An Application-Oriented Design Method for Networked Driving Simulation

Kareem Abdelgawad [1,*], Jürgen Gausemeier [1], Ansgar Trächtler [1], Sandra Gausemeier [1], Roman Dumitrescu [2], Jan Berssenbrügge [2], Jörg Stöcklein [2], and Michael Grafe [2]

[1] Heinz Nixdorf Institute, University of Paderborn, 33102 Paderborn, Germany; Juergen.Gausemeier@hni.upb.de (J.G.); Ansgar.Traechtler@hni.upb.de (A.T.); Sandra.Gausemeier@hni.upb.de (S.G.)

[2] Fraunhofer Institute for Mechatronic Systems Design IEM, 33102 Paderborn, Germany; Roman.Dumitrescu@iem.fraunhofer.de (R.D.); Jan.Berssenbruegge@iem.fraunhofer.de (J.B.); Joerg.Stoecklein@iem.fraunhofer.de (J.S.); Michael.Grafe@iem.fraunhofer.de (M.G.)

* Correspondence: Kareem.Abdelgawad@hni.upb.de; Tel.: +49-5251-606-228

Abstract: Autonomous and cooperative vehicle systems represent a key priority in the automotive realm. Networked driving simulation can be utilized as a safe, cost-effective experimental replica of real traffic environments in order to support and accelerate the development of such systems. In networked driving simulation, different independent systems collaborate to achieve a common task: multi-driver traffic scenario simulation. Yet distinct system complexity levels are necessary to fulfill the requirements of various application scenarios, such as development of vehicle systems, analysis of driving behavior, and training of drivers. With myriad alternatives of available systems and components, developers of networked driving simulation are typically confronted with high design complexity. There are no systematic approaches to date for the design of networked driving simulation in accordance with the specific requirements of the concerned application scenarios. This paper presents a novel design method for networked driving simulation. The method consists mainly of a procedure model that is accompanied by a configuration software. The procedure model includes the necessary phases for the systematic design of application-oriented platforms for networked driving simulation. The configuration software embeds supportive decision-making processes that enable developers to apply the design method and easily create different system models. The design method was validated by generating system models and developing platforms of networked driving simulation for three different application scenarios.

Keywords: autonomous and cooperative vehicle systems; connected driving simulators; systems engineering; system of systems; system-level design; application-oriented development

1. Introduction

Autonomous and cooperative vehicle technologies have attracted major attention from all key automotive players. These disruptive technologies create fascinating new mobility prospects while potentially providing more traffic safety and efficiency. As governments provide regulatory guidelines and carry out or supervise necessary infrastructure modifications, other sectors are positioning themselves firmly in this field, such as automotive manufacturers and suppliers, IT providers, insurance agencies, and logistics companies. All these key players are pursuing the economic benefits of these technologies and they must explore new business models that best suit the potential [1]. With respect to automotive manufacturers in particular, the competition to deploy these technologies onto public roads is becoming more obvious as customers' expectations rise. The technology itself turns out to be a relatively minor concern. Various automotive manufacturers have revealed their practical prowess in self-driving cars. Yet methods and tools are still required for additional refining development and test loops. For instance, it is crucial to tackle different traffic and driving strategies, as well as the interoperability between technologies of different providers.

Moreover, as human drivers are still in the loop, various related factors must be examined, such as ethical values, customer acceptance, and drivers' behavior [2]. In general, driving simulation is an effective tool that supports automotive research and industry [3]. It can be used mainly for the development and test of vehicle systems, such as advanced driver assistance systems (ADAS) [4]. Driving simulation can be used for other purposes, such as driver's training, demonstration and marketing, and studying behavior of drivers. Figure 1 shows two different driving simulator variants developed and operated with multidisciplinary expertise at the Heinz Nixdorf Institute—University of Paderborn in Germany.

(**a**)   (**b**)

Figure 1. Two driving simulator variants at the Heinz Nixdorf Institute in Germany: (**a**) Passenger car driving simulator with a motion platform; (**b**) Stationary truck driving simulator.

It is feasible and less expensive to build and operate simulations in controlled environments than conducting real field drives [3]. Various traffic scenarios involving a human-driven vehicle and programmed traffic participants can be created. Harsh environmental conditions can be reproduced easily, such as foggy or snowy roads. Furthermore, driving simulators present an inherently safe environment for experiments. There are no hazards to drivers while undergoing critical driving maneuvers or testing new vehicle systems.

However, with the introduction of autonomous and cooperative vehicle technologies, traffic systems become more complex while human drivers still represent an indispensable factor. Conventional driving simulation does not provide the realism and multi-interactivity related to these advanced automotive technologies. It provides only a rough representation of the unpredictability level associated with real traffic environments. Networked driving simulation can be used to mitigate this particular drawback. Specifically, creating a virtual driving environment that can be accessed by several human drivers provides a close approximation of real traffic interactions. There are various multi-interactive applications for networked driving simulation, such as development of vehicle systems, analysis of driving behavior, and training of drivers. A comprehensive discussion of these promising applications is presented in Reference [5]. As they focus on different aspects, these multi-interactive applications vary considerably in their system complexity requirements. This paper presents a novel method for the systematic design of networked driving simulation. The method considers the requirements of different application scenarios to determine the necessary system complexity. The rest of this paper is structured in five main sections. Section 2 specifies the problem addressed in this work; Section 3 gives an overview about two distinguished design approaches for conventional driving simulation in the literature; Section 4 presents the developed design method for networked driving simulation; the validation of the method is provided in Section 5 by means of different non-traditional application scenarios. Finally, Section 6 derives the conclusions and reveals future work.

2. Problem Description

There are various driving simulators available in the market with different fidelity levels. It is quite complicated to select a suitable driving simulator that fits a certain application scenario. Some assistance exists in the literature to support users of driving simulation while determining the necessary fidelity level of each building component [6]. Yet selecting different components to build a driving simulator requires some prior knowledge to guarantee the logical consistency and the technical interoperability. Additional work in the literature presents a method to configure driving simulation environments while assuring the compatibility of the building components [7]. However, networked driving simulation can be realized only with a more complex multidisciplinary system. This system involves typically distinct complex subsystems and components that interact together and work on a common goal. Moreover, challenging application scenarios for networked driving simulation arise with more diverging and changing requirements [5].

There has been a growing interest in a class of complex systems that themselves are composed of independent systems: systems of systems (SoS) [8]. Based on the literature review, numerous definitions exist for systems of systems. One relevant and quite simple definition is that "systems of systems are large-scale integrated systems that are heterogeneous and independently operable on their own, but are networked together for a common goal" [9]. Networked driving simulation belongs to this particular definition. Specifically, two or more driving simulators exchange information and share a common virtual environment, where human drivers interact with each other. Each participating driving simulator per se represents an independent system. A common system goal is accomplished through the collaboration within a system of systems environment. In a nutshell, the ultimate goal is to simulate multi-driver traffic scenarios close to the real traffic environment with its attendant uncertainties.

However, the complexity of designing a system of systems is daunting. One primary challenge is to pursue a synergy between the constituent systems to attain the desired system goal. Several concepts and design considerations have been addressed in the literature for the theme of SoS. The well-established principles of systems engineering can be used to overcome the pitfalls of SoS design [9]. Extending the systems engineering concepts to accommodate the SoS paradigm is discussed in Reference [10]. This led principally to the emergence of system of systems engineering. Architecting SoS environments through an evolutionary process is a crucial requirement in this regard. To that end, the open systems approach is adopted by the system of systems engineering [9]. This approach defines the general key principles for an open system architecture that is suitable for future evolution. Following this approach results in a flexible SoS that can be modified easily by exchanging the constituent systems and/or altering the characteristics of some building components. Yet building a system model before establishing the real system is one of the significant measures recommended by system of systems engineering [9]. The modeling process itself is challenging due to the complexity of the independent constituent systems. Fortunately, model-based system engineering can provide a rigorous foundation for the modeling and conceptual design of SoS [11]. In this regard, a system model is created and used as a baseline that includes the requirements, analysis, design, and verification of a target system. This system model represents a link between various disciplines, such as electrical, mechanical, software, communication, and requirements engineering [12]. That is, the system model provides a comprehensive description of the real system so that it is not specific to one particular discipline. However, a design method and a complementary software tool are required to establish different system models or configurations [12]. There is no method or tool to date for the systematic design of system models for networked driving simulation based on the determined application requirements. The following section presents two remarkable approaches from the literature for conventional driving simulation.

3. State of the Art

Manufacturers of driving simulators provide different fidelity levels to fulfill the requirements of different application scenarios. A simple classification of driving simulators into three categories based on fidelity is presented in Reference [13]: low-level, mid-level, and high-level. Low-level driving

simulators may not provide the immersion necessary for drivers to be fully involved in the simulation. High-level driving simulators may present challenges to overcome the distraction of drivers and reduce the learning time. Therefore, the selection of fidelity levels of driving simulators should properly consider the purpose of use in particular. Three generic application scenarios for driving simulation are defined in Reference [13]: driver behavioral research, vehicle design and engineering, and driver's training. These application scenarios are roughly correlated to the aforementioned classification of driving simulators [13]. Nonetheless, driving simulators are composed of many building components. Combining low-fidelity with high-fidelity components in one driving simulator can lead to effective utilization of resources and costs [14]. That is, a driving simulator may have high capability with respect to one particular component and low capability with respect to other components based on the purpose of use. However, it is challenging for non-expert users to select individual simulator components that fit their particular application scenarios. The following subsection presents guidelines from the literature to mitigate this problem.

3.1. Determining Necessary Fidelity Levels of Driving Simulators

While purchasing driving simulators, users usually undergo a selection process based on their own understanding of the capabilities of available solutions. The selection process tends usually to use the available budget to purchase simulator components with the highest possible fidelity level. This results typically in rough selections, where the end benefits are not as great as the purchase and operation costs. Negele introduced guidelines for determining the fidelity level of each primary simulator component with respect to the application scenarios [6]. Hereafter in this work, these guidelines are referred to as Negele's guidelines according to the author's name.

Negele's guidelines consider the human behavior that generally falls into one of three distinct categories: skill-based, rule-based, and knowledge-based behavior [15]. In particular, the driving behavior is affected correspondingly by the skills, experiences, and situation familiarity of drivers [16]. Skill-based responses occur in routine driving situations that require fast actions. In a driving simulator, these responses are triggered automatically only if the sensory stimuli are realistic enough for the driver. That is, high fidelity levels are required for these types of responses. Rule-based responses are invoked in driving situations that require identification and recall of previously instructed actions. The driver is fully aware of the situation and the corresponding necessary rules. In these situations, the responses are triggered moderately and the driver has some time to compensate missing cues. Therefore, a modest deviation from reality in a driving simulation environment is permitted for these types of responses. Knowledge-based responses emerge in unfamiliar driving situations that require effort and conscious attention. The driver exerts considerable intellectual effort to find an appropriate response for the situation. These responses occur slowly, so that the driver has enough time to mentally compensate necessary cues. Therefore, a large deviation from reality in a driving simulation environment is allowed for this type of responses.

Moreover, Negele's guidelines differentiate between three groups of driving tasks: primary, secondary, and tertiary. The primary tasks are further subdivided into: stabilization, guidance, and navigation tasks [6]. Maintaining the vehicle state while driving through a curve, interacting with other traffic participants, and planning an entire driving route are examples for the three types of the primary tasks respectively. While handling a driver assistance system is an example for the secondary driving tasks, adjusting the air conditioner and tuning the radio are typical tertiary driving tasks. Realistic simulation cues for appropriate vehicle control are more significant for the primary than the tertiary driving tasks [6]. The secondary driving tasks are considered intermediate with respect to the fidelity level required to control the vehicle. Figure 2 shows a matrix between the defined driving tasks and the driver response types. The mutual intersections result in 15 classes of driving simulator applications. Users have to specify the concerned driving task and response in order to determine the relevant application class [6]. Consequently, the determination of the application classes helps users to conclude the allowed fidelity deviation between the driving simulation system and reality as depicted in Figure 2.

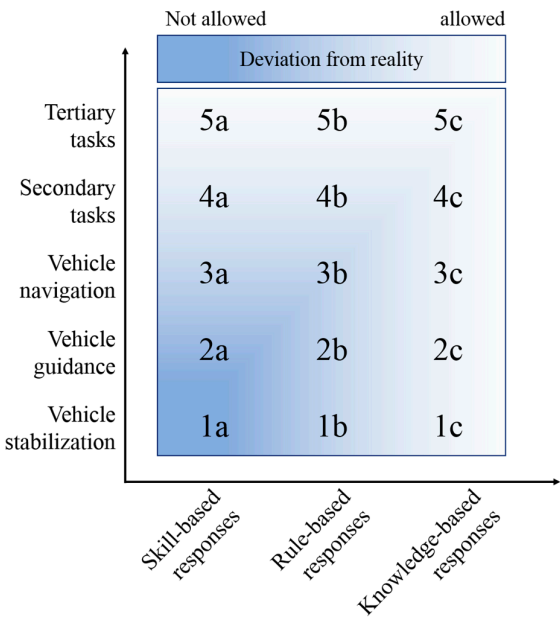



Figure 2. Scheme for classifying driving simulator applications [6].

In addition, Negele's guidelines defined the main driving simulator subsystems: scene simulation, motion simulation, driver's platform, acoustic simulation, and objects database along with traffic simulation [6]. These subsystems can be considered in different orders based on their contribution to the overall simulation fidelity for each application class. Furthermore, each subsystem has a group of features characterized by different fidelity levels. The fidelity levels are distinguished by keys that are represented as letters and ordered by numbers. As an example, Table 1 shows the features of the driver's platform along with their different fidelity levels.

Table 1. Features and fidelity levels of the driver's platform [6]

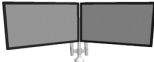
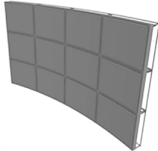
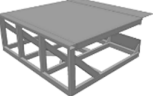
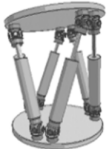
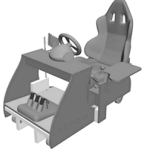
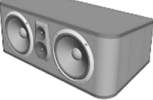
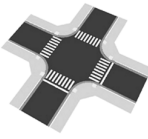
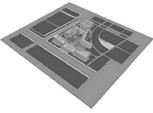
Feature	Key	Fidelity Levels
Mock-up	S1	Driving seat and HMI without chassis
	S2	Partial vehicle (quarter or half vehicle)
	S3	Complete vehicle—no modifications apparent
	S4	Series production vehicle—no modifications apparent
HMI	T1	Basic and simple HMI
	T2	Complete and realistic HMI
	T3	Complete HMI with reconfigurable display
Steering	U1	Steering moment proportional to steering angle
	U2	Electrical steering moment, damping, and friction
	U3	Electrical steering moment with high frequency
Pedals Set	V1	Passive force feedback
	V2	Adaptive force feedback—modifiable characteristic curve
	V3	Active force feedback—tangible effects of control systems

However, it may be still challenging for non-expert users to select particular fidelity levels for the features of each subsystem. Therefore, Negele's guidelines provided examples for common application classes as a means of orientation [6]. Moreover, reasonable feature fidelity levels for each of these classes were deduced and presented in the form of profile tables. As an example, Table 2 shows the profile of the application class 1a.

Table 2. Profile of driving simulator application class 1a [6]

Scene Simulation		
Viewing distance (A1)	Field of view (B2)	Stereo vision (None)
Head tracking (None)	Rear-view mirrors (E2)	Field continuity (F3)
Resolution (G2)	Frame rate (H1)	Projector type (J2)
Motion Simulation		
Motion platform < 6 DOF (K1)	Standard platform = 6 DOF (None)	Standard platform > 6 DOF (M1/M2/M3)
Vehicle dynamics (N3)	Tire (O4)	
Driver's Platform		
Mock-up (S3)	HMI (T2)	Steering (U3)
Pedals set (V2)		
Acoustic Simulation		
Primary sound P1 (P2)	Auxiliary sound (Q1)	Sound system (R1, R3)
Environment Database		
Database type (W1)	District type (Y1)	
Traffic Objects Simulation		
General traffic vehicles (Z1)	Special objects (None)	

For better visualization and easy interpretation of the fidelity levels, the defined simulator application classes are presented in the form of specification radar charts [6]. For all driving simulator subsystems, these charts depict the features and the fidelity levels in comparison to the maximum achievable fidelity levels according to the technical advancement.

In summary, Negele's guidelines present an assistance to non-expert users to determine the necessary overall fidelity levels of the driving simulators. Following these guidelines leads to the selection of driving simulators with complexity levels intended for specific application scenarios. However, users may have to alternate between different fidelity levels to address further application scenarios. This process is challenging for non-expert users as it requires technical knowledge of system structure and component compatibility and interoperability. The following section presents a method from the literature to tame this complexity.

3.2. Configuring Driving Simulation Environments

Driving simulation facilities are used in practice to possibly cover diverse application scenarios simultaneously [7]. Users may have access to various simulator components of different fidelity levels within the same driving simulation facility. A maintainable and flexible environment for driving simulation is required to easily exchange driving simulator components. Hassan presented a method to reconfigure driving simulation environments by system users [7]. Hereafter in this paper, this method is referred to as Hassan's method according to the author's name.

