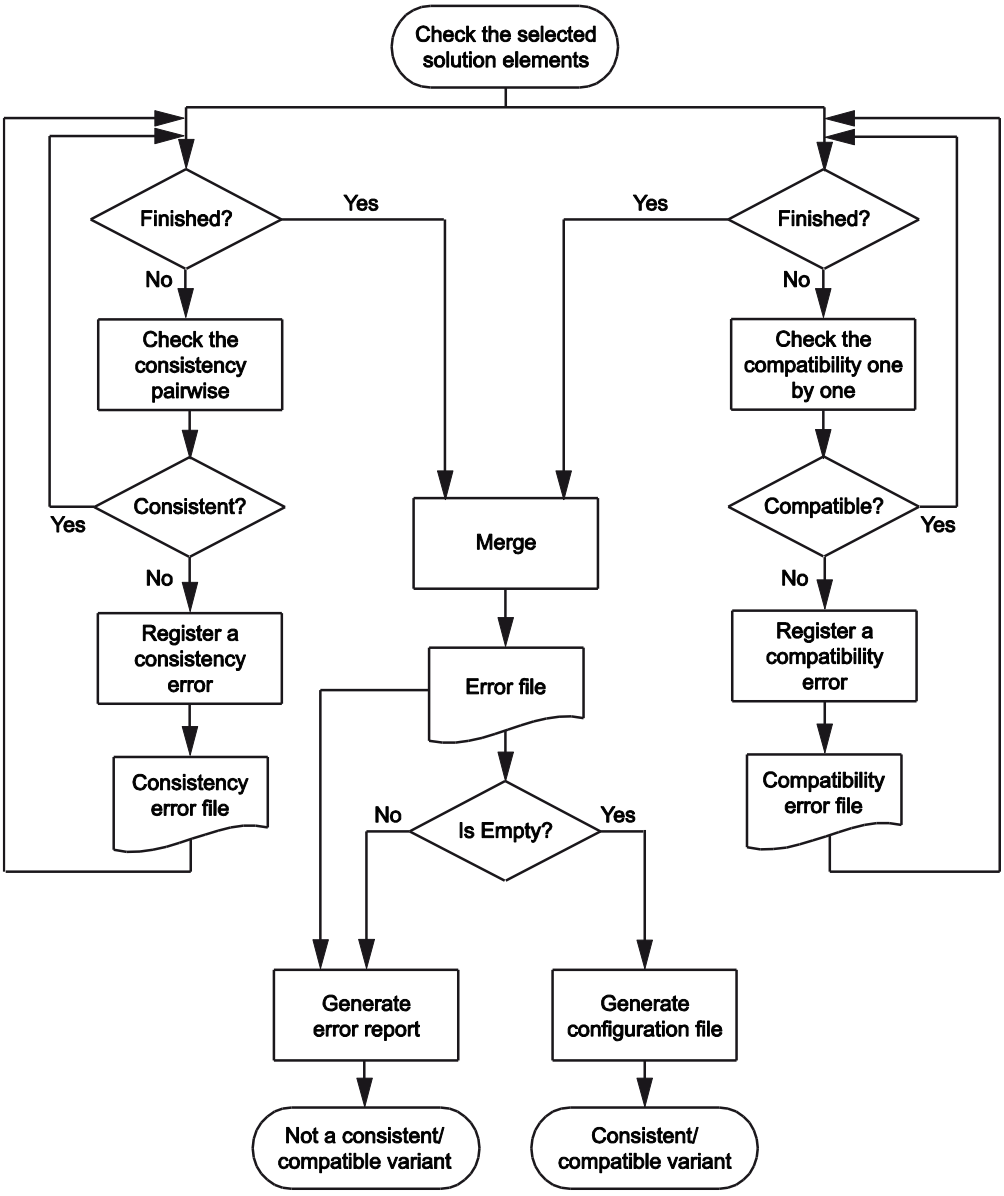


Figure 4-19: Configurations mechanism flowchart.

4.6 Phase 4 – Solution Elements Deployment

The first stage of the development procedure "System Development" was described as well as its entire three phases. The first stage has to be carried out only once by the driving simulator developer. The result of the first stage is a reconfigurable driving simulator outline, which should be extended in the **variants creation stage** by the driving simulator operator. The first stage describes the system's entire components from a solution-neutral point of view. The second stage is the concretisation stage which deals with solution elements instead of the solution-neutral components.

The second stage "variants creation" consists of three phases, starting with phase 4 "solution elements deployment". The main objective of this phase is to build a solution elements database, which contains the existing solution elements, their interfaces and attributes. This phase is an iterative process that has to be carried out each time to add or modify a solution element to the solution elements database.

The solution elements deployment is carried out in two steps. The first step is the identification and classification of the solution elements and the second step is the filling out of the solution elements database with the required attributes of each solution element.

4.6.1 Identify and Classify Solution Elements

The solution elements' identification and classification will be carried out based on the results of the first and second phases. The preferred solution elements will be carried out based on the morphological box concept according to ZWICKY [Zwi89, p. 133f.].

Figure 4-20 shows the morphological box result based on the case study. The first and the second columns of the solution elements morphological box describe the two classification levels of the components which are the result of section 4.4.2 .The third column contains the component names, the fourth column is the component corresponding function which is the result of section 4.3.5 and up to the fifth column. The columns contain the preferred solution elements based on the specification models results: application scenarios, requirements and active structure. Figure 4-20 shows the identified 11 components and their corresponding 24 solution elements.

Classification Level 1	Classification Level 2	Components	Functions	Solution Element 1	Solution Element 2	Solution Element 3	..	Solution Element n
Hardware		Input Device	Drive Virtual Vehicle	Keyboard and mouse	Logitech steering wheel G25	Logitech steering wheel G25		
		Visualization Device	Display Visual Scene	60" LED Monitor	40" LCD Monitor	Oculus Rift		
		Motion Platform	Produce Motion	AirMotion Ride	N/A	N/A		
		Acoustic Device	Generate Tones	Desktop speakers 2.0	Desktop speakers 3.1	N/A		
Software	Hardware Interfaces	Input Device Interface	Interface Input Device	Keyboard and mouse interface	Logitech G25 interface	N/A		
		Motion Platform Controller	Regulate Motion Platform	AirMotion Ride HNI Interface (C++)	AirMotion Ride HNI Interface (Matlab/Simulink)	N/A		
	Software/Models	Vehicle Model	Simulate Virtual Vehicle	Physics engine vehicle model (UNITY 3D)	HNI SVD (Matlab/Simulink)	N/A		
		Rendering Software	Produce Visual Scene	VND 1.0 (C++/OSG)	VND 2.0 (UNITY 3D)	N/A		
		Acoustic Software	Produce Sounds	HNI acoustic module (UNITY 3D)	N/A	N/A		
Resources	Computing Unit	CPU	Compute Softwares	Windows 7 PC	Windows 7 Laptop	N/A		
	Signals Processing Interfaces	CPU Interface	Interface CPU	USB Interface	UDP/IP Interface	N/A		

Figure 4-20: Case study solution elements in morphological box form.

The stated solution elements in each row can fulfil the component function, and only one solution element can be selected from each row.

4.6.2 Filling the Solution Elements Database

In order to make the configuration tool deal with the component and solution elements, there is a need to register the identified components and solution elements in a database. This database stores and organizes the components and solution elements. It also has to be readable by the driving simulator operator and accessible by the configuration tool.

The main database operations are based on CRDU classes [Bro13]: create, read, update and delete. These operations must be covered by the database.

1) **Create**: This operation could perform for both components and solution elements. The database is always extendable by adding a new component or by adding a new solution element for an existing component. This operation will be described in detail in this section.

2) **Read**: This operation can be executed for both components and solution elements. The database internal entries are accessible for the driving simulator operator, as well as for any software that would be used during the configuration process. All stored component and solution elements as well as their attributes can be accessed.

3) **Update**: This operation can be executed for both components and solution elements. Each stored component or solution element can be changed and restored.

4) **Delete**: This operation can be executed for both components and solution elements. Each stored component or solution element can be deleted from the database.

In this section, the create operation is described in detail in order to fill the solution elements database. The filling process is done in two steps: create component then create solution element.

Create a component entry: In order to create a component, the following attributes must be registered and stored in the database:

- **Component name**: This attribute is the unique name for each component, which describes the component function.
- **Component type**: This attribute is used to define the type of the component whether it is a key component or an optional component.
- **Component classification**: This attribute is used to define the type of the component: hardware, software (applications/models or hardware interfaces) or resources (computing units or signal processing interfaces).
- **Component description**: This attribute contains a brief description of the component in text form.
- **Component symbol**: This attribute contains a symbol (logo) associated with the component.
- **Component logic dependency row**: This attribute is a row which contains the logic dependency between the components and the previously added components as mentioned before in section 4.5.1. This row is part of the components dependency matrix shown in Table 4-3.

- **Component guideline[9] entry**: The guideline entry is an optional attribute which defines a preferred parameter value and condition regarding the component. For example, a guideline defines that the visualization device must have a minimum horizontal viewing angle of 100 degrees. This attribute can be added to the component in the form of the condition greater than (>) and parameter value (100 degrees).

Create a solution element entry: In order to create a solution element, the following attributes must be registered and stored in the database:

- **Solution Element Name**: This attribute is the unique name for each solution element.
- **Solution Element Path**: This attribute is the storage path on the file storage system. This is applicable only for an application/model.
- **Solution Element – Parent Component**: This attribute is the name of the corresponding parent components. Therefore, it represents the relationship between this solution element and a component.
- **Solution Element Description**: This attribute is a brief description of the solution element.
- **Solution Element Symbol**: This attribute contains a symbol (logo) associated with the solution element.
- **Solution Element Author**: This attribute is the solution element developer name, if known.
- **Solution Element Company**: This attribute is the solution element producer company name if known.
- **Solution Element Release Date**: This attribute is the date of when the solution element was released.
- **Solution Element Interface**: This attribute is a table containing all the input and output signals of the solution element. Each signal has the following attributes:
    - **Signal Name**: It contains the names of the input and output signals of the corresponding solution element.
    - **Input/Output**: It indicates the direction of the signal, i.e. whether it is an input or an output signal.

[9] Driving simulator guidelines describe preferred specification and prerequisites of the driving simulator in order to fulfill its task. Typically, there are guidelines for using the driving simulators in training tasks as stated before in section 2.2.3.

- **From**: It contains the component name from which this signal is to be fulfilled. This is applicable only for input signals.
- **Unit**: It contains the measuring unit of the corresponding signal.
- **Frequency**: It contains the sampling frequency of the corresponding signal.
- **Resolution**: It contains the resolution of the corresponding signal.
- **Protocol**: It contains the transmission protocol of the corresponding signal e.g. CAN or TCP/IP.
- **Physical Port**: It contains the physical port used to transmit the corresponding signal.
- **Mandatory/Optional**: It indicates whether the signal is mandatory or optional.
- **Description**: It contains a brief description of the corresponding signal.

- **Solution Element Consistency Row**: This attribute is a row which contains the logic consistency between the solution element and the previous added solution elements as mentioned before in section 4.5.1. This row is part of the solution elements consistency matrix shown in Table 4-4.
- **Solution Element Guideline Entry**: If the parent component has a guideline entry, the solution element inherits this entry and should define a parameter value for the entry to check the solution element confirmation with the guideline.

After registering all identified components and all preferred solution elements, which result from the metrological box in the database, the solution elements database is filled and ready to be used in the variant generation phase.

4.7 Phase 5 – Driving Simulator Variant Generation

The main objective of this phase is to define the configuration selection sequence, as well as define the configuration file structure, error reports structure and the physical connection plan.

4.7.1 Configuration Selection Sequence

In order to make a reasonable selection sequence for the solution elements, the identified components and their relationships have to be investigated. The selection sequence can be changed based on the area of use. During this phase, an example of the used case study shows how it can be determined.

The driving simulator components have been previously classified as three main classes: Hardware, software and resources. A driving simulator structure is respectively based on hardware components, software and finally, the used resources.

In order to make the selection sequence reasonable, it is not sufficient to make the selection sequence based on the classification, because of the tight correlation between some hardware and software components. Therefore, the identified components will be divided into groups of software and/or hardware based on the groups identified during the active structure specification step discussed in section 4.3.6.

As shown in Figure 4-12, the case study's active structure has the following four logic groups of components:

- **Motion System Group**, which contains the components: motion platform and motion platform controller.
- **Acoustic System Group**, which contains the components: acoustic device and acoustic software.
- **Visualization System Group**, which contains the components: visualization device and rendering software.
- **Input Device Group**, which contains the components: input device and input device interface.

The result of the active structure specification defines components clustered in logical groups, but does not give an indication about the selection sequence within the identified groups. Therefore, a further investigation is needed to get the selection sequence of the identified groups.

With the help of studying the component and its relationships based on the dependency matrix, it gives a suggestion about the reasonable selection sequence. Table 4-5 shows the dependency matrix between the components mirrored along its diagonal. In addition, two rows are added to the matrix. The first added row contains the number of relationships for each component. The number of relationships for each component is calculated by summing up its column (marked up with the small red rectangle).

The second add row contains the number of the relationships for each group of components. The number of relationships for each group is calculated by summing up the number of relationships of the entire group components (marked up with the big red rectangle).

Table 4-5: Components and groups number of relationships based on dependency matrix.

Number of the relationships between the components and the groups 0 = Independent pair 1 = Dependent pair	Motion System Group		Acoustic System Group		Visualization System Group		Input Device Group		Vehicle Model
	Motion Platform	Motion Platform Controller	Acoustic Device	Acoustic Software	Visualization Device	Rendering Software	Input Device	Input Device Interface	Vehicle Model
	A	B	C	D	E	F	G	H	I
A. Motion Platform		1	0	0	1	0	1	0	0
B. Motion Platform Controller	1		0	0	0	0	0	0	0
C. Acoustic Device	0	0		1	0	0	0	0	0
D. Acoustic Software	0	0	1		0	0	0	0	0
E. Visualization Device	1	0	0	0		1	0	0	0
F. Rendering Software	0	0	0	0	1		0	0	0
G. Input Device	1	0	0	0	0	0		1	0
H. Input Device Interface	0	0	0	0	0	0	1		0
I. Vehicle Model	0	0	0	0	0	0	0	0	
Number of Component Relationships	3	1	1	1	2	1	2	1	0
Number of Group Relationships	4		2		3		3		0

Based on the result of comparing the groups' number of relationships, a driving simulator structure is based respectively on hardware components, software and finally used resources. A reasonable selection sequence is shown in Figure 4-21 based on the calculated number of the relationships for each group. However, the visualization system and the input device group can be swapped because both have the same groups' number of relationships.

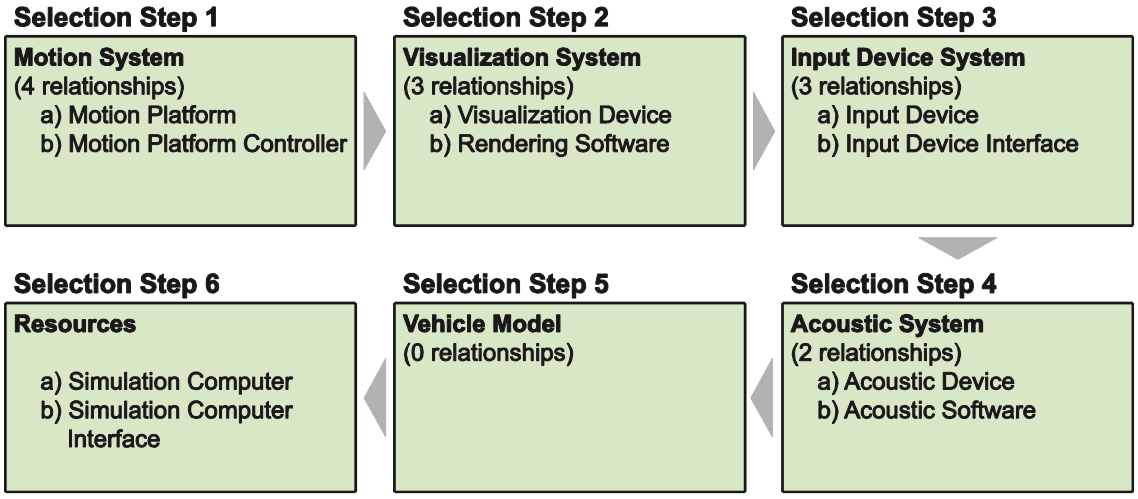


Figure 4-21: Selection steps of the case study reconfigurable driving simulator.

4.7.2 Configuration Files and Error Reports Structure

After the compilation of the solution elements' selection process, the configuration mechanism checks the selected components in terms of consistency and compatibility.

Based on the configuration mechanism check results, if the selected solution elements are consistent and compatible with each other, the configuration tool confirms that the selected solution elements can build a driving simulator variant and generates a configuration file. However, if the configuration tool finds any inconsistency or incompatibility between the selected solution elements, the configuration tool generates an error report. In the next section, the structures of the configuration file as well as the error report will be described.

Configuration File Structure

The configuration file is considered to be the result of the configuration process. It is a readable text file containing all the relative data about the selected variant. It consists of four parts: configuration data, hardware, software and resources. The configuration data is the part which describes general information about the configuration itself, e.g. configuration name, author, etc. The hardware part contains all selected hardware solution elements attributes, parent component name and detailed input/output signal descriptions. The software part contains all selected software solution elements attributes, parent component name and detailed input/output signal descriptions. The resources part contains the selected resources.

Error Report Structure

The error report is a readable text file containing warnings and errors which are detected by the configuration mechanism. It contains five parts: configuration data, hardware, software, resources and errors/warning. The first four parts are the same as in the configuration file. The error and warning part lists all detected inconsistent solution elements as well as all incompatible signals.

4.7.3 Physical Connections Plan

The configuration tool generates configuration files which contain the interfaces between the selected solution elements and the software side, but the configuration file does not contain the physical connections between the selected hardware solution elements and the selected resources. A physical connection plan is very useful for the driving simulator operator in order to prepare the driving simulator for operation. It shows in a simple way how the diverse hardware solution elements should be connected with the resource interfaces. It could be considered as a simple wiring plan.

Figure 4-22 shows an example of the physical connection plan regarding case study variant 1. The case study variant 1 consists of four hardware solution elements which have to be connected to the simulation computer interfaces. With the help of the infor-

mation stored in the solution elements database, the physical plan for the components can be generated. In this case, there were 4 connections, each hardware solution element is connected through one connection.

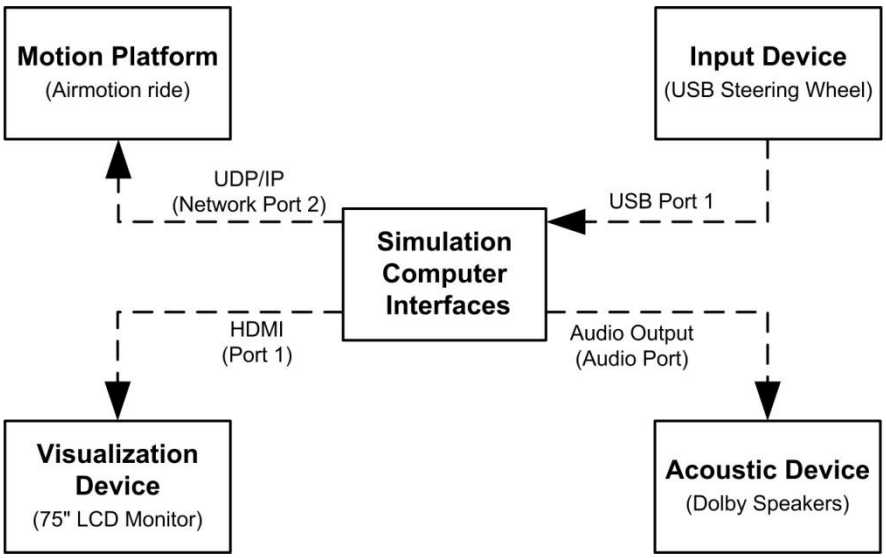


Figure 4-22: Example of a physical connection plan.

4.8 Phase 6 – System Preparation for Operation

The result of the fifth phase is the configuration file and a physical connection plan. The configuration file contains the selected solution elements, interface topology and selected resources. Additionally, the physical connection plan contains the physical interfaces between the selected hardware solution elements.

There are two preparation steps required in order to build up the selected driving simulator variant and to prepare it for the simulation. The first step is the preparation of the hardware connections and the second step is the software preparation.

4.8.1 Hardware Setup Preparation

Assuming that the selection process finished successfully and the configuration tool generated the physical connection plan, then the driving simulator operator has to plug the different hardware solution elements together. The physical connection plan makes this step easy and understandable.

For the case study example of variant one shown in Figure 4-22, the driving simulator operator has to plug in 4 cables: a USB cable between the steering wheel and the simulation computer, an HDMI cable between the 75” LCD monitor and the simulation computer, a network cable between the motion platform and the simulation computer, and an audio cable between the dolby speakers and the simulation computer. The example shows that the hardware preparation step can be easily done manually.

4.8.2 Simulation Software Preparation

To prepare the selected software solution elements for the operation, which is a complicated process (unlike the hardware preparation step) there is a need to develop software to assist this step. The software is called "Assistant[10]". The assistant software is responsible for preparing the software solution elements for the simulation by the following three steps:

1. **Read the configuration file**: The assistant software can load and phrase the configuration file. It identifies the selected applications/models and their different attributes.
2. **Fetch the applications/models**: The assistant software retrieves the storage path for each application/model. It accesses the storage file system where the applications/models are stored.
3. **Distribute the applications/models over resources**: The assistant software loads each application/model on its corresponding source selected during the selection process.

4.8.3 Communication during the Simulation Run-time

The Intelligent Interfacing Module (IIM) initializes the communication between the selected software solution elements based on the interface topology which is described in the configuration file. As soon as the user starts the simulation, the IIM ensures the communication between the simulation-related software solution elements during simulation run-time. Figure 4-23 shows the IIM function in the case study variant 1. The IIM exchanges the required input and output from and to the simulation related software solution elements during run-time. Moreover, IIM can connect the software solution elements together although a part of them runs under hard real-time conditions and the other part runs under soft real-time conditions.

[10] This software was developed during my research work in cooperation with colleagues from the Department of Control Engineering and Mechatronics at the Heinz Nixdorf Institute, University of Paderborn.

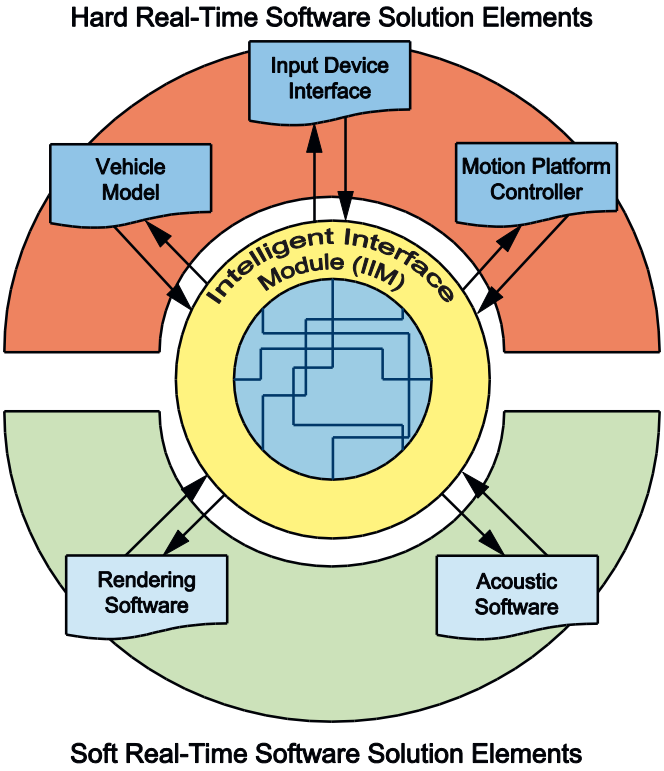


Figure 4-23: IIM function during simulation run-time of the case study variant 1.

The result of this phase is a ready-to-use driving simulator which consists of the selected software and hardware solution elements, as well as the selected resources.

5 Implementation Prototype and Validation

The previous chapter proposed a detailed description of the design framework procedure model. The procedure model described the phases and the entire tasks in each phase required to develop a reconfigurable driving simulator. The proposed procedure model showed that there is a need for a software tool to support the development process. This chapter introduces an implementation prototype of the configuration tool. Additionally, the *design framework for developing a reconfigurable driving simulator* is validated within the ADAS reconfigurable driving simulator.

