Michael Mulle, BSc.

Risk Assessment of Advanced Driver Assistant Systems using a Simulation-Based Environment

Master's Thesis

to achieve the university degree of

Master of Science

Master's degree programme: Information and Computer Engineering

submitted to

Graz University of Technology

Supervisor

Dipl.-Ing. Dr.techn. Nikolaus Furian

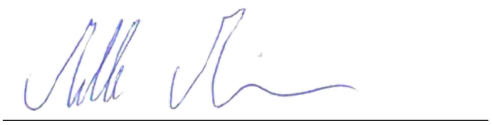
Institute of Mechanical Engineering and Business Informatics

Head: Univ.-Prof. Dipl.-Ing. Dr.techn. Siegfried Vössner

Gratwein Straßengel, October 2018

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz, 10.10.2018 

Date Signature

Eidesstattliche Erklärung[1]

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am 10.10.2018 

Datum Unterschrift

[1] Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008; Genehmigung des Senates am 1.12.2008

Abstract

One of the most significant current discussions in the technology sector is advanced driver-assistance systems and autonomous driving. Every day, there are new innovations in this field and almost every big company is doing a lot of research in this area. Recently, there has been renewed interest in functional safety in self driving cars to minimize possible accidents in future. Any accident resulting from such systems is too much, because it would effect a lower the acceptance of the population and further slow down the progress of this technology.

Due to this fact testing such components is highly important. It is impossible to develop components that work flawlessly on public roads and have thought of all sorts of situations. For this reason a tool is needed to detect critical situations in an urban environment. In order to obtain a common understanding of critical situations within vehicles, norms were published. In this thesis a theoretical background of related norms with main focus on ISO 26262 and suggested tools are given.

The main challenge to be able to find critical situations, is a suitable test environment on the one hand and suitable risk assessment methods on the other hand. This thesis develops a simulation environment, based on the procedure of the V-Model and the presented modelling process. It also shows how it is possible to integrate the behaviour of the components under test into this environment. With the help of hazard level methods, integrated in this simulation, the developed system is able to do a risk assessment of all situations in real time and find critical situations.

Finally, the results of the various hazard level functions are presented and compared. It is shown that this kind of simulation works quite well and can be used as an addition to classic tests. In order to completely replace classic tests, the application still has to be extended with features, discussed at the end of this thesis.

Contents

List of Figures

1 Introduction

Autonomous driving on public roads sounds like science fiction. Nevertheless, there is currently a real hype about this technology and it is not fitting to say that research on self driving cars is in its infancy. The first experiments were done in the 1920s. Recently, there are leverage's in this area around-the-clock. Autonomous driving is coming closer and closer and it is now important to check any systems for their reliability and safety. Introducing autonomous driving at once would be far too dangerous and simply impossible, because of the extensive research work. For this reason, for some years several car manufacturers have started to release their assistance systems step by step for road traffic. Through this gradual approach, systems can be tested in the real environment. By analysing the functionality of this components, potential developmental errors are revealed and, more importantly, cases which were not considered are often found. In this thesis an approach to find critical cases like this in a more safety and more cost-effective way is introduced. Next, a more detailed account of the current development status is given and it shows how this this thesis was proceeded.

1.1 Background and Motivation

As mentioned above, advanced driver-assistance systems (ADAS) and autonomous driving(AD) is one of the most significant current discussions in the technology sector. There are many big players doing research on this topic, such as Google with Waymo, Uber and Tesla to mention only a few. At least since the start of self-driving cars on the streets of California, everyone talks about the topic, among technicians, as well as politicians right to the normal population.

ADAS may be defined as a systematic development process, which consists of six different stages: Level 0 means that the vehicle has no automation, stage 1 is controlled by the driver supported with

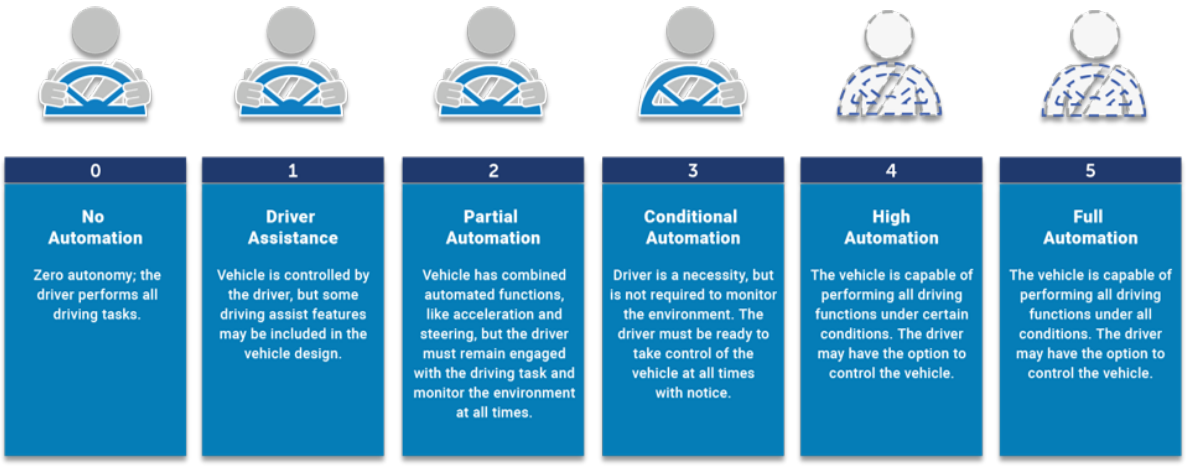


Figure 1.1: Standardized classes of autonomy [1].

some driver assist features. Also level 2 is steered by a human, even though there are automated components like lane keep systems. Meanwhile, almost every manufacturer has cars with level 2 functionalities in his assortment. Big players, like Uber or Tesla have cars in level 3, so called conditional automation. At this level the driver is not required any more, just ready to take control in critical situations at all times. The last stage before full automation is called high automation. The vehicle does not need the driver any more, but can be controlled manually if desired. At the end, level 5 reaches the full automation and the vehicle is able to perform all driving functions.
In October 2016, SAE International published the standard SAE J3016 [23] to officially define this different classes of autonomy, see 1.1.

All over the world work is being carried out on the further development. New innovations in this area will greatly affect the number of accidents. Statistics show that the main factor of the road accidents are human errors, like distraction, speed, or risk awareness [16]. Humans are responsible for approximately 90 percent of road accidents, or at least jointly responsible. Other main accident causes are environment factors, such as road designs, weather, etc. with a quote of 30 percent. In Addition to these factors there are about 10 percent of vehicle accident causes, see figure 1.2. A report from Jerry Albright calculates for 2040 a decrease in accidents in urban environments of 80%. The main reason for this prediction is the higher level of automation on the streets. Advanced driver assistant systems, which are providing automated safety features, are build to minimize the human

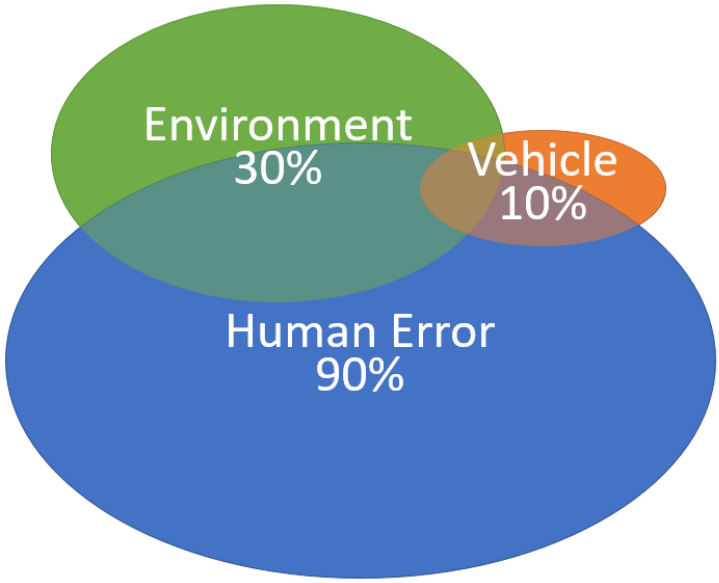


Figure 1.2: Main accident causes in Europe.

error. Additional, adaptive features like lane centring, cruise control, automate lighting etc. enable a human error close to zero.

The acceptance of the population towards self driving cars depends on the comfort on one side and the safety on the other side. For this reason, the most important part in the development process of such systems is safety. To ensure a consistent understanding of the term security worldwide, earlier the standard IEC 61508 was used to cover the electric components in vehicles. Due to the increasing number of assistant components in the last years, it became clear that this general norm for functional safety is no longer sufficient. The ISO 26262 was published in 2011, which covers the functional safety of electronic and electrical components in vehicles. The standard is important in the development process of ADAS, and plays a key role in this thesis.

Based on this standard, a V-Model has been developed to ensure that all requirements of functional safety are met. There are different safety requirements for the development of components. If a malfunction of the component is not dangerous, such as an air conditioning system, the safety requirements are accordingly low. In the contrast, when the component the functional safety influences very much, the safety requirements in the development process will accordingly high. ISO 26262 provides the Automotive Safety Integrity Level (ASIL) to calculate this risk level of the system. Currently, this ASIL is manually determined by decision matrix. This type of calculation

is therefore dependent on human perception and respectively on the qualification and experience of the deciding person. There are only a few studies that deal with a systematic calculation of an ASIL. This paper attempts to show that an scientific way to detect critical malfunctions components, whether conscious or not.
With the help of the developed risk assessment based on a simulation environment it is possible to find critical situations. Conscious malfunctions can simulated in a simulated environment and their effect can analysed using the assessed hazard level. If the impact of the malfunction is very negative on the hazard level, the ASIL must be badly valued and the development of the component must meet very high security requirements.
In order to have almost error-free ADAS systems, they need to be tested sufficiently. The method, presented in this thesis, can not replace traditional testing and it is advisable to use risk assessment in addition to classic testing. This approach can be used, at the end of the development chain, to find mistakes in the development or to find unconscious situations.

This section has introduced the causes of advanced driver assistant systems and has argued that functional safety is an important factor for acceptance in the population. The next part of this paper will represent the main objectives of this thesis and how it is possible to develop risk assessment based on a simulation environment.

1.2 Research Scope

1.2.1 Role of ISO 26262

As was pointed out in the introduction to this paper, functional safety is quite important for acceptance of self driving cars. To reach a high level of safety it's important to have knowledge about the used standards in the field of autonomous vehicles. Due to this fact, the thesis will give a short overview of the most important standard. The emerging role of ISO 26262 in the context of functional safety of advanced driver assistant systems will covered. Additionally, similar standards and safety methods are listed and contrasted to each other. In this context, the ASIL is discussed in more detail to get an basic understanding of the requirements of the V-Model provided by the ISO 26262.