Principally, Hassan's method applies a morphological box containing entries of the main driving simulator components together with the available variants [17]. These variants are called solution elements and they represent products with different fidelity levels and characteristics provided by different simulator manufacturers and developers. The driving simulator components are listed vertically and the corresponding available solution elements are listed horizontally within the morphological box.

Table 3 Table 3 shows the morphological box, where the solution elements are depicted as representative figures [7]. The morphological box of Hassan's method is modified slightly in this work. Specifically, the naming convention of the driving simulator components of Negele's guidelines is used in the morphological box of Hassan's method to maintain consistency between both approaches.

Table 3. Morphological box for driving simulator components (excerpt) [7]

Driving Simulator Components	Solution Elements		
	1	**2**	**3**
Scene Simulation System			
Motion Simulation System			
Driver's Platform			
Acoustic Simulation System			
Environment Database			
Traffic Objects Simulation			

The shown morphological box contains six driving simulator components: scene simulation system, motion simulation system, driver's platform, acoustic simulation system, environment database, and traffic objects simulator [7]. Three exemplary solution elements are provided for each driving simulator component. The morphological box can be extended horizontally to add further solution elements. System users can select simulator components and solution elements in a process similar to browsing an online catalogue to customize a product before purchasing. The core of Hassan's method incorporates a consensus check algorithm that has mainly two levels [7]. The first level is the logical dependency check between the driving simulator components. This dependency check process gives an indication whether the selection of one particular component necessitates or affects the selection of other components. For instance, the selection of the driver's platform may depend on the selection of the motion platform and vice versa. This dependency may arise due to the dimension and weight of the driver's platform in relation to the corresponding specifications of the motion platform. A two-dimensional dependency matrix is created to facilitate the dependency check process. Driving simulator components are listed in the first row and column of the matrix as shown in Table 4. The dependency matrix is mirrored along the diagonal line. The intersection of each pair of different driving simulator components determines the respective logical dependency.

Table 4. Dependency matrix of driving simulator components [7]

Dependency Scheme		Hardware				Software				
0 = Independent Components 1 = Dependent Components		Visualization System	Motion Platform	Human-Machine Interface	Acoustic System	Visualization Software	Platform Controller	Vehicle Dynamics	HMI Software	Acoustic Software
Hardware	Visualization system	x								
	Motion platform	1	x							
	Human-machine interface	0	1	x						
	Acoustic system	0	0	0	x					
Software	Visualization software	1	0	0	0	x				
	Platform controller	0	1	0	0	0	x			
	Vehicle dynamics	0	0	0	0	0	0	x		
	HMI software	0	0	1	0	0	0	0	x	
	Acoustic software	0	0	0	1	0	0	0	0	x

The second level of the consensus check algorithm is the logical consistency analysis. This consistency check process gives an indication whether the selection of one particular solution element is consistent with the selection of other solution elements. A two-dimensional consistency matrix is created to facilitate the consistency check process. The solution elements of each system component are listed in the first row and first column of the matrix. The consistency matrix is mirrored about the diagonal line. The intersection of each pair of different solution elements determines the respective logical consistency. Table 5 shows an excerpt of the consistency matrix.

Table 5. Consistency matrix of driving simulator solution elements (excerpt) [7]

Consistency Scheme			Hardware							
0 = Inconsistent Solution Elements 1 = Independent Solution Elements 2 = Consistent Solution Elements			Visualization System		Motion Platform		Human Machine Interface		Acoustic System	
			A	B	A	B	A	B	A	B
Hardware	Visualization system	A	x	x						
		B	x	x						
	Motion platform	A	2	0	x	x				
		B	0	2	x	x				
	Human machine interface	A	1	1	2	0	x	x		
		B	1	1	0	2	x	x		
	Acoustic system	A	1	1	1	1	1	1	x	x
		B	1	1	1	1	1	1	x	x

The consistency check process makes use of the preceding dependency check process. If two components are independent, the two corresponding solution elements inherit the independence. In this case, it is not necessary to check the consistency between these particular solution elements. The consistency matrix is editable to account for eventual availability change of solution elements. Both dependency and consistency matrices must be filled out by a system expert. However, Hassan's method was embedded within configuration software [7]. This can be used by non-expert system users to compose different configurations of driving simulators from the available solution elements.

In summary, Hassan's method provides a procedure to reconfigure driving simulation environments. The focus is given to the configuration process to assure the consensus of the simulator components without the consideration of the application requirements. An accompanying configuration software facilitates the configuration process. No substantial knowledge of driving simulator components or available solution elements is required from non-expert system users. However, the specifications of the driving simulator components are not correlated to the requirements of possible application scenarios. Users still need to manually determine the requirements or the necessary simulator fidelity level for their application scenarios based on some criteria, such as Negele's guidelines. Moreover, users have to manually analyze available simulator components to determine their fidelity levels. The effort increases considerably if multiple driving simulators are networked in one environment. The following section presents a new method to design networked driving simulation systems based on the specific application requirements.

4. Development Methodology

The approaches discussed in the previous section represent compelling methodological work for the field of conventional driving simulation [6,7]. However, broader design considerations are necessary for networked driving simulation as a typical system of systems (SoS) with its acknowledged complexity [5]. A multidisciplinary expertise must be involved while building system models and during system realization. The current modeling techniques for SoS are still in their infancy [9]. A domain-spanning conceptual design method and tool are required. To that end, a new systems engineering design method for networked driving simulation is presented in this section. Figure 3 depicts the fundamental components of the developed design method.

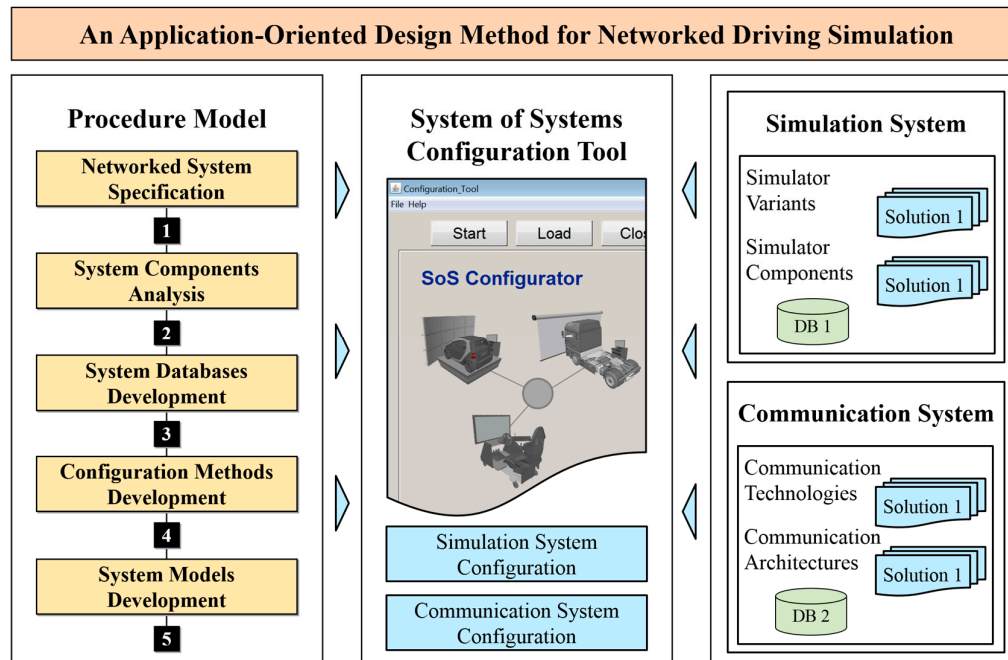



Figure 3. The fundamental components of the developed design method for networked driving simulation.

In particular, the concepts of model-based system engineering for building system models are adopted in the design method [14]. The design method consists basically of a procedure model and system of systems (SoS) configuration software as shown in Figure 3. These primary components are described as follows:

- Procedure model
  The procedure includes the necessary development phases that are arranged in a specific hierarchy towards the design of multidisciplinary system models for platforms of networked driving simulation. Each development phase contains a set of specific tasks, which shall be carried out in order to obtain the phase objectives. The procedure model specifies the methods and approaches used in each task. Moreover, the procedure model reveals the results of each individual phase. This work is concerned with the comprehensive description of the procedure model and its phases.

- SoS configuration software
  The SoS configuration software embeds the methods and approaches of the procedure model to generate application-oriented system models. The SoS configuration software guides non-expert system users in a sequential process to achieve the end objective. Non-expert users can be operators or domain-specific experts. They do not have to acquire deep multidisciplinary knowledge in order to use the SoS configuration software for system model design and generation. A comprehensive description of the design of the SoS configuration software is beyond the scope of this work. The design concepts of an analogous software tool are discussed thoroughly in Reference [18].

The ultimate goal of the design method is to assist non-expert system users in building different system models in accordance with the application scenarios of interest. The general proposed approach to taming design complexity in a systematic manner is to handle the modeling process in two major system aspects: simulation and communication. The approaches discussed in the previous section are combined and utilized in this work to address the first major aspect. Specifically, determining the fidelity levels of the constituent simulation systems is embedded within the SoS configuration software according to Negele's guidelines. Hassan's method is complemented so that

the configuration of the constituent simulation systems is carried out in accordance with the requirements of the concerned application scenarios. The second major aspect handles mainly the prioritization process of various network characteristics and functions of available competing communication systems in accordance with the requirements of the concerned application scenarios. Hence, suitable communication systems are selected to guarantee proper system operation and achieve substantial results. The following subsections describe the different phases of the procedure model and their tasks in detail.

4.1. Networked System Specification

The objective of this phase is to provide a clear understanding of the networked driving simulation system by formalizing a holistic system description. Principally, this description combines various aspects of the target system. Available system architecting and description techniques for SoS to date are not sufficient as they typically focus on specific aspects of the SoS [9]. For instance, some architecting techniques concentrate on the synergy of the constituent systems. Others focus on the communication between the constituent systems, with the argument that this particular aspect is common for all SoS types. However, the utilization of a well-established domain-spanning conceptual design method is necessary for the specification of networked driving simulation systems. This assures a broader consideration during its design as a system of systems.

To that end, the CONSENS specification technique is adopted in this phase [19]. The term 'CONSENS' is an English acronym that stands for 'conceptual design specification technique for the engineering of complex systems'. This specification technique mitigates the design complexity by describing the various aspects of the multidisciplinary systems using a set of coherent partial models. In particular, the effective usability of the CONSENS specification technique for the field of conventional driving simulation was validated in Reference [7]. Furthermore, the essential CONSENS partial models in this regard were determined and structured in a specific workflow [7]. Since the CONSENS specification technique is open for the conceptual design of newly emerging complex systems, it undergoes some minor modifications in this work for the development of networked driving simulation. The following subsection discusses the CONSENS workflow adopted in this work.

4.1.1. CONSENS Workflow for Networked Driving Simulation

The outcome of the CONSENS specification technique is represented as a principle solution that is described by seven interrelated partial models. Specifically, these partial models are: environment, application scenarios, requirements, functions, active structure, shape, and behavior [20]. Each partial model describes a specific aspect of the target system. To build a coherent system of systems model, the focus is given to the first five partial models in particular. The shape and behavior partial models are not considered in this work as they are more relevant to the development of commercial mechatronic products, such as printers and air conditioners. Figure 4 shows a specified workflow for the five relevant partial models along with their summarized results.

In this work, the CONSENS workflow is divided into three steps towards an increased system concretization. The first step includes the construction of three partial models: environment, application scenarios, and requirements. The second step depends on the outcomes of the first step to create a function hierarchy for the entire system. In the third step, an active structure is built based on the results of the previous steps. The system specification process is often carried out during expert workshops. Specifically, experts of various disciplines—such as mechanical engineering, electrical engineering, communication engineering, and requirements engineering—collaborate to specify the different aspects of the target system [19]. The following are brief discussions of the five partial models and the results with respect to networked driving simulation systems.

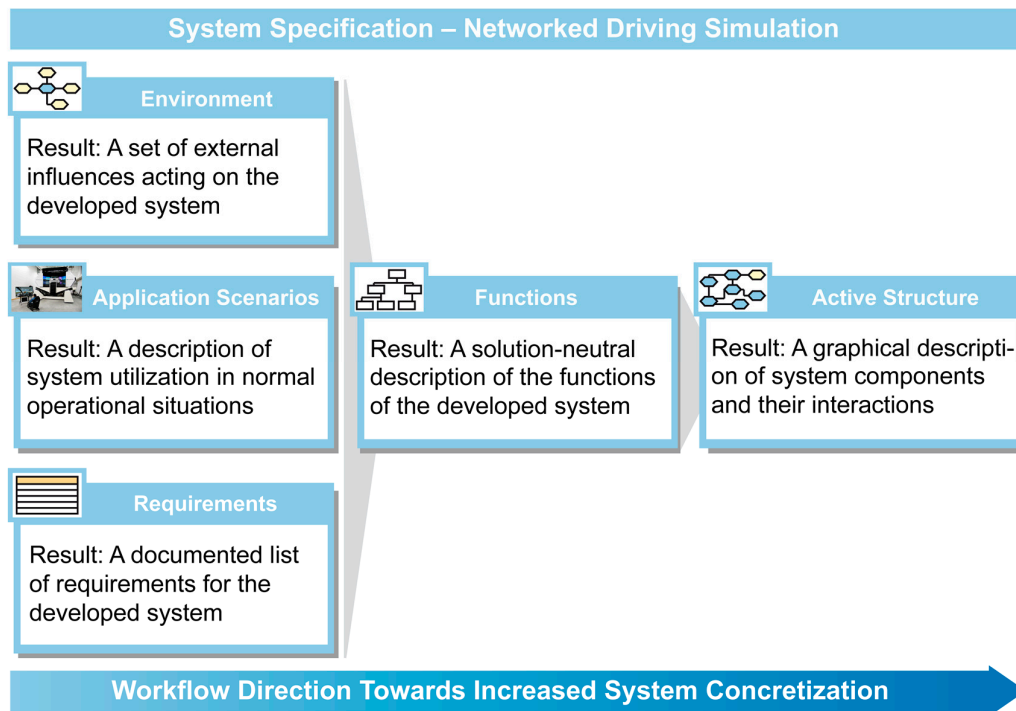



Figure 4. CONSENS workflow for networked driving simulation development.

4.1.2. Identify Environment

The environment partial model defines all possible external influences that can affect the networked driving simulation system. These external influences can be either environment elements or disturbance variables. Within the environment partial model, the networked driving simulation system is considered as a black box. That is, the internal structure and the constituent systems are not visible in this partial model. In this work, the environment partial model of the networked driving simulation system was determined based on the comprehensive analysis of typical driving simulation facilities as shown in Figure 5. Specifically, the analyses result in the identification of five main environment elements. These are described as follows:

- Drivers
  Human drivers represent a crucial environment element. They use the input devices within the driving platforms of the participating driving simulators to control a simulated vehicle in a virtual environment. The main input signals are: acceleration pedal position, brake pedal position, gear selector position, and steering wheel angle. The drivers receive feedback from their driving simulators in the form of motion or vibration, as well as visual and acoustic information. Basically, the motions and vibrations are generated by the eventually utilized motion platforms. In addition, some input devices, such as active steering wheels, can deliver relative motions to the drivers. The visual feedback is represented with virtual scenes displayed to the drivers via the visualization systems. The acoustic feedback is delivered via the acoustic systems as sound effects that accompany the 3D models. The visual and acoustic signals are generated often together by the visualization software.

- Simulation operator
  This can be a technician or a laboratory engineer, who is responsible for the general operation of the facility of networked driving simulation. Eventually, the simulation operator can be a domain-specific engineer or developer, who conducts some experiments using the networked driving simulation facility. The simulation operator can control the scenario by setting some simulation parameters. The networked driving simulation system returns simulation signals for monitoring purposes.

- Energy source
  This can be a wall outlet that provides electrical energy to the constituent systems and building components of the networked driving simulation system. Eventually, some components may require power supplies to convert the electrical power of the wall outlet to the levels suitable for their circuitry.

- Ground
  This is the physical base of the networked driving simulation system. Dynamic forces occur between the ground and the networked driving simulation system as actions and reactions, especially when the participating driving simulators are equipped with motion platforms.

- Environment
  The surrounding environment affects the networked driving simulation system through disturbing influences, such as humidity, dirt, light, and temperature. The networked driving simulation system affects the surrounding environment through the produced heat and operation noise.

Figure 5 shows the environment partial model of a networked driving simulation system. The environment elements are illustrated as yellow hexagons, while the networked driving simulation system is represented as a blue hexagon in the center of the model. The interrelations between the networked driving simulation system and the main environment components are categorized mainly as information, energy, and disturbing flows. The information flow denotes the exchange of information between the units of the whole system, such as the measured system variables or environment conditions. The energy flow denotes the transfer of energy between the units of the system, such as mechanical, thermal, or electrical energy. The disturbing flow represents any external factors affecting the normal operation of the system.