Section 5.1 presents the main functions of the configuration tool, its structure and the implementation prototype results[11]. Section 5.2 describes the design framework validation within the validation example, which is the ADAS reconfigurable driving simulator. Section 5.3 shows the different generated variants based on the procedure model and the application of the configuration tool's implementation prototype. Finally, section 5.4 describes the design framework validation based on the defined requirements.

5.1 The Configuration Tool

The previous chapter discussed the procedure model for developing a reconfigurable driving simulator from a theoretical point of view. The proposed functions and algorithms, which are described during the procedure model, need to be implemented in a software tool. This tool helps the driving simulator developers and operators during the development and variant creation processes. Moreover, the state of the art analysis shows that until now there is no such approach or software tool, which deals with a reconfigurable driving simulator based on the described procedure model and the concept behind it. Therefore, a concept and an implementation prototype of the required configuration tool are being presented in this section. The configuration tool forms the second essential component of the design framework. It realizes and supports the procedure model's second, third, fourth and fifth phases.

The main task of the configuration tool is to support the driving simulator developers and operators in developing a reconfigurable driving simulator. The configuration tool has to meet the functional requirements, which are defined in section 2.7.

The concept and the main operation of the configuration tool are described in section 5.1.1. The configuration tool's architecture is described in section 5.1.2. Section 5.1.3 briefly describes the implementation prototype of the configuration tool and its graphical user interfaces.

---

[11] The achievements in this section are carried out in cooperation with Kareem Abdelgawad during my supervision of his master thesis; "Conceptual Design of a Configuration Mechanism for a Reconfigurable Driving Simulator" of M.Eng. Kareem Abdelgawad.

5.1.1 The Configuration Tool's Main Operations

The configuration tool concept is based on the procedure model for developing a reconfigurable driving simulator which is described in chapter 4. It supports the driving simulator developer during the system development stage by organizing the identified driving simulator components, which is **the second phase result**. Additionally, the configuration check algorithms are embedded and implemented in the configuration tool. These are **the third phase results**.

Furthermore, the configuration supports the driving simulator operator during the creation stage of variants by filling out the solution elements database **during the fourth phase**. In addition, it supports the variant creation process during **the fifth phase**.

The configuration software has functional and non-functional requirements which have to be covered and implemented. In the next part, the functional and the non-functional requirements of the configuration tool are defined.

Functional Requirements

The functional requirements describe the functions and tasks which should be supported by the configuration tool:

1. The configuration tool should interact with the **reconfigurable driving simulator database**, which organizes and stores components, solution elements and variants' descriptions.
2. The configuration tool should allow the driving simulator developer to view, add, modify and remove one or more **component**.
3. The configuration tool should allow the driving simulator operator to view, add, modify and remove one or more **solution element**.
4. The configuration tool should allow the driving simulator operator to select a combination of solution elements, which are stored in the solution elements database, in order to **generate a driving simulator variant**.
5. The configuration tool should ensure that the selected combinations of solution elements are **consistent and compatible** to each other.
6. If the selected combination is consistent and compatible, the configuration tool has to generate a **configuration file** which contains all the required information about the selected solution elements and the system interface topology.
7. If the selected combination is inconsistent or incompatible, the configuration tool has to generate an **error file** which contains the inconsistent or incompatible solution elements and/or signals.

8. The configuration tool should have the ability to **load, view and modify any previously generated variant** by parsing a previously generated configuration file.

Non-Functional requirements

The non-functional requirements describe some general requirements regarding the software stability and usability. The most important non-functional requirements are defined as follows:

1. The configuration tool should have a **self-explaining graphical user interface** to ensure the usability of the software.
2. The configuration tool should assist the user in each step with a **help window**.
3. The configuration tool should be **modular and extendable**.
4. The configuration tool should be **platform-independent** which means it should be compiled in an executable package independent of the operating system.
5. The configuration tool should **prevent the user from dealing with complex processes** such as database operations or check algorithms. The user only has to deal with a graphical user interface. The configuration tool has to separate the concerns between the user interface, algorithms and database.

Based on the defined tasks, the configuration tool structure will be presented in the next section.

5.1.2 The Configuration Tool's Architecture

The configuration tool structure follows the Model-View-Controller (MVC) approach, which is a common design pattern for software architecture. The MVC architecture divides the software into three main modules: the model, the view and the controller [BY09].

Applying the MVC approach on the development of the configuration tool results in the following modules:

1. **The configuration tool model**: This is the core of the tool, which contains all required functions, algorithms and the interaction with the reconfigurable driving simulator database.
2. **The configuration tool view**: This is the graphical user interface of the tool which allows the user to execute the required operations in a simple way and without dealing with complex functions or algorithms.
3. **The configuration tool controller**: This is the events handling module of the tool. It connects the view module with the model module. As soon as the user

takes an action by using the graphical user interface, then the controller module has to execute the action's predefined operation or call a pre-programmed algorithm.

Figure 5-1 shows the different modules of the configuration tool regarding the MVC approach and the interaction between the different modules.

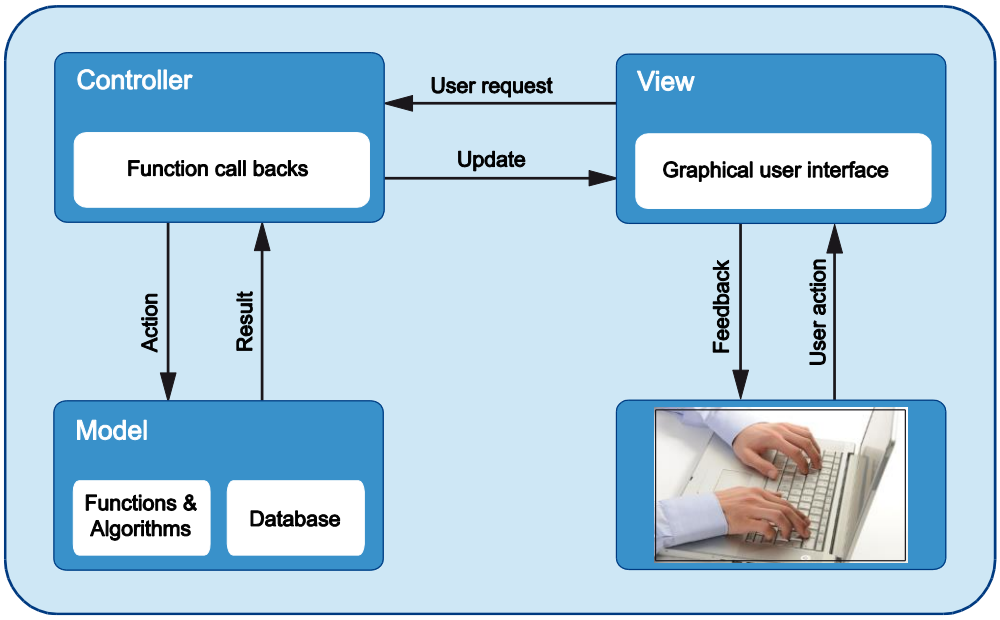


Figure 5-1: Model-View-Controller modules of the configuration tool.

5.1.3 The Configuration Tool's Implementation Prototype

A prototype of the described concept has to be implemented as a part of this work. The implemented configuration tool consists of more than 150 embedded functions. This section describes the essential components of the configuration tool, the graphical user interface and the important tasks/functions covered by the tool.

The software was implemented using two software tools: Microsoft Office Excel and Matlab. The reconfigurable driving simulator database is implemented simply in Microsoft Office Excel. Further, the functions and algorithms are implemented with the help of Matlab M-Functions and the graphical user interface is implemented with the help of Matlab-GUI utility.

The reconfigurable driving simulator database

The development of the reconfigurable driving simulator database was done based on the relational database model approach. This approach is efficient and overcomes the complexity of the relationships between the entire different database tables. The implemented database mainly contains three types of tables: the components' table, the solution elements' table and the interfaces' table. These three types of tables are connected together based on a relational model of the database.

The graphical user interface of the configuration tool

The dealing with the developed configuration tool is carried out mainly via a graphical user interface. Figure 5-2 shows the start screen which contains the main operations of the configuration tool and their correlation to the various phases of the development procedure model.

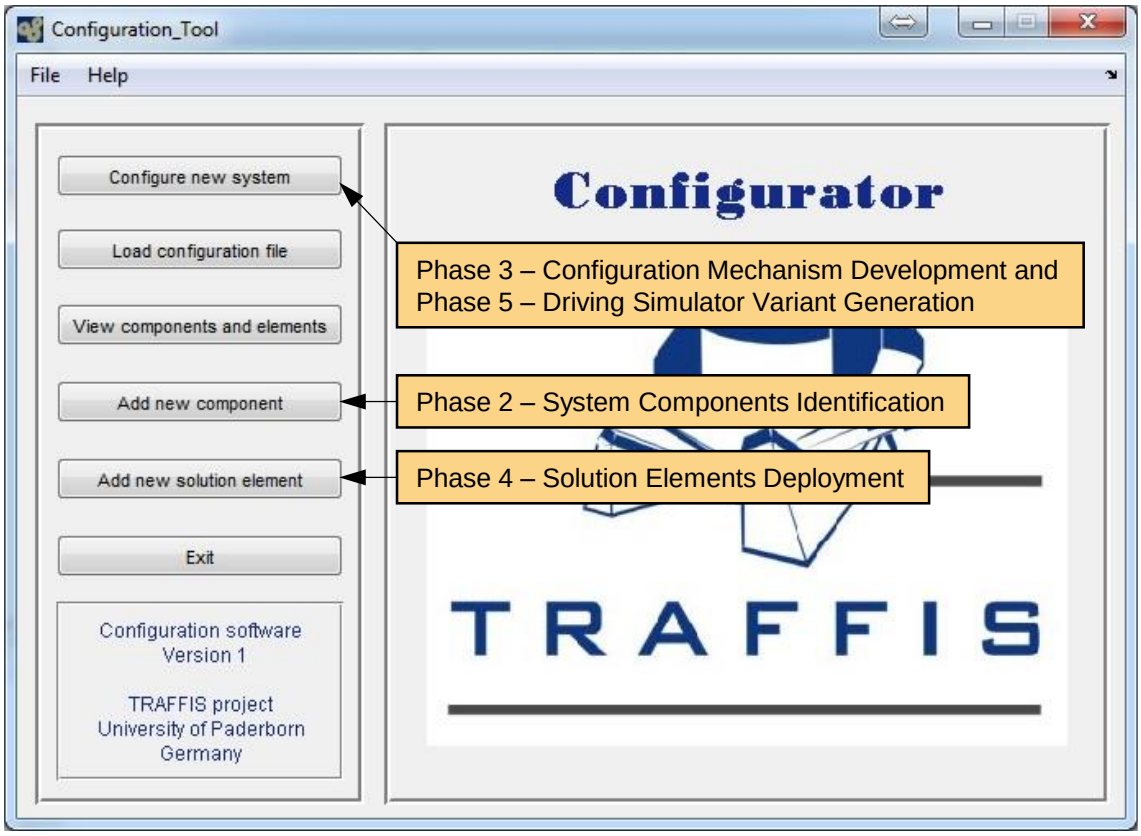


Figure 5-2: The graphical user interface of the configuration tool's implementation prototype – start screen.

The start screen operations of the configuration tool are described as follows:

- **Configure New System**: This operation is the essential task of the configuration tool. It is responsible for creating a new driving simulator variant by selecting solution elements for hardware, software and resources in a predefined sequence; so that the user is prevented from dealing with complex algorithms such as consistency and compatibility check algorithms. Firstly, the consistency check algorithm runs in the background parallel to the selection steps. The configuration tool only shows the consistent solution elements which match with the previously selected solution element. Secondly, after the selection steps end, the configuration tool executes the compatibility check algorithm to check the compatibility of the selected solution elements. After the compatibility check has finished, the configuration tool generates a configuration file if the selected

solution elements are compatible with each other or it generates an error file if the selected solution elements are not compatible with each other.

- **Load Configuration File**: This function allows the user to view and modify a previously generated configuration file. Moreover, it allows the operator to modify the previously generated configuration file by exchanging one or more of the previously selected solution elements.
- **View Components and Solution Elements**: This function allows the user to deal with the stored components and the solution elements in the database. The user can view, modify or delete one or more component or solution element.
- **Add New Component**: This function allows the user to add one new driving simulator component per execution. This function will guide the user through predefined schemes in order to register the different attributes of the new component, which have been previously described in section 4.6.2.
- **Add New Solution Element**: This function allows the user to add one new driving simulator solution element under a selected component per execution. This function will guide the user through predefined schemas in order to register the different attributes of the new solution elements, which have been previously described in section 4.6.2.

Behind each operation in the main screen, a set of panels/schemas exists to accompany the user until he accomplishes the selected function. The different panels and their functions are described in Appendix A2.2.

5.2 Design Framework Validation

In this section the *Design Framework for Developing a Reconfigurable Driving Simulator* is validated by means of developing task-specific driving simulators. The development process is described based on the presented procedure model and with the help of the implementation prototype of the configuration tool.

The validation example is described in section 5.1.2. After that, the development process will be described based on the procedure model phases from section 5.2.2 to section 5.2.7.

5.2.1 Validation Example: ADAS Reconfigurable Driving Simulator

The validation example used during this chapter is the development of ADAS reconfigurable driving simulators during the TRAFFIS project with the help of the proposed design framework. The ADAS reconfigurable driving simulator shows the design framework’s strength in the development of task-specific reconfigurable driving simulators.

Driving simulators are typically designed and built for a special purpose in order to support a specific testing or training approach in a predefined structure and preferred entire components. Adapting such systems to support new applications is very complex, time consuming, and often not feasible. Thus, with the increasing role of using driving simulators in different approaches, there is a need for highly adaptable and reconfigurable driving simulators that can be conveniently tailored to new specific test or training functions. Moreover, there are also different levels of detail of the simulation models, ranging from simple low-fidelity models to their respective complex high-end models, which provide a much more detailed simulation. Also, the hardware components range from simple to complex components, which constitute different simulator setups that are capable of simulating specific aspects of the ADAS under test.

The main objective of the TRAFFIS (German acronym for Test and Training Environment for Advanced Driver Assistance Systems) project is to build a reconfigurable driving simulator, which supports different test and training approaches related to ADAS. Therefore, using the proposed design framework to develop a reconfigurable ADAS driving simulator shows the enormous benefits of this work.

5.2.2 Phase 1 – System Specification of Driving Simulator

In this section, the specification results are presented based on the proposed phase 1 of the procedure model, which is described in section 4.3. The specification results of the ADAS reconfigurable driving simulator were carried out during several workshops by the TRAFFIS project team. The results are presented as follows:

Environment – specification results of the ADAS driving simulators

The specification of the environment influences on the ADAS driving simulator results in the identification of six important environment elements as well as three disturbing influences. Five of the six identified environment elements are identical with the identified environment elements of the case study example (described in section 4.3.2). **The new identified element is the test drive manager**. The test drive manager is the one who is responsible for setting up, observing and interacting with the system during the test/training drive simulation run-time. The instructor also has the ability to trigger some pre-defined events during the runtime (e.g. trigger the event: a child running across the road in certain situations) in order to investigate the interaction between the ADAS, the driver and the simulated vehicle systems.

Application scenarios – specification results of the ADAS driving simulators

The TRAFFIS project has five project partners, four of which plan to use the driving simulator in different areas of use as follows:

A. **Heinz Nixdorf Institute (HNI):** As the reconfigurable driving simulator developer, the main aim for HNI is to develop reconfigurable driving simulators to support small and medium-sized enterprises which work in the ADAS develop-

ment field. HNI will support these companies by providing them with the reconfigurable driving simulator which helps the enterprises during the ADAS development cycle.

B. **dSPACE GmbH (dSPACE):** As a worldwide development tools supplier of automotive control units, the main aim for dSPACE is to develop a closed loop test tool which allows their automotive customers to investigate the interaction between ADAS control units, real vehicle components and the driver in a closed loop virtual environment.

C. **Varroc Lighting Systems GmbH (Varroc):** As a worldwide automotive components supplier, Varroc (formerly Visteon) was the first producer worldwide to bring an intelligent light system e.g. AFS (Adaptive Front-lighting System) together with Ford to a series production [Sch07]. The main aim for Varroc is to use the driving simulator in the ADAS development cycle.

D. **Institut für Logistik und Verkehr (ILV) UG & Co. KG:** As one of the most modern traffic safety centres in Europe, the main aim for ILV is to develop an ADAS training driving simulator for truck and bus drivers.

The four partners defined some application scenarios individually. HNI defined one application scenario (marked with A1), dSPACE defined three application scenarios (B1, B2 and B3), Varroc defined two application scenarios (C1 and C2) and ILV defined one application scenario (D1). The short descriptions of the defined application scenarios are stated in the following part:

- **A1 – Flexible ADAS Test Tool:** The main objective of this application scenario is to build a flexible driving simulator which supports the small and medium-sized enterprises which develop ADAS. The system has to be reconfigurable so that it can match different ADAS test requirements during the different development steps. Moreover, it has to allow different test methodologies such as SiL, MiL or HiL. Additionally, the system has to allow the investigation of the interaction between driver and system.

- **B1 – HCM Man-in-the-Loop Test Tool:** Headlamp Control Module "HCM" is a driver assistance system developed by the company Varroc (formerly Visteon) [Sch07]. The main function of the HCM is to control the headlamps to ensure an optimum road illumination by using a high beam as often as possible without dazzling other traffic participants. The main objective of this application scenario is to build a driving simulator variant, which uses dSPACE existing tools to realize the test of HCM control unit in a HiL simulation environment.

- **B2 – ADAS Man-in-the-Loop Test Tool:** The main objective of this application scenario is to build a driving simulator variant, which uses dSPACE existing tools to realize the test of other ADAS control units in a HiL simulation. Simultaneously, the system has to support the investigation of the interaction between the ADAS control unit and driver.

- **B3 – Camera-Based ADAS Automated Test Tool:** The main objective of this application scenario is to build a driving simulator variant, which allows the automated test of the camera-based ADAS. Therefore, a powerful and extra-

realistic visualization system is needed. In this application scenario, there is no need to use a motion platform.

- **C1 – HCM Software-in-the-Loop Test Tool:** The main objective of this application scenario is to build a driving simulator variant, which allows the testing of the main HCM algorithms in the laboratory in a SiL simulation environment. The preferred setup is a PC-based simulator with a simple vehicle model and a visualization system.
- **C2 – HCM Hardware-in-the-Loop Test Tool:** The main objective of this application scenario is to build a driving simulator variant, which allows the testing of the HCM control unit prototype in a HiL simulation environment. The preferred setup is only a PC-based simulator with the needed HiL interfaces to test the HCM control unit in the laboratory.
- **D1 – ADAS Training in Driving Simulator:** The main objective of this application scenario is to build a driving simulator variant which allows the execution of different training scenarios in a virtual environment. The main focus of the training scenarios is to focus on ADAS usage and ADAS benefits.

Requirements – specification results of the ADAS driving simulators

The four project partners stated their requirements of the system. HNI defined its requirements regarding the general conditions of the driving simulator e.g. ergonomic, security, energy sources, etc. dSPACE defined its requirements regarding the entire simulation models. Varroc defined its requirements regarding the HCM control unit testing. ILV defined its requirements regarding the ADAS training prerequisites. These requirements were collected and organized in a large requirement list.

Functions – specification results of the ADAS driving simulators

Due to the variation in the main task, the structure and the required components of the stated application scenario, the specification of the functions also varies in its complexity and number of its entire sub-functions. Therefore, the identified functions of the stated application scenarios have to be merged together. The application scenario B1 provides all needed functions in the other application scenarios. In other words, the other application scenarios are stripped-down versions of the application scenario B1. Thus, the function and active structure specification in this section is focussing on application scenario B1. During the specification of the case study, the modelling and the analysis process have been neglected to simplify the understanding of the procedure model. Unlike the case study, the modelling and the analysis process will be considered during this validation example. Figure 5-3 shows the function hierarchy of the ADAS driving simulator.

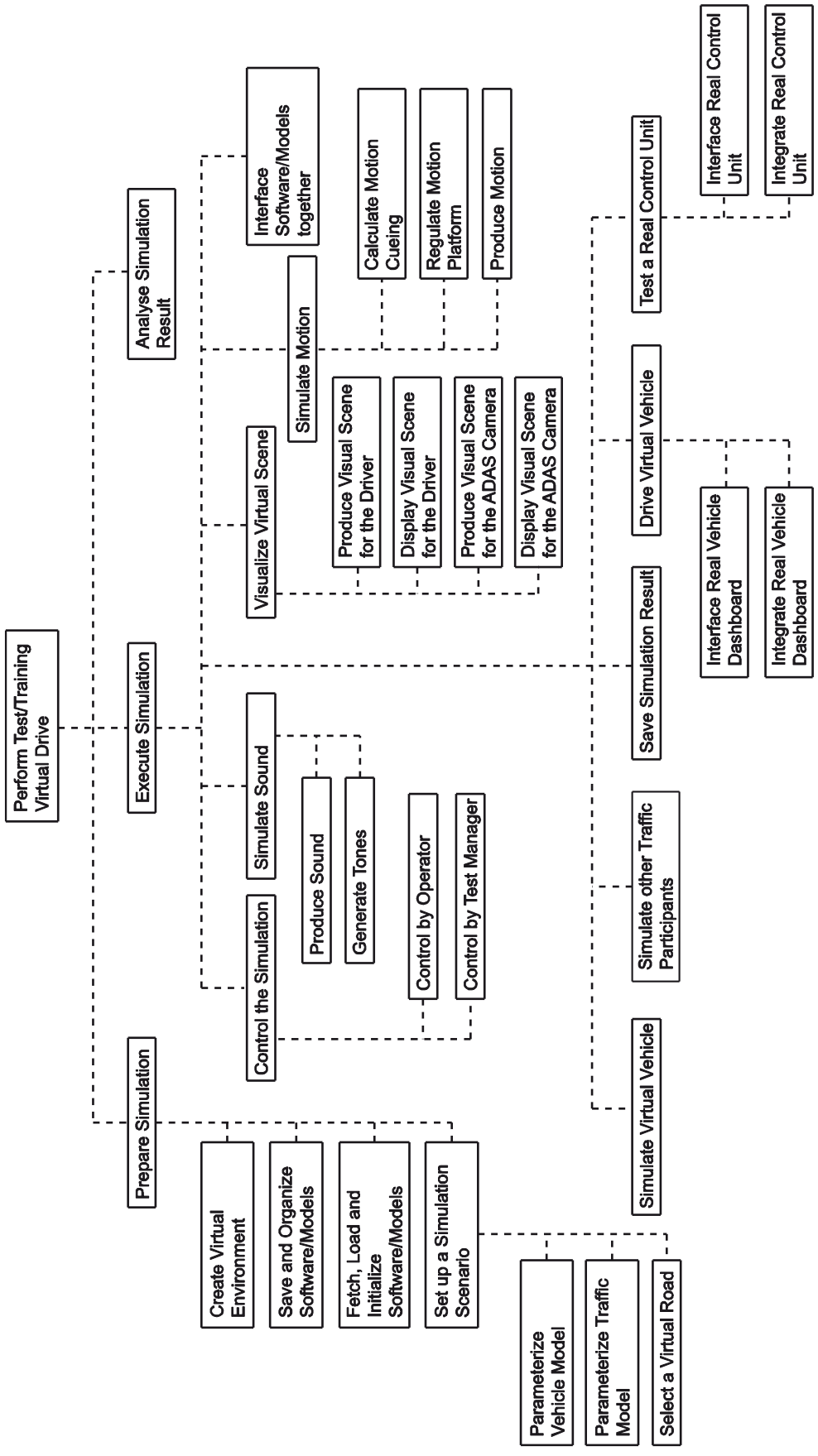


Figure 5-3: Functions' hierarchy of the application scenario B1.

The main function of the ADAS driving simulator is to *perform* a test or training virtual drive. This main function is divided into three main sub-functions: *prepare simulation* regarding the pre-processing preparation, *execute simulation* regarding the different needed functions during the simulation run-time and *analyse the simulation results* regarding the post-processing and the analysis of the simulation results. The functions' hierarchy that was modelled for the application scenario B1 consists of 26 sub-functions.

Active Structure – specification results of the ADAS driving simulators

The active structure model is built based on the function hierarchy for the scenario B1. Figure 5-4 shows the active structure model of the application scenario B1. The information flow labels have been deliberately omitted in this figure to show the system elements closer. The active structure needs to be printed out in a DIN-A2 page, in order to show all the system elements and information signals in a good readable form.