1.2.2 Development using the V-Model

Due to the importance of the V-Model, the derived model is considered in more detail. At each stage the importance of safety should be shown and how the process changes according to ASIL. Additionally, with the help of the standard and the V-Model the need of testing should be demonstrated. In order to have almost error-free ADAS systems, they need to be tested sufficiently.
For this reason, there has been renewed interest in test areas for autonomous driving vehicles to minimize possible accidents in the future and get more knowledge of various scenarios. However, the costs of testing with real environment is enormous and of course always associated with a certain risk. Due to this fact simulation of self driving cars in an virtual urban environment becomes more and more important.

1.2.3 Importance of Modelling and Simulation

This thesis should explain the term simulation in more detail and demonstrate the different kinds of models. It would go beyond the scope of this thesis, only a short overview of the modelling process will provided. It goes into a bit more detail on how to evaluate an already existing system and possibly adapt it to your own needs. This background knowledge shows how a simulation environment was selected. In this context, different simulators are checked for their characteristics and compared. The last two decades have seen a growing trend towards developing realistic simulation environment to readjust robust sensors like radar, lidar or camera and test as close to reality as possible. Recently, a considerable literature has grown up around the theme of gaming engine as a simulation environment, because of their realistic scenario building possibilities. More details will be given to Carla simulator, as it is best suited for this task. It is pointed to the benefits and how was built on this simulator.

1.2.4 Support the development of ADAS

The main aim of this study is to develop a risk assessment software based on a simulation environment. This risk assessment should be able to identify the lack of identified test scenarios. The main research question is: How it is possible to detect critical situations in an urban environment?

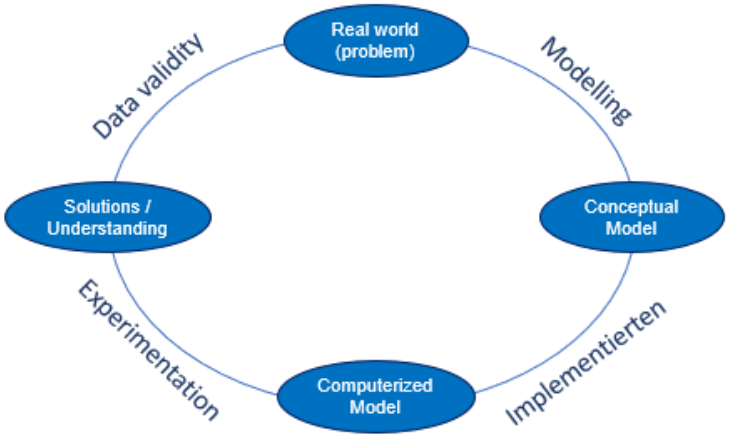


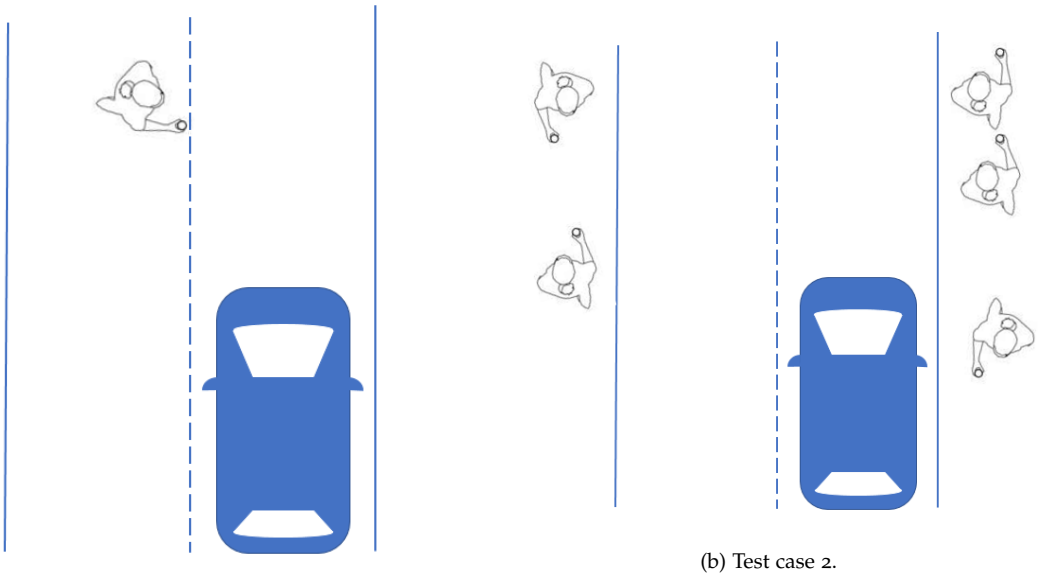
Figure 1.3: Role of simulation paradigm.

With a automatic tool to find critical situations it should be possible to find situations which have not been considered before. These cases would greatly advance the development of self driving cars. It is shown that the determination of such a situation is not easy. Defining a collision with a critical situation would be too easy and not enough. Thus, various parameters are introduced to treat a situation as critical even before a collision will happen.

In the course of the thesis different methods for the identification of such a situation are worked out. Two important cases should be evaluated and discussed for each method. The first test case is a street crossing pedestrian without other risk relevant influences, see 1.4a. In figure 1.4b the second situation is shown. This case provides a crowd of people going along a sidewalk without crossing the street. This situation is not critical and should also not be detected by the hazard level functions.

1.2.5 Improve the way of testing

Additionally, this study aims to contribute to improve the way of testing. The V-Model demonstrates the importance of testing. The considerations of risk assessment should not replace classical testing, this software should serve as support to them. The ISO 26262 expects a lot of testing, which is currently partially met by real-time testing in urban environment. Since this type of testing is very



(a) Test case 1.

(b) Test case 2.

Figure 1.4: Detection of critical situation in test case.

expensive, the new approach based on simulation should serve many resources as a preliminary test and show many problems in advance.

1.3 Outline

This thesis begins by an introduction to give some motivation why this thesis is so important for the development of safe self-driving cars and why functional safety is important for the acceptance of the population. It will then go into more detail in the functional safety standards. This will introduce the necessary background knowledge including the general norm IEC 61508 and then more details to the functional safety norm ISO 26262. In Addition to the norms the Automotive Safety Integrity Level (ASIL) will provided and how it will determined.
Chapter 3 explains the theoretical dimensions of the research, and reveals how ISO 26262 deviate the requirements within the V-Model. The structure of the V-Model will discussed in more detail and the requirements of the different phases depending on the ASIL will provided. Next the methods of risk assessment, including the used parameters and algorithms for computing the hazard level, will specified. Additionally to this, the Modelling Process will explained in more detail. A quick overview of the different validation techniques and different model and implementation types will

be shown.

The fourth section presents the findings of the research by using the research methods in a case study. Therefore, the system design and requirements will specified by using the software requirement specification proposed by the V-Model. Next, a simulation environment will be selected and the chosen simulator will discussed in more detail. A quick overview of those features and the possibility to adjust the simulation environment at own requirements will provided. In the course of this a risk assessment based on the selected simulator will be developed.

Chapter Five analyses the results of the Case Study. The different hazard level function will be compared. Additionally it will show the correlation between the findings and the automotive safety integrity level.

Finally, the conclusion will discuss the results and findings of the thesis and will give a review to challenges for further work.

2 Functional Safety Standards

In the introduction, we have explained why paying attention to functional safety in driver assistance systems is so important. ADAS can play an important role in the prevention of traffic accidents. Systems, like Collision Avoidance Systems or Lane Departure Warning Systems, provide a warning signal for the driver to minimize human error. More complex assistance systems, such Lane Keep Assistant or Autonomous Cruise Control, control the vehicle continuously. This achieves the level of autonomous driving without the need of a human driver, which will be the safest way of driving [16]. With the growing amount of features, it is highly important to detect functional failures in an early development step to minimize fatal accidents and costs. This is the purpose of the international standard IEC 61508, entitled "Functional safety of Electrical/Electronic/Programmable Electronic Safety-related Systems", which was published by the International Electrotechnical Commission in 1998.

2.1 IEC 61508

Therefore IEC 61508 [14], tries to define procedures to produce products, which meet the current state of the art related to safety conditions. It describes the most important key points of the complete safety life cycle. The design stage is taken into account, as well as the market launch and usage until the disposal of the product.
IEC 61508 covers all safety-related systems including one or more electrical or electronic or programmable electronic devices. Some classic examples are emergency shut-down systems, railway signalling systems, information-based decision support tool and many more. It takes care about possible hazards and possible effects in case of malfunction and prepares various safety precautions based on risk assessment. The higher the risk level of the product is rated, the more demands are

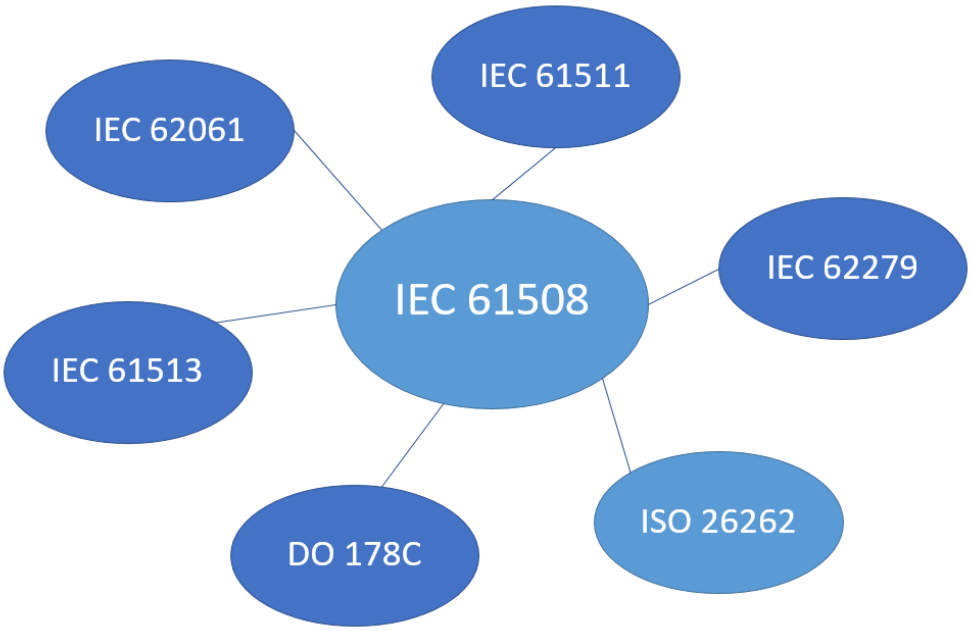


Figure 2.1: Most important derived adaptations of IEC 61508.

within the development process.
The standard finds usage in liability cases. In serious cases, the manufacturer is at least partly responsible, even if the error is in a part produced by a sub company. To avoid this liability, the manufacturer must be able to demonstrate reliable product development with the help of the established risk assessment.
For this reason, many companies require a certificate from their component suppliers, which proves that their products are valid for IEC 61508. However, it is not mandatory, it's only a recommendation.