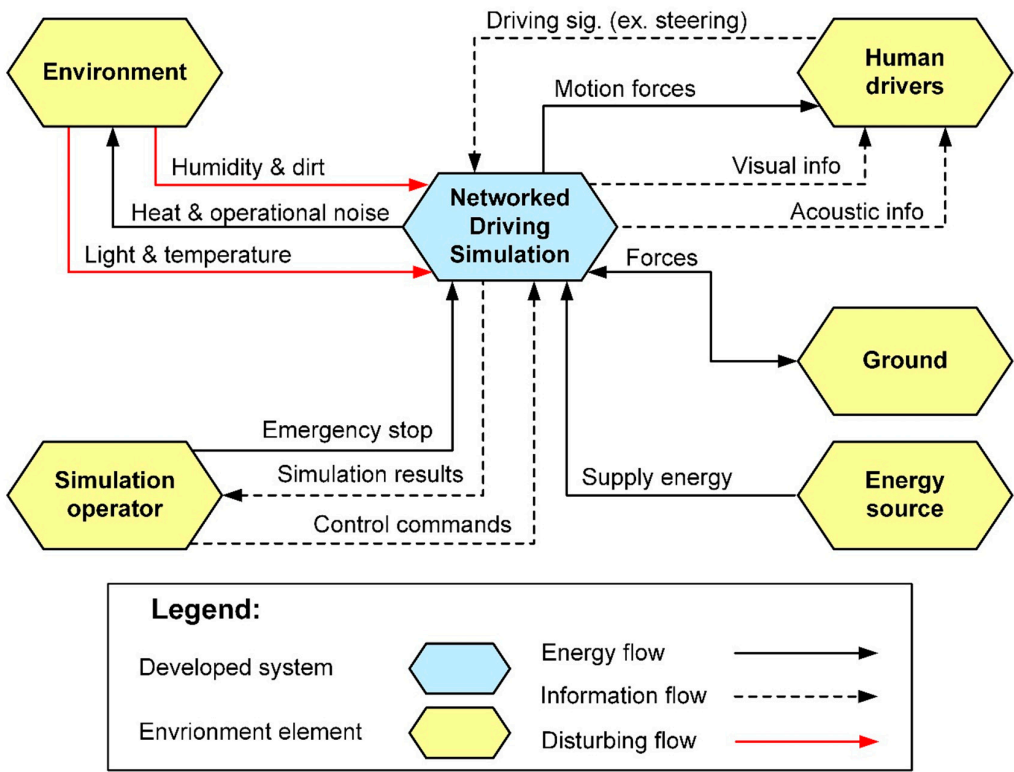
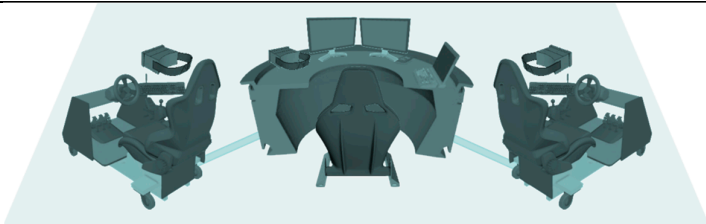



Figure 5. Example environment partial model of a networked driving simulation system.

Establishing the environment partial model ensures that all system surroundings are considered in a very early development phase. System users and maintenance personnel can use this partial

model to systematically determine and mitigate external causes of eventual future malfunctions. The following is an elaboration of the partial model of the application scenarios and its results with respect to networked driving simulation systems.

4.1.3. Define Application Scenarios

This partial model specifies the potential application scenarios of the networked driving simulation system. Each application scenario describes the target system with respect to the aim of use, operation modes, and the primary constituent systems and building components utilized in this particular application scenario. Specifically, the application scenarios are modeled using the so-called profile pages [19]. Each profile page contains characterizing information about a particular scenario, such as the title, ID, and last modification date. Moreover, each profile page provides a concise description of the application scenario [20]. Eventually, a sketch or a schematic can be added to provide better understanding of the application scenario. As a system of independent and heterogeneous systems, the partial model of the application scenarios of networked driving simulation in this work has a different form than that presented in Reference [19]. Specifically, an overall application scenario is described for the whole networked driving simulation system. This description principally highlights the ultimate goal of the developed system of systems. In addition, a purpose of use is described for each anticipated constituent simulation system. Two or more driving simulators and eventually a traffic simulator can represent a typical set of the constituent simulation systems. Furthermore, a rough description of the essential role of the communication system can be added eventually to the profile page. Table 6 shows the profile page of an example application scenario of a networked driving simulation system that is intended for use in modern driving schools. The example application scenario involves two driving simulators and a workstation for session control and monitoring.

Table 6. Example application scenario of a networked driving simulation system

Application Scenario 1	Multi-Driver Training in Driving Schools		Status 1 September 2017	Page 1
Description: An instructor at a driving school handles two trainees simultaneously in realistic and multi-interactive traffic scenarios. The trainees share a virtual traffic environment. They have to react to each other and adapt their driving behavior.				
Simulation System				
Simulation Entities			**Supplementary Components**	
Driving simulator 1	Driving simulator 2		Workstation	
Trainee 1 uses this driving simulator to experience different traffic situations in a safe virtual environment.	Trainee 2 uses this driving simulator to experience different traffic situations in a safe virtual environment.		Purpose of use: the driving instructor uses the work station to control and monitor the training session.	
Communication System				
Communication Technology		**Communication Architecture**		
It is a feasible communication technology that ensures a data exchange with little delay and loss rates.		To maintain system feasibility for driving schools, no communication architecture is utilized in this application scenario.		
Sketch				

The defined purposes of use can be used to determine and document a set of requirements for each anticipated constituent simulation system separately. This requirement description is refined in

the requirements partial model. The following is an elaboration of the requirements partial model and its results with respect to networked driving simulation systems.

4.1.4. Derive Requirements

This partial model specifies a comprehensive list of requirements of the networked driving simulation system. Principally, this list can include functional and non-functional requirements [20]. Moreover, the individual items of requirements can be denoted as demands or wishes (D/W). Table 7 shows an excerpt of an example list of requirements of a networked driving simulation system.

Table 7. Example list of requirements of a networked driving simulation system (excerpt)

ID	No.	Requirements of Networked Driving Simulation	D/W
		Requirement of Driving Simulator 1	
1	1	Scene simulation system	
	1.1	It shall cover a 120° horizontal field of view	D
	…	…	
	2	Motion simulation system	
	2.1	It shall provide three degrees of freedom	W
	…	…	
		Requirement of Driving Simulator 2	
2	1	Scene simulation system	
	1.1	It shall cover a 240° horizontal field of view	W
	…	…	
	2	Motion simulation system	
	2.1	It shall provide five degrees of freedom	D
	…	…	

As a system of independent systems, the structure of the list of requirements of networked driving simulation in this work has a different form than the standard form presented in the CONSENS specification technique [19]. Specifically, a separate set of requirements is defined for each independent constituent system and component that is denoted by a unique ID as shown in Table 7. The different sets of requirements together form the overall requirements of the networked driving simulation system. The following is an elaboration of the functions partial model and its results with respect to networked driving simulation systems.

4.1.5. Deduce Functions

The functions of the networked driving simulation system are defined based on system requirements and application scenarios. Interactive simulation, traffic simulation, operation management, data collection, and network communication are the fundamental system functions identified based on a comprehensive analysis of the networked driving simulation. Each of these defined functions may undergo further top-down hierarchical subdivisions [20]. Figure 6 shows the functions and sub-functions of a networked driving simulation system.

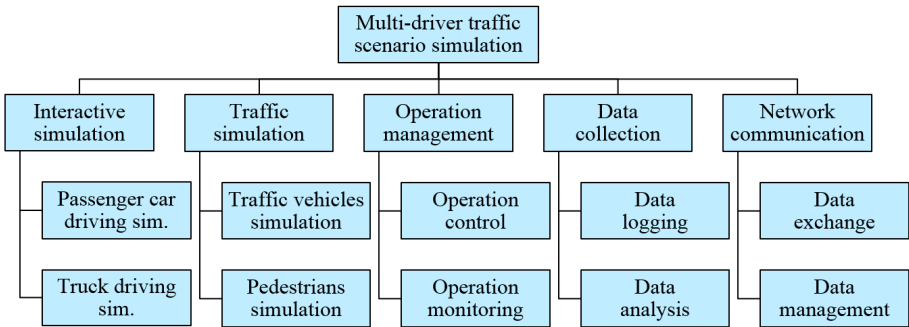



Figure 6. Example functions partial model of a networked driving simulation system.

The defined system functions are realized by solution patterns towards system concretization. For instance, the interactive simulation function can be carried out by two or more driving—like passenger car or truck—simulators of different or equal complexity grades. The simulation of traffic vehicles and pedestrians can be performed with an independent traffic simulator. A workstation can provide capabilities of operation control and monitoring. A database console can carry out the data logging function and serve for subsequent simulation session analysis. A communication technology, for instance, such as Ethernet, can carry out the data exchange between the constituent systems and building components. Data management can be achieved by communication architecture, like high-level architecture [21]. As a lot of solutions may be available, a classification scheme (morphological box) can be utilized to facilitate the systematic combination of available solutions [17]. According to the SoS definition adopted in this work, networked driving simulation systems are composed of further heterogeneous constituent systems and building components. Therefore, combining only compatible solutions does not apply in this context in contrast to the design of typical mechatronic systems [17]. The following is an elaboration of the active structure partial model and its results with respect to networked driving simulation systems.

4.1.6. Build Active Structure

The active structure partial model is created based on the defined system functions and the possible constituent systems and building components of networked driving simulation. In contrast to the environment partial model that considers the whole system as a black box, the active structure partial model concretizes the system by illustrating its internal structure in more detail [20]. Specifically, it shows the main system components and their primary interrelationships in the form of information and energy flows. Figure 7 shows the active structure of a networked driving simulation system including all possible (yet not all necessary) constituent systems and building components. These correspond to the particular system functions defined previously.

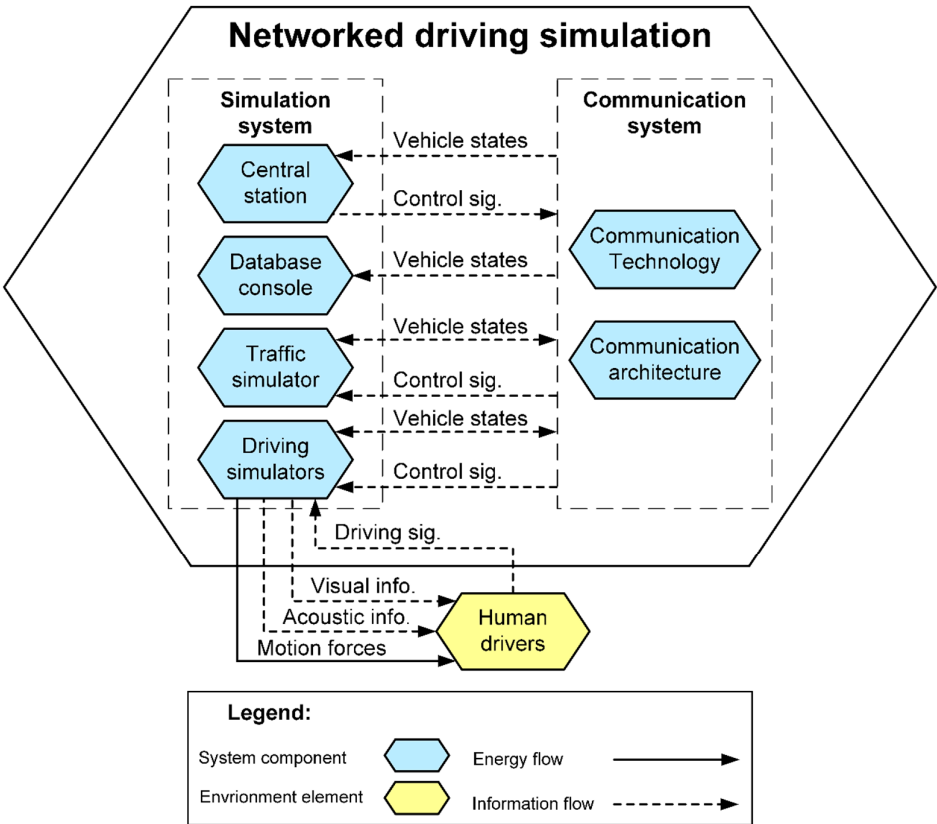



Figure 7. Example active structure partial model of a networked driving simulation system.

The presented active structure partial model includes six constituent systems and building components that belong to two main member groups: simulation system group and communication system group. On the one hand, the driving simulators and the traffic simulator are constituent systems that belong to the simulation system group. Similarly, the workstation and database station are considered as supplementary system components that belong to the simulation system group. On the other hand, the communication technology and the communication architecture belong to the communication system group. To illustrate, Figure 7 depicts the drivers that represent a crucial environment element to illustrate the interaction with the driving simulators that act as central constituent systems of the networked driving simulation system.

In summary, the collective results of the five specified partial models form together a principle solution that acts as a communication and cooperation basis between the experts of the involved development domains [20]. This basis is used for the subsequent design and development phases of the networked driving simulation system in this work. The following subsection presents a comprehensive analysis of system components as a further step towards concretization.

4.2. System Components Analysis

The objective of the second development phase is the identification, description, and classification of the components of the networked driving simulation system. This development phase depends mainly on the results of the system specification phase. Nonetheless, prior to the identification of the system components, a distinction must be clear between the terms 'constituent systems' and 'building components'. On the one hand, the constituent systems are independent participants within the networked driving simulation system. They can carry out meaningful tasks of their own, even if they are not networked to an entire system. On the other hand, the building components can provide services to the system of systems. However, they cannot carry out meaningful tasks of their own, more specifically, when they are not networked to one or more constituent systems. The following is an elaboration of the system components identification task and its results.

4.2.1. Identify System Components

The active structure partial model initially revealed the possible five system components of the networked driving simulation system. These system components can be identified further based on the presented distinction between the constituent systems and building components of the networked driving simulation. On the one hand, the driving and traffic simulators are constituent systems. They can be used separately for useful, independent purposes. On the other hand, workstations, database consoles, and the communication system are building components. They present services to the entire system, but they are not useful if utilized independently without constituent systems. Table 8 presents the five system components and the clear distinction between the constituent systems and the building components.

Table 8. Distinction between constituent systems and building components

Distinction	System Components of Networked Driving Simulation				
	Driving Simulators	Traffic Simulators	Workstations	Database Consoles	Communication Systems
Constituent System	x	x			
Building Component			x	x	x

Based on the results of the identification task, the following is an elaboration of the components description task and its results. The concrete role of each system component within the networked driving simulation is highlighted.

4.2.2. Describe System Components

The five main identified system components of the networked driving simulation system can be characterized as essential and optional components based on their roles. Essential system components are vital to achieve the central purpose of networked driving simulation: multi-driver traffic scenario simulation. However, the system of networked driving simulation can operate without the optional components and still achieve this central purpose. The following is a concise description and a role characterization of each system component of the networked driving simulation system from a solution-neutral perspective.

- Driving simulators

  They are operated by human drivers to control the respective simulated vehicles. The driving simulators can be of different types, such as a passenger car simulator or a truck simulator. Moreover, driving simulators of different complexity grades can principally participate within the networked driving simulation system. By any means, the participation of at least two driving simulators is necessary not only to achieve the central purpose, but also to establish a system of networked driving simulation. If a third driving simulator is added to the system, one of the driving simulators can be eventually considered as an optional component. However, driving simulators are characterized as essential constituent systems in general.

- Traffic simulators

  They generate traffic participants, such as programmed vehicles and pedestrians, to add more complexity to the multi-driver traffic scenario. One traffic simulator is often sufficient for the system of networked driving simulation. However, more than one traffic simulator can be integrated within the system to provide different granularity levels of traffic simulation, such as macroscopic and microscopic traffic flows [22]. The system of networked driving simulation can operate without the utilization of traffic simulators. In this case, the multi-driver traffic scenario simulation depends only on the participating interactive driving simulators. Hence, traffic simulators are characterized as optional constituent systems.

- Workstations

  A workstation is utilized to provide control and monitoring operations on the networked driving simulation system. That is, the simulation operator can make commands to stop/start the system and control particular building components. Moreover, the simulation operator can monitor various signals that give indications about the operation and performance of the system and its building components. Principally, the system of networked driving simulation can operate without the use of a workstation. Hence, the workstation is characterized as an optional building component.

- Database consoles

  A database console is utilized to capture and save the simulation data. Moreover, operators and developers can conduct simulation analysis or generate after-action-review reports. However, the system of networked driving simulation can operate without the use of a database console. Hence, the database console is characterized as an optional building component.

- Communication systems

  In this work, a communication system includes two categories of building components: communication technologies and communication architectures. On the one hand, the communication technologies are responsible for information exchange, such as Ethernet, CAN, and FlexRay. These communication technologies differ mainly through the provided networking characteristics. The system of networked driving simulation cannot operate without the use of a communication technology. Hence, communication technologies are characterized as essential building components. On the other hand, the communication architectures are responsible for networked simulation management, such as Distributed Interactive Simulation (DIS) and High-Level Architecture (HLA) [23,21]. These communication architectures differ mainly through the provided functions and services that can be useful for networked simulation.

Unlike the communication technologies, the system of networked driving simulation can operate without the use of communication architectures. Hence, the communication architectures are characterized as optional building components. Table 9 shows the identified and described system components together with their role significance within the networked driving simulation system.