This active structure consists of 24 system elements (components). 16 of these are software components labelled with (SW), 6 are hardware components labelled with (HW) and 2 are database components labelled with (DB). 21 components can be grouped into 9 groups. The description of each system element is stated in the identified components section 5.2.3

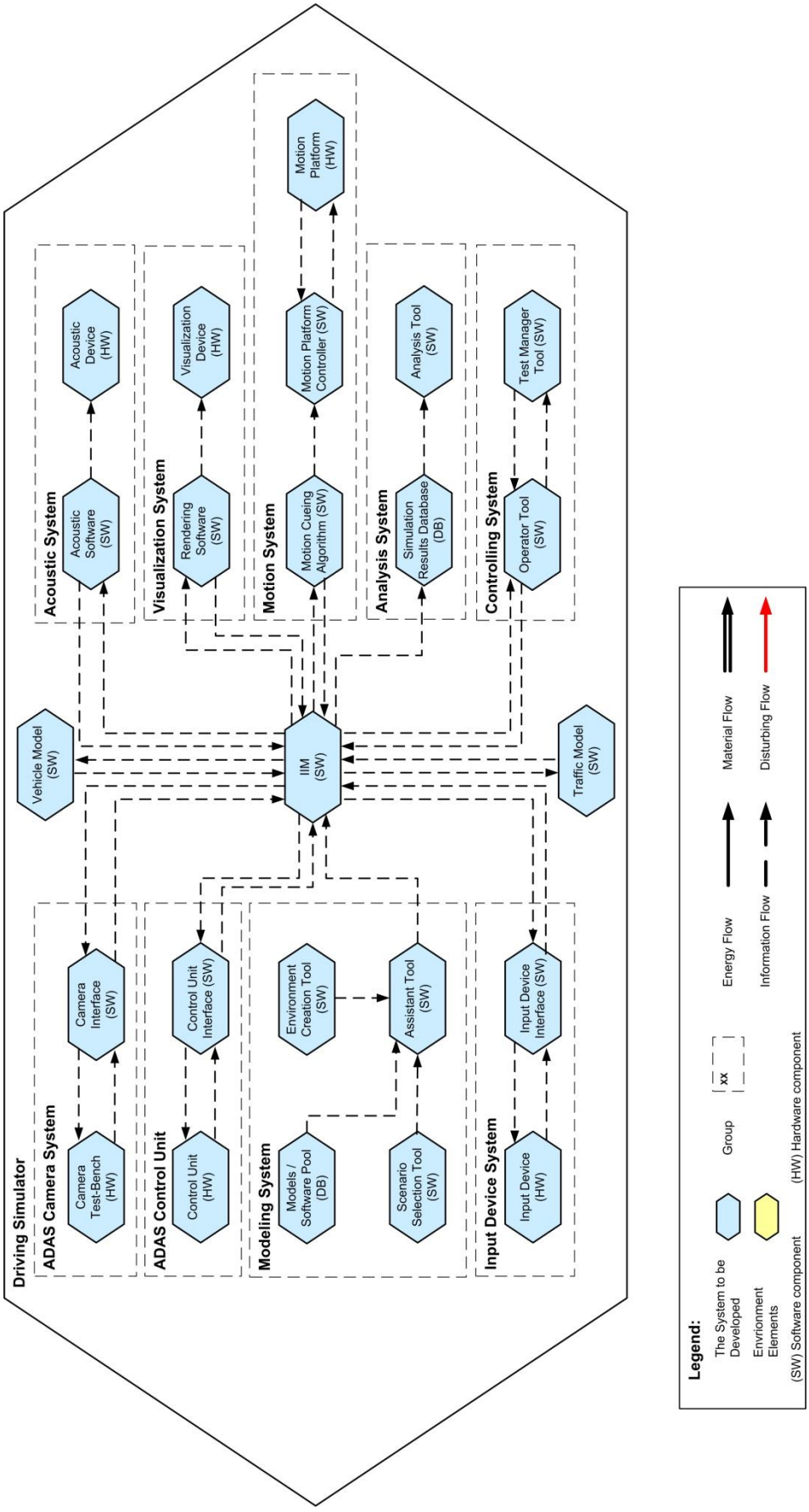


Figure 5-4: Active structure model of the application scenario B1.

5.2.3 Phase 2 – Main Components Identification

Based on the first phase results, the reconfigurable driving simulator components and key components have to be identified, classified and described in this phase. This phase follows the proposed procedure model phase 2 which is described in section 4.4. In this phase, 29 reconfigurable driving simulator components are identified, classified and described. Figure 5-5 shows the classification of the identified components.

Twelve of the ADAS reconfigurable driving simulator identified components are identical to the previously identified components of the case study variants. The identical components are: Input Device, Input Device Interface, Intelligent Interfacing Module (IIM), Vehicle Model, Rendering Software, Visualization Device, Motion Platform, Motion Platform Controller, Acoustic Software, Acoustic Device, Simulation Computer and Simulation Computer Interface. These components have been previously described in section 4.4.3. The key components are also identical to the case study key components. The 15 identified new components are described as follows:

ADAS Control Unit: This is the real ADAS control unit in its prototype or serial production hardware form. With the help of HiL technology, the real control unit can be integrated and tested within the simulated components.

ADAS Control Unit Interface: This is the interface between the control unit hardware, and the simulated virtual components. With the help of special equipment called “digital signal conditioning”, the physical interface of the real control unit can be integrated in the simulation environment. Moreover, a model is required in order to map, code and decode the information exchange between the real ADAS control unit and the simulated components.

ADAS Camera Test-Bench: The test of camera-based ADAS typically focuses on the control unit function. That is done by modelling and simulating the camera and the image processing algorithms. Testing by using a simulated camera does not allow the ADAS developer to test the image processing algorithms, and an intensive test with the real camera is needed in a real environment. The idea behind the camera test-bench is to achieve a visual interface between the camera and the simulated components. The camera will be positioned in front of an LCD display, and the simulated scene according to the camera perspective will be continuously displayed on the LCD display. The camera will detect the simulated environment and the simulated object and then send the captured images to the image processing algorithms [TH13a], [TH13b].

ADAS Camera Test-Bench Interface: This is the interface between the visualization software and the ADAS camera test-bench. A special visualization program is implemented in order to adapt the visualization software to fit the camera detection parameters. Moreover, the image processing detection results are prepared and forwarded to the other simulation components [TH13a], [TH13b].

Traffic Model: This software model is required for the simulation of other traffic participants. This model allows the simulation of various traffic situations and complex traffic scenarios. These simulated traffic scenarios greatly support the development and evaluation of the ADAS under test.

Motion Cueing Algorithm: This algorithm calculates the presentation of haptic information (cues) with the aim of resembling real movements in the virtual environments [Slo08].

Environment Creation Tool: This is a software tool that allows the automatic generation of virtual roadways. The virtual roadways are represented with a logic model, which contains mathematical road descriptions and attributes for the simulation models, as well as the graphics model, which contains the visual representation of the environment [KGG+11a], [KGG+11b].

Scenario Selection Tool: This software tool allows the driving simulator operator to set up a specific simulation scenario by selecting a combination of a predefined vehicle parameters set, a pre-generated virtual environment and a predefined traffic scenario.

Models/ software Pool: This is a file storage system or a database, which contains the models and software solution elements.

Assistant Tool: This software operates in a pre-processing step, reads the configuration file, allocates the selected software components in the model/ software pool, then loads and distributes them onto the selected resources.

Operator Tool: The operator software is a user-friendly graphical user interface which allows the driving simulator operator to operate the driving simulator by means of selecting the simulation scenarios, starting the simulation, controlling the simulation and stopping the simulation session.

Test Manager Tool: This software component allows the test manager or the driving instructor to observe the simulation during simulation run-time and to trigger some predefined events. These events could be training-relevant events, e.g. a child running in front of the vehicle, or testing-related events, e.g. changing some ADAS control unit parameters.

Simulation Results Database: During the simulation run-time, some selected data and simulation results are logged and stored in a database for analysis. These selected logged data is stored in a simulation results database.

Analysis Tool: The logged simulation results are later used in the post-processing phase for a visual analysis of the driving session. It can be used to replay and prolong the driving session, in order to facilitate the visual analysis of the interplay between vehicle, ADAS and driver in a specific test situation and to track down problems or errors when using ADAS in such a virtual prototyping environment.

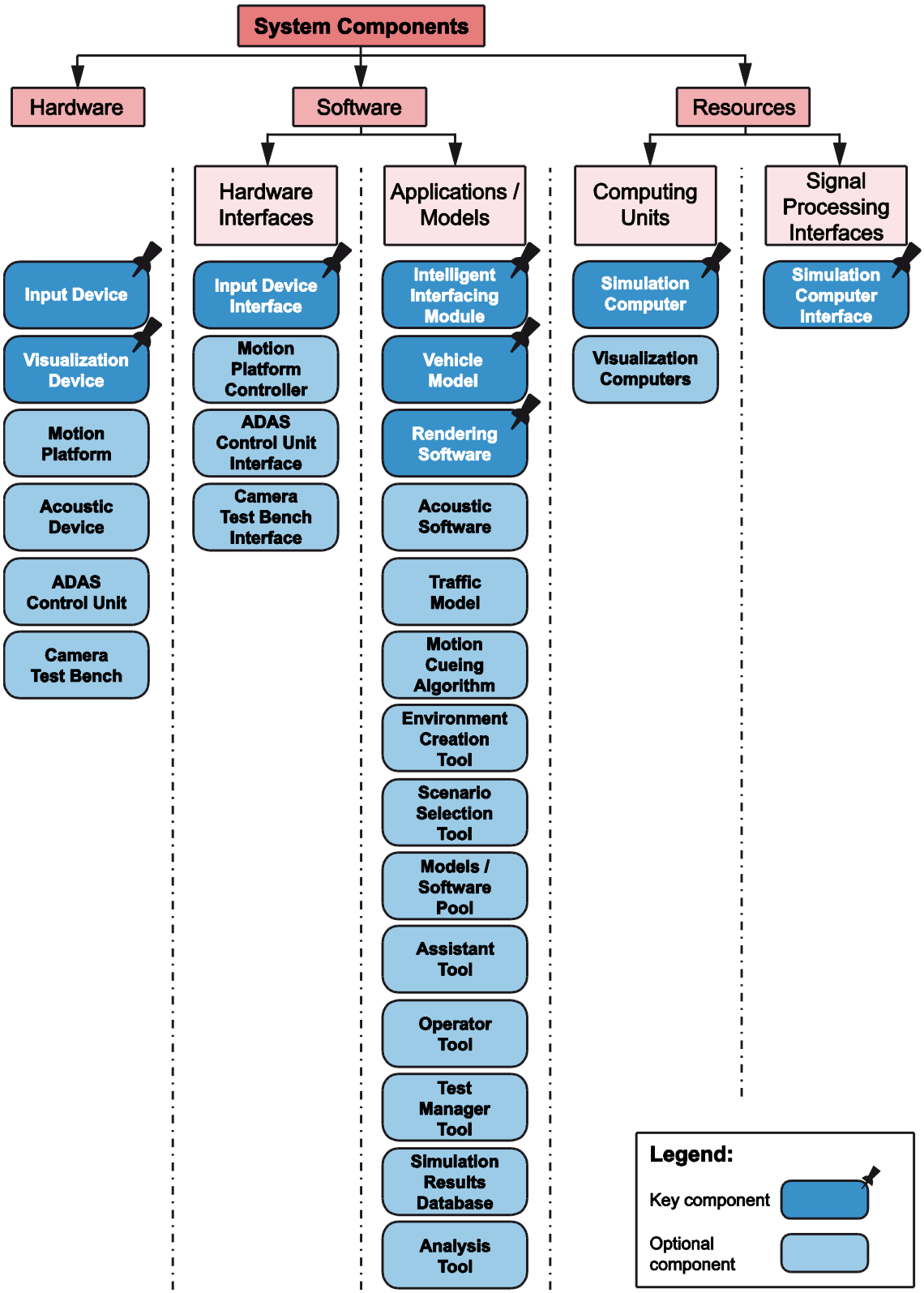


Figure 5-5: The classification of the ADAS reconfigurable driving simulator components.

5.2.4 Phase 3 – Configuration Mechanism Development

The configuration mechanism which has been previously discussed in section 4.5 fits exactly in order to be used within the ADAS reconfigurable driving simulator. In this section, the logic relationships between the reconfigurable driving simulator components are described.

ADAS Reconfigurable Driving Simulator Dependency Matrix

The dependency matrix describes the logic dependency between the identified components of the ADAS reconfigurable simulator. The matrix has a size of 26x26 rows and columns regarding the number of the identified software and hardware components. With the help of the configuration software operation "Add New Component", the dependency matrix is generated, based on the components attribute "Component Logic Dependency row".

Table 5-1: Part of the 26x26 dependency matrix of the identified ADAS reconfigurable driving simulator components.

Dependency Matrix 0 = Independent pair 1 = Dependent pair		Hardware Components										
		Input Device	Visualization Device	Motion Platform	Acoustic Device	ADAS Control Unit	Camera Test-Bench	Vehicle Model	Rendering Software	Acoustic Software	Input Device Interface	Motion Platform Controller
		A	B	C	D	E	F	G	H	I	J	K
Hardware	A. Input Device											
	B. Visualization Device	0										
	C. Motion Platform	1	1									
	D. Acoustic Device	0	0	0								
	E. ADAS Control Unit	0	0	0	0							
	F. Camera Test-Bench	0	0	0	0	0						
	G. Vehicle Model	0	0	0	0	0	0					
	H. Rendering Software	0	1	0	0	0	0	0				
	I. Acoustic Software	0	0	0	1	0	0	0	0			
	J. Input Device Interface	1	0	0	0	0	0	0	0	0		
	K. Motion Platform Controller	0	0	1	0	0	0	0	0	0	0	

ADAS Reconfigurable Driving Simulator Consistency Matrix

The Consistency Matrix describes the logic consistency between the available solution elements. With the help of the configuration software operation "Add New Solution Element", the consistency matrix will be generated based on the solution element attribute "Solution Element Consistency row".

5.2.5 Phase 4 – Solution Elements Deployment

The main objective of the fourth phase is to identify and classify the desired solution elements. Moreover, the solution elements database has to be filled with the required attribute of each solution element and each component. The deployment of the components and solution elements is supported by the configuration software tool.

The identification of the solution elements results in a total number of 67 existing solution elements. These solution elements are deployed in the reconfigurable solution elements database with the help of the configuration tool. The 67 solution elements contain 18 hardware solution elements. For example, the component *input device* has 5 solution elements: "*Mercedes-Benz Actros Tuck cabin", "Smart for Two Cabin", "Logitech USB steering wheel G25"*, *"Logitech USB steering wheel G27"* and *"Keyboard and mouse".* The deployed solution elements also contain 31 software solution elements. For example, the component vehicle model has 3 solution elements: *"dSPACE ASM"*, *"HNI Simple Vehicle Dynamic Model"* and *"Physics Engine based Vehicle Model"*. The deployed solution elements also contain 18 resource solution elements. For example, the component simulation computer has 3 solution elements: *"Simulation Control PC"*, *"dSPACE Real-Time Processor Board"* and *"Stand Alone Simulation Computer".*

5.2.6 Phase 5 – Driving Simulator Variant Generation

The main objective of this phase is to define the configuration selection sequence, as well as the generation of the configuration file and the physical connection plan with the help of the configuration software. By applying the method for defining the configuration selection sequence, which is described in section 4.7.1, this results in that the ADAS reconfigurable driving simulator has 12 selection steps. Figure 5-6 shows the identified selection steps and the number of relationships which have been calculated for each group.

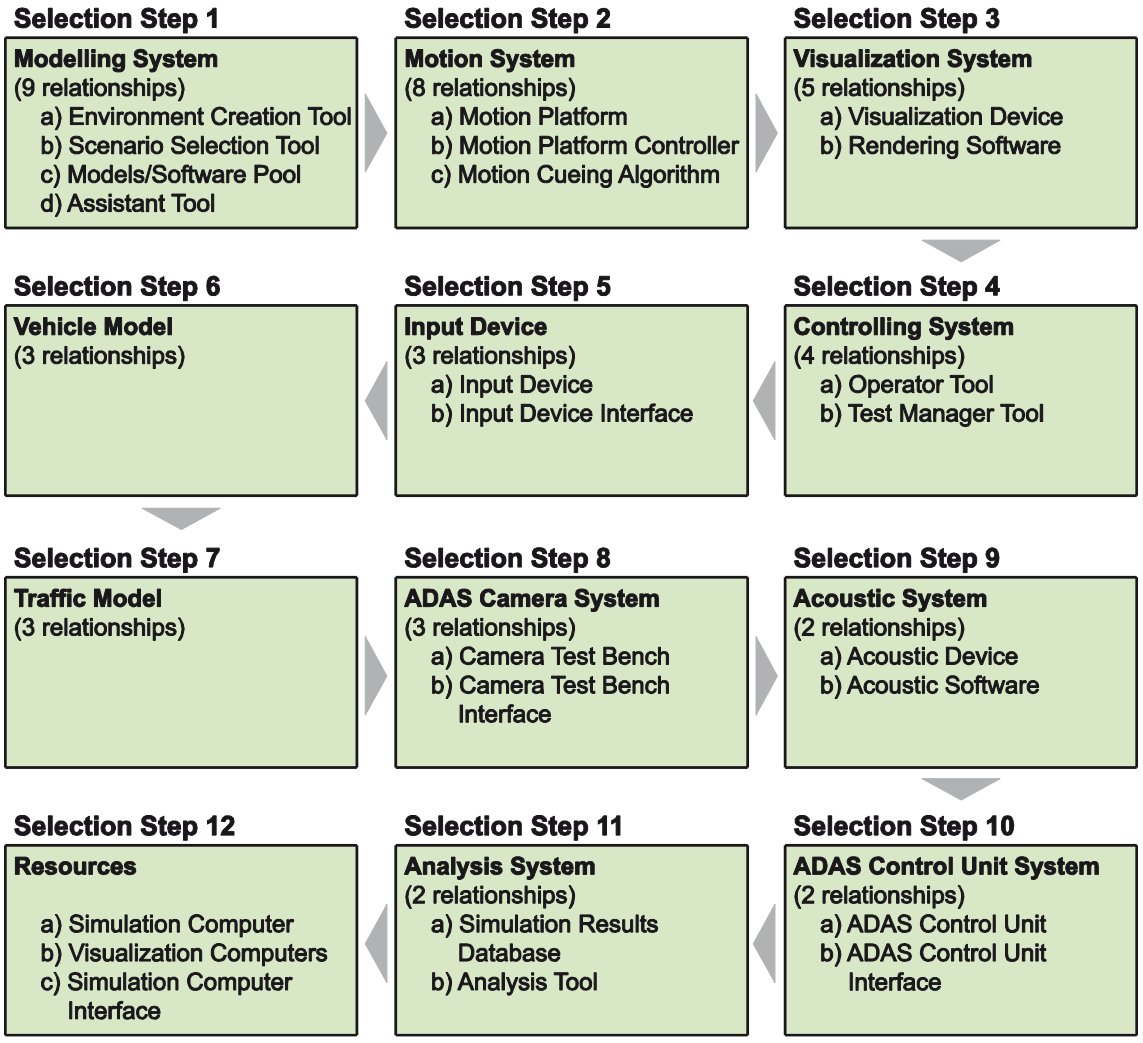
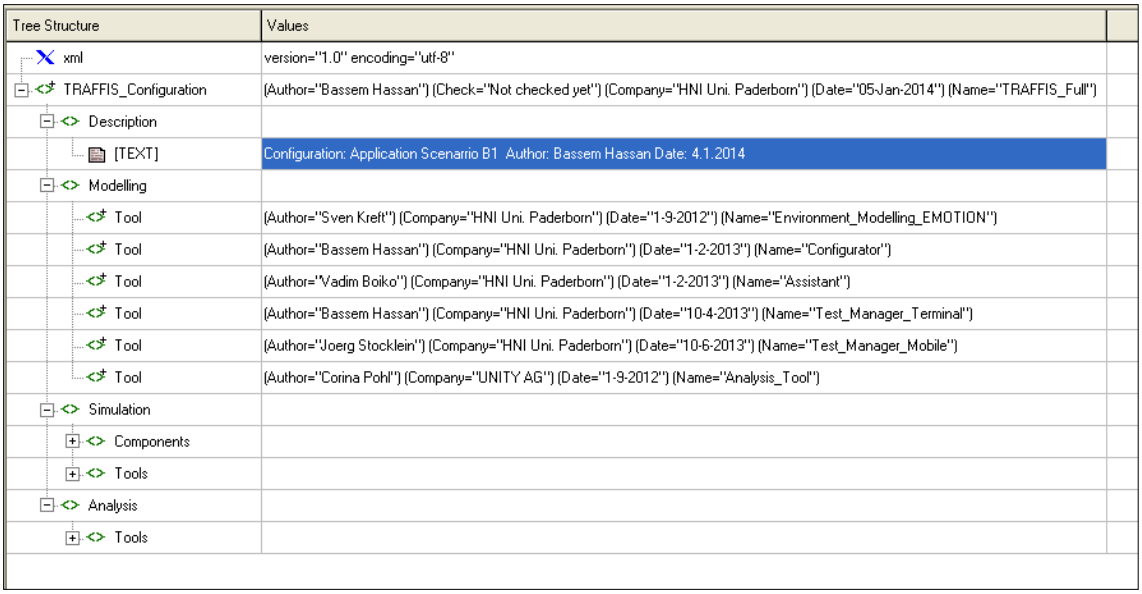


Figure 5-6: Selection steps of the ADAS reconfigurable driving simulator.

Configuration File Structure

In chapter 4 and the case study example, the modelling and analysis tools have been neglected. In this section, the modelling and analysis tools are taken into consideration. Therefore, the configuration file structure for the ADAS reconfigurable driving simulator consists mainly of four sections. The first section contains the general information of the configuration, e.g. configuration name, author, etc. The second section contains information about the selected modelling tools, e.g. the environment creation tool name and version, models/ software pool storage path, etc. The third section contains information about the selected hardware solution elements, software solution elements, interface topology and resources. The fourth part contains information about the analysis tool, simulation results database and a list of the analysis related signals which have to be logged during simulation run-time and have to be analysed. The configuration file will be generated by the configuration tool in an Extensible Mark-up Language (XML) file. Figure 5-7 shows a screenshot of the configuration file of the application scenario B1.



Tree Structure	Values
xml	version="1.0" encoding="utf-8"
TRAFFIS_Configuration	(Author="Bassem Hassan") (Check="Not checked yet") (Company="HNI Uni. Paderborn") (Date="05-Jan-2014") (Name="TRAFFIS_Full")
Description	
[TEXT]	Configuration: Application Scenario B1 Author: Bassem Hassan Date: 4.1.2014
Modelling	
Tool	(Author="Sven Kreft") (Company="HNI Uni. Paderborn") (Date="1-9-2012") (Name="Environment_Modelling_EMOTION")
Tool	(Author="Bassem Hassan") (Company="HNI Uni. Paderborn") (Date="1-2-2013") (Name="Configurator")
Tool	(Author="Vadim Boiko") (Company="HNI Uni. Paderborn") (Date="1-2-2013") (Name="Assistant")
Tool	(Author="Bassem Hassan") (Company="HNI Uni. Paderborn") (Date="10-4-2013") (Name="Test_Manager_Terminal")
Tool	(Author="Joerg Stocklein") (Company="HNI Uni. Paderborn") (Date="10-6-2013") (Name="Test_Manager_Mobile")
Tool	(Author="Corina Pohl") (Company="UNITY AG") (Date="1-9-2012") (Name="Analysis_Tool")
Simulation	
Components	
Tools	
Analysis	
Tools	

Figure 5-7: Configuration file of the application scenario B1.

Physical connections plan

The physical connection plan of the application scenario B1 is illustrated in Figure 5-8. It shows the selected hardware solution elements and the selected resources. The application scenario B1 has 6 hardware solution elements, 7 visualization computers, a real-time processor board and a real-time digital signal processing unit. The selected hardware elements are: the ATMOS motion platform, the truck cabin as an input device, the Headlamp Control Module “HCM” as an ADAS control unit, the HNI camera test-bench as an ADAS camera test-bench, dolby desktop speakers as an acoustic device, as well as 8 projectors and 3 flat screens as display devices. The selected resources are: a real-time processor board, a real-time digital signal processing unit and 7 visualization computers (one master and 6 slaves). Figure 5-8 shows the connection between the hardware and the resources.