Due to the fact that IEC 61508 is a general and no harmonized standard, its fulfilment is not enough. However, it can be taken as a basis for more specific standards, see figure 2.1. The following list contains the most important derived adaptations:

- IEC 61511: Functional safety - Safety instrumented systems for the process industry sector
- IEC 62279: Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems
- ISO 26262: Road vehicles – Functional safety

- IEC 61513: Nuclear power plants - Instrumentation and control important to safety - General requirements for systems
- DO 178C: Software Considerations in Airborne Systems and Equipment Certification
- IEC 62061: Safety of machinery: Functional safety of electrical, electronic and programmable electronic control systems

In the following pages, I will present the adaptation ISO 26262, entitled Road vehicles – Functional safety, which is an important norm for advanced driver assistance systems.

2.2 ISO 26262

As discussed above, there is a highly increasing number of electronic components in vehicles. As advanced driver assistance systems become more and more complex, the risk of a malfunction increases. To guarantee, that the functional safety is up to date, the standard ISO 26262 was introduced by the International Organization of Standardization in 2011 [15]. This standard is mainly used by the automotive industry and its suppliers to ensure a high safety standard. For legal security an external audit office can be engaged to test the product according to ISO 26262.

ISO 25252 suggests a product life cycle including management, development, production, operation, service and decommissioning phase, see figure 2.2. Therefore it covers the earliest concept steps up to the disposal of the product. The main focus of this thesis is the development phase, because in this phase the software design, development and testing is done.

This product life cycle is also reflected in the structure of the standard, which consists of 9 parts and the guideline. Part one explains the terms and abbreviations used in the following pages of the document.
The second part, Management of function safety, gives a overview of the required safety management during the concept phase right up to the decommissioning of the product. The amount of management, which is required to fulfil the claims, is defined by the automotive safety integrity level (ASIL) of the embedded product.
The concept phase is responsible for hazard analysis and risk assessment. Firstly the related hazards have to be analysed. With the help of a specified classification concept, the ASIL can be defined. To

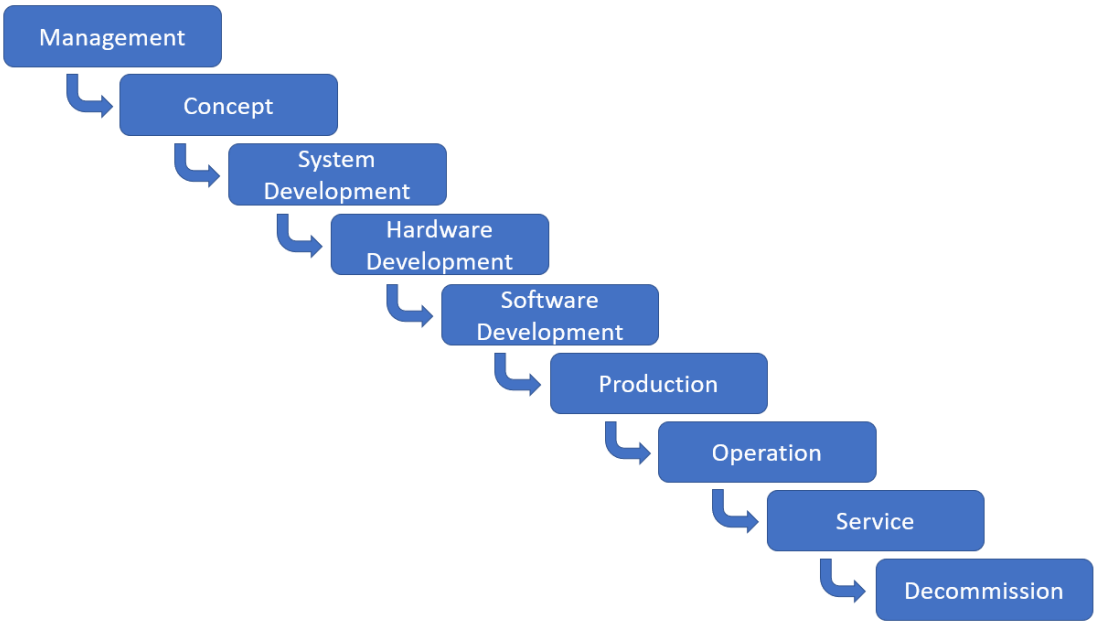


Figure 2.2: Product life cycle defined by ISO 26262.

do this, every identified hazard is analysed on severity, exposure and controllability of the situation. A given table helps to assign the hazard situation to the relevant level, more details in section 2.3. Parts four to six cover the development phase of the product life cycle and is divided by system, hardware and software product development. All these parts contain working steps similar to the V-Model. There are methods listed, which, depending on the ASIL, must be committed, recommended or implemented as an option. In section 3.1 the V-Model used in ISO 26262 is described in more detail. Part seven is responsible for the production and operation phases and declares demands to the safety conditions in these steps.
The next part, called supporting processes, contains interfaces within distributed developments, specification and management of safety requirements, configuration and change management, verification, documentation and assessments tools for software and hardware components. Almost arrived at the end, the rules of ASIL decomposition and criticality analysis are given.
Last but not least, there is a guideline on ISO 26262, how to use the standard and some examples are listed.

In the following pages, Part 6, Product development at the software level will be specified in more detail. ISO 26262-6 consists of 7 clauses, starting with number 5 and ending with 11. More details

about these clauses are listed bellow.

Clause 5: Initiation of software development investigates the type and tools used within the development process of the software. In this phase the development plan will be created with decisions like using model-based development or not, or which languages will be used.

Clause 6: Safety requirements specification identifies the hazard functions of the software. Dangers can happen directly by a desired cause, such as loss of traction during braking, or indirectly by malfunctions. For these hazards the safety requirements have to be specified, including connections within the components.

Clause 7: Architectural design declares a high level architecture for each software component. This phase again consists of 18 requirements. First of all the notations and the principle architectural design have to be done, like design considerations, identification of software units, component categorization, just to name a few requirements. The important part of this clause is the allocation of ASIL level and the safety analysis as well as defining safety mechanism and error handling. Finally it is important to do some verification of the architectural design.

Clause 8: Software unit design and implementation provide a list of requirements to design and implement each subsystem on it's own. At the end of this phase, there is a unit verification to guarantee safety.

Clause 9: Unit testing is responsible for testing the small components on a flawless function. ISO 26262 [15] lists following requirements: safety-related components, test planning and execution, unit testing methods, test cast generation and test coverage metrics.

Clause 10: Integration testing provides a requirement list for testing connected components. If no errors are found in unit testing, it is still possible to find errors with the help of integration testing, because of failures within interfaces and communication between the components. So the list of requirements is similar to unit testing, but one level above it.

Clause 11: Safety requirements verification obtain the correct functionality on the target environment. To do this, the first proposed requirement is to verify the planning and execution. After that test the environments and target hardware. Finally the evaluation of the results is very important.

Overall, this standard highlights the importance of funcional safety and lists some prepared tools and how to use it. However, risk assessment is not a one-time process, it has to be done more often in the development phase. In the next section the evaluation of the ASIL will be shown.

2.3 Automotive Safety Integrity Level

Automotive Safety Integrity Level is a type of risk assessment method and a key concept of ISO 26262 [15]. It is a risk based approach to classify the examined component in different levels. There are four standard ASILs. The lowest risk level is indicated by ASIL A and ASIL D indicates the highest one, with ASIL B and ASIL C in-between. If there is no risk associated, the assigned level is set to quality management (QM) with no relevant risk measure.

2.3.1 Hazard Analysis and Risk Assessment

This approach uses three identifiers called Severity (S), Exposure (E) and Controllability (C).

- **Severity** indicates the expected severity of injuries in the event of the malfunction and can be divided by S0 (No Injuries) up to S3 (Fatal Injuries).
- **Exposure** defines the probability of occurring an injury. The Classification starts with E0 (Incredibly unlikely) up to E4 (High probability).
- **Controllability** defines the probability that the driver can prevent the injury. It is divided into four levels starting with C0 (Easy to Control) up to C3 (Uncontrollable).

Once these identifiers are declared, the ASIL can be assigned with the help of the ISO 26262 ASIL allocation table, see 2.1.

Below is an example helps to get a basic understanding of the ASIL calculation.
First the malfunction must be defined, for example a failure with the ABS system. The next task is to analyse the possible hazards, like losing the stability. Now the identifiers Severity, Exposure and Controllability have to be defined. For example if S2, C2 and E3 are identified, the resulting ASIL is ASIL A.

Table 2.1: The ISO 26262 ASIL Allocation table (Source ISO 26262-3 [15])

		Exposure	Controllability		
			C1	C2	C3
Severity	S1	E1	QM	QM	QM
		E2	QM	QM	QM
		E3	QM	QM	A
		E4	QM	A	B
	S2	E1	QM	QM	QM
		E2	QM	QM	A
		E3	QM	A	B
		E4	A	B	C
	S3	E1	QM	QM	A
		E2	QM	A	B
		E3	A	B	C
		E4	B	C	D

2.3.2 Comparison with other Hazard Level Standards

The way other hazard level standards assess risk is comparable to the ASIL way of ISO 26262. There is a quick comparison in table 2.2. What stands out in the table is that IEC 61508 uses the Safety Integrity Level (SIL), a reliability valuation method, instead of ASIL. It provides a acceptable number of failures per time. First IEC 61508 is divided in three different modes, low demand, high demand, or continuous mode. With the help of these modes and the specified limits of probability failures it is possible to assign a SIL between 1 and 4. For example a subsystem in a low demand mode and an assigned SIL 3 is required to have a probability failure limit smaller than 10^{-4}. In summery, ISO 26262 evaluates the level using 3 parameters, severity, exposure and controllability and IEC 61508 only use one value, the probability of failure on demand.
The Design Assurance Levels (DAL) defined by ARP 4754 [22] is comparable to ASIL. DAL provides five levels from E to A, which is similar with QM to ASIL D. Components with rating QM or DAL-E do not need additional functional safety arrangements. Otherwise ASIL D and DAL-A require the highest attention in the area of functional safety.

Table 2.2: Comparison with similar standards.

	Safety Level				
General (IEC 61508)	-	SIL-1	SIL-2	SIL-3	SIL-4
Automotive (ISO 26262)	QM	ASIL A	ASIL B/C	ASIL C/D	-
Aviation (DO 178C)	DAL-E	DAL-D	DAL-C	DAL-B	DAL-A
Railway (IEC 62279)	-	SIL-1	SIL-2	SIL-3	SIL-4

2.3.3 Benefit of Automotive Safety Integrity Level

With the help of the ASIL it is possible to assign the required tools and methods, which should be used in the software development process. The higher the level the more requirements have to be met.