Table 9. Role significance of constituent systems and building components

| Role Significance | Components of Networked Driving Simulation Systems | | | | | |
| | Constituent Systems | | | Building Components | | |
	Driving Simulators	Traffic Simulators	Workstations	Database Consoles	Comm. Technologies	Comm. Architectures
Essential Component	x				x	
Optional Component		x	x	x		x

The identification and description of system components provided more understanding towards system concretization. Using the results of the identification and description tasks, the following is an elaboration of the components classification task. Main categories of system components are specified as an essential preparation step for the subsequent development phases.

4.2.3. Classify System Components

Based on the functions and active structure established in the previous development phase, system components can be classified into two main groups: the simulation system group and communication system group as shown in Figure 8.

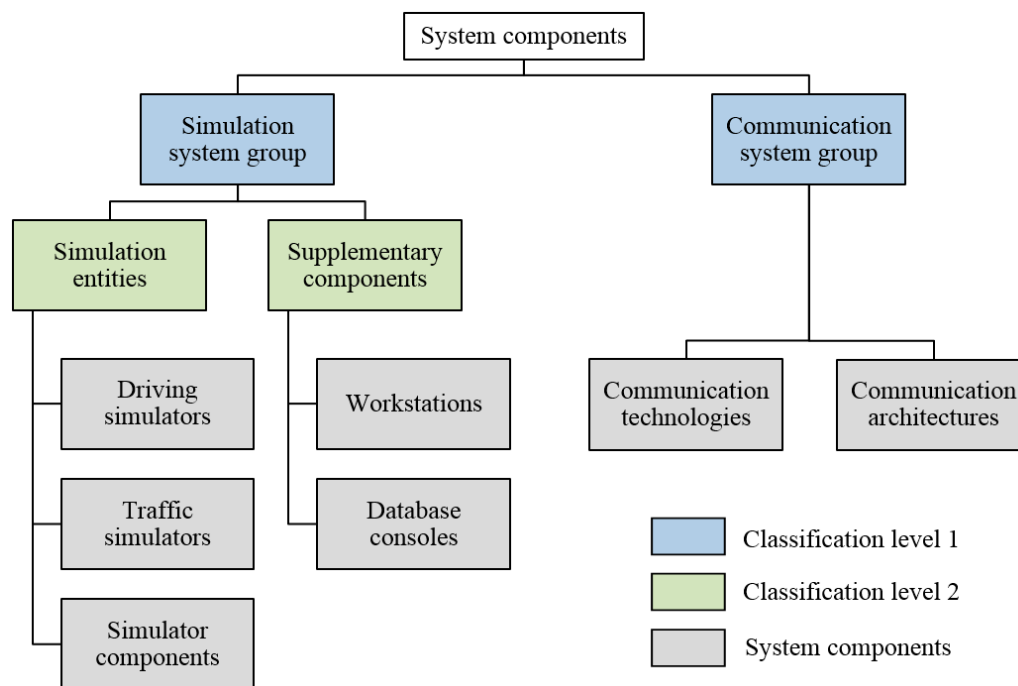



Figure 8. Classification of networked driving simulation system components.

On the one hand, the simulation system group is classified further into simulation entities and supplementary components. The driving and traffic simulators are assigned to the simulation system group under the simulation entities. Moreover, individual components of driving simulators, such as visualization systems and motion platforms, belong to the simulation entities as well. Comprehensive identification, description, and classification of the individual components of driving

simulators are presented in References [6,7]. The workstations and database consoles can be assigned to the simulation system group. However, these belong to the supplementary components to indicate their relative uncritical role within the system of networked driving simulation. On the other hand, the communication system group includes various communication technologies and communication architectures. This particular classification reflects the functional role of the five main system components within the networked driving simulation as a system of systems. It is used as a basis for the next development phases. The following section presents the development of system databases and the deployment of solution elements.

4.3. System Databases Development

The third development phase depends on the results of the preceding phases and presents another step towards system concretization. The objective of this development phase is to build system databases that contain entries of the analyzed system components, which are concretized as solution elements. While system components are solution-neutral, the solution elements represent concrete products of the system components provided from different developers and manufacturers. In addition, an approach to filling the system databases with entries of the solution elements is presented in this development phase. The system databases are accessible and editable from the system of systems (SoS) configuration software. This is necessary for the subsequent development phases that address the configuration of the networked driving simulation system and the generation of system models. The following is an elaboration of the structure of the system databases.

4.3.1. Build System Databases for Solution Elements

In this task, system databases are developed based on the classification of system components presented in the previous phase. More specifically, two system databases are built for the two main groups of system components: simulation system database and communication system database.

On the one hand, the simulation system database includes only solution elements of components related to the simulation task of the overall system of networked driving simulation. The simulation system database has four tables representing the four component categories that belong to the simulation system group: driving simulators, traffic simulators, workstations, and database consoles. These four database tables are filled with entries of the corresponding solution elements. A database for the solution elements of the individual driving simulator components has been created and filled within Hassan's method [7]. It has tables for solution elements of three categories of driving simulator components: hardware, software, and resources. This particular database is merged with the simulation system database developed in this work. Its entries can be used eventually during the next phase of system configuration.

On the other hand, the communication system database includes solution elements of components related to the communication task of the overall system of networked driving simulation. The communication system database has two tables representing the two components that belong to the communication system group: communication technologies and communication architectures. Similarly, these two database tables are filled with entries of the corresponding solution elements. Figure 9 depicts the two developed system databases and the main associated tables.

The system databases can be implemented with different database development tools. However, the selected database development tool and the implementation approach must allow the fundamental database operations: create, read, update, and delete [24]. These basic database operations are typically summarized using the acronym 'CRUD' according to the first letters of the four operations, respectively. This particular feature is necessary to make the system databases accessible and editable from the SoS configuration software. The following is an elaboration of the specified attributes of the main database tables.

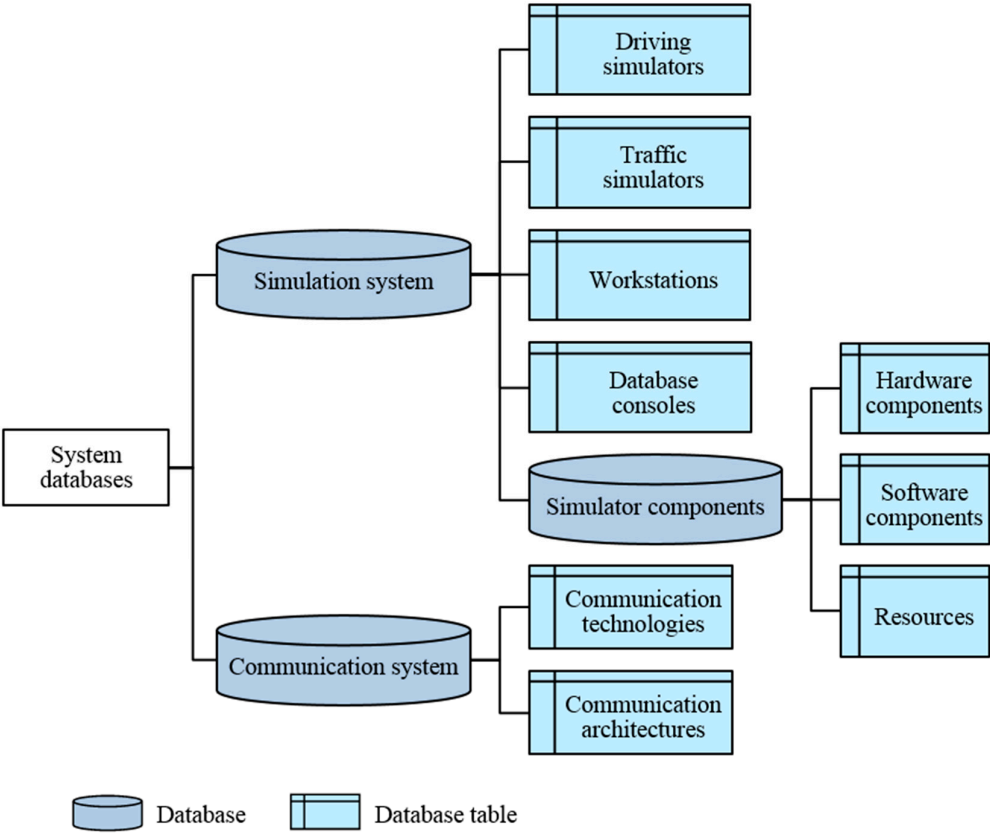
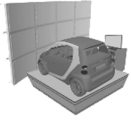
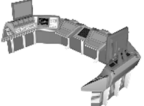
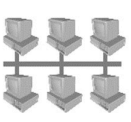
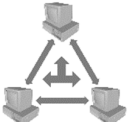



Figure 9. The developed system databases and the main tables.

4.3.2. Fill System Databases with Solution Elements

Apart from the database tables used in Hassan's methods, six database tables were identified in the previous development task to include solution element entries of six system component categories: driving simulators, traffic simulators, workstations, database consoles, communication technologies, and communication architectures. These database tables must be specified further with attributes before registering entries of corresponding solution elements. The table attributes presented in this development task can be extended arbitrarily so that the design conforms to the open systems approach of the system of systems engineering [9].

Table 10 shows the database table created to include entries of existing driving simulators as available solution elements. For example, entries of three driving simulators are included: ATMOS (Atlas Motion System) driving simulator, Airmotion_ride driving simulator, and HNI (Heinz Nixdorf Institute) PC-based driving simulator. These driving simulators were developed at the Heinz Nixdorf Institute in Paderborn, Germany within a previous research project [25]. They have different fidelity levels and can be used for different application scenarios. A comprehensive description of the technical specifications of these driving simulators is presented in Reference [25].

Table 10. Database table of driving simulators with building components of specified fidelity levels

ID	Name	Visualization System					Motion System					Acoustic System			Driver Platform			
		Eye Distance (A)	Field of View (B)	Rear-View Mirrors (E)	Continuity (F)	Resolution (G)	Motion Standard (L)	Motion < 6 DOF (K)	Motion > 6 DOF (M)	Vehicle Dynamics (N)	Tire Model (O)	Primary Sound (P)	Auxiliary Sound (Q)	Sound System (R)	Mock-Up (S)	HMI (T)	Steering (U)	Pedals Set (V)
1	ATMOS simulator	2	3	2	3	2	–	3	–	3	3	1	1	1	4	2	2	1
2	Airmotion _ride	2	1	3	1	2	1	–	–	2	2	1	1	1	1	1	1	1
3	HNI PC simulator	1	1	3	1	1	–	–	–	2	2	1	–	1	1	1	1	1

In addition to the ID and name attributes, this database table has four more attributes in accordance with the four driving simulator components identified in Negele's guidelines [6]: visualization system, motion system, acoustic system, and driver's platform. Reference [6] provides a detailed description of these components together with all possible fidelity levels. In accordance with Negele's guidelines, specific fidelity levels are assigned to the individual components of the driving simulators in this work as shown in Table 10. The subsequent application-oriented configuration and model generation processes will make use of these fidelity level assignments. To conform to the SoS definition adopted in this work, the traffic simulation is considered as a separate task that is independent of the driving simulators in contrast to both Negele's guidelines and Hassan's method. The database table of traffic simulator entries mainly has four attributes in addition to the ID: environment database, objects simulation, granularity level, and visualization type. A detailed description of these characteristics is provided in Reference [6]. The third member of the simulation system database is the workstations table. This database table includes solution element entries of workstations that have different capabilities or specifications. The set of attributes can include the ID, name, manufacturer, number of monitors, computer specifications, etc. The fourth member of the simulation system database is the database consoles table. Similarly, this database table includes entries of solution elements of database consoles that have different capabilities or specifications. The set of attributes can include the ID, name, developer, design software, interfaces, storage capacity, and computer specifications. Some of the solution elements of the individual driving simulator components can be characterized based on the features identified in Negele's guidelines [6]. Specifically, nine driving simulator components were identified in Hassan's method: visualization system, motion platform, driver's platform, acoustic system, visualization software, motion controller, vehicle model, HMI interface, and acoustic software. The visualization software, motion platform controller, and HMI interface in Hassan's method do not have corresponding features in Negele's guidelines. Hence, the remaining six driving simulator components from Hassan's method are correlated to features from Negele's guidelines. This correlation step represents the link established in this work between Hassan's method and Negele's guidelines. The database tables of the individual driving simulator components are extended in this work to allow for the fidelity level assignments of the solution elements. With respect to the communication system, a database table includes entries of all available communication technologies as solution elements. These communication technologies differ mainly through network characteristics. In addition to the ID and name attributes, this database table has one main attribute representing significant characteristics of the communication technologies [26]. For instance, this attribute is divided into eight sub-attributes: bandwidth, latency, jitter, packet loss rate, determinism, error rate, transmission mode, and segment length. Another database table includes entries of all available communication architectures as solution elements. These communication architectures differ mainly through the provided functions and services for networked simulation. In addition to the ID and name attributes, this database table has one main attribute representing significant characteristics, functions, and services of the communication architectures. Further attributes can be added to the database table of communication architectures, such as the provider and the version of the underlying standard. The following subsection presents a central development phase of the design method.

4.4. Configuration Methods Development

The previous phases provided a comprehensive understanding of the networked driving simulation system. The system components were identified, described, and classified. Moreover, system databases were developed in accordance with the results of the system components analysis. Table 11 shows a morphological box that includes the identified system components and symbolic, exemplary solution elements.

Table 11. Morphological box of the networked driving simulation system (excerpt)

System Components	Solution Elements		
	1	**2**	**3**
Driving Simulators			
Traffic Simulators	SUMO	Aimsun	HNI
Workstations			
Database Consoles	SQL Server	ORACLE	IDERA
Communication Technologies	Ethernet 1 Gb	CAN Bus	InfiniBand
Communication Architectures			

In general, the morphological box represents a well-established approach that can be used particularly when it comes to system composition [17]. In this work, the system components and the corresponding available solution elements are inserted into the rows of a morphological matrix as shown in Table 11. While the first four system components belong to the simulation aspect of the networked driving simulation system, the latter two system components belong to the communication aspect. The solution elements of system components must be combined systematically to obtain an overall solution. The networked driving simulation system is composed of independent, heterogeneous systems. Hence, the combination of solution elements is not governed by their consistency or compatibility in this work in contrast to Hassan's method [7]. The solution elements are selected based on the offered capabilities and functionalities with respect to the requirements of the concerned application scenarios. The following is a discussion of a configuration method for the simulation aspect of the networked driving simulation system.

4.4.1. Simulation System Configuration Method

The simulation system aspect in this work includes four system components: driving simulators, traffic simulators, workstations, and database consoles. The application-oriented selection of the participating driving simulators is the central task of the simulation system configuration. However, available driving simulators must be classified according to their capabilities to account for the subsequent phase of application-oriented system model configuration and generation.

Selection Approach for Available Driving Simulators

Classifying driving simulators into three categories (low-level, mid-level, and high-level) collectively is not practical [13]. A driving simulator may have high capability for one particular component and low capability for other components based on the purpose of use [14]. Hence, driving simulators are classified in this work in accordance with the 15 application classes defined in Negele's guidelines [6]. These application classes give more insight into the fidelity levels of the individual driving simulator components. While seven application classes are considered as not common or practical, eight application classes are fully specified in Negele's guidelines [6]. Nonetheless, the specifications of available driving simulators may not exactly fulfill the whole requirements of the application classes. Therefore, a cost function is defined to give a quantified indication of the specification/requirement deviations.

The relative significances of the individual driving simulator components are specified for each application class in Negele's guidelines [6]. For instance, the motion system is more significant than the visualization system for the application class 1a (skill-based responses and vehicle stabilization). In this work, the relative significance is quantified, where each driving simulator component takes a unique integer number from 1 to 4. Higher numbers mean more relative significances. With respect to the application class 1a for example, the significance numbers: 4, 3, 1, and 2 are assigned to the simulator components: motion system, visualization system, acoustic system, and driver's platform respectively. Based on the specified and quantified relative significances, Equation (1) presents the cost function developed in this work to give an indicative measure of the deviation between the specifications of the available driving simulators and the requirements of the application classes:

$$\text{Fidelity level deviation} = \sum_{m=1}^{4} \frac{\text{Significance}_m}{N} \times \left(\sum_{n=1}^{N} |\text{Level}_{\text{Req}(n)} - \text{Level}_{\text{Spec}(n)}| \right), \qquad (1)$$

where:

m	Designation of the driving simulator component
Significance_m	Specified relative significance value of a simulator component
n	Designation of the feature of a driving simulator component
N	Maximum number of features of a driving simulator component
$\text{Level}_{\text{Req}}$	Requirement feature fidelity level of a simulator component
$\text{Level}_{\text{Spec}}$	Specification feature fidelity level of a simulator component

This cost function is applied to each available driving simulator with respect to each application class. The minimal deviation value among all available driving simulators with respect to a particular application class indicates a best possible match between the specifications and the requirements. With respect to any particular application class, Equation (2) presents a simple function used to find the minimal deviation value among all driving simulators.

$$\text{Best matching simulator} = \text{Min} (\text{Deviation}_1, \text{Deviation}_2, \text{Deviation}_3, \dots, \text{Deviation}_n), \qquad (2)$$

where n is the number of available driving simulators. The resulting minimal deviation value is used to select the driving simulator, whose specifications best match the requirements of a concerned application class. Further cost functions can be eventually developed, provided that they can give unique selections of driving simulators. The developed cost function has been applied to the three exemplary driving simulators of Table 10 and showed very good results, where no ambiguous selections were provided. **Error! Not a valid bookmark self-reference.**Table 12 shows these results with respect to the eight specified application classes.