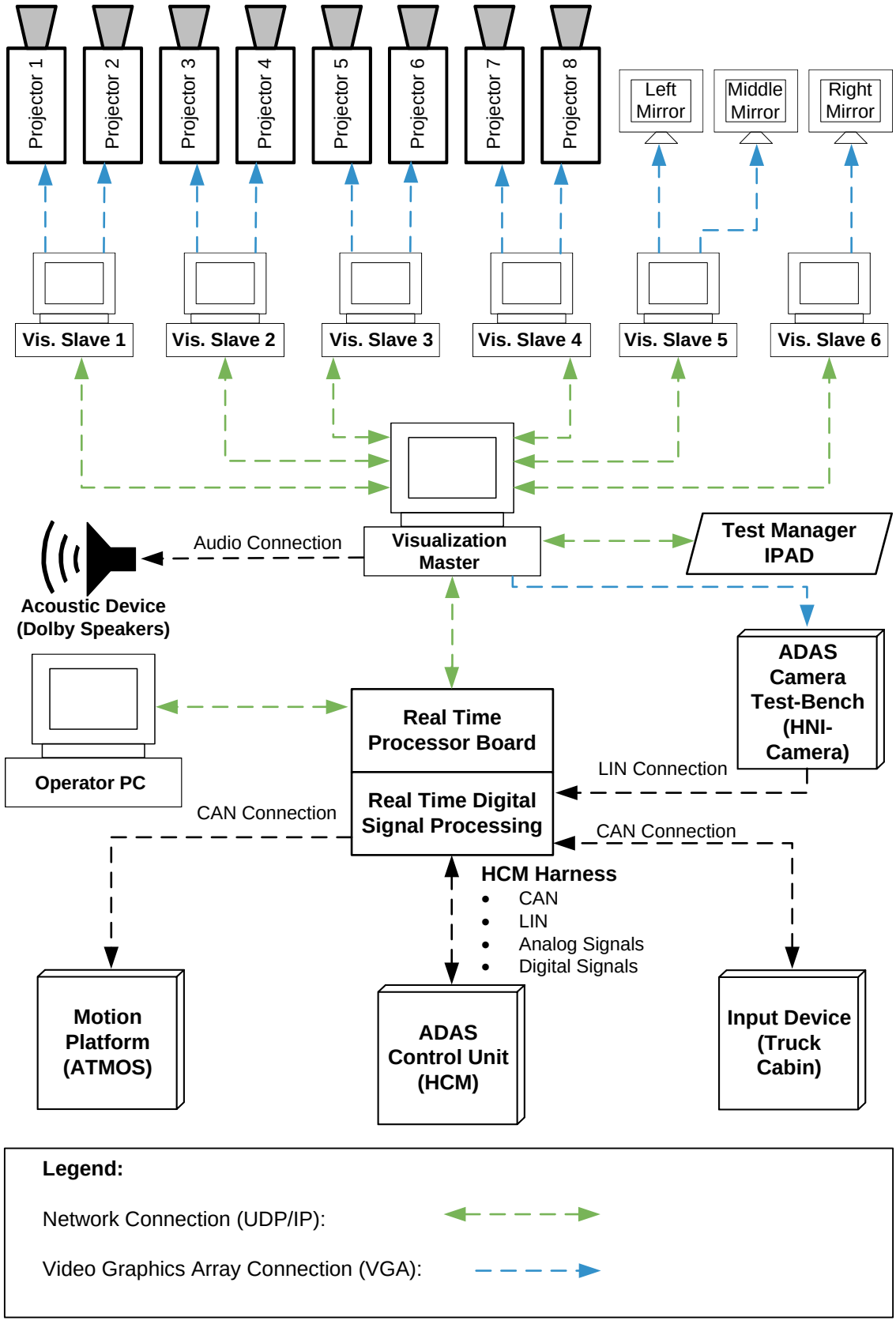


Figure 5-8: Physical connections plan of the application scenario B1.

5.2.7 Phase 6 – System Preparation for Operation

The generated configuration file from phase 5 has to be loaded to the assistant software. The software parses the configuration file, then fetches the selected applications/models components and then loads them on the selected computers. Figure 5-9 shows the graphical user interface of the assistant software.

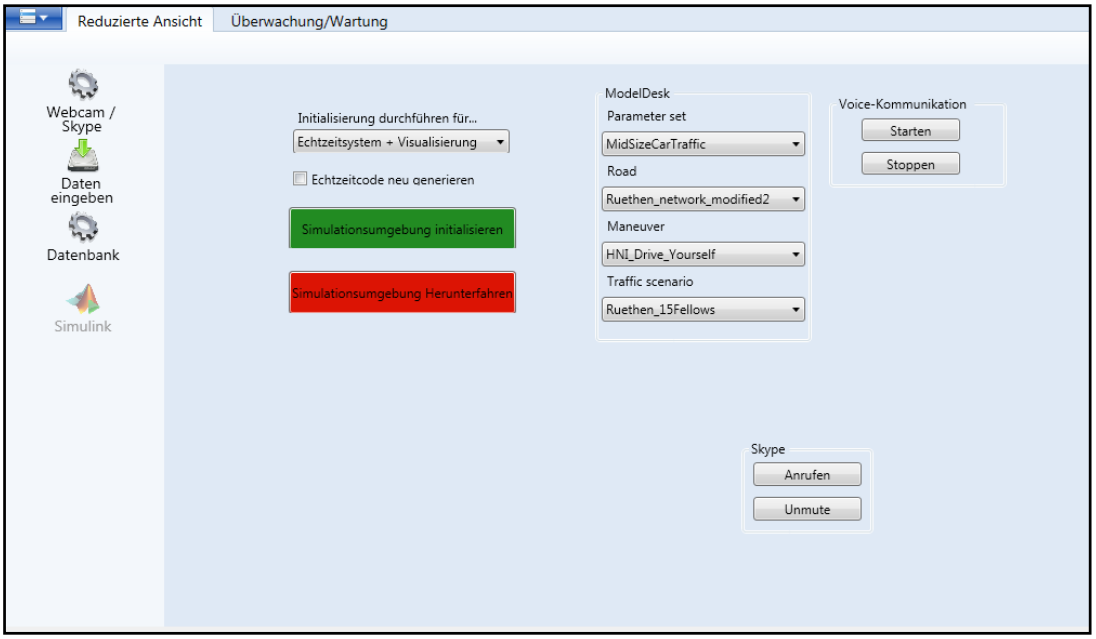


Figure 5-9: Graphical user interface of the assistant software.

Communication during the Simulation Run-time

Intelligent Interfacing Module (IIM) initializes the communication between the selected solution elements based on the interface topology, which is described in the configuration file. As soon as the user starts the simulation, the IIM ensures the communication between all system components during simulation run-time. Figure 5-10 shows the IIM function in the application scenario B1. The IIM exchanges the required input and output from and to the simulation-related software solution elements during run-time. Moreover, IIM connects the software solution elements together, although a part of them runs under hard real-time conditions and other parts runs under soft real-time conditions.

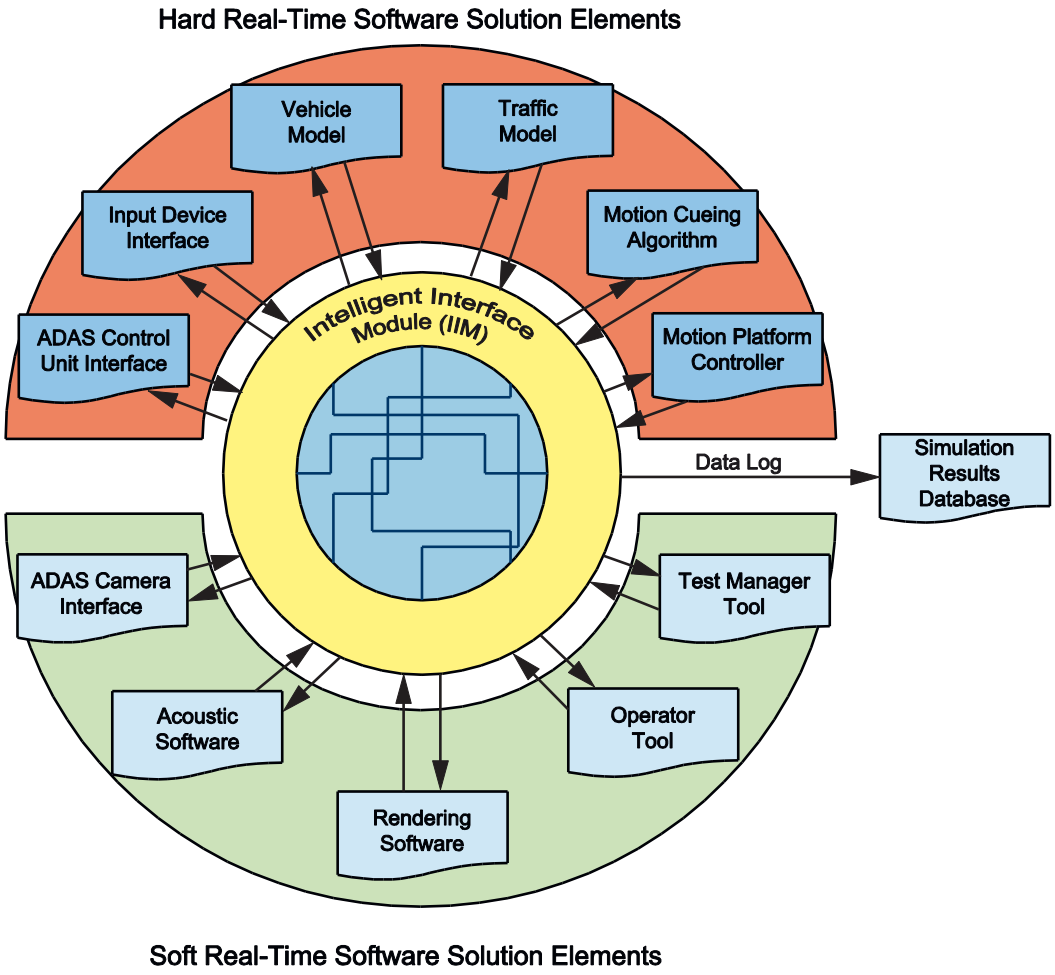


Figure 5-10: Graphical user interface of the assistant software.

5.3 The Created Variants with the Help of the Design Framework

In order to validate the design framework, three ADAS driving simulator variants have been generated with the help of the described procedure model and the implementation prototype of the configuration tool. The three generated ADAS driving simulator variants were generated simply by selecting their desired components and their preferred solution elements.

5.3.1 Configuration 1 – TRAFFIS-Full

The name of the first generated variant is “TRAFFIS-Full”. This variant has the most complex structure and it contains most of the ADAS reconfigurable driving simulator components. This variant is based on the application scenario B1 which is described in section 5.2.2. The main objective of the TRAFFIS-Full variant is testing the real headlamp control module “HCM” control unit in HiL environment. Additionally, the driving simulator motion platform and the real vehicle cabin allow the investigating of the interaction between the driver and the HCM control unit in a Human-in-the-Loop environment. Figure 5-11 shows the TRAFFIS-Full variant.

Figure 5-11: The TRAFFIS-Full variant.

The motion platform which is used in this variant is the ATMOS motion platform. It consists of two dynamical parts with 5 degrees of freedom (DOF). The first dynamical part is the moving platform. It has 2 DOF and is used to simulate the lateral and longitudinal accelerations of the vehicle. It can move in the lateral plane and at the same time, it has the ability to tilt around its lateral axis with a maximum angle of 13.5 degrees and around the longitudinal axis with a maximum angle of 10 degrees. Four linear actuators are used to control the movements in both directions. The second dynamical part is the shaker system, which has 3 DOF to simulate the roll and pitch angular velocities and the vertical acceleration of the vehicle. It is driven by a three drive crank mechanism (three actuators) [HBA+13].

The TRAFFIS-Full driving simulator variant consists of the following simulation-related components and solution elements:

Table 5-2: TRAFFIS-Full driving simulator variant components and solution elements

Hardware	Software	Resources
Motion Platform: ATMOS motion platform	**Motion Platform controller:** ATMOS motion platform controller	**Simulation Computer:** dSPACE Quad-Core real time processor board (ds1006)
	Motion Cueing Algorithm: HNI classical motion cueing	**Visualization Computers:** 7 commercial windows PCs
Visualization Device: 8 Projectors cover 240 degrees horizontal field of view and 3 LED displays to visualize the 3 mirrors	**Rendering Software:** Virtual Night Drive "VND 2.0" visualization module	**Simulation Computer Interface:** dSPACE DSP-Board (ds2211)
Acoustic Device: Dolby Desktop speakers	**Acoustic Software:** Virtual Night Drive "VND 2.0" acoustic module	
ADAS Control Unit: HCM real control unit	**ADAS Control Unit Interface:** HCM interface	
Input Device: SMART-for-two real vehicle	**Input Device Interface:** SMART-for-two interface	
	Vehicle Model: dSPACE ASM	
	Traffic Model: dSPACE ASM-Traffic	

5.3.2 Configuration 2 – TRAFFIS-Portable

The name of the second generated variant is "TRAFFIS-Portable". This driving simulator variant is a stripped-down version of the TRAFFIS-Full variant, which is based on the application scenario D1 which is described in section 5.2.2. The main objectives of the TRAFFIS-Portable variant are traffic safety training as well as illustrating the bene-

fits of ADAS functions. The traffic safety trainings typically take place on site at logistic agencies. Therefore, a portable driving simulator variant with a simple motion platform was needed. Figure 5-12 shows the TRAFFIS- Portable variant.

Figure 5-12: The TRAFFIS-Portable variant.

The TRAFFIS-Portable driving simulator variant consists of the following simulation-related components and solution elements:

Table 5-3: TRAFFIS-Portable driving simulator variant components and solution elements

Hardware	Software	Resources
Motion Platform: Airmotion ride motion platform	**Motion Platform controller:** Airmotion ride motion platform controller	**Simulation Computer:** Windows PC
Visualization Device: Oculus Rift	**Rendering Software:** Virtual Night Drive “VND 2.0” visualization module	**Simulation Computer Interface:** Standard USB and network interfaces
Acoustic Device: Dolby Desktop speakers	**Acoustic Software:** Virtual Night Drive “VND 2.0” acoustic module	
Input Device: Logitech USB steering wheel and pedals (G25)	**Input Device Interface:** Logitech USB steering wheel and pedals interface	
	Vehicle Model: A simple vehicle model based on UNITY 3D physics engine	

	Traffic Model: A simple traffic model based on UNITY 3D physics engine	

5.3.3 Configuration 3 – TRAFFIS-Light

The name of the third generated variant is “TRAFFIS-Light”. This variant has the simplest structure and contains the smallest number of ADAS reconfigurable driving simulator components. This variant is based on the application scenario C1 which is described in section 5.2.2. The main objective of the TRAFFIS-Light variant is testing the main HCM algorithms in the laboratory in a SiL simulation environment. The generated setup is a PC-based simulator with a simple vehicle model and a visualization system. Figure 5-13 shows the TRAFFIS-Light variant.

Figure 5-13: The TRAFFIS-Light variant.

The TRAFFIS-Portable driving simulator variant consists of the following simulation-related components and solution elements:

Table 5-4: TRAFFIS-Light driving simulator variant components and solution elements

Hardware	Software	Resources
Visualization Device: 23" LED screen	**Rendering Software:** Virtual Night Drive "VND 2.0" visualization module integrated with the main HCM function as a SiL	**Simulation Computer:** Windows PC
Acoustic Device: Dolby Desktop speakers	**Acoustic Software:** Virtual Night Drive "VND 2.0" acoustic module	**Simulation Computer Interface:** Standard USB and network interfaces
Input Device: Logitech USB steering wheel and pedals (G27)	**Input Device Interface:** Logitech USB steering wheel and pedals interface	
	Vehicle Model: A simple vehicle model based on UNITY 3D physics engine	
	Traffic Model: A simple traffic model based on UNITY 3D physics engine	

5.4 The Design Framework Validation based on the Requirements

The *Design Framework for Developing a Reconfigurable Driving Simulator*, which was developed during this work, has to be evaluated based on the requirements defined in section 2.7. The next section briefly describes the evaluation of the developed design framework according to each requirement.

R1 – Systematic Procedure: The developed procedure model ensures the systematic of the defined development phases. It consists of two stages: the first one describes the development phases which should be carried out by the developer of the driving simulator and the second one describes the variants' creation phases, which should be carried out by the operator of the driving simulator. Each stage is divided into three phases. Each phase describes the individually required tasks within the phase. Moreover, it describes the used functions and algorithms and the results of each phase.

R2 – Complexity Reduction: The developed procedure model has reduced the system complexity by identifying the main components of the driving simulator. Moreover, the

identified components have been classified in three categories: hardware components, software components and resources. The task-specific variant creation processes, which have to be carried out by the operator of the driving simulator, have been simplified by selecting only the desired solution elements which are like building blocks.

R3 – Domain-Spanning: The developed procedure model has considered the different domains of mechatronic by using the mechatronic systems specification techniques "CONSENS" in order to specify the reconfigurable driving simulator. Based on the specification results, a reconfigurable driving simulator has been developed as a mechatronic system. Additionally, the design framework has been easily used by our interdisciplinary designers during several workshops.

R4 – High Potential for Automation: The developed procedure model has been developed to allow a high potential for automation. The configuration tool's implementation prototype shows that four of the six phases could be implemented in a computer-aided software tool (see Figure 5-2).

R5 – Driving Simulator Reconfigurability: The developed design framework has been easily used to generate task-specific driving simulator variants without in-depth know-how of the structure. By separating the development phases from the variant creation phases, the driving simulator operator can create task-specific driving simulator variants just by selecting the desired solution elements. The variant creation phases do not require any in-depth expertise in the system's entire structure. Moreover, the interfacing between the different solution elements during the simulation run-time is automatically performed based on the configuration file and with the help of the intelligent interface module.

R6 – Reengineering of Existing Driving Simulators: With the help of both case study examples in chapter four, the developed design framework has proven the feasibility of the reengineering of existing driving simulators. The two existing driving simulators of the case study have been redeveloped and their entire solution elements had been used within the variant creation process of the reconfigurable driving simulator.

R7 – Supporting the Development of ADAS: The developed design framework has shown that, with the help of ADAS validation examples in chapter five, the ADAS development, testing and training are fully supported. Additionally, the three created variants of the TRAFFIS project have presented different test environments of the ADAS such as SiL within the TRAFFIS-Light version and HiL within the TRAFFIS-full version.

R8 – Separation of Concerns: The developed configuration tool is an easy-to-use tool. It prevents the operator from dealing with complex procedures such as consistency and compatibility check algorithms. Moreover, it allows the driving simulator developer to easily perform complex tasks (e.g. dealing with the solution elements database) through a graphical user interface. Additionally, it allows the driving simulator operator to per-

form complex tasks (e.g. creating a task-specific driving simulator variant) by selecting the desired solution element through a graphical user interface.

R9 – Modular and Extendable System Structure: The structure of the configuration tool's implementation prototype is modular and extendable. The configuration tool was implemented with the help of the model-view-controller approach (see section 5.1.2). This made the implemented software modular, extendable and easy-to-use.

Conclusion

The *Design Framework for Developing a Reconfigurable Driving Simulator* which was developed during this work completely fulfils all of the defined requirements. Additionally, the design framework has been successfully used to reengineer two existing fixed-structure driving simulators into reconfigurable ones. Moreover, the design framework has been successfully used to develop an ADAS reconfigurable driving simulator and three task-specific driving simulator variants. The developed design framework represents a practicable approach for developing a new generation of driving simulators which have the ability to be easily reconfigured to fulfil different tasks.

6 Summary and Outlook

Driving simulators have been used successfully for decades in different application fields. They vary in their structure, fidelity, complexity and cost from low-level driving simulators to high-level driving simulators. Nowadays, driving simulators are usually developed individually by suppliers and they are developed with a fixed structure to fulfil a specific task. Nevertheless, using a driving simulator in an application field, such as ADAS development, requires several variants of a driving simulator. These variants differ in their structure, in the used solution elements and in the level of detail of the entire models. Therefore, there is a need to develop a reconfigurable driving simulator which allows its operator to easily create different variants without in-depth expertise in the system structure and without the help of the driving simulator's manufacturer.

Driving simulators are complex, interdisciplinary mechatronic systems. Therefore, the development of a reconfigurable driving simulator is a challenge. During the problem analysis, this challenge was analysed, the reconfigurable driving simulator term was defined and the essential requirements of the design framework were identified.

The extensive analysis of the state of the art has shown an existing method for the selection of the driving simulator and previous approaches towards developing reconfigurable driving simulators. The method named "Application Oriented Conception of Driving Simulators for the Automotive Development", developed by NEGELE, allows automotive engineers to formulate the requirements and specifications of a driving simulator for a specific application. Further to this, many driving simulators were investigated, but only seven of them could be identified as possible previous approaches towards developing a reconfigurable driving simulator. The seven identified driving simulators were classified into four categories: low-level, mid-level driving simulators, high-level and multi-level driving simulators. The investigation of the existing methods and driving simulators has shown that, there is no existing method or a developed driving simulator to date which covers all the design framework requirements. Therefore, a need for action was identified.

In order to solve the challenge of developing a reconfigurable driving simulator, *A Design Framework for Developing a Reconfigurable Driving Simulator* was developed to meet the defined requirements and to fulfil the need for action. The design framework consists mainly of the procedure model and the configuration tool. They are briefly described as follows:

- The **procedure model**: This defines the required phases in a hierarchy, in order to develop a reconfigurable driving simulator. Each phase contains complete tasks. These tasks have to be carried out in order to achieve the phase objectives. The procedure model organizes the required tasks in each phase and describes what method or algorithm should be used to fulfil each task. The used methods and algorithms contain existing approaches as well as new approaches, which

were developed during this work. Moreover, the procedure model defines the result of each phase. This is needed as input for the following phases.

- **Configuration tool's** implementation prototype: This organizes the existing driving simulator software and hardware components and their corresponding solution elements in a solution elements database. Moreover, it supports operators of the reconfigurable driving simulator, in order to create driving simulator variants or to reconfigure the existing variants.

The description of the development procedure has been illustrated with the help of two case study driving simulators, which have been developed with a fixed structure. During this work, the case study variants were reengineered and they were converted into reconfigurable driving simulators.

The design framework has been validated with the help of a validation example. The validation example was the development of ADAS reconfigurable driving simulators. They are task-specific driving simulators which are used for the testing and training of ADAS. During the validation, three variants of the reconfigurable driving simulator were successfully developed.

In summary, the developed *Design Framework for Developing a Reconfigurable Driving Simulator* is a comprehensive framework which supports the driving simulator developers in their development of reconfigurable driving simulators. Moreover, it allows the driving simulator operators to easily create task-specific driving simulator variants.

Outlook

The developed *Design Framework for Developing a Reconfigurable Driving Simulator* has considered the driving simulator as a mechatronic system. The procedure model and the configuration tool have been kept general, in order to be applicable for other mechatronic systems. The usage of the developed design framework for other mechatronic systems still has to be investigated.

Additionally, there are some enhancements that have to be carried out for the implementation prototype of the configuration tool. These enhancements are described as follows:

- **Interfacing the configuration tool with the Mechatronic Modeller**: In the first phase of the procedure model, the reconfigurable driving simulator was specified with the help of CONSENS – "*Conceptual Design Specification Technique for the Engineering of Complex Systems".* The driving simulator specification phase was done without the support of the configuration tool. There is a software tool for the computer-aided-modelling called "Mechatronic Modeller", which is programmed based on CONSENS. The Mechatronic Modeller supports the specification's partial models used in this work. These partial models are environment, application scenarios, requirements, functions and active structure [GDN10]. Moreover, the Mechatronic Modeller generates a meta-file which

contains the specification results. The configuration tool should be able to load the generated meta-file by the Mechatronic Modeller, interpret its contents and load the identified driving simulator automatically in the solution elements database.