For each clause a table exists with tools classified by the ASIL and a specific value, named highly recommended, recommended and no recommendation, for each tool.

- **No Recommendation (o):** The technique is optional and can be used if required, but there is no recommendation.
- **Recommended (+):** The method should be used, but it is not obligatory.
- **Highly Recommended (++):** The method is mandatory and has be used.

3 Background Information

This chapter discusses the impact of ISO 26262, on the development of software components. Part 6, product development at the software level, provides approaches on how to develop functional safety within software components. The V-Model will be the main focus and it is explained in more detail in section 3.1. There will always be cross-references to ISO 26262 and the V-Model will be presented by this area of application.

Gradually it will become apparent that defining critical situations is very important to ISO 26262 and it's V-Model. Due to this fact section 3.2 will demonstrate different ways on how to define hazardous phases in traffic of an urban environment. It explains why the question, "What is a critical situation in an urban environment?", is not easy to answer.

The second objective of this thesis is to represent a software in a loop model. For this reason, modelling and simulation will be discussed in section 3.3. Different types of "in a Loop"-models are shown and their advantages and disadvantages are reviewed in more detail.

3.1 V-Model

The V-Model is a standard proceed model to organize the project to minimize development errors in all process steps. The main objectives can be listed as follows: Minimize the project risks, improve the guarantee and quality, minimize system life circle methods and improving the communication between the related components as well as the involved developers [20].

Like every method, the V-Model also has advantages and disadvantages [2]. The model is fits good for restricted projects. Due to the fact of the clear defined procedure, the model is very popular in projects with a clear deadline and defined milestones. It provides a simple and easy usage and

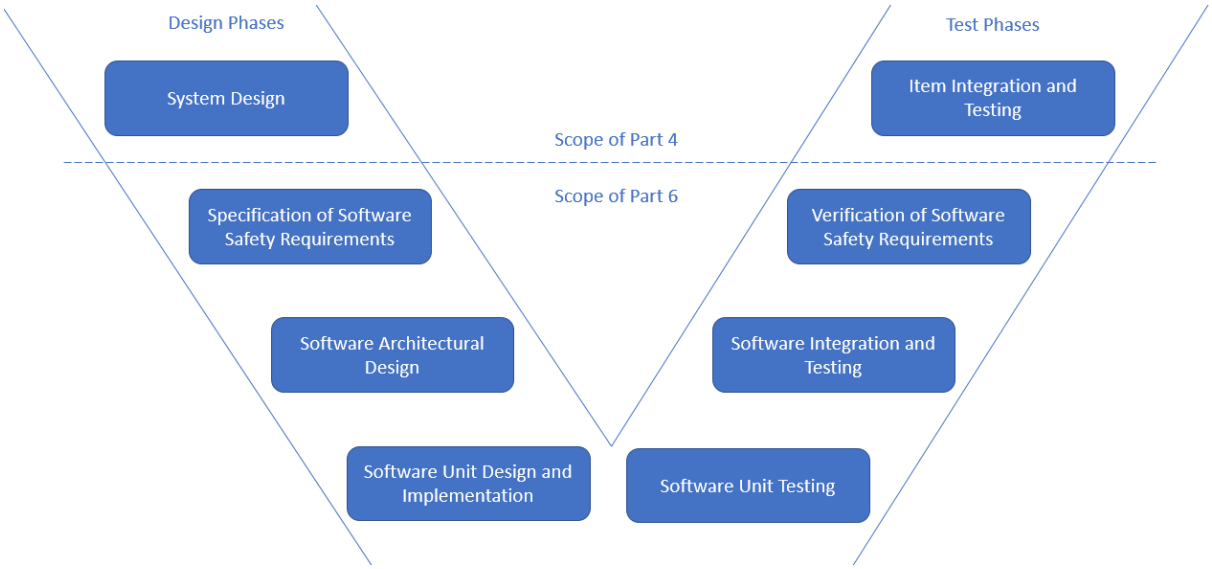


Figure 3.1: V-Model of software product development.

the planning activity as well as designing test cases happens well before implementation phase. For this reason it is ideal for time management projects. The active defect tracing detects possible malfunctions in a early stage and avoids a downward flow of the mistakes.
The most problematic disadvantage of the V-Model is the missing extendibility of software systems in a later development stage. Software design and requirements are defined in an early phase and can not be changed later. Additionally, there is no early prototype developed. The first software development is done in the implementation phase, which maybe is to late for indicating forgotten functions. The model works quite fine for small components, because of this reason it is recommended to divide a complex system in smaller units, like it is done in ADAS [9].

As stated above, the V-Model is used as the basic approach in software development of ISO 26262. In figure 3.1 the derived V-Model is presented. The figure is structured in a design phase on the left and a test phase on the right side of the V. The items within this model are nearly the same like the clauses in ISO 26262-6. As figure 3.1 shows, the four design steps on the left side, are system design, specification of software safety requirements, software architectural design and software unit design and implementation. The test phases consist of software unit testing, software integration and testing, verification of software safety requirements and item integration and testing. Each individual step will be discussed in more detail below.

3.1.1 System Design

The main goal of the system design is to create a general view to develop a clear structured and well defined software component. Firstly the client and contractor perform a requirement engineering to declare all functional and non functional requirements and create a product requirement document. Defining clear structured requirements is not easy for humans. Due to this fact Cimatti at al. defined a list of conditions [7]. Each requirement is classified to a specific categories using specific conditions. Table 3.1 represent the conditions and their categories. This classification helps to get a better overview of the domain and should clarify which tools are used to represent each of them.

Table 3.1: Conditions of the functional requirement defined by Cimatti et al. [7].

Category	Condition
Glossary requirement	Does the text fragment define a specific concept of the domain?
Architecture requirement	Does the requirement introduce some system's modules and describe how they interact?
State requirement	Does the requirement describe the steps a particular module performs or the states where a module might be in?
Communication requirement	Does the requirement describe messages modules exchange?
Property requirement	Does the requirement describe expected properties of the domain or constraints of the system-to-be?
User requirement	Does the requirement describe actions or constraints which have to be considered, satisfied or performed by the user?
Safety requirement	Does the requirement describe necessary safety constraints?
Annotation	Is the text fragment a note that does not add any information about the ontology or the behavior of the specified system?

Once this document has been created, an overview of the function should be given and an assignment by ASIL should be possible.

Table 3.2 shows the impact of the specified ASIL on the prescribed techniques. As can be seen from the table the usage of defensive implementation techniques is hgihly recommended on ASIL C & D, but there is no recommendation on ASIL A. As expected, there is no attribute for QM.

Table 3.2: Coding and modelling guideline by ISO 26262-6

	ASIL			
	A	B	C	D
Enforcement of low complexity	++	++	++	++
Use of language subsets	++	++	++	++
Enforcement of strong typing	++	++	++	++
Use of defensive implementation techniques	o	+	++	++
Use of established design principles	+	+	+	++
Use of unambiguous graphical representation	+	++	++	++
Use of style guides	+	++	++	++
Use of naming conventions	++	++	++	++

Now with the help of the suggested tools a concept including program structure, programming techniques and algorithms can be created.

3.1.2 Specification of Software Safety Requirements

The specification of Software Safety Requirements critically examines all functions which could lead to a hazardous situation. It specifies the functional safety requirements and extends the interfaces between software and an other component. In terms of the specifications, limits must be taken into account to avoid problems later.

One of the most well-known methods to find safety requirements is the iterative Hazard analysis and risk assessment process [29]. Figure 3.2 shows a quick overview of the method and how to find a safety goal.
The first step in this process is to define a preliminary feature description, which specifies the drivers benefit and known limitations. The next step is to perform the situation analysis with the help of a generic operational situation tree, where all possible situations will be divided in levels like a tree. After defining the tree a specific critical situation will defined and the hazard identification can be done to specify all possible reasons of possible failures. Once all possible hazards are specified, the dimensioning of hazard situations with the ASIL method can be done followed by the function

refinement process to break down hazardous events and summarize it's safety requirement goals. Due to the fact that the system design phase needs ASIL specification this phase overlaps with the design phase.

The advantage of defining trees is to have a structured knowledge. It is easy to expand with new findings and helps to set the focus on the important facts. The disadvantage of doing it manually is the coverage. It's difficult to find all possible hazardous situations manually and set a limit of possible situations. To date, only a limited number of methods to find critical situations are available. Therefore, this thesis discusses how to find critical situations automatically and helps to fill the tree without a knowledge gap.

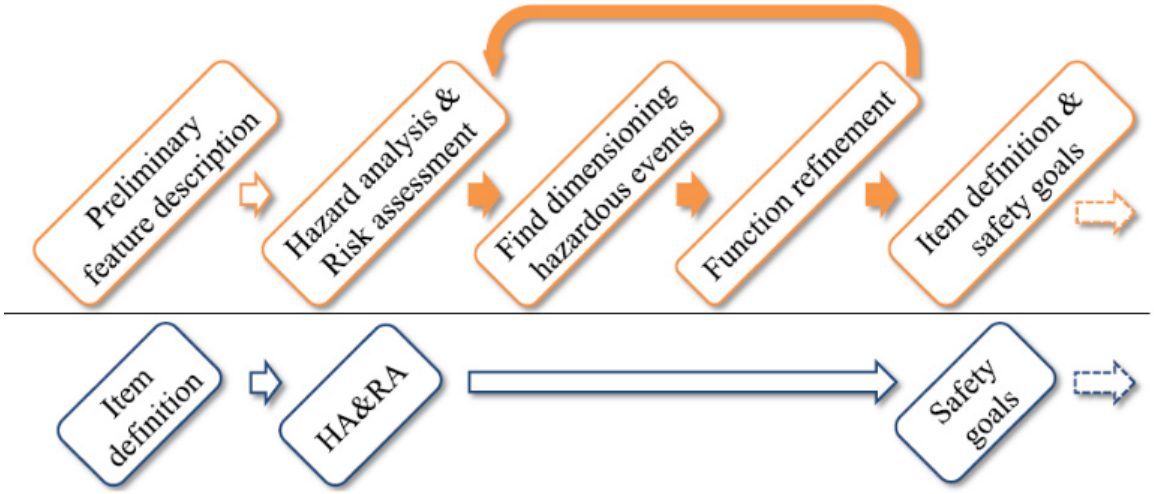


Figure 3.2: Iterative HA&RA and function refinement process [29].