Table 12. Results of the developed cost function for three example driving simulators

Example Driving	Specified Driving Simulator Application Classes							
Simulators	1a	2b	3b	3c	4b	4c	5b	5c

ATMOS driving simulator	2.04	2.15	3.22	3.22	2.00	**1.81**	2.00	2.00
Airmotion_ride driving simulator	3.26	2.23	1.48	1.48	**1.38**	2.38	1.80	1.80
HNI PC-based driving simulator	3.15	2.30	**1.30**	**1.30**	1.96	2.96	1.78	1.78

Example minimal deviation values are highlighted in Table 12 supposing that the database table contains entries of only three driving simulators. For instance, the HNI PC-based driving simulator can be used for the application class 3b (navigation driving tasks with rule-based responses) and the application class 3c (navigation driving tasks with knowledge-based responses). The Airmotion_ride driving simulator can be used for the application class 4b that addresses secondary driving tasks with rule-based responses [6]. The ATMOS driving simulator can serve the application class 4c that addresses secondary driving tasks with knowledge-based responses [6]. If other entries of driving simulators are added to the database table and the cost function is applied, the results of the minimal deviation function will differ with respect to each application class.

Selection Approach for Further Available Simulation System Components

The selection of solution elements of traffic simulators, workstations, and database consoles is more convenient and straightforward due to the limited number of characterizing features. A similar approach can be applied to these simulation system components, however, without the use of a particular predetermined significance factor. That is, the deviation between the specifications of the available solution elements and the requirements of the application scenarios can be determined through any simple cost functions based on comparison tables.

Selection Approach for Available Driving Simulator Components

The available driving simulators may not be satisfactory if the deviations between their specifications and the requirements of the concerned application classes are large. In such cases, new driving simulators can be built through combining solution elements of the different driving simulator components. Hassan's method can be utilized for this particular purpose [7]. In this regard, the previous development phase presented an approach to assign feature fidelity levels from Negele's guidelines to the solution elements of the driving simulator components specified in Hassan's method. Moreover, the associated database table has been modified to include these assignments. According to Negele's guidelines, driving simulator components are characterized by a set of features. For instance, the visualization system is characterized by five features: eye distance, field of view, rear-view mirrors, continuity, and resolution.

Nonetheless, the features of an available solution element, together, may not have the exact fidelity levels that fulfill the corresponding requirements of a particular application class. Therefore, a cost function must be defined to give a quantified indication of the deviation. No relative significances for the features of the individual driving simulator components are specified in Negele's guidelines. Hence, no particular significance factor is used within the cost function. Equation (3) presents the cost function developed in this work to give an indicative measure of the deviation between the specifications of individual driving simulator components and the corresponding requirements of the application classes:

$$\text{Fidelity level deviation} = \sum_{n=1}^{N} \left| \text{Level}_{\text{Req}(n)} - \text{Level}_{\text{Spec}(n)} \right|, \tag{3}$$

where:

n   Designation of the feature of a driving simulator component
N   Maximum number of features of a driving simulator component
$\text{Level}_{\text{Req}}$ Requirement feature fidelity level of a simulator component
$\text{Level}_{\text{Spec}}$ Specification feature fidelity level of a simulator component

This cost function is applied to all available solution elements of each driving simulator component with respect to the corresponding requirements of each application class. Among all solution elements of a particular driving simulator component, the minimal deviation value indicates a best possible specifications match to the corresponding requirements of the concerned application class. With respect to any particular driving simulator component and an application class, Equation (4) presents a simple function used to find the minimal deviation value among all solution elements.

$$\text{Best matching component} = \text{Min}(\text{Deviation}_1, \text{Deviation}_2, \text{Deviation}_3, \dots, \text{Deviation}_n), \qquad (4)$$

where n is the number of available solution elements of any particular driving simulator component. This approach complements the process of driving simulator configuration introduced in Hassan's method [7]. Specifically, the selection of solution elements of driving simulator components is governed by their capability to fulfill the corresponding requirements of the concerned application scenarios. The following is a discussion of a configuration method for the communication aspect of the networked driving simulation system.

4.4.2. Communication System Configuration Method

The communication system aspect in this work includes two system components: communication technologies and communication architectures. While using a communication technology is essential for the operation of the networked driving simulation system, the communication architectures are classified as optional building components. There are a lot of solution elements for both building components from different providers. Moreover, the available solution elements are usually subjected to continuous development to establish variants of different characteristics and functions. A careful selection of the communication system is necessary to reach the expected outcomes of the networked driving simulation system.

Determining Priority of Communication Characteristics and Functions

Communication technologies are characterized typically by a set of network characteristics, such as bandwidth and latency. However, no particular communication technology can provide the best possible specifications regarding all network characteristics [27]. Therefore, it may be difficult to find an absolute optimal solution element for all application scenarios due to the presence of various conflicting network characteristics and myriad choices of available communication technologies. This leads to a typical multi-criteria decision-making problem [28]. Thereby, the network characteristics represent the criteria and the communication technologies represent the alternatives. In general, there are different methods in the literature to handle multi-criteria decision making problems [28]. Nonetheless, some of these methods require a prior assignment of priority weights to the different criteria. This is necessary to ultimately reach a compromised solution. The compromised solution in this regard refers to a choice that satisfies the most important criteria to a sufficient extent while partially satisfying the less important criteria.

Hence, the network characteristics in this context must be prioritized based on the requirements of the concerned application scenarios. Table 13 shows an excerpt of a priority analysis matrix that can be used to assign priority weights to the network characteristics of the communication technologies. At this stage, the priority weighting process is solution-neutral, where no particular communication technologies are considered.

Table 13. Priority analysis matrix including example network characteristics

Priority Scheme 1 = Equally Important 5 = More Important 10 = Much More Important 0.2 = Less Important 0.1 = Much Less Important	Network Characteristics					Sum of Weights	Priority Weight (%)
	Bandwidth	Packet Loss Rate	Determinism	Latency	Segment Length		
Bandwidth	x	5	0.2	1	10	16.2	29.35
Packet loss rate	0.2	x	0.2	1	5	6.4	11.59
Determinism	5	5	x	5	10	25	45.29
Latency	1	1	0.2	x	5	7.2	13.04
Segment length	0.1	0.2	0.1	0.2	x	0.4	0.730

The relevant network characteristics are listed vertically and horizontally in the priority analysis matrix. Based on the requirements of a concerned application scenario, a relative priority weight is assigned to each pair of different network characteristics according to a priority scheme. The priority scheme used in this work includes five levels as shown in

Table **13**. Exemplary relative priority weights are presented in

Table **13** for a set of five network characteristics. For instance, the bandwidth can be less important than the determinism, but much more important than the segment length for a particular application scenario. The overall relative priority weight of each network characteristic is calculated as a sum of weights. The final priority weighting percentages of each network characteristic are calculated according to Equation (5).

$$\text{Priority weighting}_n \ (\%) \ = \ (\text{Sum of weights}_n \ \times \ 100) \ / \left(\sum_{m=1}^{M} \text{Sum of weights}_m \right), \qquad (5)$$

where:
  n and m   Designations of the network characteristics n and m
  M   Total number of network characteristics

The priority weighting percentages reflect the unique significances of the individual network characteristics with respect to a concerned application scenario. Although the shown example priority analysis matrix includes only five network characteristics, it can be extended vertically and horizontally to include more network characteristics as desired.

Similarly, the communication architectures are characterized typically by a set of functions for networked simulation. However, no particular communication architecture can provide the best possible specifications regarding all functions. Hence, the communication functions must be prioritized with respect to the application scenarios to reach a compromised solution. A similar approach can be used to assign priority weights to the functions of the communication architectures based on the requirements of the concerned application scenarios.

Selection Approach for Available Communication Systems

After determining the relative priorities of the communication characteristics and functions, a decision-making method is required to avoid an exhaustive and impractical search among all available solution elements of the communication technologies and communication architectures. The cost–benefit analysis method is used in this work as a well-established decision-making process recognized by systems engineering [29]. Based on this method, Table 14 shows an exemplary assessment of three communication technologies with respect to five network characteristics.

Table 14. Example assessment according to the cost-benefit analysis method

Network Characteristics	Priority Weight (%)	Communication Technologies					
		Ethernet 10 Mbps		CAN Bus		InfiniBand	
		Fulfillment (%)	Partial Assessment (%)	Fulfillment (%)	Partial Assessment (%)	Fulfillment (%)	Partial Assessment (%)
Bandwidth	29.35	80	23.5	20	5.87	100	29.4
Packet loss rate	11.59	10	1.16	100	11.6	100	11.6
Determinism	45.29	00	0.00	100	45.3	00	0.00
Latency	13.04	00	0.00	100	13.0	00	0.00
Segment length	0.730	100	0.73	50	0.37	30	0.22
Final assessment (%)		25.39		76.14		41.22	

The network characteristics are listed vertically and the available communication technologies are listed horizontally within the shown assessment matrix. Principally, the assessment uses the results of the priority analysis scheme presented earlier in this sub-section. More specifically, each network characteristic is assigned to its priority weighting percentage that is calculated using the priority analysis scheme. The priority weighting percentages differ according to user preferences for the concerned application scenarios. For each communication technology, the extent of fulfillment of each network characteristic is determined using the values given by the user. Consequently, all available communication technologies are assessed partially with respect to the individual network characteristics using Equation (6).

$$\text{Partial assessment (\%)} \; = \; \frac{\text{Priority weight (\%)} \;\times\; \text{Fulfillment (\%)}}{100}, \tag{6}$$

The final assessment of each communication technology is calculated as the summation of all partial assessments according to Equation (7).

$$\text{Final assessment}_n \; (\%) \; = \; \sum_{m=1}^{M} \text{Partial assessment}_m \; (\%), \tag{7}$$

where:

  n   Designation of the communication technology n
  m   Designation of the communication characteristic m
  M   Total number of communication characteristics

Equation (8) presents a simple function that is used to find the best matching communication technology, which has the highest final assessment value.

$$\text{Best matching communication technology} \; = \; \text{Max}(\text{Final assessment}_1, \dots, \text{Final assessment}_n), \tag{8}$$

where n is the number of available communication technologies.

A quite similar approach can be used for the selection of the communication architectures. That is, the user prioritizes the functions and services based on the concerned application scenarios to calculate priority weighting percentages. The available communication architectures are assessed according to the prioritized functions and services. Consequently, a simple function can be used to select the best matching communication architecture that has the maximal assessment value.

4.5. System Models Development

The previous development phases and their tasks focused on the comprehensive analysis of the whole system and the development of selection approaches for the simulation and communication aspects. The outcomes are embedded in the SoS configuration software to save efforts and time of non-expert system users. The last development phase is concerned with the actual creation of application-oriented system models based on the outcomes of the previous phases. The following is an illustration of the system configuration sequence to compose application-oriented system models for networked driving simulation.

4.5.1. Specify Configuration Sequence

The selection sequence of system components is specified in this task. The system components have been classified into two groups in the second development phase of the procedure model: the simulation system group and communication system group. Basically, the components of the simulation system group are selected before the components of the communication system group. The number of the selected simulation system components—and hence, the amount of the exchanged data packets—can affect the determination of some network characteristics, such as the bandwidth. Figure 10 illustrates the selection sequence of the simulation and communication system components. The user is guided by the SoS configuration software through this configuration sequence. The configuration process is based on the selection approaches developed in the previous phase of the procedure model. The system user can navigate back and forth arbitrarily along the configuration sequence to modify the selections.

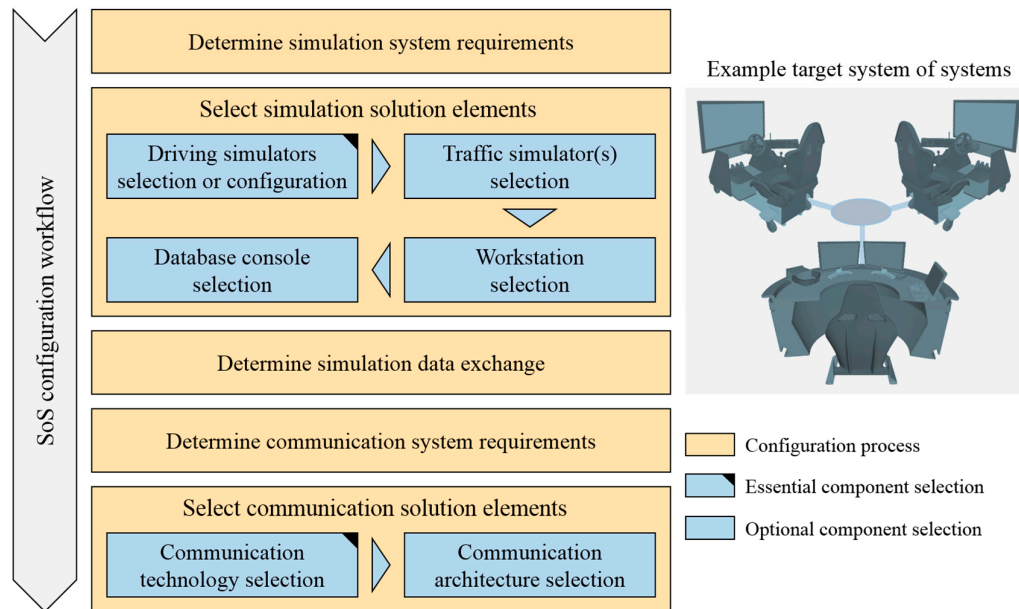



Figure 10. System configuration workflow and selection sequence of system components.

The user starts the configuration process by determining the concerned response type and driving task for each participating driving simulator. The SoS configuration software determines the corresponding application classes. Consequently, the SoS configuration software finds the best matching driving simulators within the entries of the corresponding database table. Alternatively, the user can configure new driving simulators from the individual simulator components using the SoS configuration software. Similarly, the SoS configuration software finds the best matching traffic simulator within the entries of the corresponding database table according the concerned application classes. The user can skip this suggestion as traffic simulators are optional components for the networked driving simulation. After that, the user can select a workstation and a database console based on the desired requirements. Similarly, the user can skip this step as the utilization of these system components is optional for networked driving simulation. As an intermediate step before the selection of the communication system components, the user determines the information exchange between selected simulation system components. After that, the SoS configuration software finds a best matching communication technology according to specified and prioritized requirements regarding the network characteristics. Some of these requirements—e.g., bandwidth—can be only estimated as worst-case scenario based on the amount of exchanged data packets [30]. Consequently, the result is an initial proposed communication technology with the best matching network characteristics. Finally, the SoS configuration software finds a best matching communication architecture according to the eventually specified and prioritized requirements regarding the communication functions and services. The user can skip this step as the utilization of a communication architecture is optional for networked driving simulation. Before the final creation of a system model, the following task is performed to assure the appropriate selection of the communication technology.

4.5.2. Examine Network Behavior

In this task, the initially selected communication technology is examined to make sure that the eventually estimated requirements of network characteristics still can guarantee proper system operation. A network simulator can be utilized for this purpose as supplementary software [31]. Principally, network simulators are used to simulate the network behavior using mathematical formulas and models of the network protocols. There are commercial network simulators, such as OPNET and QualNet. Other network simulators are available as free and open source packages, such as NS2, NS3, J-Sim, SSFNet, and OMNeT++.

The Automotive Network Diagnoser (ANDi) from Technica Engineering GmbH in Munich, Germany is utilized particularly in this work. It is a user-friendly test and simulation environment that supports various communication technologies and operation platforms. Using this network simulator, users can construct a virtual network by creating nodes connected by network segments in a form that replicates a desired real network topology. The characteristics of the initially selected communication technology are provided along with the estimated amount of data packets to start a simulation. According to the observed network behavior, the initial requirements of network characteristics are eventually modified. In this case, the selection step of communication technologies is recalled to select a more appropriate solution element. This process is repeated until the simulated network behavior exactly meets or comes very close to the requirements of the concerned application scenario regarding the communication technology. The following task discusses the creation of a system model after finishing the configuration process.

4.5.3. Generate System Models

In this task, a system model is generated by the SoS configuration software in the form of a comprehensive report that contains concise information about the selected solution elements. In general, model-based systems engineering relies on models to progress from the level of requirements to the level of system realization [32]. That is, it follows a model-centric approach rather than a traditional document-centric approach. In practice, the SoS design and modeling cannot be carried out through a conventional system development process. Yet the complexity of the SoS design can be reduced considerably by following a model-centric approach [32]. Agent-based modeling can provide a practical tool in this regard [32]. It is a relatively new approach for modeling complex systems that are composed of further interacting systems. The agent-based modeling method defines the roles of all system components from a bottom-up perspective [33]. Hence, the created agent-based model can describe the emerging system as a whole based on the roles and interactions of the constituent systems and components. There is no universal agreement on a precise definition for the term 'agent'. In this work, an agent is a constituent system or a building component as identified, described, and classified in the second development phase of the procedure model. A formal visual scheme is typically required to apply the agent-based modeling principles.