- **Compatibility check for the resources**: Currently, the compatibility check algorithm checks input and output signals of the software solution elements if they are compatible with each other or not. A further enhancement is to extend the algorithm by checking the resources solution elements by means of the available physical interfaces and the available computing power against the software solution elements which have to run on the resource.
- **Solution elements database**: Currently, the solution elements database is implemented with the help of Microsoft Office Excel, which makes the implementation simple. However, it has some disadvantages. For example, the file size of the solution elements database increases rapidly and the database access time is long. The recommendation is to implement the solution elements database in a professional database tool such as SQL.
- **Selection sequence panels**: Currently, the selection panels of the solution elements have to be manually implemented based on the identified selection sequence. The configuration tool has to calculate the selection sequence and generate their panels automatically.

List of Abbreviations

3D	3-dimensional
ACC	Adaptive Cruise Control
ADAS	Advanced Driver Assistance System
AFS	Adaptive Front-lighting System
ATMOS	Atlas Motion System
CAD	Computer Aided Design
CAN	Controller Area Network
Ch.	Chapter
CONSENS	Conceptual Design Specification Technique for the Engineering of Complex Systems
CPU	Central Processing Unit
CRDU	Create, Read, Update and Delete
D/W	demands and wishes
DAS	Driver Assistance System
DB	Database
DiL	Driver-in-the-Loop
DOF	Degrees of Freedom
DSRC	Dedicated Short Range Communication
ECU	Electronic Control Unit
e.g.	exempli gratia – for example
ESP	Electronic Stability Program
et al.	et alii – and others
etc.	et cetera – etcetera
FBB	Functional Building Block
FDMU	Functional Digital Mock-Up
FFAS	*Fortgeschrittene Fahrerassistenzsysteme*
FMECA	Failure Models, Effects and Critically Analysis

FPGA	Field Programmable gate Arrays
GPU	Graphical Processing Unit
GUI	Graphical User Interface
HCM	Headlamp Control Module
HDMI	High Definition Multimedia Interface
HiL	Hardware-in-the-Loop
HMI	Human-Machine-Interface
HNI	Heinz Nixdorf Institute
HW	Hardware
i.e.	id est – that is
IIM	Intelligent Interfacing Module
ILV	*Institut für Logistik und Verkehr*
LCA	Lane Change Assistance
LCD	Liquid Crystal Display
LED	Light Emitting Diode
LKA	Lane Keeping Assistance
M/O	mandatory/optional
MiL	Model-in-the-Loop
MVC	Model-View-Controller
MMI	Man-Machine Interface
N/A	Not Available
NADS	National Advanced Driving Simulator
OBD	On-Board Diagnostics
OSG	Open Scene Graph
RCP	Rapid Control Prototyping
RMS	Reconfigurable Manufacturing System
RMT	Reconfigurable Machine Tools
RPM	Rounds per Minute

SiL	Software-in-the-Loop
SQL	Structured Query Language
SVD	Simple Vehicle Dynamics
SW	Software
TCP/IP	Transmission Control Protocol / Internet Protocol
TMC	Traffic Message Channel
TRAFFIS	Test- und Trainingsumgebung für fortgeschrittene Fahrerassistenzsysteme - German acronym for Test and Training Environment for Advanced Driver Assistance Systems
TRDS	Toyota Research Driving Simulator
UDP/IP	User Datagram Protocol / Internet protocol
USB	Universal Serial Bus
VDI	*Verein Deutscher Ingenieure* – Association of German Engineers
VGA	Video Graphics Array
VHiL	Vehicle-Hardware-in-the-Loop
VND	Virtual Night Drive
VTI	The Swedish National Road and Transport Research Institute
WiFi	Wireless Fidelity
XML	Extensible Mark-up Language

Bibliography

[All09] ALLERTON, D.: Aerospace Series – Principles of Flight Simulation. Wiley, Chichester, Great Britain, 2009 – ISBN: 978-0-470-75436-8

[And11] ANDERSEN, G.: Sensory and Perceptual Factors in the Design of Driving Simulation Displays. In: Fisher, D.; Caird, J.; Rizzo M.; Lee, J. (Eds.): Handbook of Driving Simulation for Engineering, Medicine, and Psychology. CRC Press Taylor & Francis Group, USA, 2011, pp. 8.1-8.11 – ISBN 978-1-4200-6100-0

[ARC11] ALLEN, R.; ROSENTHAL, T.; COOK, M.: A Short History of Driving Simulation. In: Fisher, D.; Caird, J.; Rizzo M.; Lee, J. (Eds.): Handbook of Driving Simulation for Engineering, Medicine, and Psychology. CRC Press Taylor & Francis Group, USA, 2011, pp. 2.1-2.16 – ISBN 978-1-4200-6100-0

[AWH10] ALTENDORFER, R.; WIRKERT, S.; HEINRICHS-BARTSCHER, S.: Sensor Fusion as an Enabling Technology for Safety-critical Driver Assistance Systems. SAE International Journal of Passenger Cars - Electronic and Electrical Systems, October 19 2010, SAE International, USA, 2010, pp.183-192 – ISSN 0148-7191

[AYR+07] ABDEL-ATY, M.; YAN, X.; RADWAN, E.: Using the UCF driving simulator as a test bed for high risk locations – Technical Report. Florida State Department of Transportation, Tallahassee Research Center, University of Central Florida and Center for Advanced Transportation System Florida department of transportation, Orlando-Florida, USA, 2007

[Bed10-ol] BDTI: Vision-Based Advanced Driver Assistance –TI Hopes You'll Give Its Latest SoCs a Chance, October 22, 2013, http://www.bdti.com/InsideDSP/2013/10/23/TI

[Bir80] BIRKHOFER, H.: Analyse und Synthese der Funktionen technischer Produkte. VDI-Verlag Fortschritts-Bericht VDI-Z, Reihe 1, Nr. 70, Düsseldorf, Germany, 1980

[BKG08] BERSSENBRÜGGE, J.; KREFT, S.; GAUSEMEIER, J.: Using a Virtual Reality-based Night Drive Simulator as a Tool for the Virtual Prototyping of an Advanced Leveling Light System. In: Proceedings of IDETC/CIE 2008 ASME 2008 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, August 3 - 8 2008, New York City, USA

[Bro13] BROWN, M.: Developing with Couchbase Server. O'Reilly Media, Sebastopol, California, USA, February 2013 – ISBN 978-1-4493-3116-0

[BY09] BUCK, E. M.; YACKTMAN, D. A.: Cocoa Design Patterns. Pearson Education 1 September 2009, Boston, USA, 2009

[Car14-ol] CARSIM – QuadDS and HexDS mechanical simulation http://www.carsim.com/, 2014

[CDF+07] COLDITZ, F.; DRAGON, L.; FAUL, R.; MELJNIKOV, D.; SCHILL, V.; UNSELT, T.; ZEEB, E.: Use of Driving Simulators within Car Development. In: Proceedings of Driving Simulation Conference North America, September 12-14 2007, Iowa City, USA

[CH11] CAIRD, J.; HORREY, W.: Twelve Practical and Useful Questions about Driving Simulation. In: Fisher, D.; Caird, J.; Rizzo M.; Lee, J. (Eds.): Handbook of Driving Simulation for Engineering, Medicine, and Psychology. CRC Press Taylor & Francis Group, USA, 2011 – ISBN 978-1-4200-6100-0

[Cha08] CHALLEN, J.: Reality Bytes – John Challen takes a look at Toyota's latest driving simulator. In: Traffic Technology International, UIP, Great Britain, April/Mai 2008 – ISSN 1356-9252

[CM11] CASTRONOVO, A.; MÜLLER, C.: A Schema of Possible Negative Effects of Advanced Driver Assistant Systems. In: Proceedings of the 6th International Driving Symposium on Human Factors in Driver Assessment, Training and Vehicle Design (Driving Assessment-2),

June 27-30 2011, Lake Tahoe, California, USA, Public Policy Center, University of Iowa, 2011

[DAG12] DUMITRESCU, R.; ANACKER, H.; GAUSEMEIER, J.: Design Framework for the Integration of Cognitive Functions into Intelligent Technical Systems. In: Proceedings of 1st Joint International Symposium on System-integrated Intelligence 2012: New Challenges for Product and Production Engineering, 2012, June 17 - 19 2012, Hannover, Germany, pp. 17-20, 2012 – ISBN: 978-3-943104-59-2

[DDS01] DEPARTMENT OF DEFENSE SYSTEMS MANAGEMENT COLLEGE (Ed.): Systems Engineering Fundamentals. Defense Acquisition University Press, Fort Belvoir, Virginia, USA, 2001

[Dum11] DUMITRESCU, R.: Entwicklungssystematik zur Integration kognitiver Funktionen in fortgeschrittene mechatronische Systeme. Dissertation, der Fakultät Maschinenbau der Universität Paderborn, HNI-Verlagsschriftenreihe, Band 286, 2011

[EFG+03] ESPIÉ, S.; FOLLIN, E.; GALLÉE, G.; GANIEUX, D.; HERPERT, J.-M.: Automatic Road Networks Generation Dedicated to Night-Time Driving Simulation. In: Proceedings of Driving Simulation Conference North America, 8.-10. October 2003, Dearborn, Michigan – ISSN 1546-5071

[Eng08] ENGEN, T.: Use and Validation of Driving Simulators. Dissertation, Norwegian University of Science and Technology, Faculty of Engineering Science and Technology, Department of Civil and Transport Engineering, 2008

[EUP03] EUROPEAN UNION PARLIAMENT: Directive 2003/59/EC of the European Parliament and of the Council on the initial qualification and periodic training of drivers of certain road vehicles for the carriage of goods or passengers, amending Council Regulation (EEC) No 3820/85 and Council Directive 91/439/EEC and repealing Council Directive 76/914/EEC, Official Journal of the European Union, 15 July 2003

[FCR+11] FISHER, D.; CAIRD, J.; RIZZO M.; LEE, J.: Handbook of Driving Simulation for Engineering, Medicine, and Psychology – An Overview. In: Fisher, D.; Caird, J.; Rizzo M.; Lee, J. (Eds.): Handbook of Driving Simulation for Engineering, Medicine, and Psychology. CRC Press Taylor & Francis Group, USA, 2011, pp. 1.1-1.16 – ISBN 978-1-4200-6100-0

[Fes14-ol] FESTO AG & CO. KG: Airmotion ride – Fahren und Fliegen mit einem 6-Achsen Full Motion Simulator: http://www.festo.com/rep/de_corp/assets/pdf/Airmotion_ride_de.pdf, 2014

[FSA+11] FISCHER, M.; SEHAMMAR, H.; AUST, M. L.; NILSSON, M.; LAZIC, N.; WEIEFORS, H.: Advanced driving simulators as a tool in early development phases of new active safety functions. In: Proceedings of the 3rd International Conference on Road Safety and Simulation September 14-16 2011, Indianapolis, USA, 2011

[FSS+13] FILIPPO, F.; STORK A.; SCHMEDT, H.; BRUNO, F.: A modular architecture for a driving simulator based on the FDMU approach. International Journal on Interactive Design and Manufacturing (IJIDeM), Springer-Verlag, March 09 2013, Paris, France, 2013, ISSN 1955-2513

[GAC+13] GAUSEMEIER, J.; ANACKER, H.; CZAJA, A.; WAßMANN, H.; DUMITRESCU, R.: Auf dem Weg zu intelligenten technischen Systemen. In: Gausemeier, Jürgen; Dumitrescu, Roman; Rammig, Franz-Josef; Schäfer, Wilhelm; Trächtler, Ansgar (Hrsg.): 9. Paderborner Workshop Entwurf mechatronischer Systeme, April. 2013, HNI-Verlagsschriftenreihe, Paderborn Band 310, Paderborn, 2013, pp. 11-47 – ISBN 978-3-942647-29-8

[GB11] GREENBERG, J.; BLOMMER, M.: Physical Fidelity of Driving Simulators. In: Fisher, D.; Caird, J.; Rizzo M.; Lee, J. (Eds.): Handbook of Driving Simulation for Engineering, Medicine, and Psychology. CRC Press Taylor & Francis Group, USA, 2011, pp. 7.1-7.34 – ISBN 978-1-4200-6100-0

[GCB+06] GREENBERG J.; CURRY, R.; BLOMMER, M.; KOZAK, K.; ARTZ, B.; CATHEY, L.; KAO, B.: The validity of last-second braking and steering judgments In: Proceedings of Driving Simula-

tion Conference Europe, October 4-6 2006, Paris, France, 2006, pp. 143-153 – ISBN 2-85782-641-9

[GDN10] GAUSEMEIER, J.; DOROCIAK, R.; NYßEN, A.: The Mechatronic Modeller – A Software Tool for Computer-Aided Modeling of the Principle Solution of an Advanced Mechatronic System. In: Proceedings of the 11th International Workshop on Research and Education in Mechatronics, September 9-10 2010, Ostrava, the Czech Republic, 2010 – ISBN 88-88412-33-6

[GEK01] GAUSEMEIER, J.; EBBESMEYER, P.; KALLMEYER, F.: Produktinnovation – Strategische Planung und Entwicklung der Produkte von morgen. Carl Hanser Verlag München, 2011 – ISBN 3-446-21631-6

[GFD+09] GAUSEMEIER, J.; FRANK, U.; DONOTH, J.; KAHL, S.: Specification technique for the description of self-optimizing mechatronic systems. Research In: Research in Engineering Design, November 2009, Volume 20, Issue 4, Springer, London, 2009, pp. 201-223 – ISSN 0934-9839

[GGT13] GAUSEMEIER, J.; GAUKSTERN, T.; TSCHIRNER, C.: Systems Engineering Management Based on a Discipline-Spanning System Model. In: C.J.J. Paredis, C. Bishop, D. Bodner (Eds.): Conference on Systems Engineering Research (CSER'13), 19-22 March 2013, Atlanta, USA, 2013

[GKR03] GUE, D.; KLEE, H.; RADWAN, E.: Comparison of Lateral Control in a Reconfigurable Driving Simulator. In: Proceedings of Driving Simulation Conference North America, October 8-10 2003, Dearborn, USA, Michigan, 2003, ISSN 1546-5071

[GPD+06] GIETELINK, O.; PLOEG, J.; DE SCHUTTER, B.; VERHAEGEN, M.: Development of advanced driver assistance systems with vehicle hardware-in-the-loop simulations. In: Vehicle System Dynamics, July 2006, Volume 44, Issue 7, 2006, pp. 569–590 – ISSN: 0042-3114

[Grä04] GRÄßLER, I.: Kundenindividuelle Massenproduktion. Springer-Verlag Berlin, 2004 – ISBN 978-3-642-18681-3

[GYA02] GOLIAS, J.; YANNIS, G.; ANTONIOU, C.: Classification of driver assistance systems according to their impact on road safety and traffic efficiency. In Transport Reviews – Journal of Intelligent Transportation Systems, Taylor & Francis, Volume 22, Issue 2, 2002, pp. 179-196

[He06] HE, Y.: NADS miniSim Driving Simulator – Technical Report, The University of Iowa – National Advanced Driving Simulator, Document ID: N06-025, September 2006, Iowa City, USA, 2006

[HKB+11] HUMMEL, T.; KÜHN, M.; BENDE, J.; LANG, A.: Advanced Driver Assistance Systems – An investigation of their potential safety benefits based on an analysis of insurance claims in Germany. German Insurance Association – Insurers Accident Research, Research Report FS 03, Berlin, 2011

[HN07] HUESMANN, A.; NAUDERER, J.: Applications to driving simulation and their requirements to the tool In: Proceedings of the Second Motion Simulator Conference, September 19-20 2007, Braunschweig, Germany, 2007

[HS11] HANCOCK, P.; SHERIDAN, T.: The Future of Driving Simulation. In: Fisher, D.; Caird, J.; Rizzo M.; Lee, J. (Eds.): Handbook of Driving Simulation for Engineering, Medicine, and Psychology. CRC Press Taylor & Francis Group, USA, 2011, pp. 4.1-4.10 – ISBN 978-1-4200-6100-0

[HTF96] HARASHIMA, F.; TOMIZUKA, M.; FUKUDA, T.: Mechatronics – „What Is It, Why and How?" An Editorial. In: IEEE/ASME Transactions on Mechatronics 1(1), 1996

[IHK07] IHK-PROJEKTGESELLSCHAFT MBH (Der regionalen Dienstleister für Bildung und Projekte): Fragen-Antwortkatalog zur Umsetzung des Berufskraftfahrer-Qualifikationsgesetzes

(BKrFQG) und der Berufskraftfahrer-Qualifikations-Verordnung (BKrFQV), Ergebnis eines Ländergesprächs am 27.11.2007 in Duisburg, 2007, Germany

[Jam11] JAMSON, H.: Cross-Platform Validation Issues. In: Fisher, D.; Caird, J.; Rizzo M.; Lee, J. (Eds.): Handbook of Driving Simulation for Engineering, Medicine, and Psychology. CRC Press Taylor & Francis Group, USA, 2011, pp. 12.1-12.13 – ISBN 978-1-4200-6100-0

[Kan11] KANTOWITZ, B.: Using Driving Simulators Outside of North America. In: Fisher, D.; Caird, J.; Rizzo M.; Lee, J. (Eds.): Handbook of Driving Simulation for Engineering, Medicine, and Psychology. CRC Press Taylor & Francis Group, USA, 2011, pp. 3.1-3.14 – ISBN 978-1-4200-6100-0

[Kau03] KAUßNER, A.: Dynamische Szenerien in der Fahrsimulation. Dissertation, Fakultät für Mathematik und Informatik, Julius-Maximilians-Universität Würzburg, 2003

[KCF+10] KOSCHER, K.; CZESKIS, A.; ROESNER, F.; PATEL, S.; KOHNO, S.; CHECKOWAY, S.; MCCOY, D.; KANTOR, B.; ANDERSON, D.; SHACHAM, H.; SAVAGE, S.: Experimental Security Analysis of a Modern Automobile. In: Proceedings of 31st IEEE Symposium on Security and Privacy, May 16-19 2010 – Berleley/Oakland, California, USA, 2010, pp. 447-462 – ISBN 978-0-7695-4035-1

[KG11] KEARNEY, J.; GRECHKIN, T.: Scenario Authoring. In: Fisher, D.; Caird, J.; Rizzo M.; Lee, J. (Eds.): Handbook of Driving Simulation for Engineering, Medicine, and Psychology. CRC Press Taylor & Francis Group, USA, 2011, pp. 6.1-6.12 – ISBN 978-1-4200-6100-0

[KGG+11a] KREFT, S.; GAUSEMEIER, J.; GRAFE, M.; HASSAN, B.: Automatisierte Trassierung virtueller Straßen auf Basis von Geo-Informationssystemen. In: Gausemeier, J.; Grafe, M. (Eds.): 10. Paderborner Workshop "Augmented & Virtual Reality in der Produktentstehung", 19. - 20. Mai 2011, HNI-Verlagsschriftenreihe, Band 295, Paderborn, 2011, pp. 253-266 ISBN 978-3-942647-14-4

[KGG+11b] KREFT, S.; GAUSEMEIER, J.; GRAFE, M.; HASSAN, B.: Automated Generation of Virtual Roadways based on Geographic Information Systems. In: Proceedings of the ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, August 29 - 31 2011, Washington DC, USA

[KOM+09] KLEIN, T.; ORTMANN, S.; MÜLLER, J.; RADIMIRSCH, M.; HAUPTVOGEL, A.: Funktionsentwicklung für Fahrerassistenzsysteme – Modellbasierte Entwicklung und innovative Simulationswerkzeuge zur Sicherung des Wettbewerbsvorsprungs. In: 14. Internationaler Kongress Elektronik im Kraftfahrzeug der VDI-Gesellschaft Fahrzeug- und Verkehrstechnik, 7. - 8. Oktober 2009, Baden-Baden, VDI Verlag, VDI-Berichte 2075, 2009 – ISSN 0083-5560

[Kre12] KREFT, S.: Systematik zur effizienten Bildung geospezifischer Umgebungsmodelle für Fahrsimulationen. Dissertation, der Fakultät Maschinenbau der Universität Paderborn, HNI-Verlagsschriftenreihe, Band 305, 2012

[Lan00] LANGLOTZ, G.: Ein Beitrag zur Funktionsstrukturentwicklung innovativer Produkte. Forschungsberichte aus dem Institut für Rechneranwendung in Planung und Konstruktion RPK der Universität Karlsruhe, Shaker Verlag, 2000

[Mau09] MAURER, M.: Entwurf und Test von Fahrerassistenzsystemen. In: Winner, H.; Hakuli, S.; Wolf, G. (Hrsg.): Handbuch Fahrerassistenzsysteme – Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort. Vieweg+Teubner Verlag, Wiesbaden, 2009

[MDR91] MCCONAILL, P.; DREWS, P.; ROBROCK. P. (Eds.): Mechatronics and Robotics. ICS Press Amsterdam, 1991

[Mey07] MEYWERK, M.: CAE-Methoden in der Fahrzeugtechnik. Springer-Verlag, Berlin, 2007 – ISBN 978-3-540-49866-7

[MSK+11] MÄKINEN, T.; SCHULZE, M.; KRAJZEWICZ, D.; GAUGEL, T.; KOSKINEN, S.: Driving Implementation and Evaluation of C2X Communication Technology in Europe – DRIVE C2X

methodology framework. Technical Report Version 1.0, Deliverable D22.1, June 30 2011, Sindelfingen, Germany, 2011

[MV34] MILES, G.H.; VINCENT D.F.: The Institute's tests for motor drivers. The Human Factor, Volume VIII (7-8), 1934, pp. 245-257 – ISSN 0301-7397

[NAD10] NATIONAL ADVANCED DRIVING SIMULATOR – Overview 2010, The University of Iowa – National Advanced Driving Simulator, Iowa City, USA, 2010

[NAD14-ol] NATIONAL ADVANCED DRIVING SIMULATOR: http://www.nads-sc.uiowa.edu/, 2014

[NDW09] NEUMANN-COSEL, K.; DUPUIS, M.; WEISS, C.: Virtual Test Drive Provision of a consistent tool-set for [D,H,S,V]-in-the-loop. In: Proceedings of Driving Simulation Conference Europe, February 4-6 2009, Monaco, France – ISBN 978-2-85782-671-2

[Neg07] NEGELE, J.: Anwendungsgerechte Konzipierung von Fahrsimulatoren für die Fahrzeugentwicklung. Dissertation, Fakultät für Maschinenwesen der Technischen Universität München, 2007

[Nor94] NORDMARK, S.: Driving simulators, trends and experiences. In: Proceedings of Driving Simulation Conference, January 11-14 1994, Paris, France, 1994

[Ocu14-ol] OCULUS RIFT – The Developer Kit: http://www.oculus-rift.com, 2014

[Ook14-ol] OOKABO – Free Picture of Everything on Earth (The first known flight simulator): http://ookaboo.com/o/pictures, 2014

[Oxf14-ol] OXFORD DICTIONARY OF ENGLISH: http://www.oxforddictionaries.com/, 2014

[PBF+07] PAHL, G.; BEITZ, W.; FELDHUSEN, J.; GROTE, K.-H.: Konstruktionslehre – Grundlagen erfolgreicher Produktentwicklung – Methoden und Anwendung. Springer- Verlag, Berlin, 7. Auflage, 2007

[RF09] RÖGLINGER, S.; FACCHI, C.: How Can Car2X-Communication Improve Road Safety A Statistical Based Selection and Discussion of Feasible Scenarios. Heft Nr. 15 aus der Reihe „Arbeitsberichte – Working Papers", Ingolstadt, Germany, April 2009 ISSN 1612-6483

[Res09] RESPONSE 3 CONSORTIUM (Ed.): Code of Practice for the Design and Evaluation of ADAS. Information Society Technologies, Version 5.0, 2009

[Sch07] SCHMIDT, C.: How to Make an AFS System Predictive: ADASIS Interface Implementation. In: Proceedings of the 7th International Symposium on Automotive Lighting, Darmstädter Lichttechnik Series, Volume 12, September 25-26 2007, Darmstadt, Germany. 2007, pp. 482-484 – ISBN 978-3-8316-0711-2

[Sch89] SCHWEITZER, G.: Mechatronik-Aufgaben und Lösungen. Fortschrittsbericht, VDI Nr. 787. VDI, Düsseldorf, 1989

[Slo08] SLOB, J.: State-of-the-Art Driving Simulators – a Literature Survey, Internal Report DCT 2008.107, Department of Mechanical Engineering, Eindhoven University of Technology, Netherlands, 2008

[Str05] STRAUS, S.: New, improved, comprehensive, and automated driver's license test and vision screening system, Final Report 559(1), Arizona Department of Transportation, May 2005, Phoenix, Arizona, USA, 2005