3.1.3 Software Architectural Design

The Architectural Design phase states as the name indicates the high level architecture of the software as well as evaluation and verification of it. The software architecture describes the interfaces between the software components, the act in combination, dynamic aspects, like the work-flow of the processes. Additionally, it defines the functional and non-functional safety requirements, such support, function, usage and so on. This architecture clarifies the complexity and helps in the understanding of the whole software system. As mentioned above, there are also proposals for the methods related to ASIL. One table defines suggestions for notations, the others describes

principles and recommendations for mechanisms of error detection of the software component. Table 3.3 provides an overview of the recommendations to mechanisms for error detection at the software architecture level. For example an external monitoring facility is not compulsory in ASIL A, but highly recommended in ASIS D.

Table 3.3: Mechanisms for error detection by ISO 26262-6

	ASIL			
	A	B	C	D
Range checks of input and output data	++	++	++	++
Plausibility check	+	+	+	++
Detection of data errors	+	+	+	+
External monitoring facility	o	+	+	++
Control flow monitoring	o	+	++	++
Diverse software design	o	o	+	++

3.1.4 Software Unit Design and Implementation

The main objective in this phase of the V-Model is to specify the software units with the help of the software architecture. The following implementation of each component should be done with the help of the prepared design. The final stage of this phase verifies the implementation using static analytic methods. The goal of this analysis is to find runtime errors, memory errors and get a general understanding of code as well as the complexity.

3.1.5 Software Unit Testing

The software unit testing step is the first one in the test phase of the V-Model of software product development, shown in figure 3.1. When this step is finished the low-level components should be free of unwanted functionality. Before starting the testing, a software verification plan must created. The unit test has to fulfil the defined requirements of the software design, robustness and should show that there is no unintended functionality.

Figure 3.3 illustrates some test methods and test-case generation steps for unit and integration testing. On the left side of the figure the software unit testing steps are presented.

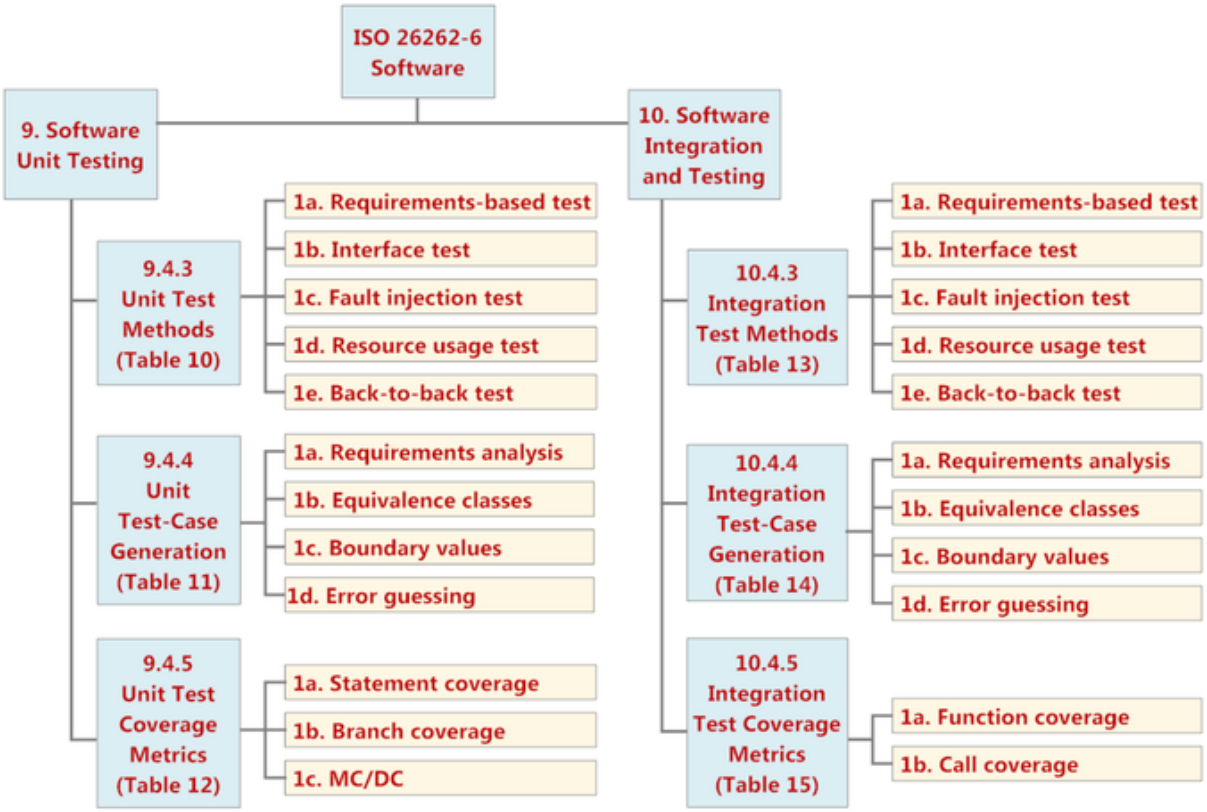


Figure 3.3: Software testing methods and metrics [19].

Software units are the smallest sub-system in software development.Testing these units makes it easier to find failures, detect the reason for it and solve it. Testing the whole system at once leads to great complexity and becomes very confusing very quickly. As can be seen from the table 3.4, the recommended requirements becomes more extensive when the ASIL gets higher. The upper part of the table 3.4 provides the suggestions for methods to test the sub components, like recommendation for interface test or a resource usage test. The lower part of the table represents the test-case generation methods, which should be used. For example the requirement analysis from the phases before should be definitely used.

Table 3.4: Recommendations of software unit testing methods and unit test-case generation methods by ISO 26262-6.

	ASIL			
	A	B	C	D
Unit testing methods				
Requirements-based test	++	++	++	++
Interface test	++	++	++	++
Fault injection test	+	+	+	++
Resource usage test	+	+	+	++
Back-to-back test	+	+	++	++
Unit test-case generation				
Requirements analysis	++	++	++	++
Equivalence classes	++	++	++	++
Boundary values	+	+	+	++
Error guessing	+	+	+	++

3.1.6 Software Integration and Testing

The following stage is the software integration and testing. The first goal is to integrate and connect the software units in a whole system. The integration can be done with the help of simulation and a ”In a Loop”-concept, like ”Software in a Loop” or ”Hardware in a Loop”, see section 3.3 or in reality with a prototype or a test bench approach.

On completion of integration the testing can be started. A flawless and mature unit testing does not mean that the integration testing has no errors. Due to new connections, interfaces and communications between the components new errors can occur. For this reason Integration testing is very similar to the unit testing, see figure 3.3 and compare the right, integration testing side with the left side of the unit testing. From this the table 3.4 is identical for the integration test.

3.1.7 Verification of Software Safety Requirements

The main task of verification of software safety requirements is to proove the fulfilment of the requirements specification provided in the design document. This task has to be done on the target hardware to test under real conditions.

Table 3.5: Software safety verification requirements by ISO 26262-6.

	ASIL			
	A	B	C	D
Verification planning and execution	++	++	++	++
Test environments	+	+	++	++
Target hardware	+	++	++	++
Evaluation of results	+	++	++	++

3.1.8 Item Integration and Testing

The last phase in the V-Model integrates the components to the highest level of the system, respectively to the end-system, hardware and software, which provides the function for the vehicle. At the end of this stage the integrated system has to be tested for the defined safety requirement found in the design phase. One way to validate the defined requirements is acceptance testing defined by Cimperman [8]. Each requirement defined in the software design should have at least one test case. At the end all test cases have to be successful to finish the project.

3.2 Risk Assessment

As discussed in the previous chapters, functional safety is very important for ADAS systems. To achieve this the ISO 26262 requirements must be respected. Most of it concerns testing the system. To fulfil SO 26262 the product must be tested in a real urban environment, but this approach has too many disadvantages. To avoid going beyond the scope of this thesis, only a few reasons are mentioned below. First of all, it would be too dangerous for the driver in the vehicle as well as all

other road users. Next, it would be too cost intensive and in addition not allowed, due to the law. These points are all very important, but just testing on real roads alone would not lead to a good result. The reason is very simple. Roads in the urban environment including all traffic participants and obstacles contains an extremely large amount of data. To test all different situations based on this dataset, infinite resources are needed. In order to obtain the most reliable result, you would have to test every innovation for a very long time on the road and that would take too much time and cost to complete.

For these reasons simulation becomes more and more important and reduces the cost factors by a multiple of its value. Unfortunately, it is also not possible to test all sorts of situations in a simulation, because it would cost too much resources, because of the hight amount of test data. Due to this fact the number of test-cases must be restricted. In order to guarantee a certain degree of security, all situations must nevertheless be covered by the restricted number of test cases. This will work with so-called extreme cases, in particular critical situation.

The specification of a critical situation is the key topic of this thesis. It can be determined by spatial or temporal proximity. A collision with another road participant or static object is also critical, but the situation starts already before this crash and is therefore not sufficient as a definition. Some possible indicators for the detection would be ”Deceleration to avoid Crash”, ”Collision Probability” or ”Time To Collision (TTC)”.

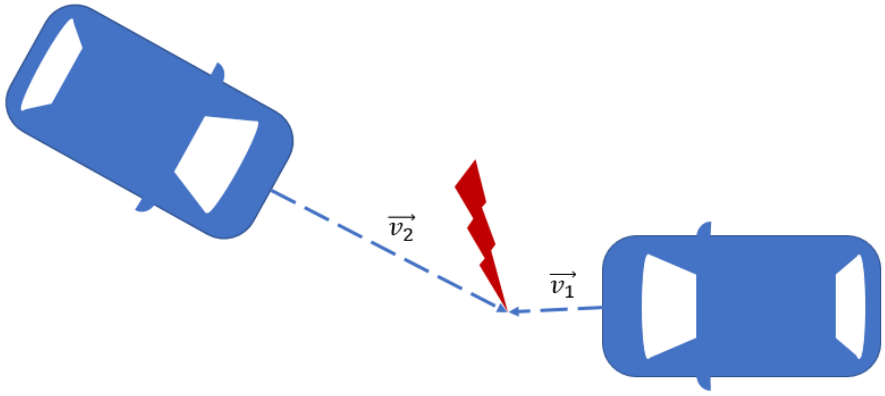
3.2.1 Time To Collision

In this thesis the focus is on the method of Time To Collision and how to handle traffic in an urban environment. Time to Collision is a well known method and was first introduced by John Hayward with the words ”the time required for two vehicles to collide if they continue at their present speed and on the same path” [12]. Hayward mentioned the shorter the TTC the more serious the situation and this approach is still valid. Figure 3.4a shows a collision of two vehicles with different velocities and different start location. Now it is possible, based on these parameters, to find an abstract TTC solution for this situation. The velocity v and the start location s_0 from equation 3.1 is known for both cars. To find the time where both vehicles are on the same point s the formulas must be equated, see equation 3.2. The received time is equal to the time to collision. In this simplified form of calculation, many factors, such as changeable speed, size and shape of the vehicle, etc., have been

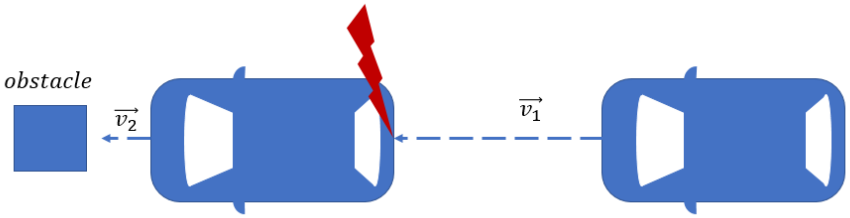
neglected. More detailed algorithms can be found in section 3.2.2 where the different used methods and approaches are explained.

$$s = v * t + s_0 \tag{3.1}$$

$$v_1 * t + s_{1,0} = v_2 * t + s_{2,0} \Rightarrow t = \frac{s_{2,0} - s_{1,0}}{v_1 - v_2} \tag{3.2}$$



(a) Collision with stable velocity.