The unified modeling language (UML) is a practical means for graphical visualization during system conception [34]. It provides an abstract modeling level that can be used for the system-level design. Specifically, UML uses a set of well-defined elements that are independent of any particular programming language to describe a system as high-level structures. Reference [34] discusses the UML diagrams that can be used practically for agent-based modeling: sequence, state, activity, and class diagrams. The sequence diagrams can be used if the sequential interaction of agents over time is a significant design aspect. The state diagrams can be used if the change of the internal states of the agents is of particular interest. The activity diagrams are similar to the traditional flow charts. They can be used if it is important to analyze the activity progress of system components. The class diagrams consist typically of classes and different types of relationships, such as association, composition, and inheritance [34]. They can be used when the focus is given to system composition and the relationships between its components. A detailed discussion about the UML class diagrams is presented in Reference [35].

The application-oriented modeling of networked driving simulation in this work is concerned particularly with the system components and their relationships. Hence, it adopts the agent-based modeling principles together with the concepts of the UML class diagrams. Specifically, the system model consists of three parts that are constructed automatically by the SoS configuration software based on the results of the configuration process.

System Model Components—Part I

The first part contains a UML class diagram that gives a holistic overview about the configured networked driving simulation system. The UML class diagram is further divided into two groups, which are called packages in accordance with the UML notations [35]. While the first package

addresses the simulation system aspect, the second package addresses the communication system aspect. Each package encompasses various blocks. The parent blocks are classes that represent the utilized system components. The parent blocks within the simulation system package have inheritance relationships with an upper parent simulation system class. Similarly, the parent blocks within the communication system package have inheritance relationships with an upper parent communication system class. The child blocks are instances that represent the selected solution elements. The instances have abstraction relationships to the corresponding classes (parent blocks). Each block within the class diagram is typically composed of three compartments stacked vertically. The top compartment includes the name. The middle compartment lists the significant specifications and characteristics. The bottom compartment lists the main operations performed by the system component or the solution element.

System Model Components—Part II

The second part consists of two sections containing radar charts. Generally, the radar chart is a demonstration method for the visualization of data sets that include multiple quantitative variables. These variables are represented on axes that start from an origin point. Particularly, the first section of this part contains radar charts illustrating the deviation between the specifications of each selected simulation solution element and the corresponding application requirements. Analogously, the second section contains radar charts illustrating the deviation between the specifications of each selected communication solution element and the corresponding application requirements. This part gives an overview of the eligibility of each selected solution element with respect to the concerned application scenario. This visual demonstration of the specifications and the requirements enables users to decide whether to proceed or to navigate back to select other alternatives.

System Model Components—Part III

The third part contains a list of the simulation data that each selected solution element sends and/or receives. The simulation data are characterized by different attributes, such as the sender, unit, and type. This list of exchanged simulation data can be used during the preparation of the constituent systems and building components for system realization. Moreover, it can be used as a basis if a communication architecture is utilized to particularly provide a data distribution management service [36]. The Extensible Markup Language (XML) is chosen for the construction of this part. The XML defines a set of rules to encode the information in an easy format that can be understood by humans as well as software programs. Furthermore, the hierarchical structure supported by the XML format makes it convenient to trace and find the concerned information. A comprehensive discussion about the XML and its rules is presented in Reference [37].

Based on the information demonstrated through the created system model, the user can still navigate back to alter particular selections when necessary. Finally, the user can save the created system model and/or print it to start the system realization. The generated model is used at this stage to easily communicate information about the system that shall be built. Three system models are shown together with example application scenarios in the next section.

5. Validation

Three example multi-interactive application scenarios are presented in this section in order to validate the design method. A comprehensive description is provided for each example application scenario. Corresponding system models are generated using the developed SoS configuration software. Specifically, the solution elements of the simulation and communication system components are selected using the approaches adopted in the developed method and embedded in the SoS configuration software. The actual application requirements and the preferences of the system user play a considerable role during the configuration process. Beside the validation purpose, the presented application scenarios deliver an example line of thoughts to illustrate how to use the SoS configuration software for creating system models.

5.1. Multi-Interactive Training with ADAS

5.1.1. Scenario Definition

Although ADAS are designed to reduce the burden on drivers, the complexity of user interface grows with increasing the number of automated functionalities. This demands some of the driver's attention and introduces a considerable cognitive load. Therefore, conventional training with driving simulators must be adapted for more immersion and a capability for interactive supervision and instruction [5]. The application scenario of this validation example is to perform multi-interactive, supervised training sessions to learn ADAS functions and avoid eventual overestimation of their capabilities.

5.1.2. Configuration Process

Based on this application scenario, a simulation environment consisting mainly of two driving simulators is defined. While one driving simulator is used by a trainee, the other driving simulator is used by a training instructor. The trainee is introduced to various ADAS functions—e.g., blind spot and congestion assistance systems—while driving through an unfamiliar road network. The trainee activates and handles the settings of the ADAS functions and responds to the dashboard indicators. This purpose of use falls within the driving simulator application class 4c that addresses knowledge-based responses and secondary driving tasks. The training instructor has a simple navigation driving task. The aim is to drive interactively within the same virtual environment to observe the traffic situation and to eventually introduce unexpected driving maneuvers. The response of the training instructor is not of concern as a matter of course. The purpose of use can be represented by the driving simulator application classes 3a, 3b, and 3c that address different responses with navigation driving tasks. The application class 3c is chosen for convenience. The determined application classes of the two participating driving simulators are used together as the first actual input to the SoS configuration software. Among the available driving simulators, the SoS configuration software suggests two particular driving simulators that best fulfill the requirements of application classes 4c and 3c.

In this application scenario, no traffic simulator is chosen as the trainee has to react only to the maneuvers introduced by the training instructor. No workstation is required for this platform as the training instructor already participates interactively in the virtual traffic scenario. A database console can be used to capture, save, and replay the simulation data for analysis and after-action review. A lot of data must be exchanged between the participating simulators. Not only the position and orientation data of the simulated vehicles are exchanged through the network, but also additional information for better immersion and engaging training sessions, such as front and rear lamps state, turning indicator lamps state, and front wheels orientation. A bandwidth of 10 Mbps (megabit per second) is required initially according to a worst-case calculation of the number of exchanged data packets [30]. Lower priority levels are assigned to the other communication characteristics, such as real-time data delivery or data loss rate. Accordingly, a 10 Mbps Ethernet with User Datagram Protocol is suggested by the SoS configuration software for this application scenario. The simulated network behavior using the ANDi software confirmed the eligibility of this selection. No standard architecture for networked simulation is selected as networked simulation management functions are not required for this application scenario.

5.1.3. Generated System Model

A system model is created by the SoS configuration software based on the scenario analysis and the configuration process. Figure 11 shows a simplified version of the simulation package of the UML class diagram containing the selected simulation system components.

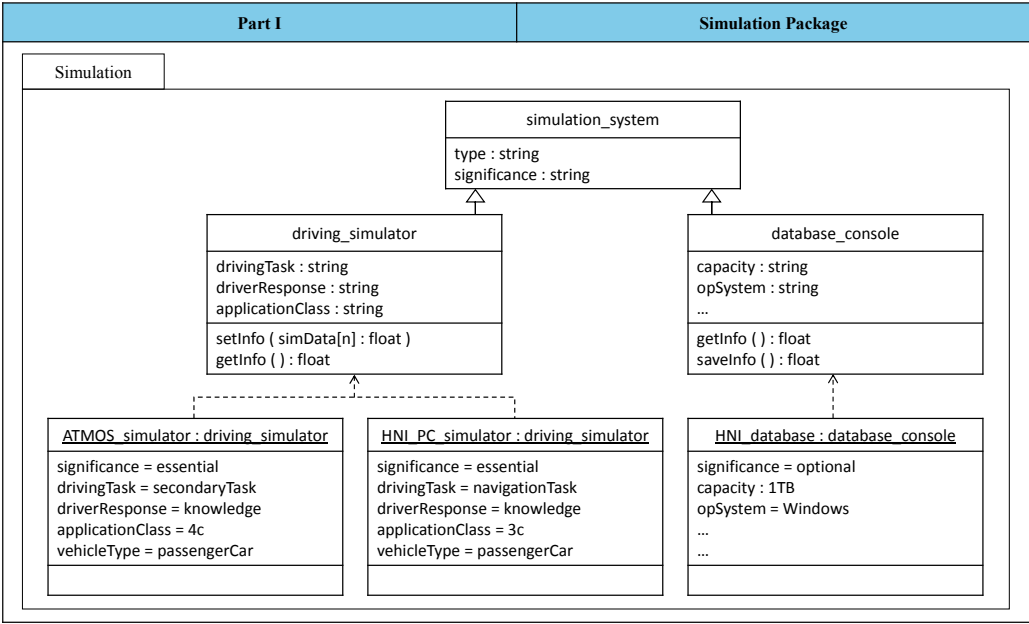



Figure 11. UML class diagram of the selected simulation system components—package 1.

The simulation package includes a main class named simulation_system. Two classes inherit the simulation_system class: driving_simulator and database_console. The driving_simulator class has two instances representing the two selected driving simulators: ATMOS_simulator and HNI_PC_simulator. The database_console class has one instance representing the selected database console: HNI_database. Figure 12 shows a simplified version of the communication package of the UML class diagram containing the selected communication system components.

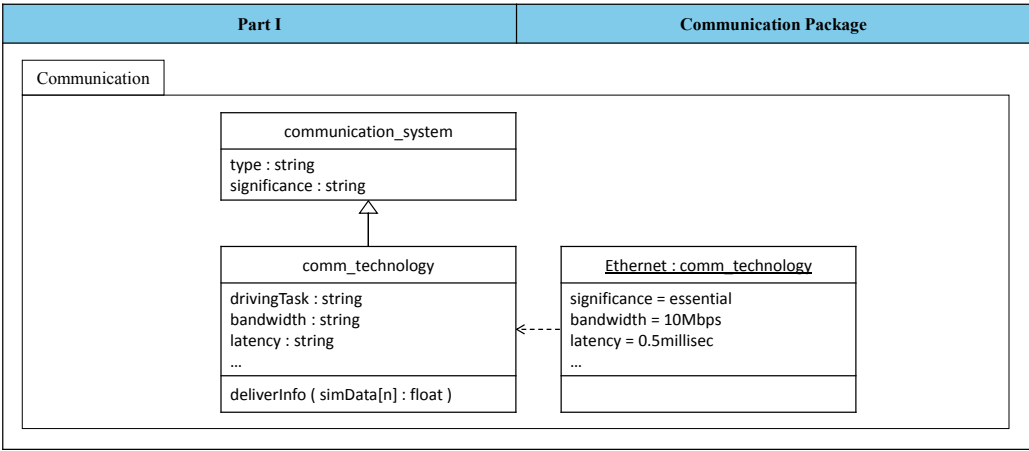



Figure 12. UML class diagram of the selected communication system components—package 2.

The communication package includes a main class named: communication_system. There is only one class that inherits the communication_system class: comm_technology. It has one instance representing the selected communication technology. Figure 13 shows the first section of the second part of the generated system model. This section contains the specification/requirement radar charts of the two selected driving simulators. The definitions of the depicted features and their fidelity levels are provided in Reference [6].

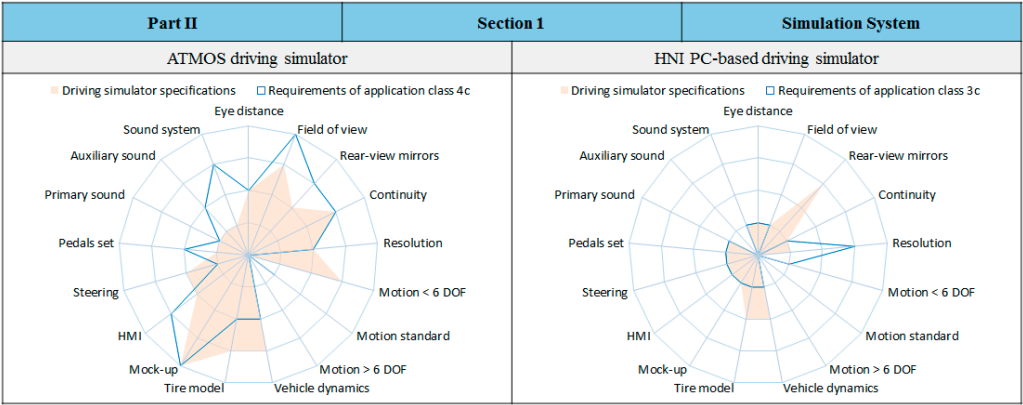



Figure 13. Specification/requirement radar charts of the selected system components.

Analogously, the second section of the second part contains the specification/requirement radar chart of the selected communication technology. Figure 14 shows the two networked driving simulators of the first example application scenario.

Figure 14. Established platform of networked driving simulation for the first application scenario.

The HNI PC-based driving simulator shown at the left side within Figure 14 has no motion platform. This driving simulator has a commercial wheel–transmission–pedals set and a simple driving seat. It utilizes a 60-inch high-definition screen. The HNI PC-based driving simulator is operated by a software package developed with MATLAB/Simulink. The ATMOS driving simulator shown at the right side within Figure 14 has a complex motion platform consisting of two dynamical components. The motion platform provides five Degrees Of Freedom (DOF) to fully simulate vehicle lateral and longitudinal accelerations. The ATMOS driving simulator has an eight-channel cylindrical projection system. It is powered by eight LCD-projectors to cover a horizontal field of view of 240 degrees. In addition, three small displays are used to simulate the side and rear mirror views. The ATMOS driving simulator is operated by a software package developed by the company dSPACE in Germany. The platform shown in Figure 14 adopts the mixed-fidelity concept [38]. That is, two

driving simulators of two different complexity grades and component fidelity levels are networked together to achieve a common goal.

5.2. Multi-Interactive Demonstration of an Autonomous Driving System

5.2.1. Scenario Definition

Demonstrating the capabilities of autonomous driving contributes significantly to raising the awareness about its benefits and attracting more customers [39]. Automotive manufacturers organize demonstration events to show the magnificent features and enable broader audience to be familiar with the new technologies. Demonstration with the help of drives on test roads delivers an impressive experience to potential customers. However, driving with other traffic participants is typically not permitted to date. Networked driving simulation can complement the demonstration purpose by adding an interactive factor to the simulated traffic environment. This delivers a comprehensive experience with the capabilities as well as the limitations of the autonomous driving system. The application scenario of this validation example is to interactively demonstrate different autonomous driving technologies.

5.2.2. Configuration Process

A simulation environment consisting mainly of two driving simulators is defined. One driving simulator is equipped with a simulation model for an autonomous driving system [25]. It is used by customers to interactively experience and test the system in a safe simulation environment. It is assumed that the customers are introduced theoretically to the autonomous driving system in advance. That is, they know about the features of the system and how to handle its settings. This purpose of use falls within the application class 4b that addresses rule-based responses and secondary driving tasks. The second driving simulator is used by a marketing representative, who has a simple navigation driving task. The aim is to drive interactively within the same virtual environment to subject the autonomous driving system to different traffic conditions—e.g., car following or emergency brake scenarios. The response of the marketing representative is not of concern as a matter of course. This purpose of use falls within the application classes 3a, 3b, and 3c that address different responses with navigation driving tasks. The application class 3c is chosen for convenience. Similar to the previous example application scenario, the determined application classes of the two participating driving simulators are used together as the first actual input to the SoS configuration software. Among the available driving simulators, the SoS configuration software suggests two particular driving simulators that best fulfill the requirements of application classes 4b and 3c.

In this application scenario, no traffic simulator is chosen so that customers are not overwhelmed or confused by the interaction with other programmed traffic participants at this system introductory level. No workstation is required for this platform as it used simply for quick demonstration purposes in exhibitions, where no control or monitoring tasks are necessary. Similarly, no database console is used as typically no analysis process is required after the demonstration sessions.

Only the position and orientation data of the simulated vehicles must be exchanged in this application scenario. No high priority is given to the bandwidth of the communication system. The simulation model of the autonomous driving system incorporates sub-models of various sensors [25]. Another sub-model takes decisions for rapid actions that influence the vehicle dynamics—e.g., acceleration, braking, and steering. Hence, deterministic data exchange between the driving simulators is essential for reliable system operation. A bandwidth of 1 Mbps (Megabit per second) is required initially based on a worst-case calculation of the number of exchanged data packets [30]. Other network characteristics are less relevant—e.g., secure data exchange, or length of transmission medium. Accordingly, the CAN bus technology is suggested by the SoS configuration software for this application scenario as a deterministic communication technology [40]. If the utilization of the CAN bus technology is expensive and relatively complex as it requires special network cards, the user can alternatively choose the FireWire (IEEE 1394). The FireWire (IEEE 1394) is a serial bus for

high-speed communication that can be utilized as a more feasible alternative [41]. The high performance provided by the IEEE-1394 connection reduces the likelihood of bandwidth saturation and data collisions. Although the IEEE-1394 standard is typically used to connect computers to peripherals—e.g., digital cameras and external hard drives—it can be used to carry network data as well. The utilization of FireWire with the Internet Protocol provides a very near deterministic data delivery [41]. In this case, the IEEE-1394 standard can be mapped to the lower three layers of the widely-known Open Systems Interconnection (OSI) network model: physical, link, and network layers. The traditional IEEE-1394 connection supports a bandwidth of 400 Mbps. Newer versions support potential bandwidth up to 3 Gbps. In addition to its high transfer rate, the IEEE-1394 connection supports isochronous data transfer, i.e., delivering data at a deterministic rate. This makes it suitable for applications that need to transfer large amounts of data under real-time requirements. The simulated network behavior using the ANDi software confirmed the eligibility of the FireWire (IEEE 1394) as an alternative selection. Similar to the previous validation example, no standard architecture for networked simulation is used in this application scenario as networked simulation management functions are not required.