[SW08] SIDDIQI, A.; DE WECK, O.L.: Modeling Methods and Conceptual Design Principles for Reconfigurable Systems. Journal of Mechanical Design, Volume 130, Issue 10, October 2008, ASME, College Park, Maryland, USA, 2008

[Trä05] TRÄCHTLER, A.: Integrierte Fahrdynamikregelung mit ESP, aktiver Lenkung und aktivem Fahrwerk. at – Automatisierungstechnik 53(1), Oldenbourg Wissenschaftsverlag, 2005, pp. 237-251 – ISBN 978-3-8348-0109-8

[TH13a] TAN, Y.; HASSAN, B.: A Method for Testing Camera Based Advanced Driving Assistance Systems. In: Proceedings of ISAM (IEEE International Symposium on Assembly and Task Planning), July 2 – August 2 2013, Xi'an, China, 2013, pp. 151-154 – ISBN 978-1-4799-1656-6

[TH13b] TAN, Y.; HASSAN, B.: A Concept of Camera test-bench for testing Camera Based Advanced Driver Assistance Systems. In: Proceedings of IDETC/CIE 2013 ASME 2013 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, August 4 - 7 2013, Portland, USA, 2013 - ISBN: 978-0-7918-5586-7

[TTV14-ol] TRANSPORTATION TECHNOLOGY VENTURES – OSS Driving Simulators SimWiki, Other simulators: http://www.transportationtechnologyventures.com, 2014

[VDI2206] VDI- GESELLSCHAFT ENTWICKLUNG KONSTRUKTION VERTRIEB (Hrsg.): VDI-Richtlinie 2206 „Entwicklungsmethodik für mechatronische Systeme“, Beuth-Verlag, Berlin, 2004<Angabe zur Norm bzw. Richtlinie>

[VDI4005] VDI- GESELLSCHAFT ENTWICKLUNG KONSTRUKTION VERTRIEB (Hrsg.): VDI-Richtlinie 4005: Einflüsse von Umweltbedingungen auf die Zuverlässigkeit technischer Erzeugnisse – Mechanische Einflüsse der Umwelt, November 1983

[VG12] VAßHOLZ, M.; GAUSEMEIER, J.: Cost-Benefit Analysis – Requirements for the Evaluation of Self-Optimizing Systems. In: Proceedings of the 1st Joint International Symposium on System Integrated Intelligence 2012 – New Challenges for Product and Production Engineering, June 27-29 2012, Hannover, Germany, 2012, pp. 14-16

[Wei92] WEIßMANTEL, H.: Mechatronik-Elektromechanik-Feinwerktechnik. In: VDI-Workshop, Braunschweig, Germany, 1992

[WH09] WEINBERG, G.; HARSHAM, B.: Developing a Low-Cost Driving Simulator for the Evaluation of In-Vehicle Technologies. In: Proceedings of the First International Conference on Automotive User Interfaces and Interactive Vehicular Applications (AutomotiveUI 2009) September 21-22 2009, Essen, Germany, 2009

[WW71] WEIR, D.H.; WOJCIK, C.: Simulator studies of the driver´s dynamic response in steering control tasks. Highway Research Record, Washington DC, USA, 1971, 364, pp. 1-15 – ISSN 0073-2206

[YS08] YOSHIMOTO K.; SUETOMI, T.: The History of Research and Development of Driving Simulators in Japan. Journal of Mechanical Systems for Transportation and Logistics, Volume 1, Issue 2, 2008, pp. 159-169

[Zee10] ZEEB, E.: Daimler’s new full-scale, high-dynamic driving simulator – A technical overview. In: Proceedings of Driving Simulation Conference Europe 2010, September 9-10 2010, Paris, France, pp. 157-165 – ISBN 978-2-85782-685-9

[Zwi89] ZWICKY, F.: Morphologische Forschung – Wesen und Wandel materieller und geistiger struktureller Zusammenhänge. Schriftenreihe der Fritz-Zwicky-Stiftung, Band 4, Verlag Baeschlin, Glarus, 1989 – ISBN 978-3-8135-0314-2

Appendix

Contents

Page

A1 Amendments to the Design Framework (Chapter 4)

In this appendix some additional figures and tables are presented regarding the results of the procedure model's phases which are explained in chapter four.

A1.1 Specification Results of the Case Study – Variant 2

In this section, the specification results (phase 1) of the case study variant 2 are presented.

A1.1.1 The Environment Model of the Variant 2

Figure A-1 illustrates the environment model of the case study – variant 2. The system under development is illustrated as a blue hexagon in the centre of the figure. The four environment elements are illustrated as yellow hexagons, and the interaction flows between the system and its environment are illustrated as arrows.

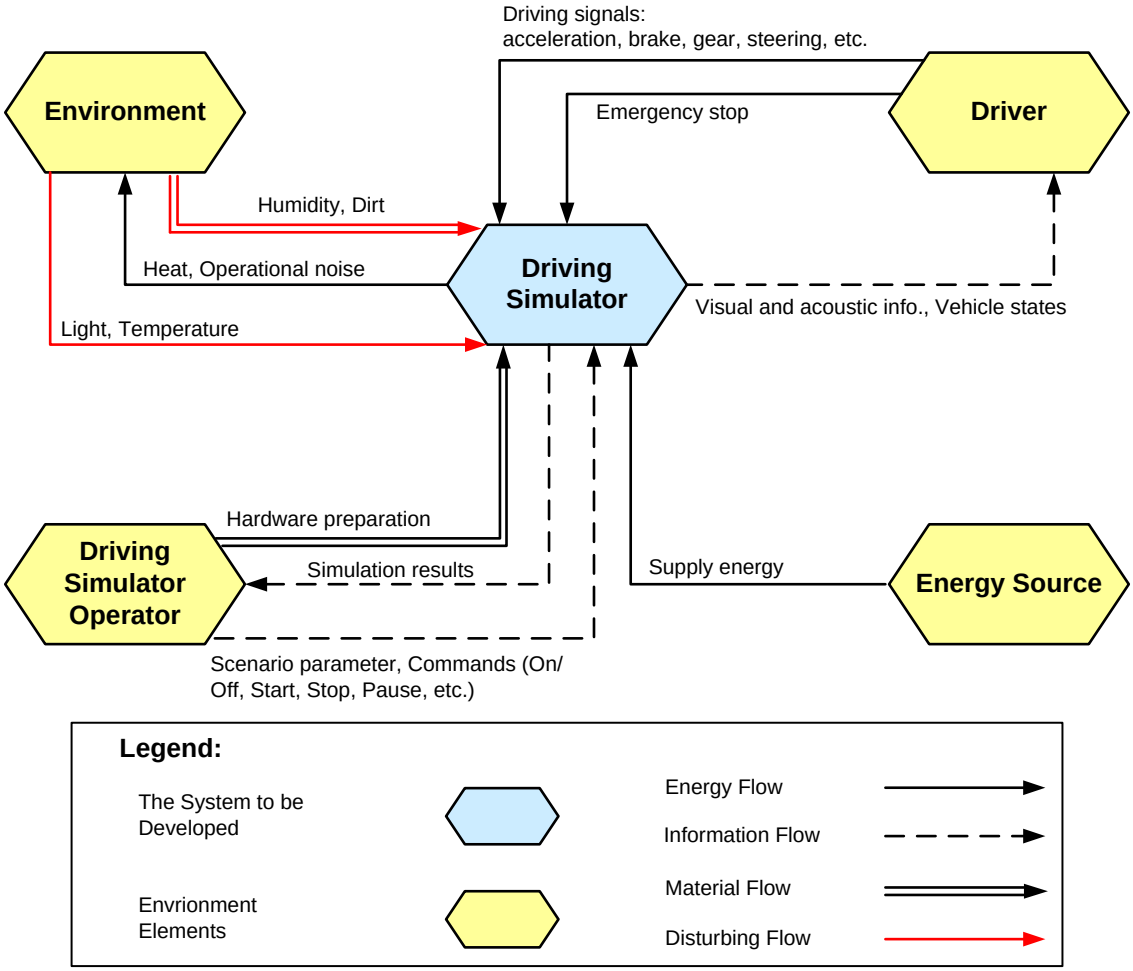


Figure A-1: Environment model of the case study – variant 2.

A1.1.2 The Application Scenarios of Variant 2

Figure A-2 shows the second application scenario's profile page "Virtual Drive on a PC" regarding the case study variant 2. This variant is a fixed-base driving simulator that means it does not have a motion platform. The application scenario is illustrated in a profile page which is adapted to the reconfigurable driving simulator approaches. It contains a short description of the system's normal operation, as well as the desired set-up in form of solution-neutral hardware and software components and a simple sketch.

Status: 1.12.2013	**Application Scenario:** Virtual Drive on a PC	A2	Page: 1

Description:

This is a virtual test drive in a driving simulator. The driver sits in front of a PC. The PC has an input device, which has a steering wheel and three pedals (acceleration, brake and clutch pedals). This input device allows the driver to drive and control a virtual vehicle in a virtual environment. As soon as the simulation starts, based on the driver inputs through the input device the vehicle model calculates the vehicle movements in the virtual environment. During each sampling cycle the vehicle model updates the position and orientation of the vehicle chassis and calculates the engine speed. Based on the new position and orientation calculated by the vehicle model. The rendering software visualizes the virtual environment based on the new vehicle position and orientation perspective and displays the rendered frame on a display device. The acoustic software calculates the engine sound based on the engine speed and generates tone by the acoustic device.

Setup description:

Hardware	Software
Input Device	Vehicle Model
Visualization Device	Visualization Rendering Software
Acoustic System	Acoustic Software

Sketch:

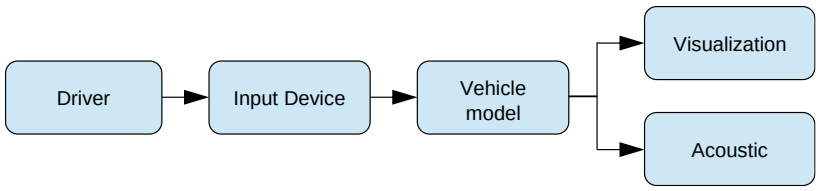


Figure A-2: Application scenario example for the case study – variant 2.

A1.1.3 The Functions Hierarchy of Variant 2

The main function of the case study – variant 2 is to perform a test drive. In order to achieve this function, the driving simulator has to visualize virtual scenes, simulate sound, simulate a virtual vehicle and drive the virtual vehicle through a virtual environment. Figure A-3 shows the functions hierarchy of the first variant.

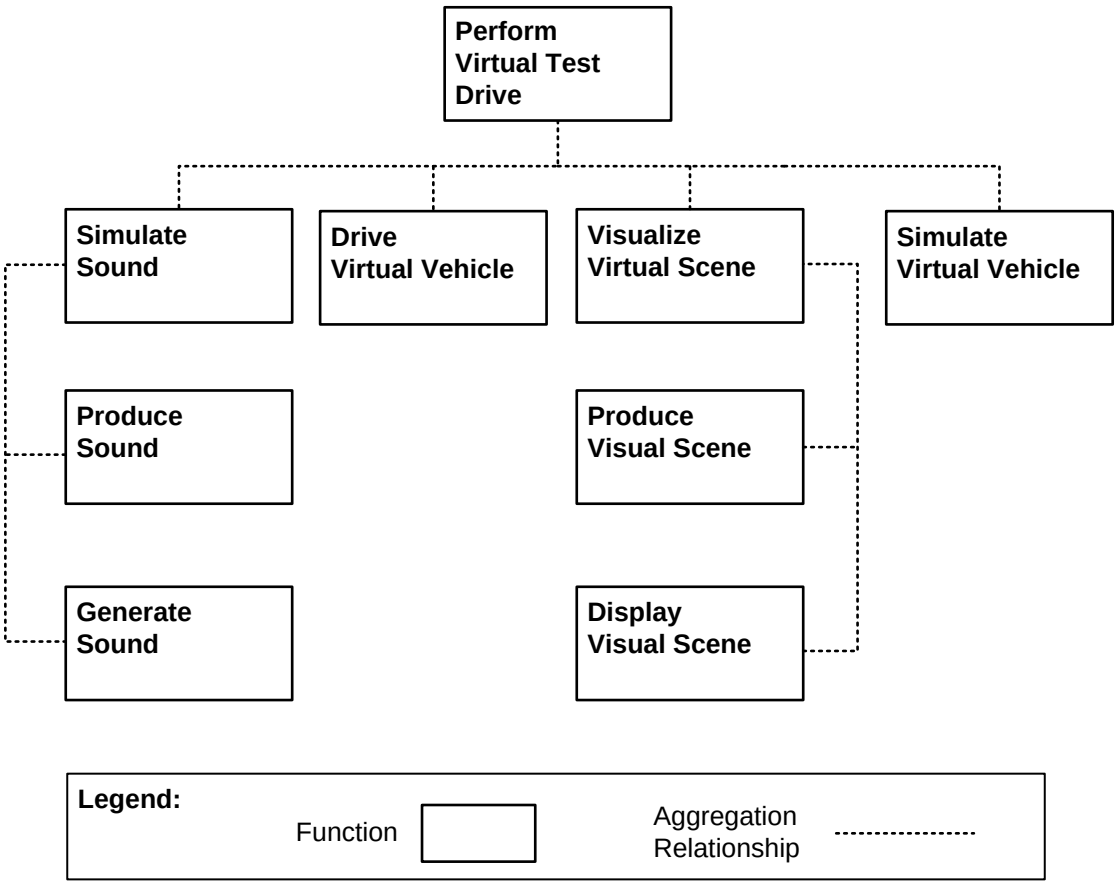


Figure A-3: Functions model of the case study – variant 2.

A1.1.4 The Active Structure of Variant 2

Figure A-4 shows the active structure specification results for the case study variant 2. The active structure consists of eight system elements (components): five of them are software components labelled with (SW), and three hardware components labelled with (HW). Moreover, one of the environment elements (Driver) illustrates an example of the interaction between the entire components of system and an environment element. Six of the eight components could be grouped into 4 groups e.g. the rendering software and the visualization device hardware compose the visualization system group.

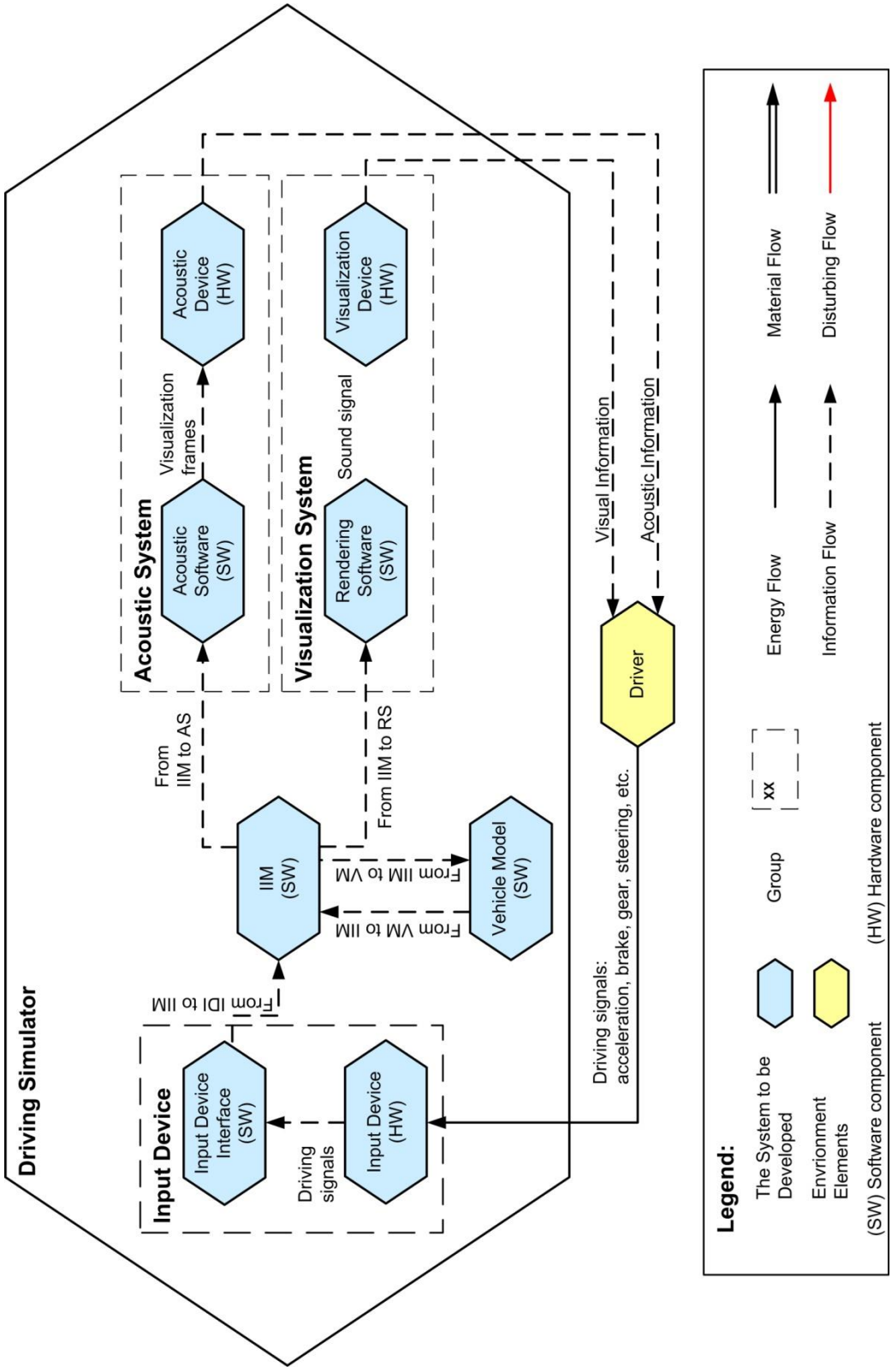
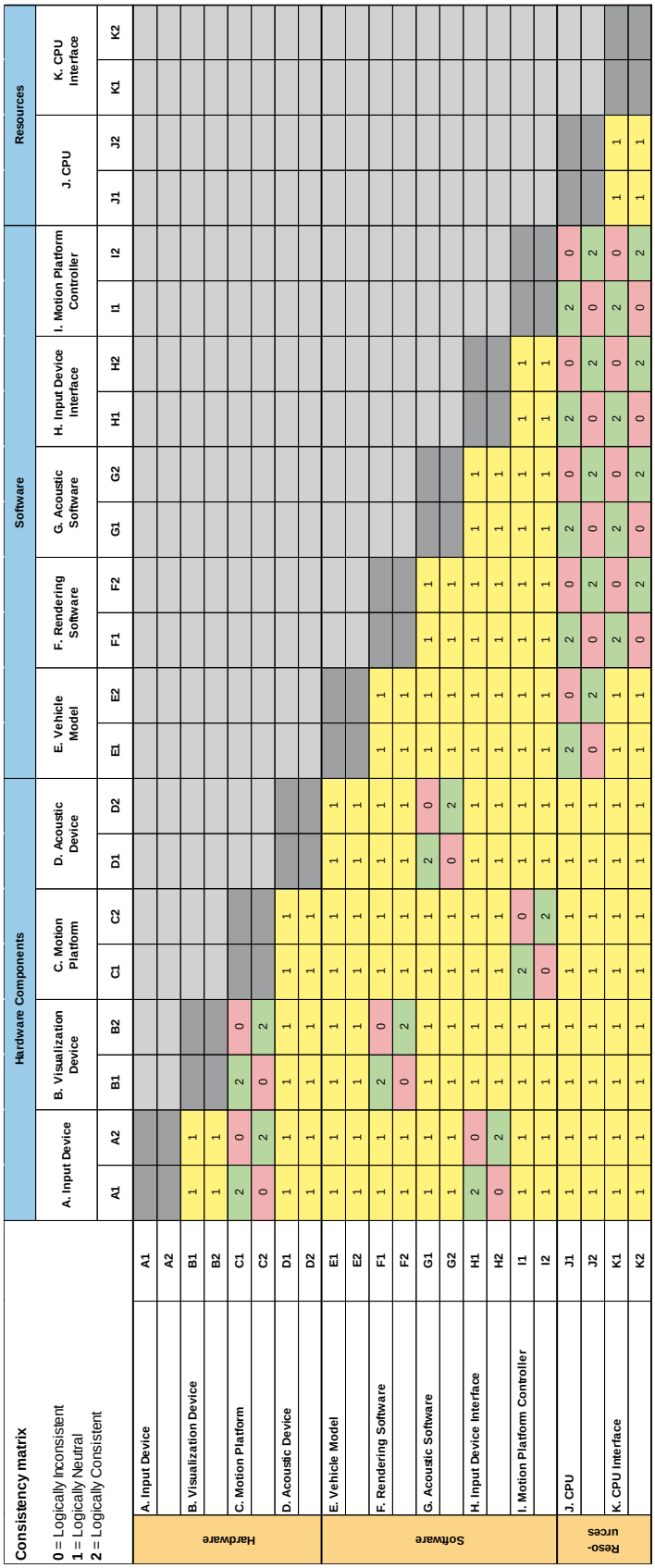


Figure A-4: Active Structure model of the case study – variant 2.

A1.2 The Consistency Matrix of the Case Study

Table A-1 shows the consistency matrix based on the result of the case study with the assumption that each component has two solution elements. This table shows the configuration mechanism development (phase 3).

Table A-1: The consistency matrix – example of case study solution elements.



A2 Amendments to the Implementation and Validation (Chapter 5)

In this appendix, some additional figures are presented regarding the implementation prototype of the configuration tool, which is explained in chapter five.

A2.1 Configuration Tool – Main Operations

Figure A-5 shows the main start screen of the configuration tool's graphical user interface.

Figure A-5: The graphical user interface of the configuration tool's implementation prototype – start screen.

As described in chapter 5, the main operations of the configuration tool are stated as follows:

- **Configure New System**
- **Add New Component**
- **Add New Solution Element**
- **Load Configuration File**
- **View Components and Solution Elements**

The configuration tool's main operation and the graphical user interfaces of each operation are described and illustrated in the next sections.

A2.2 Configure New System

The main aim of the configuration tool is to create variants of the reconfigurable driving simulator. The variant creation is done by the operation “Configure New System”. This task is done during the 12 selection steps identified during phase 5. Each selection step has its own panel.

Figure A-6 shows the first selection step for the “Modelling System” which contains the following components for selection: environment creation tool, scenario selection tool, models/ software tool location and assistant tool.

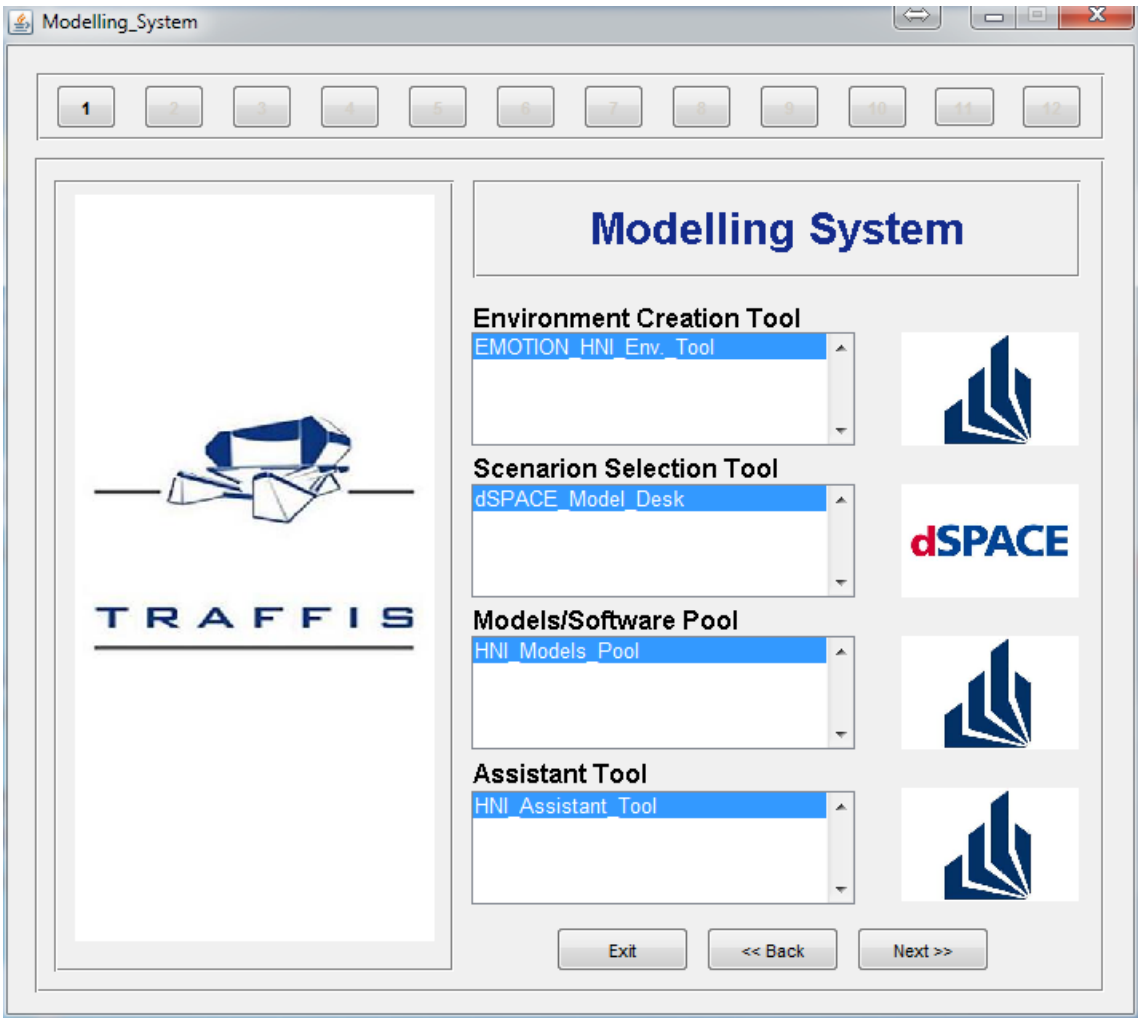


Figure A-6: Configure new system first selection step – modelling system.