(b) Collision due to unexpected velocity change.

Figure 3.4: Collision of two Vehicles.

The predicament gets more dangerous the closer the vehicles come together or the faster they drive. These factors are also covered by the time to collision. However, as mentioned above, the speed and rotation of the vehicles may also change. An extension of the formula above, will also include this change, but taking the acceleration into the account of TTC is not enough. It's quite possible that another traffic participant will accelerate unexpectedly. This can be for example, a change of direction or braking because there is an obstacle in front of the other vehicle. Figure 3.4b

represents a standard situation with a vehicle followed by another one. Without expectation an obstacle appears in front of the car ahead and it has to slow down extremely fast. Due to this fact a critical situation appears, because the following vehicle does not react on the brake event and might will crash. Situations like this can constantly crop up and should be taken into account of the risk level. With the help of a predefined probability distribution, the change of the forward speed and rotation, can be included.

By using a weighting function, see 3.2.2, concerning TTC, forward acceleration and rotation change, a certain risk value can now be determined with respect to one other vehicle.

3.2.2 Algorithms for computing the Time To Collision

As mentioned in the previous section, determining the time to collision is one way to get a value of danger. As explained earlier, the smaller the TTC the more dangerous the situation becomes. The difficulty is to calculate this in real time. The first task is to check if there is a collision between the vehicle to be checked and the object within a certain time. In order to do this and to accelerate the calculation process, a distinction is made between the object classes. They are classified in static objects, speed limit signs, traffic lights and dynamic objects, such as pedestrians and vehicles. An exception is the road line intersection, which will be included in the hazard level of the situation. Before proceeding to examine the risk level for the specific objects, it is important to give a short introduction of the used calculation methods, introduced by Jia Hou, George F. List and Xiucheng Guo [13]. We assume that all parameters, like position, velocity, acceleration and dimensions of the objects are given for these calculation methods.

Circle Method

This method is the simplest and fastest approach. Figure 3.6a represents the circle method with two colliding cars. An overlap of the two circles corresponds to a collision and happens when the distance of the two vehicles is less than the sum of the radii. A glance at the right of the formula 3.3 shows the squared sum of the radii (r_i, r_j). With the help of the pythagoras a collision can now be determined. The first part of the left side represents the squared distance on the x-axis of the vehicles to each other depending on the time t. The second part is responsible for the y-axis. x_i, x_j,

y_i and y_j are the center-coordinates (x&y-axis) of the vehicles at time t equals the start time zero. $v_{x,i}$, $v_{x,j}$, $v_{y,i}$, and $v_{y,j}$ corresponds to the velocities of the vehicles and $a_{x,i}$, $a_{x,j}$, $a_{y,i}$ and $a_{y,j}$ are the uniform acceleration of the vehicles. At the end, if there is a valid time t, then there is a collision of the vehicles and time t is equal to the time to collision ttc.

$$\begin{aligned}&[x_i + v_{x,i} * t + \frac{1}{2} * a_{x,i} * t^2 - x_j - v_{x,j} - \frac{1}{2} * a_{x,j} * t^2]^2 + \\ &[y_i + v_{y,i} * t + \frac{1}{2} * a_{y,i} * t^2 - y_j - v_{y,j} - \frac{1}{2} * a_{y,j} * t^2]^2 \\ &= (r_i + r_j)^2\end{aligned} \tag{3.3}$$

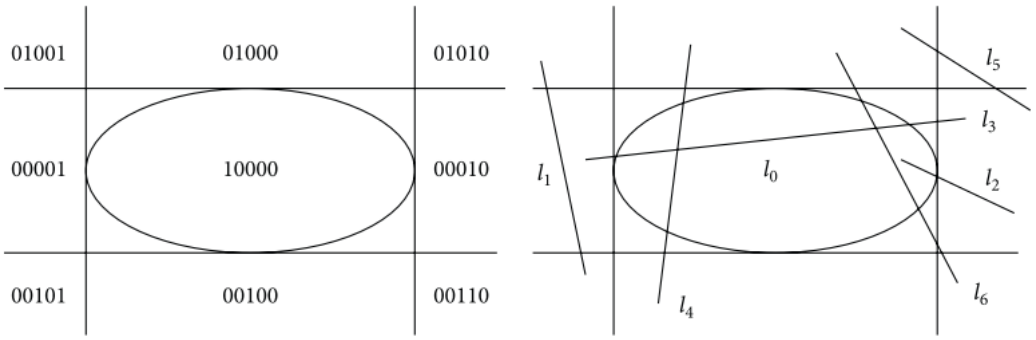
Ellipse Method

There are some differences compared to the previous methods. As the name suggests, an ellipse is used instead of a circle, see figure 3.6b. Due to this fact the angle α_i of the car is important for the calculation, see equation 3.4. Additionally, to check an object on a collision, every point of the object must be checked separately. For this reason this method is much slower than the circle method, but in some cases it's important to check each detected point. The advantage of this approach is the included buffer. In some cases it's useful to pay more attention on the longitudinal than to the transversal direction of the vehicle. The major semi-axis a_i is recommended to define with the half length of the vehicle multiplied with the factor 1.3. The minor semi-axis b_i can be defined with the width of the car time 1.1. Equation 3.4 obtains if the point with coordinates x_j, y_j is on the ellipse with the centroid coordinate x_i, y, i.

$$\begin{aligned}&\frac{1}{a_i^2} * [(x_j - x_i) * cos\alpha_i + (y_j - y_i) * sin\alpha_i]^2 + \\ &\frac{1}{b_i^2} * [(y_j - y_i) * cos\alpha_i - (x_j - x_i) * sin\alpha_i]^2 \\ &= 1\end{aligned} \tag{3.4}$$

Ellipse-Rectangle Method

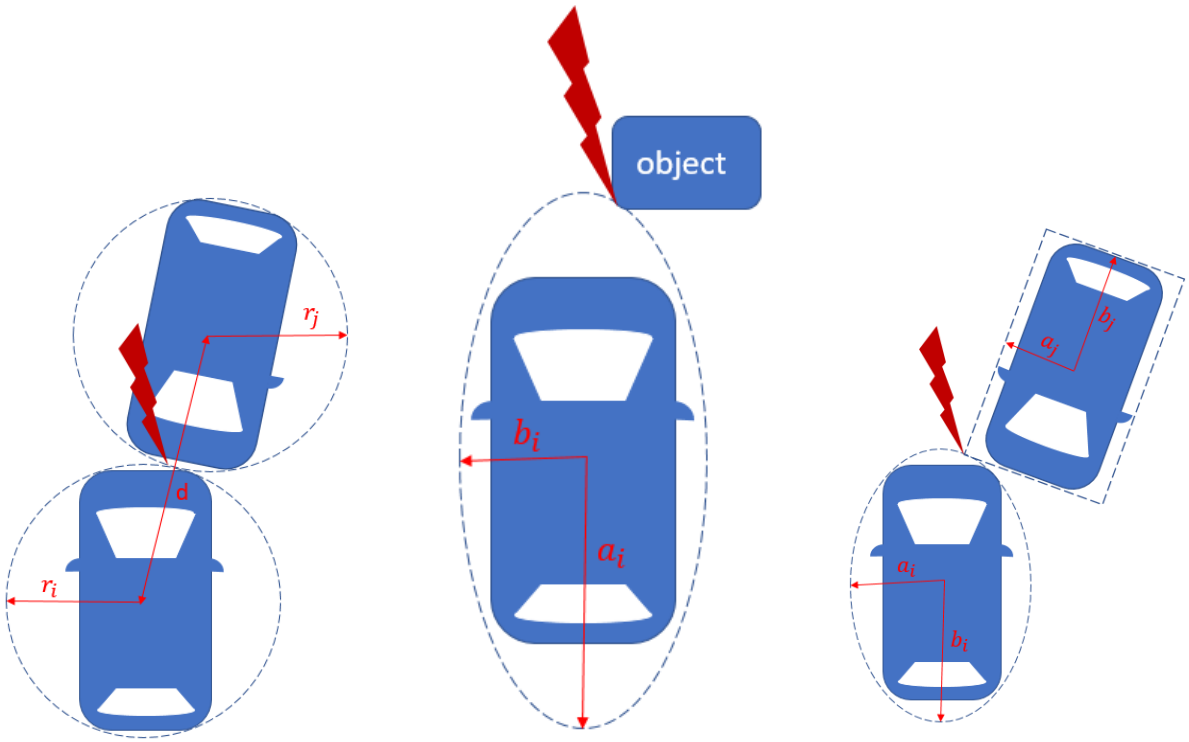
The most complex and accurate method for calculating the TTC is the ellipse-rectangle method. Figure 3.6c demonstrates this method and shows that there is one vehicle surrounded by a ellipse and one vehicle surrounded by a more detailed dimensioned rectangle. This method uses the ellipse approach and extends it with a rectangle instead of determining a single point.
To speed up the algorithm, the calculation is split in several steps. First the ellipse is divided into nine regions, see on the left of figure 3.5. In addition, each side of the rectangle is considered on it's own. The second step is a validation of line intersections. To check for intersection possibilities the start and the endpoint of a line is important. The rectangle overlaps with the ellipse if both points are in the ellipse, one point is in the ellipse or if the points are on the opposite of the ellipse to each other. If both endpoints of the rectangle are on the same side, for example both are over the highest point of the ellipse, there is no overlap. A further review is needed if one point is on the side of the ellipse and the other one is above or under the ellipse, for example the start point is on the left side of the ellipse and the end point is on the top right region.



(a) Nine regions for the ellipse (b) Lines and the ellipse

Figure 3.5: Detection of a intersection between the ellipse and a line segment of the rectangle [13].

Next, the further review is similar to the formula 3.4 where each point on the line of the rectangle is tested separately. If there is a side of the rectangle with an overlap an collision exists.
This approach is a very detailed method with many process steps. In some cases the calculation is very fast, however there are some cases where a further review is needed and then the approach is a slow one and should only be used in exceptional cases.