5.2.3. Generated System Model

A system model is created by the SoS configuration software based on the scenario analysis and the configuration process. Figure 15 shows a simplified version of the simulation package of the UML class diagram containing the selected simulation system components.

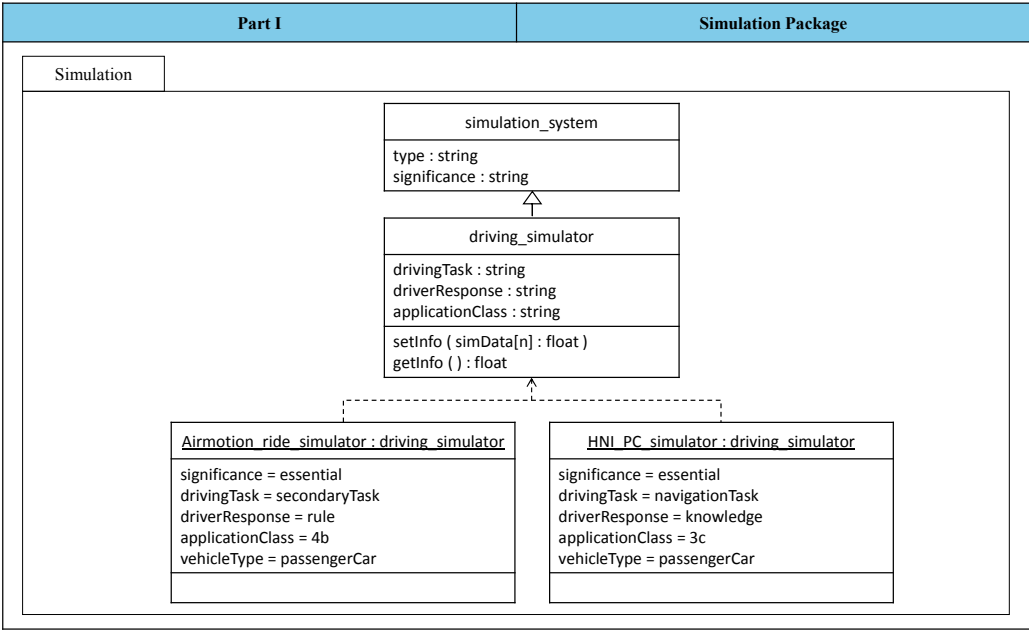



Figure 15. UML class diagram of the selected simulation system components—package 1.

The simulation package includes a main class named simulation_system. There is only one class that inherits the simulation_system class: driving_simulator. The driving_simulator class has two instances representing the two selected driving simulators: Airmotion_ride_simulator and HNI_PC_simulator. Figure 176 shows a simplified version of the communication package of the UML class diagram containing the selected communication system components.

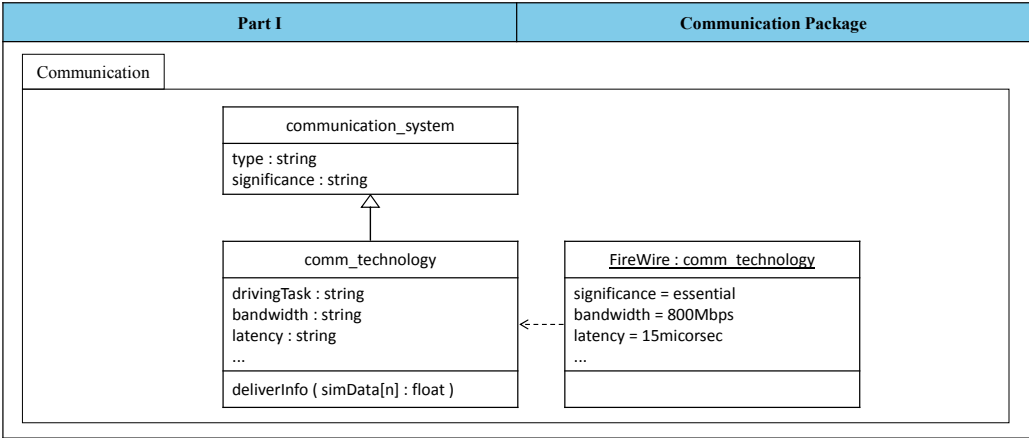



Figure 16. UML class diagram of the selected communication system components—package 2.

The communication package includes a main class named: communication_system. There is only one class that inherits the communication_system class: comm_technology. It has one instance representing the selected communication technology. Figure 17 shows the first section of the second part of the generated system model. This section contains the specification/requirement radar charts of the two selected driving simulators. The definitions of the depicted features and their fidelity levels are provided in Reference [6].

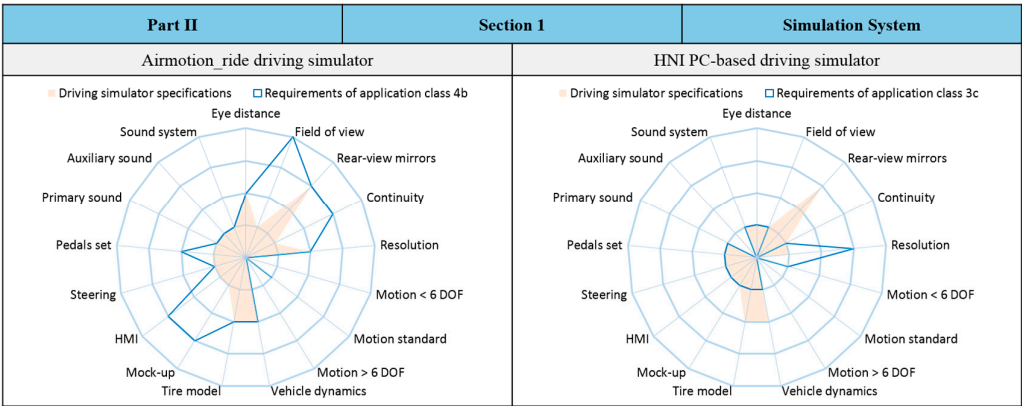



Figure 17. Specification/requirement radar charts of the selected system components.

Analogously, the second section of the second part contains the specification/requirement radar chart of the selected communication technology. Figure 18 shows the two networked driving simulators of the second example application scenario.

The HNI PC-based driving simulator is utilized as shown at the left side within Figure 18. It is the same simple driving simulator variant of the previous example application scenario. The Airmotion_ride driving simulator shown at the right side within Figure 18 has a pneumatic motion platform consisting of an inverted hexapod system that provides six DOF. A simple motion controller regulates the extension and contraction of six pneumatic elastic tubes based on the position and orientation of the simulated vehicle. Similar to the HNI PC-based driving simulator, the Airmotion_ride driving simulator has a 60-inch high-definition screen and a low-cost commercial wheel-transmission-pedals set. The Airmotion_ride driving simulator is operated by a software package developed with MATLAB/Simulink by the Heinz Nixdorf Institute. Similar to the platform of the first application scenario, the platform shown in Figure 18 follows the mixed-fidelity concept [38]. Specifically, two driving simulators of two different complexity grades and component fidelity levels share the same virtual environment to achieve a common goal.

Figure 18. Established platform of networked driving simulation for the second application scenario.

5.3. Multi-Interactive Analysis of Advanced Traffic Systems

5.3.1. Scenario Definition

Promising applications are emerging with the utilization of Vehicle-to-Infrastructure (V2I) and Vehicle-to-Vehicle (V2V) communication technologies. The common target of these technologies is to improve traffic efficiency while reducing the probability of collisions [42]. Yet analyzing different strategies is important to compare their efficiency and benefits, as well as to detect possible shortcomings early. Microscopic and macroscopic traffic modeling and simulation are effective tools to study the causes of traffic problems in general and to evaluate the solutions of potential traffic strategies in particular [43].

However, the human drivers still represent an important factor. They may be assisted by various connected systems that offer different information and service levels [44]. Drivers may take different decisions according to the received information—e.g., changing the route. Adding human drivers to the simulation environment helps to conduct analysis in a multi-interactive traffic environment, and hence, to deliver more substantial results. The application scenario of this validation example is to analyze different traffic strategies while taking the human driver factor into consideration.

5.3.2. Configuration Process

A simulation environment consisting mainly of two driving simulators is defined. Both driving simulators are used by test persons, who are familiar with the simulated road network and the features of the addressed connected vehicle technology. The test persons have to plan the route and drive from an origin to a target location. They can change the planned route based on the received information about the current traffic situation. This purpose of use falls within the application class 3b that addresses rule-based responses and navigation driving tasks. Similar to the previous example application scenarios, the determined application classes of the two participating driving simulators are used together as the first actual input to the SoS configuration software. Among the available driving simulators, the SoS configuration software suggests a particular driving simulator that best fulfills the requirements of application class 3b.

In this application scenario, a traffic simulator is chosen in accordance with the requirements of the application class 3b. This traffic simulator allows for changing the traffic density and defining the behavior of individual traffic participants. More information about this particular traffic simulator is presented in Reference [0]. A workstation is required for this application scenario to perform monitoring and control operations. Moreover, a database console is necessary to capture, save, and replay the simulation data for analysis and after-action reviews.

In this application scenario, a considerable amount of data must be exchanged through the communication system. Not only position and orientation data of simulator vehicles are exchanged, but also those of each programmed traffic participant. Moreover, data messages of the addressed connected vehicle technology are exchanged between the simulator vehicles. Hence, this application scenario is concerned particularly with the bandwidth for data exchange. The other characteristics of the communication technology have lower priority levels, such as, real-time data delivery or data loss rate. A bandwidth of 1 Gbps (Gigabit per second) is required initially based on a first worst-case calculation of the number of exchanged data packets [30]. Accordingly, a 1 Gbps Ethernet with User Datagram Protocol is suggested by the SoS configuration software for this application scenario. The simulated network behavior using the ANDi software confirmed the eligibility of this selection. More driving simulators may be added to the system to increase the complexity of traffic scenarios. In some scenarios, one of the driving simulators may be used to represent a special-purpose vehicle—e.g., an ambulance or an emergency vehicle. This special-purpose simulated vehicle may have different requirements regarding the type of exchanged data. It may be necessary to declare the generated and required data separately for each traffic participant. Therefore, a standard architecture for networked simulation is required unlike the previous validation examples. HLA standard is selected for this application scenario, especially, due to the provided declaration management function [46].

5.3.3. Generated System Model

A system model is generated by the SoS configuration software based on the scenario analysis and the configuration process. Figure 19 shows a simplified version of the simulation package of the UML class diagram containing the selected simulation system components.

The simulation package includes a main class named simulation_system. Four classes inherit the simulation_system class: driving_simulator, traffic_simulator, workstation, and database_console. The driving_simulator class has two instances representing the two selected driving simulators: HNI_PC_simulator1 and HNI_PC_simulator2. The traffic_simulator class has one instance representing the selected traffic simulator: HNI_traffic. Similarly, the workstation and the database_console classes have instances representing the respective selected solution elements. Figure 20 shows a simplified version of the communication package of the UML class diagram containing the selected communication system components.

The communication package includes a main class named: communication_system. Two classes inherit the communication_system class: comm_technology and comm_architecture. Each of these classes has an instance representing the selected solution element. Figure 21 shows the first section of the second part of the generated system model. This section contains the specification/requirement radar charts of the two selected driving simulators. The definitions of the depicted features and their fidelity levels are provided in Reference [6].

Analogously, the second section of the second part contains the specification/requirement radar charts of the selected communication technology and communication architecture. Figure 22 shows the two networked driving simulators of the third example application scenario.

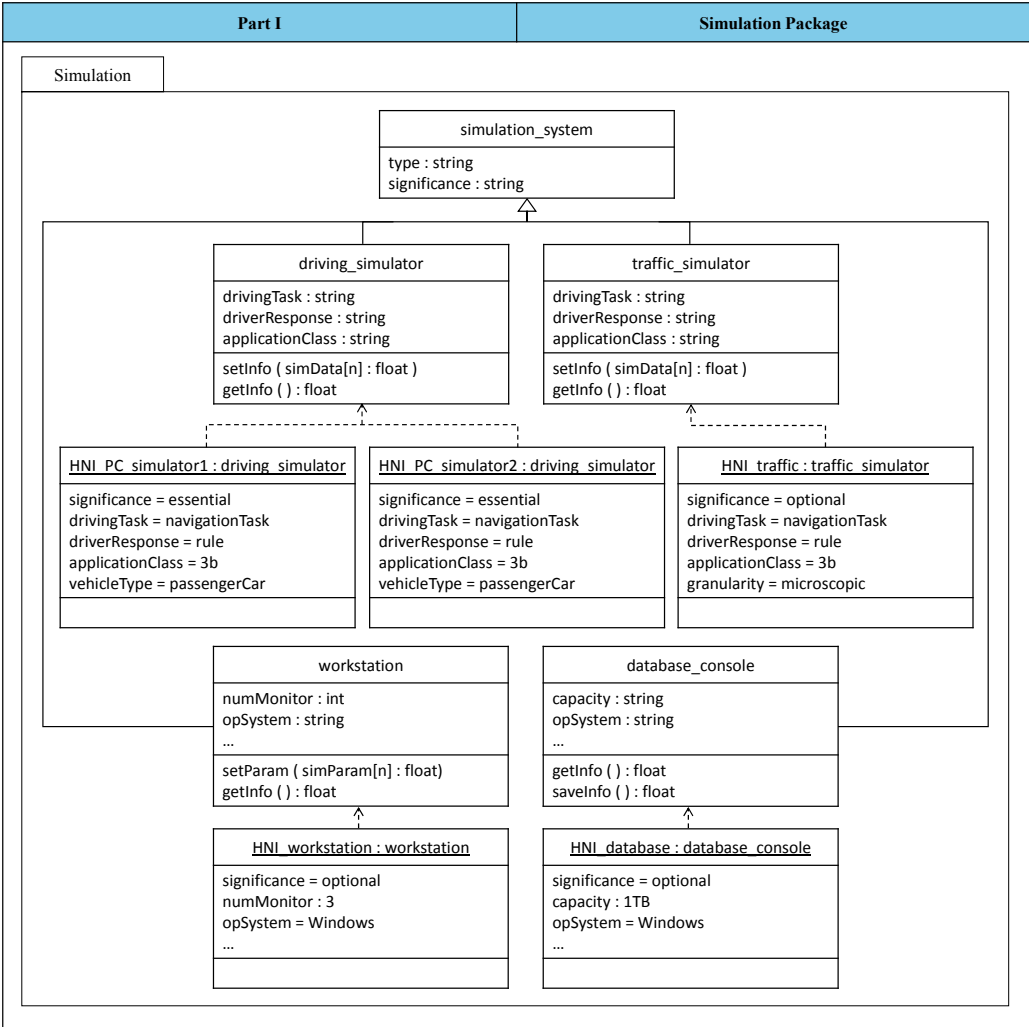



Figure 19. UML class diagram of the selected simulation system components—package 1.

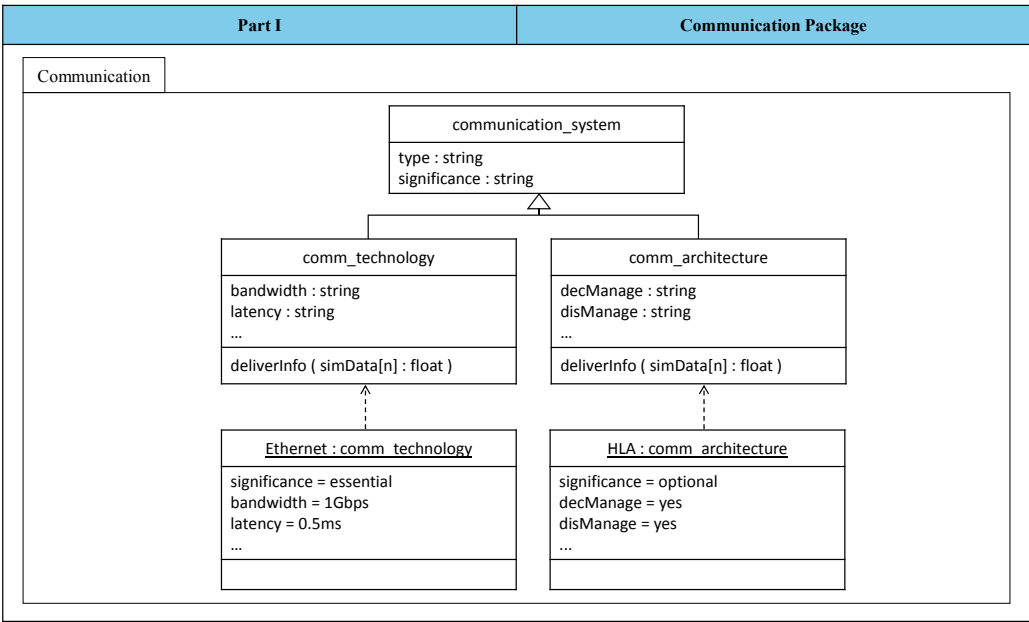



Figure 20. UML class diagram of the selected communication system components—package 2.