Figure A-7 shows the second selection step for the "Motion System" which contains the following components for selection: the motion platform, the motion platform controller and the motion cueing algorithm.

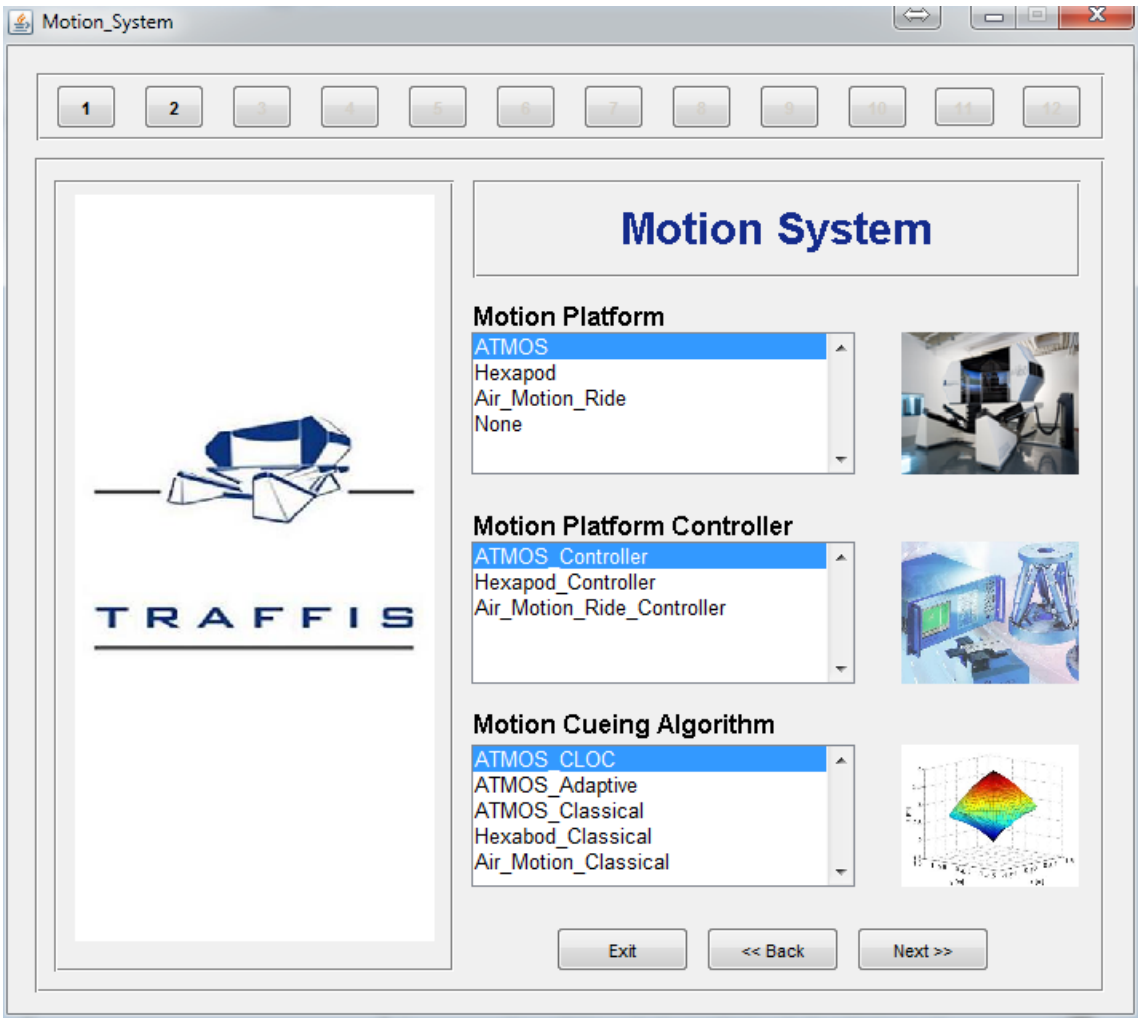


Figure A-7: Configure new system second selection step – motion system.

Figure A-8 shows the third selection step for the "Visualization System" which contains the following components for selection: the visualization device, and the rendering software.

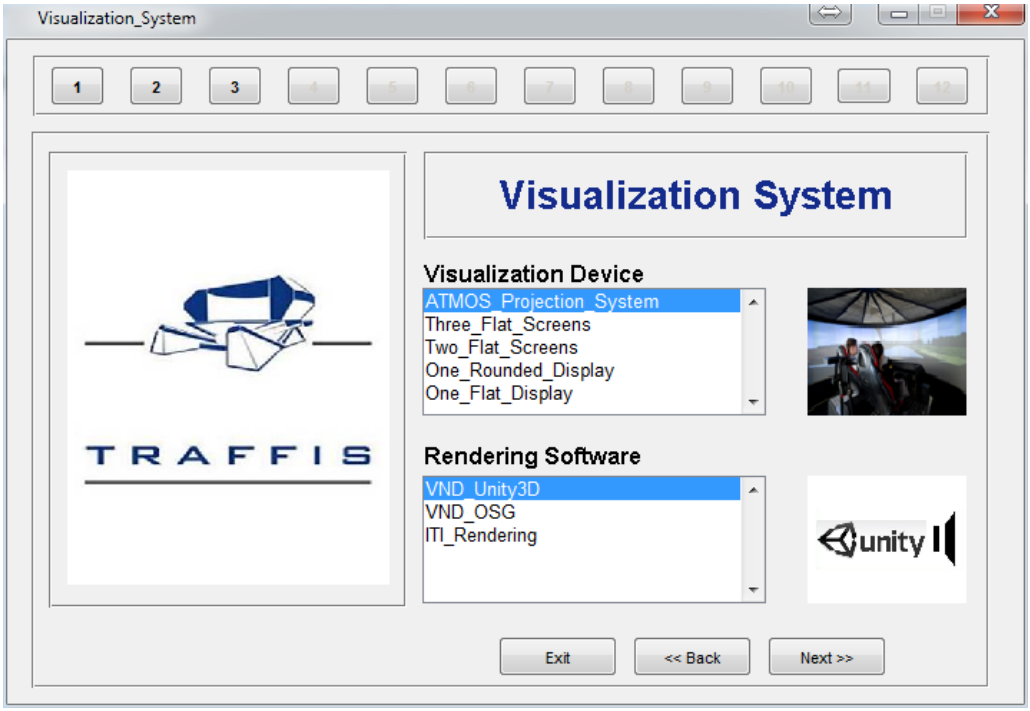


Figure A-8: Configure new system third selection step – visualization system.

Figure A-9 shows the fourth selection step for the "Controlling System" which contains the following components for selection: the operator tool, and the test manager tool.

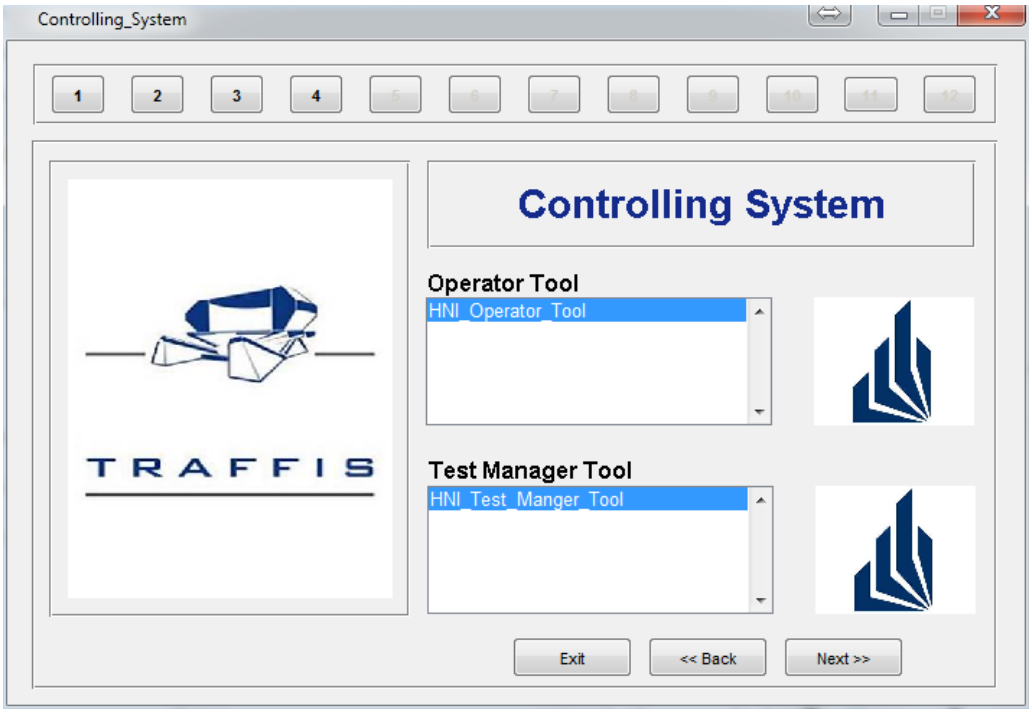


Figure A-9: Configure new system fourth selection step – controlling system.

Figure A-10 shows the fifth selection step for the "Input Device" which contains the following components for selection: the input device, and the input device interface.

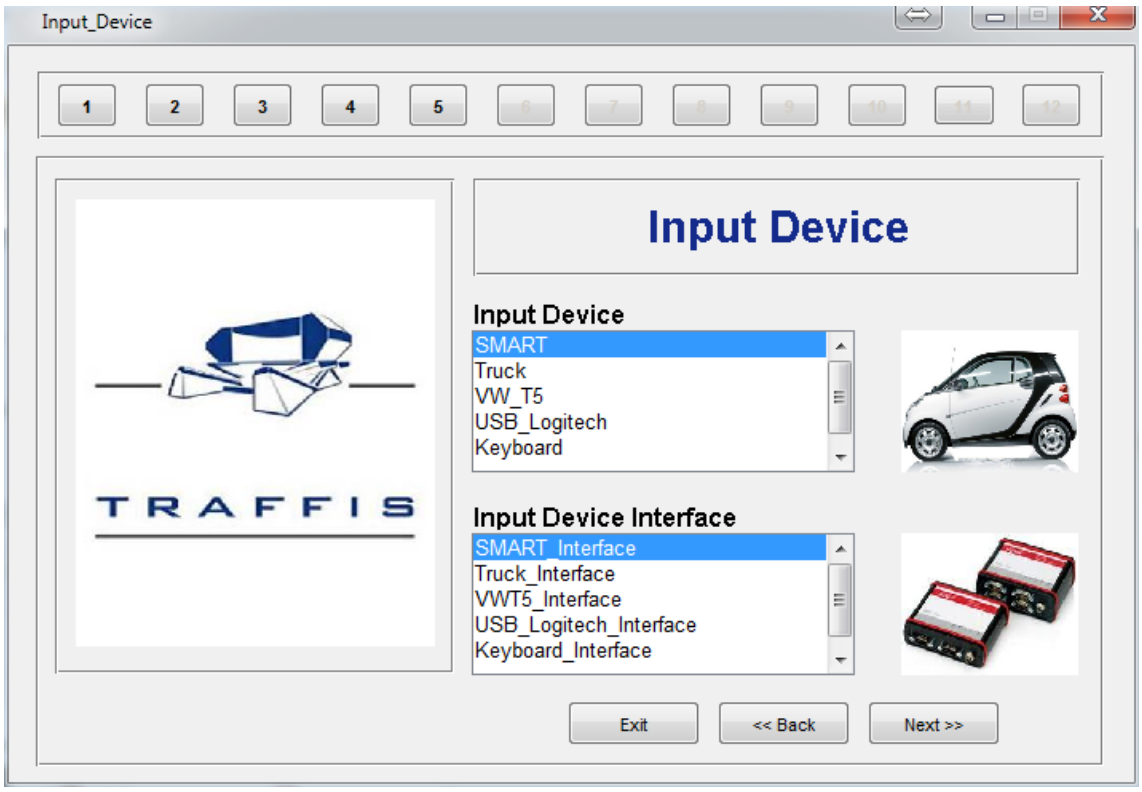


Figure A-10: Configure new system fifth selection step – input device.

Figure A-11 shows the sixth selection step for the "Vehicle Model".

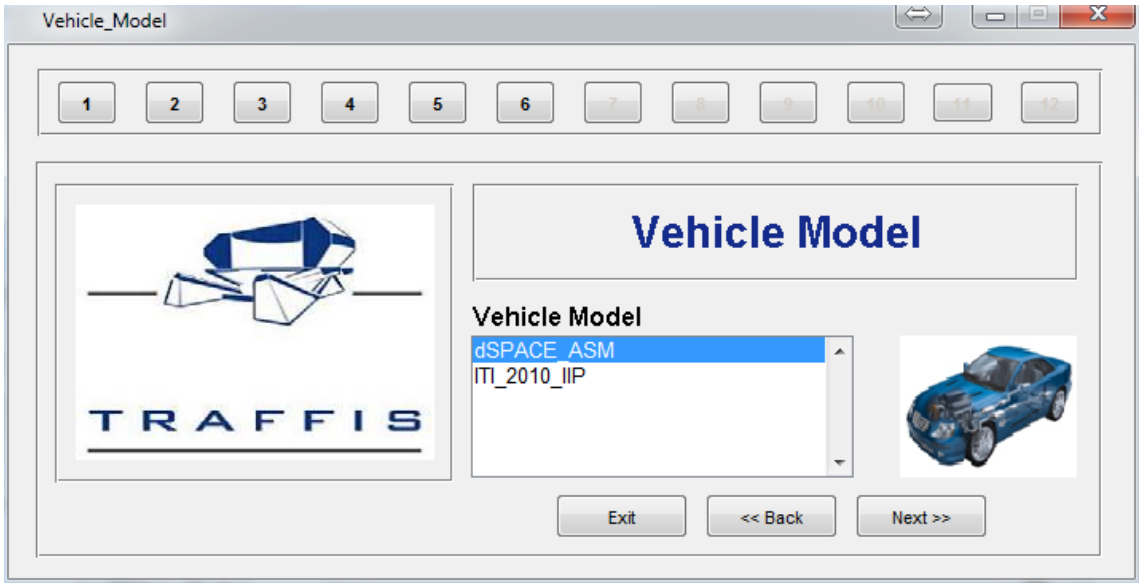


Figure A-11: Configure new system sixth selection step – vehicle model.

Figure A-12 shows the seventh selection step for the “Traffic model”.

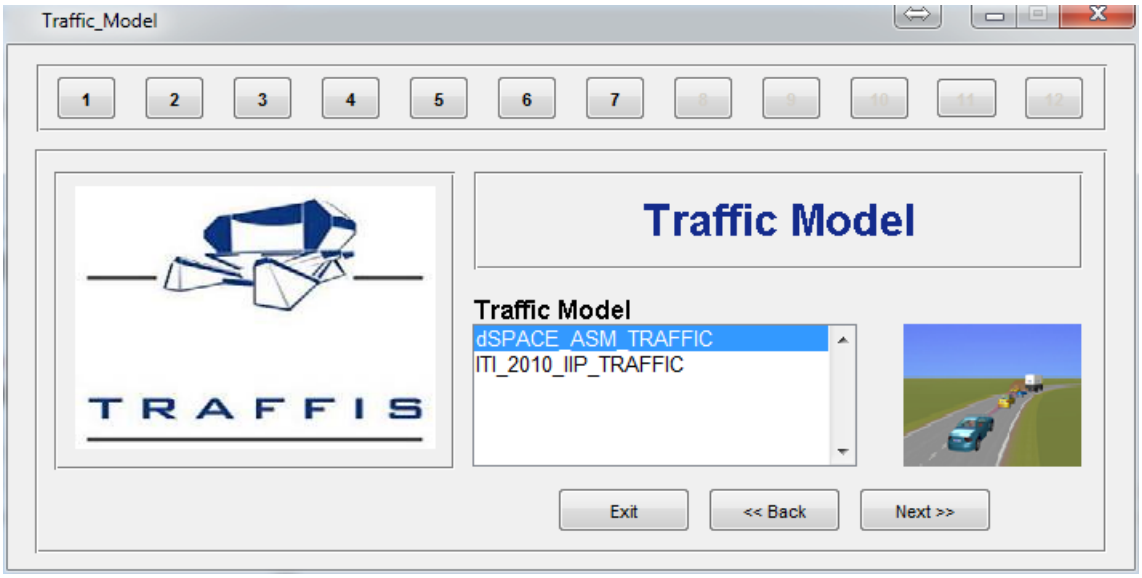


Figure A-12: Configure new system seventh selection step – traffic model.

Figure A-13 shows the eighth selection step for the “ADAS Camera System” which contains the following components for selection: the camera test-bench, and the camera test-bench interface.

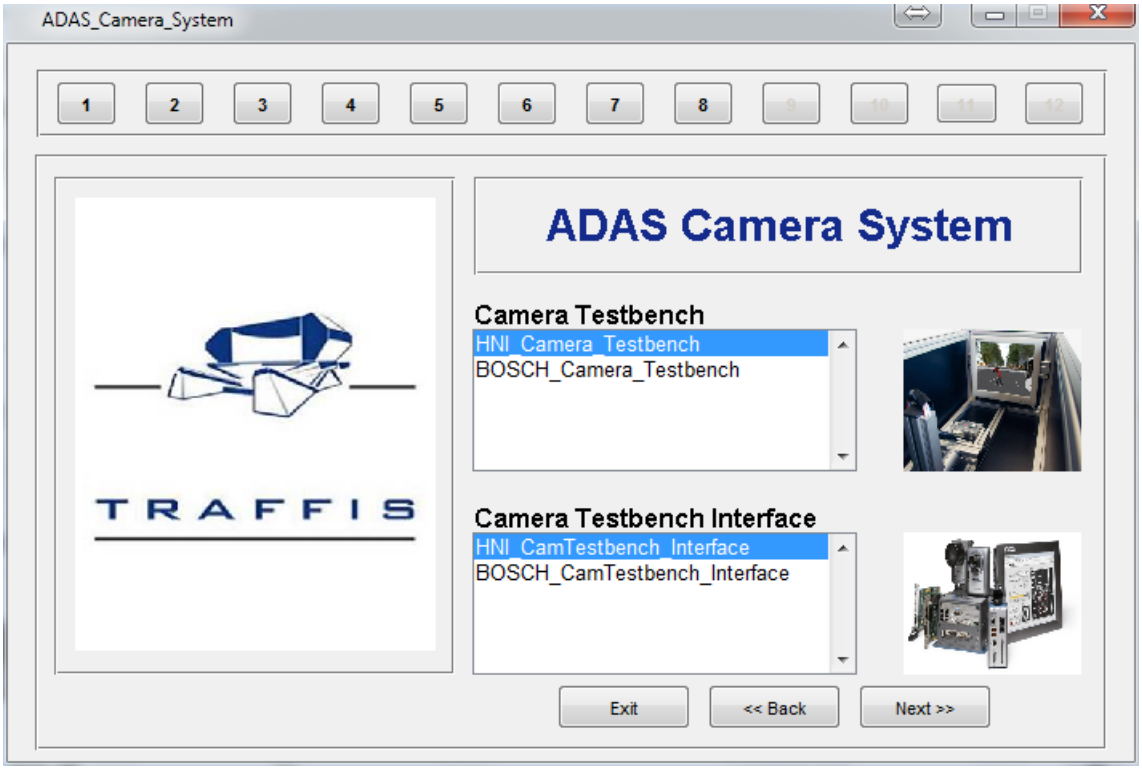


Figure A-13: Configure new system eighth selection step – ADAS camera system.

Figure A-14 shows the ninth selection step for the "Acoustic System" which contains the following components for selection: the acoustic device, and the acoustic software.

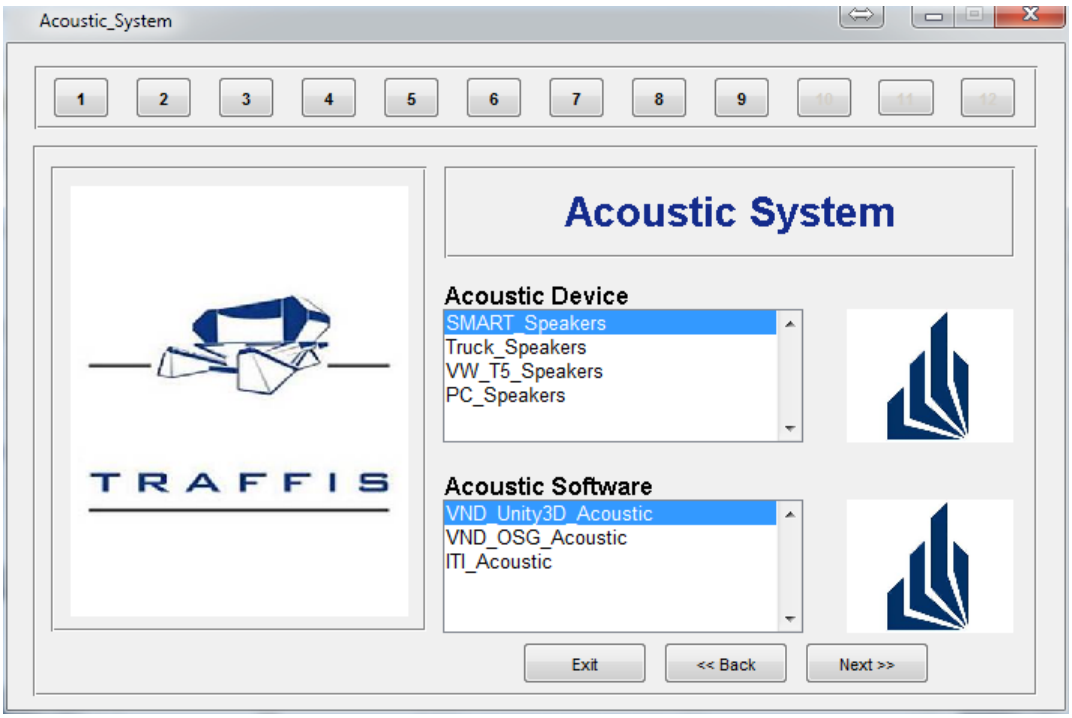


Figure A-14: Configure new system ninth selection step – acoustic system.

Figure A-15 shows the tenth selection step for the "ADAS Control Unit System" which contains the following components for selection: the ADAS control unit and the ADAS control unit interface.