(a) Collision detection with the help of the circle method.

(b) Collision detection with the help of the ellipse method.

(c) Collision detection with the help of the ellipse-rectangle method.

Figure 3.6: Collision of two vehicles with various calculation methods.

3.2.3 Object Classification

In general, a situation consists of several influencing factors and not just two vehicles. The influences are classified in static, dynamic and hybrid objects. Static objects have a fixed position and therefore no velocity or acceleration change. Some static objects are listed below:

- Buildings
- Fences
- Poles
- (Roads)
- Sidewalks
- Vegetation
- Walls
- others

To determine the risk level for these objects calculating with TTC can be done again. Due to the fact that the position does not change all the time and the assumption that we know our speed and acceleration, we are not required to handle a probability distribution. All parameters are known and the risk level is equal to the TTC and it's weighting.
The next class provided in the following list includes dynamic objects:

- Pedestrians
- Vehicles
- Animals
- others

The risk level of all these dynamic objects can be calculated with the help of the vehicle calculating technique. The difference between this object is only the probability distribution and the weighting. For example, an animal will change its direction or speed much more likely than a car. Therefore, the expected value of a rough change of direction of an animal must be much grater than the value of a car.
The next class are some atypical cases like traffic lights, road lines or speed limit signs. A traffic light for example has a fixed position, but sometimes the status is green, which equals ready to go and can be ignored. In the other case the traffic light is on the same position and the status is red, which equals don't go ahead and can be taken into account like a static object with a virtual line over the road. In contrast, speed limit signs always have an impact. The speed should not be greater than the speed limit valid from the position of the sign. There is a so-called scope of validity.

3.2.4 Hazard Level Functions

As stated above, it is possible to calculate a risk level for all individual objects. In order to evaluate a situation in the whole it is important to bring the different evaluations to a common denominator. This section provides three different techniques to determine a hazardous level to find critical situations, which are deployed in the case study.

Summed Function: The first approach sums up the individual levels of the objects. Additionally, the different objects will be weighted in a different level. With the help of this weighting the Severity of damage from ASIL will included into the risk assessment of the situation. Equation 3.5 illustrates

a possible calculation including the different classified objects. w_x identifies the weights and $risk_x$ the determined risks. The main advantage of this method is that at any time all dangerous objects can be found in the calculated level. Using equation 3.6 the value can be restricted between 0 and 100 where B is equal to the risk level where the hazard level should be 50.

$$\begin{aligned} risk = & w_{staticobjects} * risk_{staticobjects} + w_{vehicles} * risk_{vehicles} + \\ & w_{pedestrians} * risk_{pedestrians} + w_{speedlimitsigns} * risk_{speedlimitsigns} + \\ & w_{trafficlights} * risk_{trafficlights} + w_{roadintersections} * risk_{roadintersections} \end{aligned} \tag{3.5}$$

$$hazardlevel = 100 * (1 - \frac{2}{e^{risk/B} + 1}) \tag{3.6}$$

Highest Value Function: The major disadvantage of the "Summed Function" is how to handle cases with a low and a large number of traffic participants with the same parameters at once. For example, if a group of persons go on the side of a road, this method calculation leads to a more dangerous situation than a street crossing person. The reason is that the small risk levels of the individuals added may yield to a greater value than the crossing person. To solve this kind of problem the "Highest value function" was introduced, see equation 3.7. With the help of searching the maximum risk level of all objects the hazard level is defined. The disadvantage of this method is that only the largest risk factor is represented in this level.

$$hazardlevel = \max_{k \in objects} (w_k * risk_k) \tag{3.7}$$

Top Average Function: To combine the advantages of the previous methods the "Top Average Function" was developed. The first step is to determine the average value of the risk, defined in 3.5. The following step is to select all objects with a risk level higher than the average one. With the help of this the small values, like those from the group of persons, can filtered out. Now, it's possible to calculate the "Summed function" with the related risk values again. This form of calculation method has now the advantages of both method. It has information to all relevant object and there is a higher chance of receiving a qualitative hazard value.

3.3 Modelling and Simulation

As mentioned in the introduction, experimenting in the real world is quite expensive. Due to this fact, the idea of modelling and simulation (M&S) with the help of computers is a cheaper and safer way of testing. Computer based simulation dates back to the second world war to get a basic understanding of the behaviour of neurons. Louis G. Birta and Gilbert Arbez introduced modelling and simulation with the following words: ”Modelling and simulation is a tool that provides support both for the planning, design and evaluation of dynamic systems as well as the evaluation of strategies for system transformation and change” [17].
Today M&S is often used as a replacement of the physical experiments. To do this, first the model have to be designed. Decisions have to be made in what depth the mathematical and physical laws will be included to get a sufficiently good model to represent the real system. If the system is sufficiently defined, it can be simulated and displayed with the help of computers. Now, the behaviour of the system can be studied more easily and inexpensively without taking the risk of physical experiments into account.

3.3.1 Modelling Process

To develop a simulation environment the type of simulation model must be specified. Sargent R. defined in ”Types of Models”[4] four main model types. There is the *mathematical model*, which formulate the system by using mathematical expressions, relations and logistics. Mathematical models consists out of one or more equations which simulate a deviation of the real world. The *analog model* determines out of some analog input characteristics a system of interest. *Graphical Model* is the next model type and represent the word objectives in scope of graphs. The graphs often have directed edges to connect the nodes, so called directed graphs. The last model type is the *iconic model*, which looks like the real system and take usage of physical laws. Example systems are a wind tunnel for testing air plains or testing the behaviour of vehicles in a urban environment with real physic laws.

The development process, illustrated in figure 3.7, starts with a requirement analysis to define the goal objectivities of the project. This is important to compare existing components to be honest not to start from scratch [6]. Testing ADAS will need a virtualization of many traffic scenarios. There

are already many different approaches with a lot of know how. To re-implement these would mean a great deal of work and it is often easier to incorporate parts of these simulations. To define the software design the requirements are important to decide which existing components can be used and which components must be developed.

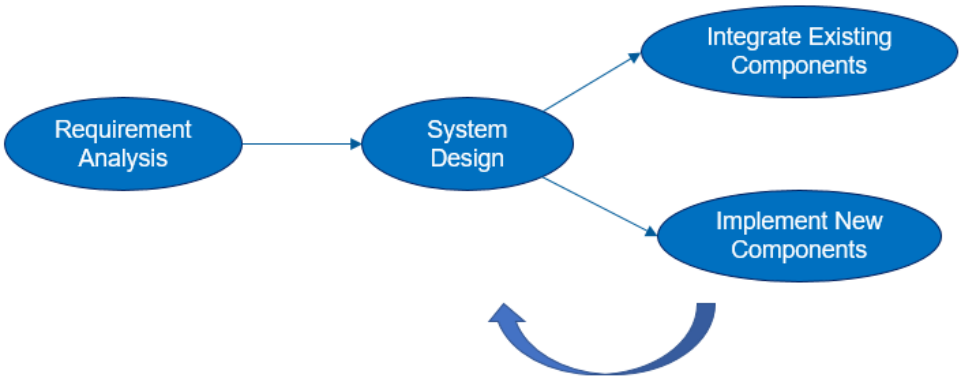


Figure 3.7: Development and Modelling a Simulation System.

To clarify whether existing components can be used or not the model should be verified and validated. The Modelling and Simulation life circle must be modular and structured to make the project successfully [18]. Figure 3.8 demonstrates a simplified verification modelling process. On the top of the figure is the problem entity, the real word with the idea specified by the requirements. Next the conceptual model is defined as the mathematical, logical reconstruct of the problem entity. At the end the computerized model is the implementation of the conceptual model.
Sargent [25] defines the process to develop the conceptual model as Analysis and Modelling. Develop the computerized model can be done through computer programming and implementation. Next the experimentation process checks with the help of tests and experiments the relation to the problem and checks the deviations from the problem entity.
When these components are defined, the new or already existing components can verified and validated. The conceptual model validation obtains the accuracy and differences between the problem entity and the conceptual model. The computerized model verification observes the implementation of the computerized model. At the end the operational validation monitors the result of the computerized model and compares the output with the real system.

Next the the modeling and simulation process in more detail will be descripted adapted from Benjamin M. [5]. The starting point is the real system or problem entity. The first step is to establish the goal objectives and the scope of the environment, which can be done with the help of the requirement analysis. The next task is to formulate the conceptual model. After defining the model, the data should be analysed and acquired using the conceptual model validation. Next the implementation of the computerized model can be done with a final verification. The last step in the developing process circle is the design and execution of experiments. At the end of the process a decision must be made if the simulation is good enough. If the simulation is not good enough, the whole process must restarted from the beginning. If the simulation is good a documentation must be written. It can be seen, that this process is very close to the simplified validation and verification modelling p

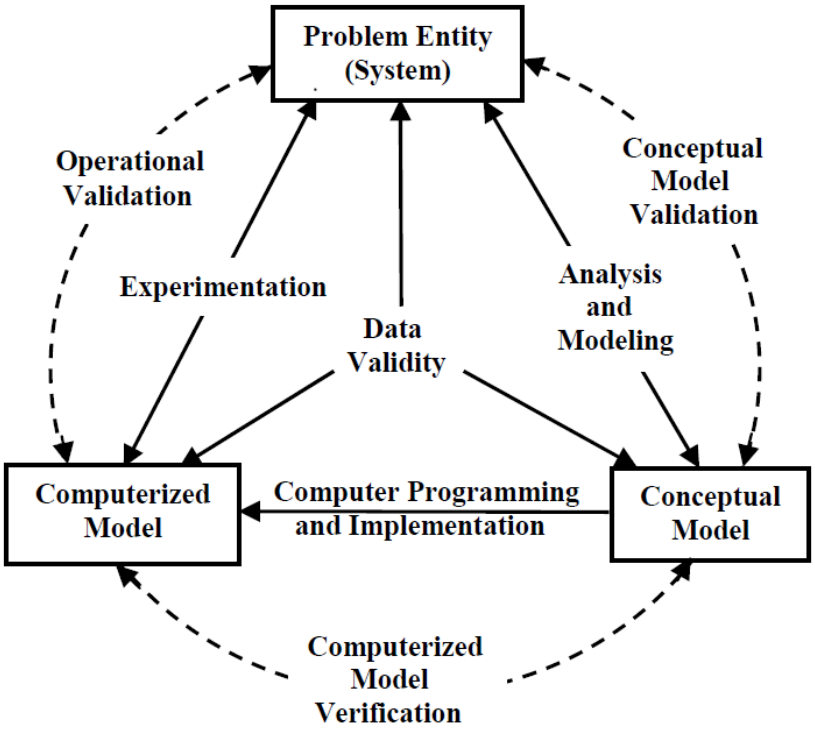


Figure 3.8: Simplified validation and verification modelling process [24].