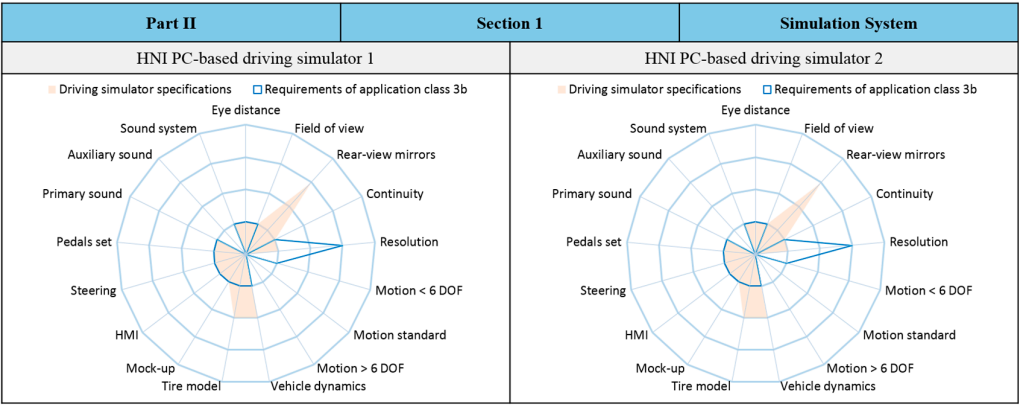



Figure 21. Specification/requirement radar charts of the selected simulation system components.

Figure 22. Established platform of networked driving simulation for the third application scenario.

The platform shown in Figure 22 utilizes two variants of the same driving simulator: HNI PC-based driving simulator. In this application scenario, the HNI PC-based driving simulator has a small screen and a normal seat in contrast to the configuration used within the previous two example application scenarios. Consequently, the shown platform does not impose special space requirements in comparison to the platforms of the previous validation examples. The platform shown in Figure 22 does not impose special space requirements in comparison to the platforms of the previous validation examples. It has been demonstrated successfully during the FMB 2016 exhibition (a German acronym that stands for "Forum of Mechanical Engineering") in Bad Salzuflen in Germany. The exhibition visitors got insight into the developed platform of networked driving simulation and its intended application scenario. The following section outlines the conclusions of the presented work, emphasizes the novelty of the developed method, and reveals the future work.

6. Conclusions and Future Work

This work presented a new method for the systematic design of networked driving simulation systems. The design method consists mainly of a procedure model accompanied by a configuration software. With its concrete phases, the procedure model analyzes the system thoroughly and addresses all the necessary tasks for the system modeling process. The design process is embedded in the configuration software to aid non-expert system users while selecting system components in accordance with the requirements of the concerned application scenarios. In particular, the design method considers the whole system of networked driving simulation in two main aspects: simulation and communication. The novelty of the developed method can be summarized in the following three concrete aspects:

- Combining and using two distinguished approaches from the literature for the selection of the simulation components of the networked driving simulation system [6,7].
- Utilizing a well-established decision-making method for the selection of the communication components of the networked driving simulation system [29].
- Creating system models with a structure that follows the principles of the agent-based modeling technique and uses the concepts of the UML class diagrams [32,35]. Thereby, the system model acts as simple communication basis for subsequent system realization.

Together, these unique aspects form the first methodological work for networked driving simulation to date. The presented validation examples emphasized the flexible usability of the developed method by designing three different system models. These system models make use of existing driving simulators in accordance with the requirements of the concerned application scenarios. The utilized driving simulators exhibit different complexity grades (mixed-fidelity levels). That is, they have different technical specifications and serve different purposes of use. However, new application scenarios were achieved by the integration in environments of networked driving simulation. Moreover, the utilized communication systems have different characteristics and capabilities. These were tailored to the data delivery requirements of the respective application scenarios. Three platforms of networked driving simulation have been built in accordance with the designed system models. With the help of the accompanying configuration software, non-expert users can apply the developed method to design further application-oriented system models for networked driving simulation.

As potential future work, further non-traditional application scenarios for driving simulation will be introduced and analyzed. The purpose of these non-traditional application scenarios is to keep up with advancements in the automotive field. The requirements of these application scenarios will be determined based on the demands of future autonomous and cooperative driving systems. These requirements will be used with the presented method and the accompanying SoS configuration software to create further system models for networked driving simulation. Users and developers will be able to use these ready system models to establish corresponding application-oriented platforms for networked driving simulation.

Acknowledgments: The presented work was carried out by the authors during the collaborative research and development activities at the Heinz Nixdorf Institute and the Fraunhofer Institute in Paderborn, Germany. No necessary special or external funding resources were acquired to produce or publish the research and development outcomes of this work. The authors acknowledge the valuable consultation and support of dSPACE GmbH, HELLA KGaA Hueck & Co., Varroc Lighting Systems GmbH, TU Dortmund University, Oregon State University, German Aerospace Center (DLR), Aerosoft GmbH, VDL Bus & Coach GmbH, UNITY AG, Fahrerakademie Paderborn, Isaak driving school, Hainer driving school, and Ringhoff driving school.

Author Contributions: Kareem Abdelgawad examined the related methods in the literature, developed the new method, designed the validation examples, and wrote the paper. Jürgen Gausemeier and Ansgar Trächtler were consulted regarding the call for action and directed the research topic. Sandra Gausemeier contributed to the analysis and evaluation of the related methods in the literature. Roman Dumiterscu and Jan Berssenbrügge contributed to design of the validation examples. Jörg Stöcklein and Michael Grafe built the example platforms for networked driving simulation. The latter seven authors examined the work and reviewed the paper.

Peer-reviewed version available at *Designs* **2017**, *1*, 6; doi:10.3390/designs1010006

Conflicts of Interest: All authors declare no conflict of interest.

References

1. Wall, M.; Gausemeier, J.; Peitz, C. Technology Push-based Product Planning—Future Markets for Emerging Technologies. *Int. J. Technol. Mark.* **2011**, *8*, 61–81, ISSN 1741-878X.
2. Maurer, M.; Gerdes, J.C.; Lenz, B.; Winner, H. *Autonomous Driving: Technical, Legal and Social Aspects*, 1st ed.; Springer: Heidelberg, Germany, 2016, ISBN 978-3-662-48845-4.
3. Arioui, H.; Nehaoua, L. *Driving Simulation*, 1st ed.; John Wiley & Sons: Hoboken, NJ, USA, 2013, ISBN 978-1-84821-467-5.
4. Winner, H.; Hakuli, S.; Lotz, F.; Singer, C. *Handbook of Driver Assistance Systems: Basic Information, Components and Systems for Active Safety and Comfort*, 1st ed.; Springer: Cham, Switzerland, 2015, ISBN 978-3-319-12351-6.
5. Abdelgawad, K.; Gausemeier, J.; Dumitrescu, R.; Grafe, M.; Stöcklein, J.; Berssenbrügge, J. Networked Driving Simulation: Applications, State of the Art, and Design Considerations. *Designs* **2017**, *1*, 4.
6. Negele, H.J. Anwendungsgerechte Konzipierung von Fahrsimulatoren für die Fahrzeugentwicklung. Ph.D. Thesis, Faculty of Mechanical Engineering, University of Munich, Munich, Germany, 2007.
7. Hassan, B. A Design Framework for Developing a Reconfigurable Driving Simulator. Ph.D. Thesis, Faculty of Mechanical Engineering, University of Paderborn, Paderborn, Germany, 2014.
8. DiMario, M.J. *System of Systems Collaborative Formation*, 1st ed.; World Scientific Publishing: Singapore, 2010; pp. 29–62, ISBN 978-981-4313-88-9.
9. Jamshidi, M. *System of Systems Engineering: Innovations for the Twenty-first Century*, 1st ed.; John Wiley & Sons: Hoboken, NJ, USA, 2008, ISBN 978-0-470-19590-1.
10. Johnson, M.A. From engineering to system engineering to system of systems engineering. In Proceedings of the IEEE World Automation Congress (WAC), Hawaii, HI, USA, 28 September–2 October 2008, ISBN 978-1-889335-38-4.
11. Keating, C.; Rogers, R.; Unal, R.; Dryer, D.; Sousa-Poza, A.; Safford, R.; Peterson, W.; Rabadi, G. System of systems engineering. *J. Eng. Manag.* **2010**, *15*, 36–45, doi:10.1080/10429247.2003.11415214.
12. Gausemeier, J.; Czaja, A.; Wiederkehr, O.; Dumitrescu, R.; Tschirner, C.; Steffen, D. Survey: Systems Engineering in Industrial Practice. In Proceedings of the Tag des Systems Engineering, Stuttgart, Germany, 6–8 November 2013.
13. Fisher, D.; Caird, J.; Rizzo, M.; Lee, J. *Handbook of Driving Simulation for Engineering, Medicine, and Psychology*, 1st ed.; CRC Press Taylor & Francis Group: Boca Raton, FL, USA, 2011, ISBN 978-1-4200-6100-0.
14. Porter, B.E. *Handbook of Traffic Psychology*, 1st ed.; Academic Press, Elsevier: Waltham, MA, USA, 2011, ISBN 978-0-12-381984-0.
15. Cacciabue, P.C. *Guide to Applying Human Factors Methods: Human Error and Accident Management in Safety-critical Systems*, 1st ed.; Springer: London, UK, 2004, ISBN 978-1-84996-898-0.
16. Walker, G.H.; Stanton, N.A.; Salmon, P.M. *Human Factors in Automotive Engineering and Technology*, 1st ed.; Taylor and Francis Group: Boca Raton, FL, USA, 2015, ISBN 978-1-4094-4757-3.
17. Pahl, G.; Beitz, W.; Feldhusen, J.; Grote, K.-H. *Engineering Design: A Systematic Approach*, 3rd ed.; Springer: Heidelberg, Germany, 2007, ISBN 978-1114243064.
18. Hassan, B.; Gausemeier, J.; Abdelgawad, K.; Berssenbrügge, J.; Grafe, M. Systematik für Die Entwicklung von Rekonfigurierbaren Fahrsimulatoren. In Proceedings of the 12th Workshop on Augmented & Virtual Reality in Product Development, Paderborn, Germany, 23–24 April 2015; HNI Publication Series; Volume 342, pp. 213–229, ISBN 2195-5239.
19. Gausemeier, J.; Frank, U.; Donoth, J.; Kahl, S. Specification technique for the description of self-optimizing mechatronic systems. *J. Res. Eng. Des.* **2009**, *20*, 201–223, ISSN 0934-9839.
20. Gausemeier, J.; Rammig, F.J.; Schäfer, W. *Design Methodology for Intelligent Technical Systems: Develop Intelligent Technical Systems of the Future*, 1st ed.; Springer: Heidelberg, Germany, 2014; pp. 117–171, ISBN 978-3-642-45434-9.
21. Wenguang, W.; Yongpinq, X.; Xin, C.; Qun, L.; Weiping, W. High level architecture evolved modular federation object model. *J. Syst. Eng. Electron.* **2009**, *20*, 625–635, e-ISSN 1004-4132.

22. Potuzak, T. Comparison of Road Traffic Network Division Based on Microscopic and Macroscopic Simulation. In Proceedings of the 13th International Conference on Computer Modelling and Simulation (UKSim), Cambridge, UK, 30 March–1 April 2011, ISBN 978-1-61284-705-4.

23. Xu, C.; Song, J.; Chen, M.; Chen, J.; Yu, L. Research on Adaptive State Update Strategy of Distributed Interactive Simulation. In Proceedings of the 3rd IEEE International Conference on Multimedia Information Networking and Security (MINES), Shanghai, China, 4–6 November 2011, ISBN 978-1-4577-1795-6.

24. Stephens, R. *Beginning Database Design Solutions*, 1st ed.; John Wiley & Sons: Hoboken, NJ, USA, 2008, ISBN 978-0-470-38549-4.

25. Abdelgawad, K.; Hassan, B.; Berssenbrügge, J.; Stöcklein, J.; Grafe, M. A Modular Architecture of an Interactive Simulation and Training Environment for Advanced Driver Assistance Systems. *Int. J. Adv. Softw. IARIA* **2015**, *8*, 247–261.

26. Mir, N.F. *Computer and Communication Networks*, 1st ed.; Prentice Hall: Upper Saddle River, NJ, USA, 2006, ISBN 978-0131389106.

27. Doganata, Y.N.; Tantawi, A.N. Analysis of communication requirements for intelligent transportation systems: methodology and examples. In Proceedings of the 45th IEEE Vehicular Technology Conference, Chicago, IL, USA, 25–28 July 1995, ISBN 0-7803-2742-X.

28. Triantaphyllou, E. *Multi-criteria Decision Making Methods—A Comparative Study*, 1st ed.; Springer International Publishing: Cham, Switzerland, 2000; Volume 44, pp. 5–21, ISBN 978-1-4419-4838-0.

29. Brent, R.J. *Applied Cost-benefit Analysis*, 2nd ed.; Edward Elgar Publishing: Cheltenham, UK, 2006, ISBN 9781843768913.

30. Meinel, C.; Sack, H. *Internetworking: Technological Foundations and Applications*, 1st ed.; Springer: Berlin, Germany, 2013, ISBN 978-3-642-35391-8.

31. Wehrle, K.; Gunes, M.; Gross, J. *Modeling and Tools for Network Simulation*, 1st ed.; Springer: Cham, Switzerland, 2010, ISBN 978-3-642-12330-6.

32. Achesona, P.; Daglia, C.; Kilicay-Ergin, N. Model Based Systems Engineering for System of Systems Using Agent-based Modeling. In Proceedings of the Conference on Systems Engineering Research (CSER'13), Atlanta, GA, USA, 19–22 March 2013; doi:10.1016/j.procs.2013.01.002.

33. Banos, A.; Lang, C.; Marilleau, N. *Agent-Based Spatial Simulation with NetLogo*, 1st ed.; ISTE Press, Elsevier: London, UK, 2015, ISBN 9781785480553.

34. Bersini, H. UML for ABM. *J. Artif. Soc. Soc. Simul.* **2012**, *15*, doi:10.18564/jasss.1897.

35. Rumpe, B. *Modeling with UML: Language, Concepts, Methods*, 1st ed.; Springer: Cham, Switzerland, 2016, ISBN 978-3-319-33932-0.

36. Sarbazi-Azad, H.; Zomaya, A.Y. *Data Distribution Management*, 1st ed.; Wiley-IEEE Press, Hoboken, NJ, USA, 2014, ISBN 9781118640708.

37. Rambhia, A.M. *XML Distributed Systems Design*, 1st ed.; Sams Publishing: Indian-apolis, IN, USA, 2002, ISBN 978-0672323287.

38. Cutler, M.; Walsh, T.J.; How, J.P. Reinforcement learning with multi-fidelity simulators. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014, ISBN 978-1-4799-3686-1.

39. Planing, P. Innovation Acceptance: The Case of Advanced Driver-Assistance Systems. Ph.D. Thesis, Faculty of Mechanical Engineering, Leeds Metropolitan University, Leeds, UK, 2007.

40. Voss, W. *A Comprehensible Guide to Controller Area Network*, 2nd ed.; Copperhill Media Corporation: Greenfield, MA, USA, 2015, ISBN 978-0976511601.

41. Anderson, D. *FireWire System Architecture: IEEE 1394A*, 2nd ed.; Addison-Wesley Professional: Boston, MA, USA, 1998, ISBN 978-0201485356.

42. Wuthishuwong, C.; Traechtler, A.; Bruns, T. Safe Trajectory Planning for Autonomous Intersection Management by Using Vehicle to Infrastructure Communication. *J. Wirel. Commun. Netw.* **2015**, *33*, doi:10.1186/s13638-015-0243-3.

43. Barcelo, J. *Fundamentals of Traffic Simulation*, 1st ed.; Springer Science & Business Media: New York, NY, USA, 2010, ISBN 978-1-4419-6141-9.

44. Stevens, A.; Brusque, C.; Krems, J. *Driver Adaptation to Information and Assistance Systems*, 1st ed.; Institution of Engineering and Technology: London, UK, 2013, ISBN 978-1-84919-639-0.

45. Abdelgawad, K.; Henning, S.; Biemelt, P.; Gausemeier, S.; Trächtler, A. Advanced traffic simulation framework for networked driving simulators. In Proceedings of the 8th IFAC Conference on Advances in Automotive Control (ACC), Norrköping, Sweden, 20–23 June 2016; Volume 49, pp. 101–108.

46. Abdelgawad, K.; Gausemeier, J.; Grafe, M.; Berssenbrügge, J. Interest Manager for Networked Driving Simulation Based on High-Level Architecture. *Designs* **2017**, *1*, 3.