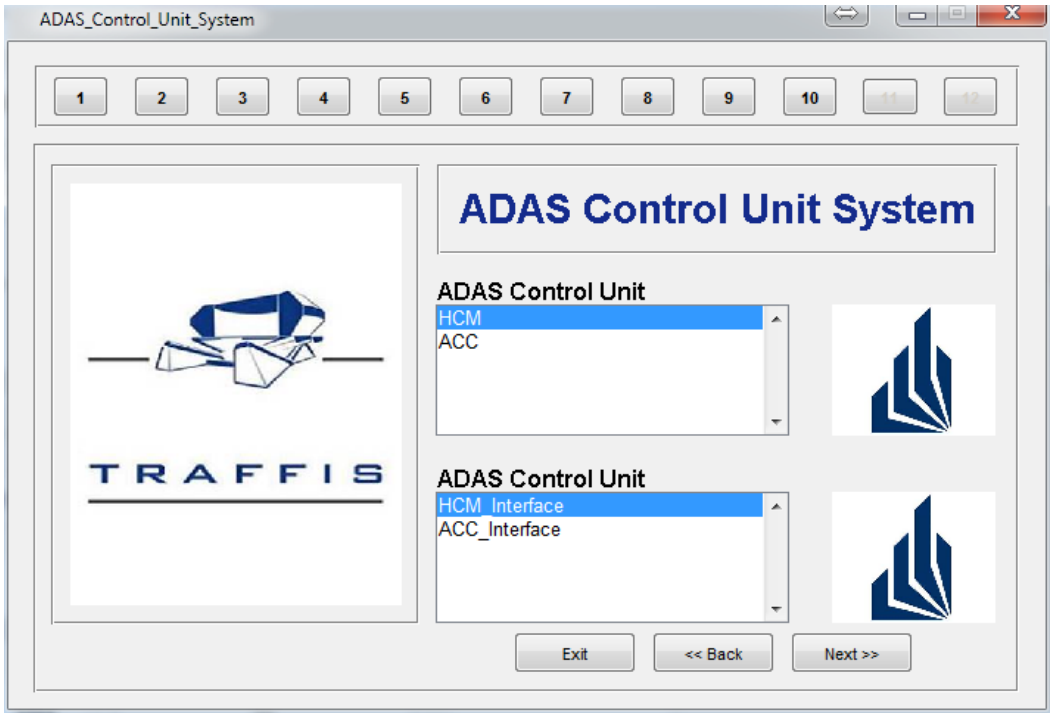


Figure A-15: Configure new system tenth selection step – ADAS control unit system.

Figure A-16 shows the eleventh selection step for the "Analysis System" which contains the following components for selection: the simulation results database and the analysis tool.

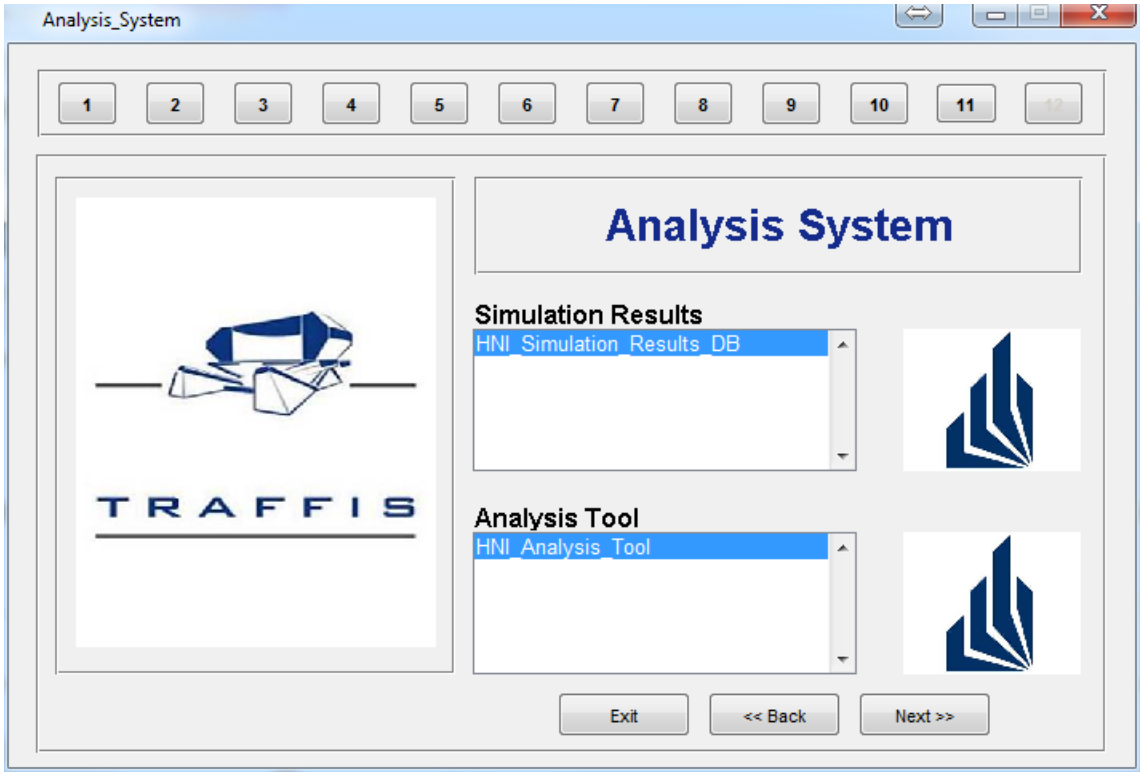


Figure A-16: Configure new system eleventh selection step – analysis system.

Figure A-17 shows the twelfth selection step for the "Resources" which contains the following components for selection: the simulation computer, the visualization computers and the simulation computer interfaces.

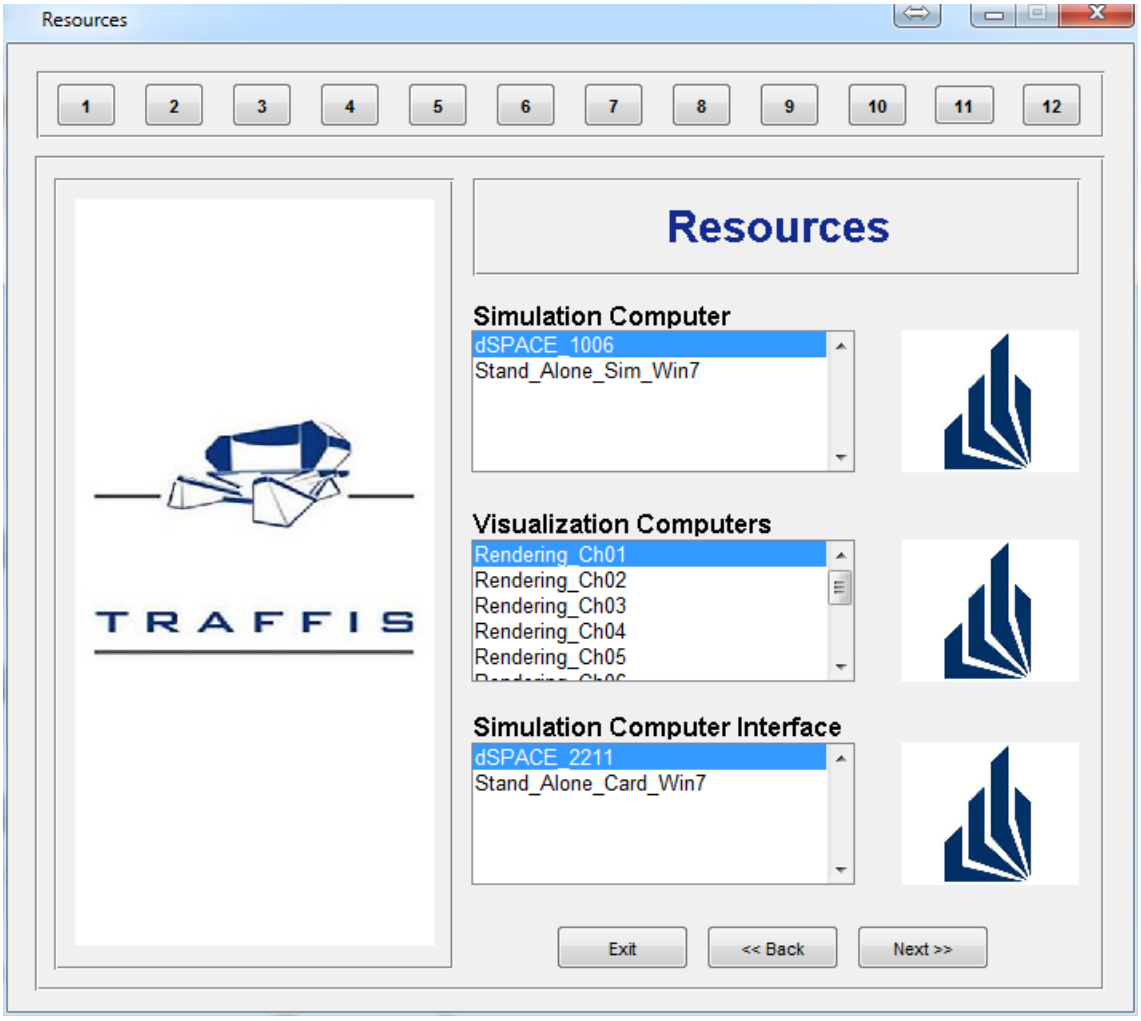


Figure A-17: Configure new system twelfth selection step – resources.

Figure A-18 shows the "Compatibility Check" step for the interfaces of the selected solution elements. The configuration check could check the compatibility of the interfaces by means of the three following aspects: signal name, signal unit and signal frequency.

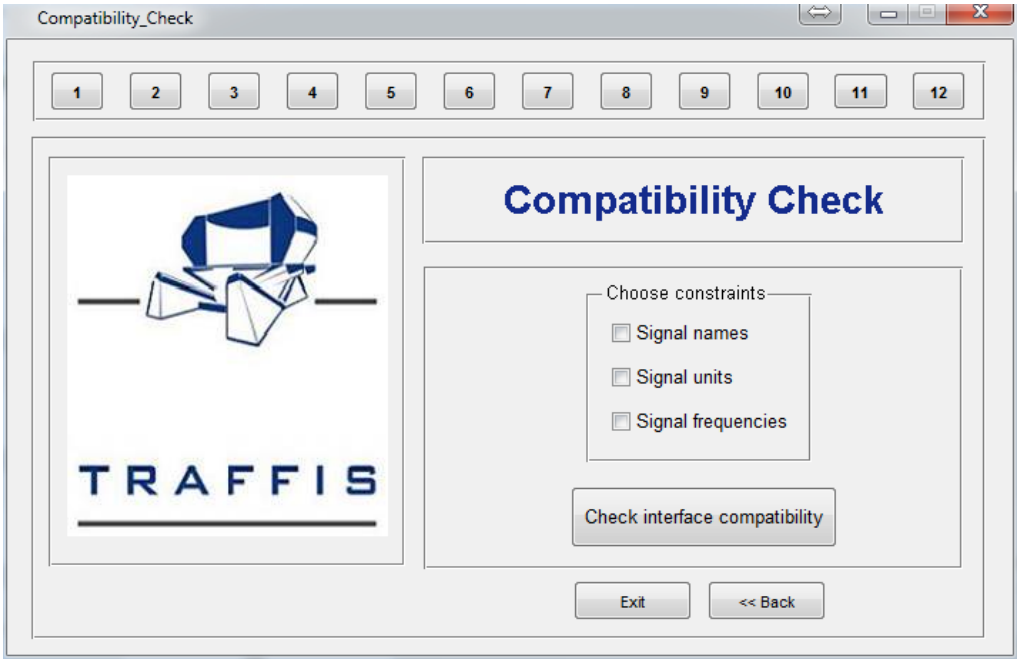


Figure A-18: Configure new system – compatibility check step.

Figure A-19 shows the last step to configure a driving simulator variant, which is the "Configuration File Generation". In this panel, each selected solution element can be viewed and edited.

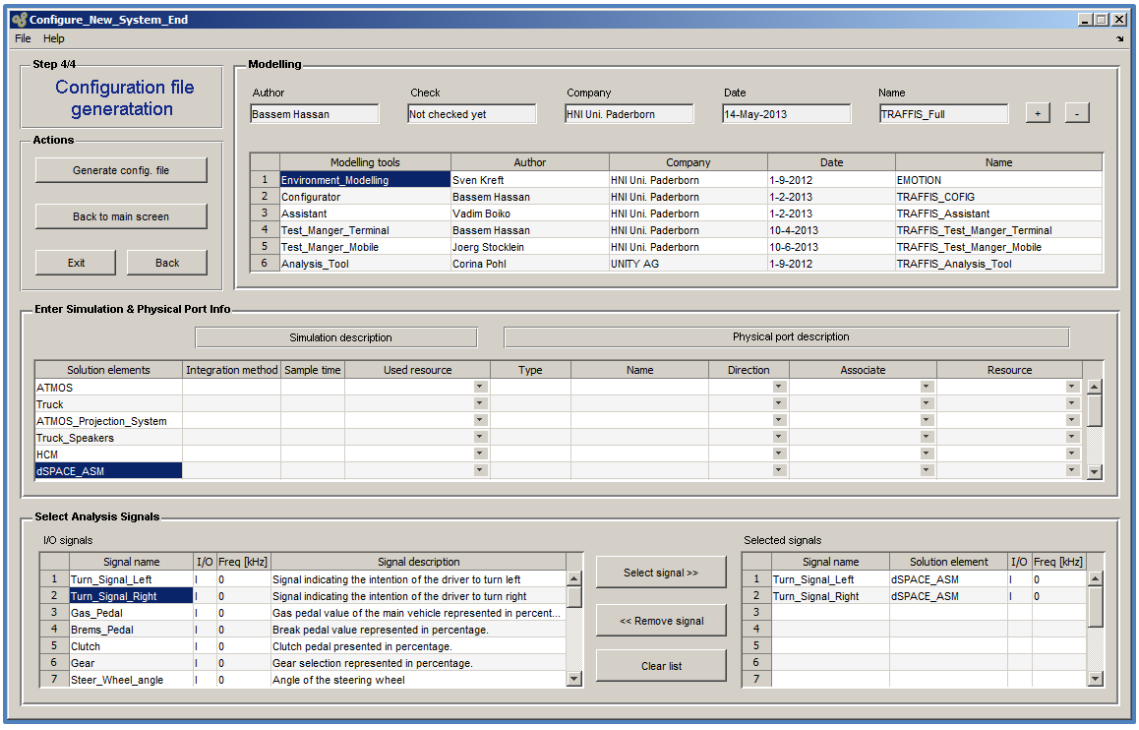


Figure A-19: Configure new system – configuration file generation step.

A2.3 Add New Component

Figure A-20 shows the "Add New Component" panel. The new component can be added by the means of its name, description, symbol, category and its desired signal interface, if this exists.

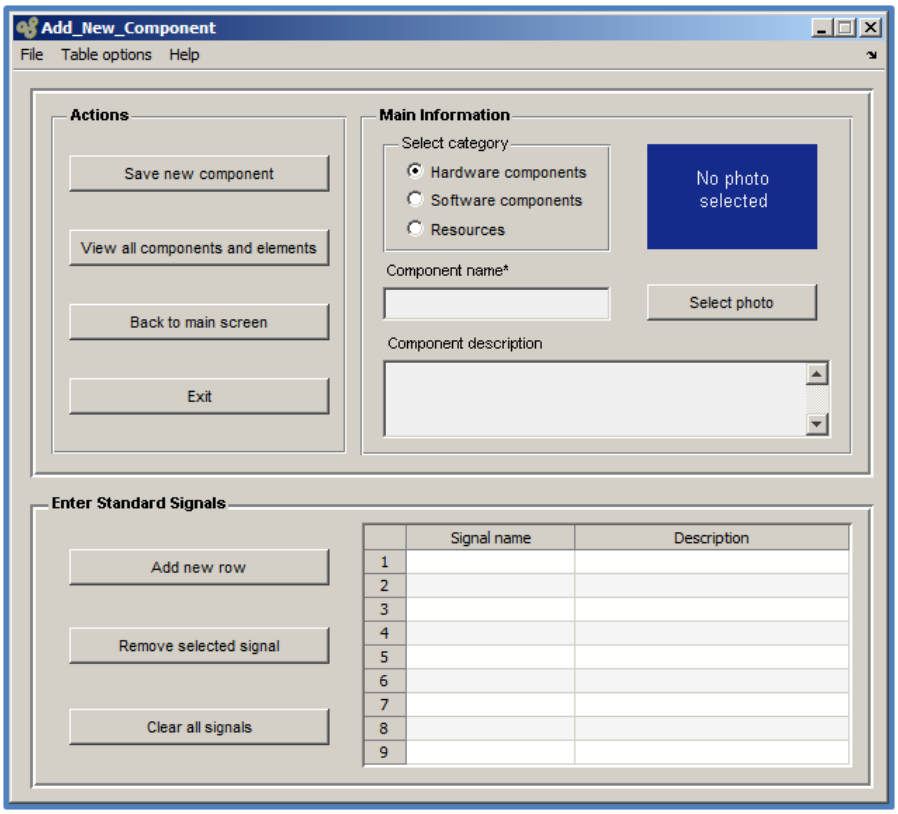


Figure A-20: Add new component panel.

A2.4 Add New Solution Element

Figure A-21 shows the "Add New Solution Element" panel. The new solution element can be added by means of its name, description, symbol, category, detailed list of inputs/outputs and logical consistency with the other solution elements.

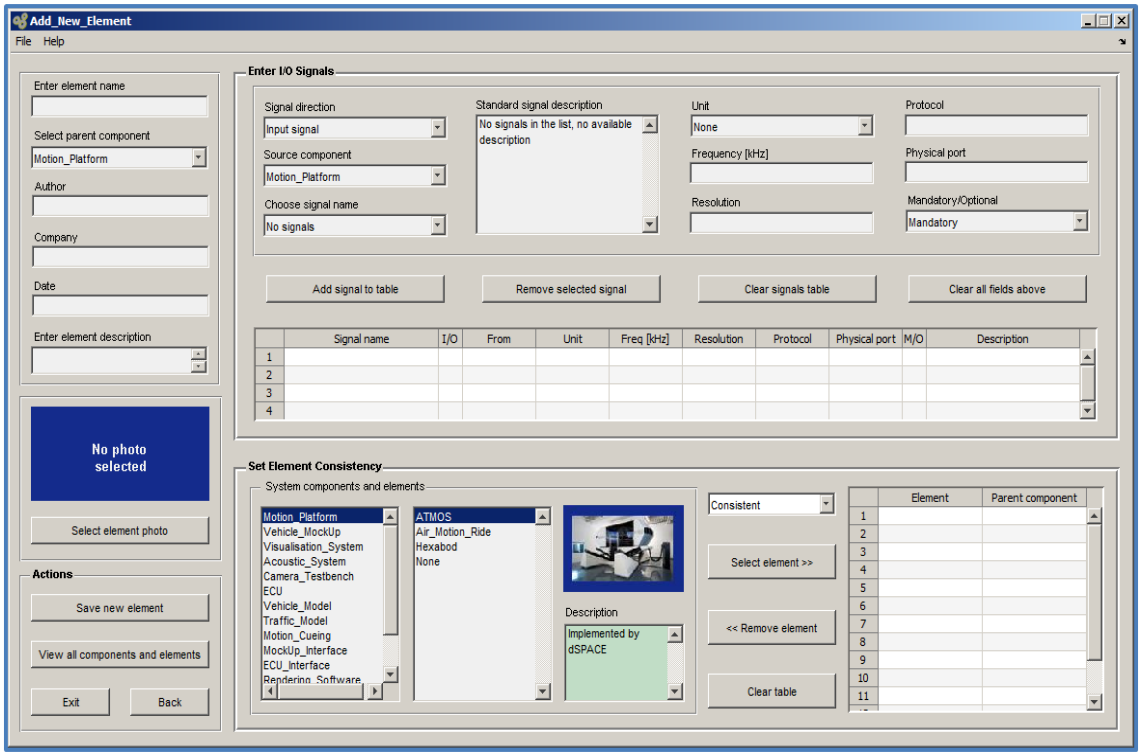


Figure A-21: Add new solution element panel.

A2.5 Load Configuration File

Figure A-22 shows the "Load Configuration File" panel. With the help of this panel, a previously generated configuration file could be loaded, edited and saved after editing.

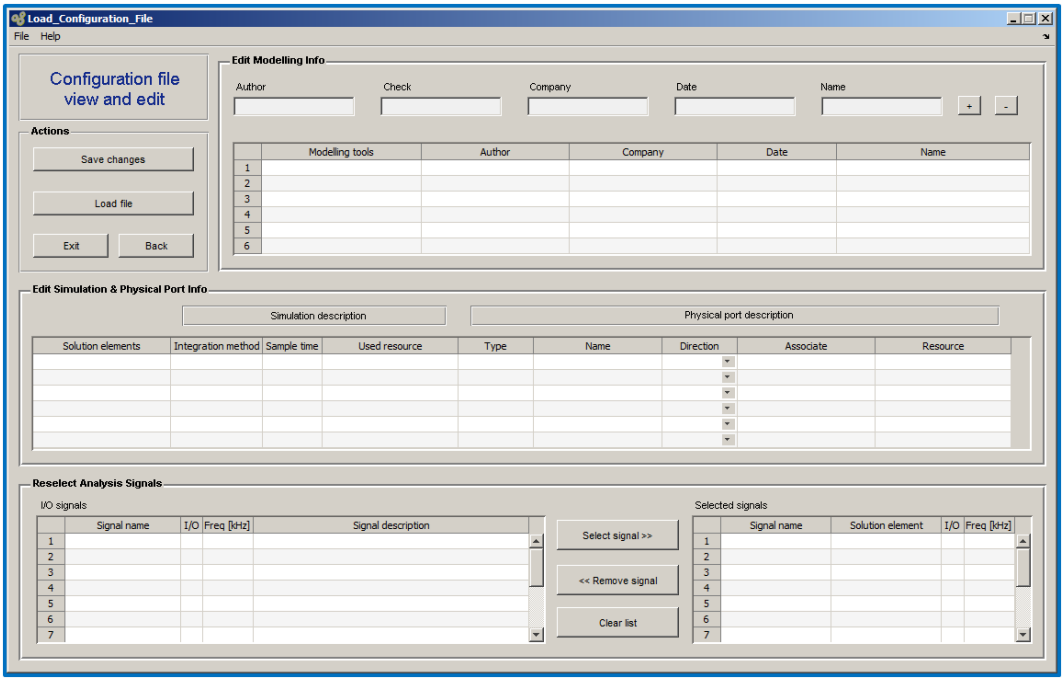


Figure A-22: Load configuration file panel.

A2.6 View Components and Solution Elements

Figure A-23 shows the "View Components and Solution Elements" panel. With the help of this panel, a previously added component or solution element could be viewed, edited and deleted.

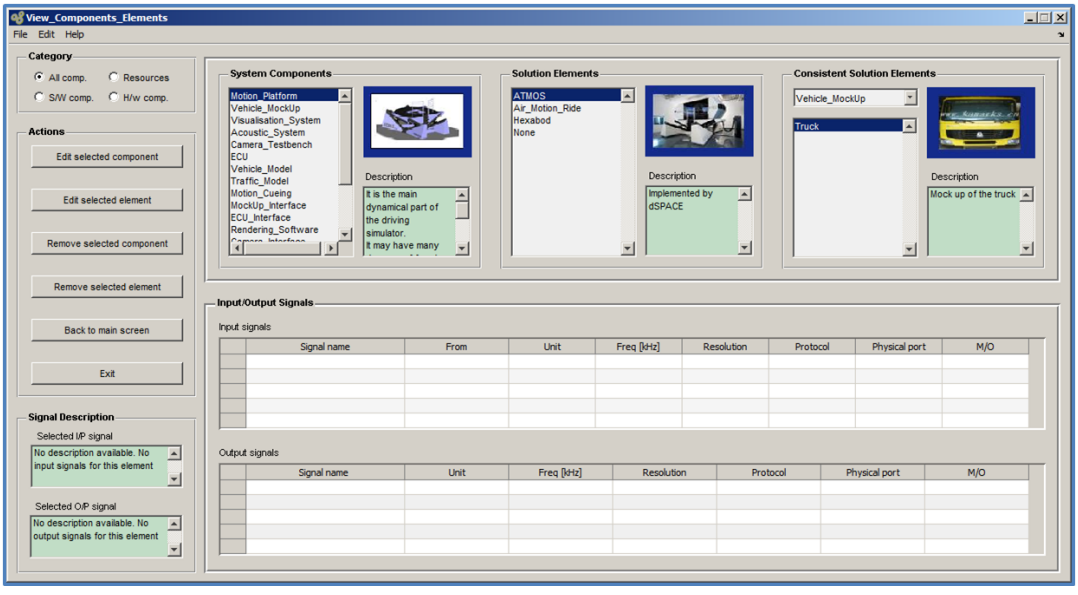


Figure A-23: View components and solution elements panel.