3.3.2 Validation Techniques

What follows is a brief description of validation techniques to check the accuracy of the computerized model an relation to the problem entity. This methods can used in the simplified modelling process and provided by Sargent [25].

- Animation: The behaviour is displayed graphically.
- Comparison to Other Models: Results can compared with other models.
- Degenerate Tests: The model behaviour is tested with the help of specific values and compared with expected outputs.
- Event Validity: Event triggered actions will be tested based on specific events.
- Extreme Condition Tests: Model behaviour is tested by extreme conditions of one or more levels.
- Face Validity: Knowledge of individual behaviour of the model will asked.
- Historical Data Validation: The results of the model will tested with historical data of other sources.
- Internal Validity: Several stochastic test runs with known results of individual components will tested.
- Multi-stage Validation: A combined method of the historical methods is used.
- Operational Graphics: Important values will shown in form of graphical output to compare the behaviour more easier.
- Predictive Validation: A forecast of the systems behaviour is used and compared with the model.
- Turing Tests: Knowledgeable persons are asked to check accuracy.

3.3.3 In the Loop

M&S is used in a variety of areas in the economy as well as in research. Some example domains are medical, research, military, gaming as well as in industrial usage.
Car manufacturers are using simulations more and more. On the mechanical side simulations can be applied for example to develop better driving behaviour, getting a better traction with or without a spoiler. By arising software supported components in vehicles modelling and simulation

of such systems becomes more and more important. For this reason and the increasing number of advanced assistance systems, simulating a system for vehicles and their components is a big development field. In order to simulate the system, the system itself as well as its environment and their influences must interacting in the simulation. This is named "In the Loop"-Modelling and Simulation. There are three different kinds of "In the Loop" approaches.

Human in the Loop: The main goal of this simulation is to train people. Applications are for example: air plane, military, emergency management, traffic control simulations and many others. The system is designed to interact with the human in it and simulate his actions in the system. The human should see the result of its interactions to learn with the help of a virtual system. With the help of these systems its easy to evaluate the performance of the human and the system, when interacting together.

Hardware in the Loop is used to test and verify the function of hardware systems under simulated input values. The system is designed to provide the specific hardware system with a nearly realistic input and take care of the output of the hardware again to react on possible influences. For example a vehicle with ADAS systems can operate with simulated input values. Objectives of this kind of simulation is the operational testing and evaluation of hardware components.

Software in the Loop: The software system will run under a simulated environment to provide inputs, like sensor signals. This simulation environment is interacting with the software system and acts on it's output. A Lane Keeping Assistant (LKA) for example is providing the needed steering to keep in the lane of the road. The simulated environment has to react on this new steering value and provide a sensor signal with a new distance from the center line. Now again the LKA can react on the new distance value of the simulated sensor signal. This order will be processed until the end of the simulation, see figure 3.9. At the end the verification of the LKA in a simulation environment can be done and the question on the functionality can be answered right now. In practice the LKA needs much more sensor signals and needs to provide more output values.
Section 4.6.1 provides more detail to the implementation of a Software in the Loop system using Model.CONNECT. Model.CONNECT, developed by AVL List GmBH [3], is a tool to combine different tools together. With drag and drop options it's easy to define a large system out of small components from different tools.

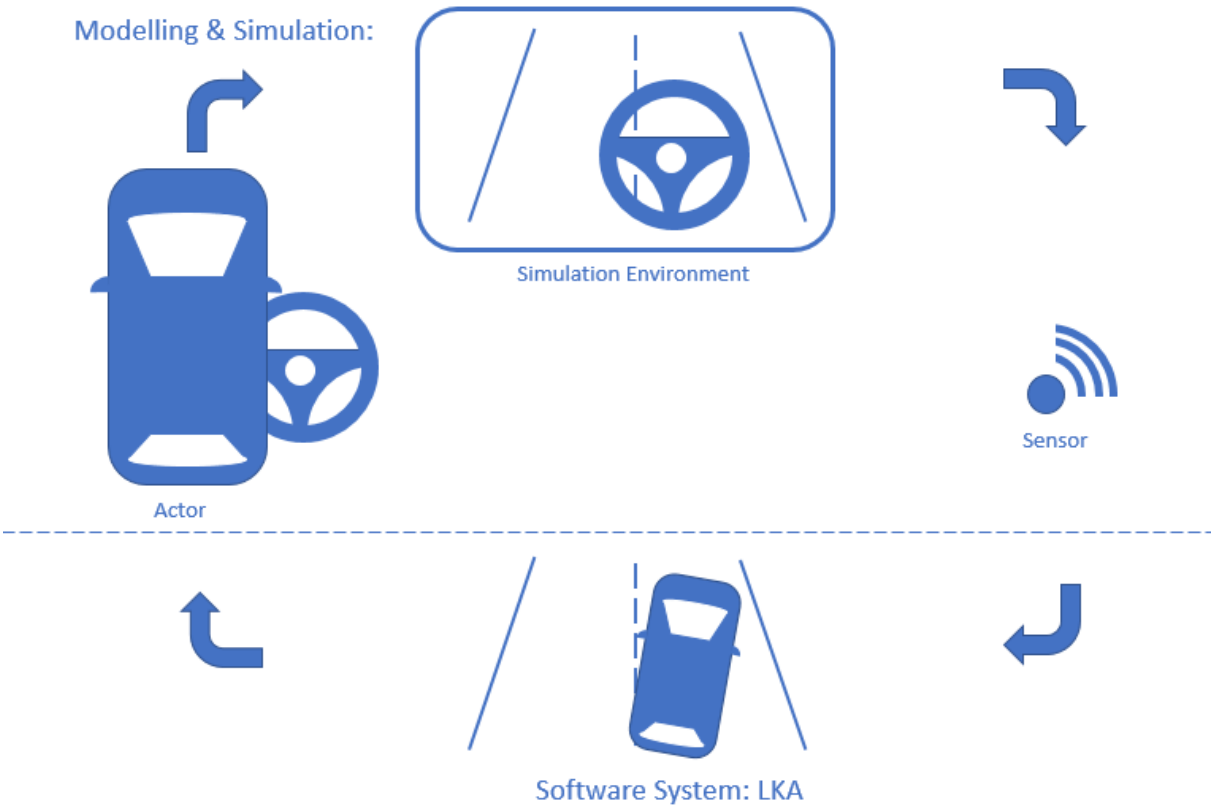


Figure 3.9: Software in the Loop Modelling and Simulation of Lane Keeping Assistant (LKA).

3.3.4 Agent-Based Simulation

Important to test ADAS is the interacting behaviour of different dynamic objects, for example other vehicles, pedestrians, animals and others, see figure 3.10.

Agent-based modelling and simulation (ABMS) is one approach to test this interactions [30]. Dynamic objects are represented as agents, which interacting with the system. A agent is a software part which executes tasks and decision making on it's own. Agents can communicate with other agents or humans and should designed predictable and within a set of constraints. A ABMS consinsts out of one or more agents to determine and analyse the interacting between the agents and the rest of the environment.
To build a agent-based simulation some questions must be answered. Important questions defined by Whitaker [30]: "What real-world characteristics or behaviors will the agents need to represent in the simulated world?" and "What are the characteristics or attributes of each agent (or class

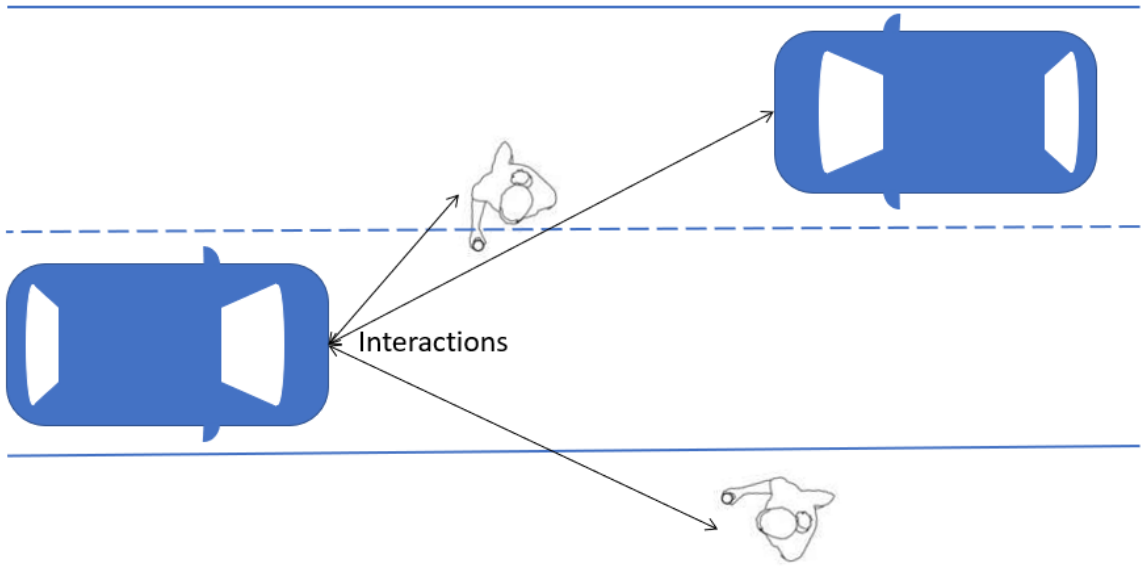


Figure 3.10: Interactions of Agents and vehicle controlled by ADAS.

of agents)?". Also good to know is which input values and which output values are available for agents. Based on this questions and other requirements agents can developed or if existed integrated in the simulation environment.

4 System Design and Implementation

This part of this thesis describes in greater detail the practical part. Regarding to the V-Model, discussed in section 3.1, this chapter focuses on the testing phase.
With the developed algorithms its possible to find and create new test cases as well as testing some new software components of a vehicle. Finding new test cases, or better finding critical situations is also the main task of this work. The main objective is to find these situations in an urban environment when driving with a more or less ADAS supported vehicle.

To implement this, the Software in the Loop method, described in section 3.3, is used. In order to validate ADAS, very high demands are put on a simulation environment. In section 4.2 the requirements are listed and some simulators are compared to each other. To anticipate it, the CARLA simulator is used, because this simulator meets the requirements best. The structure and function of CARLA will be explained in the section 4.3.

As far as section 3.2 is concerned, it's not easy to find critical situations. These considerations have been adapted to the environment in this thesis. To get a hazard value, first the risk for each individual object has to be calculated. In order to achieve this, it must be checked whether the objects meet certain conditions. Additionally, the Time To Collision (TTC) must be determined. With the help of some extensions and algorithms, like ”circle method”, ”ellipse method” or ”ellipse-rectangle method” this concept is implemented in the practical work, refer to section 3.2.2.

At the end of this chapter, some use cases will be demonstrated with the focus on critical situations.

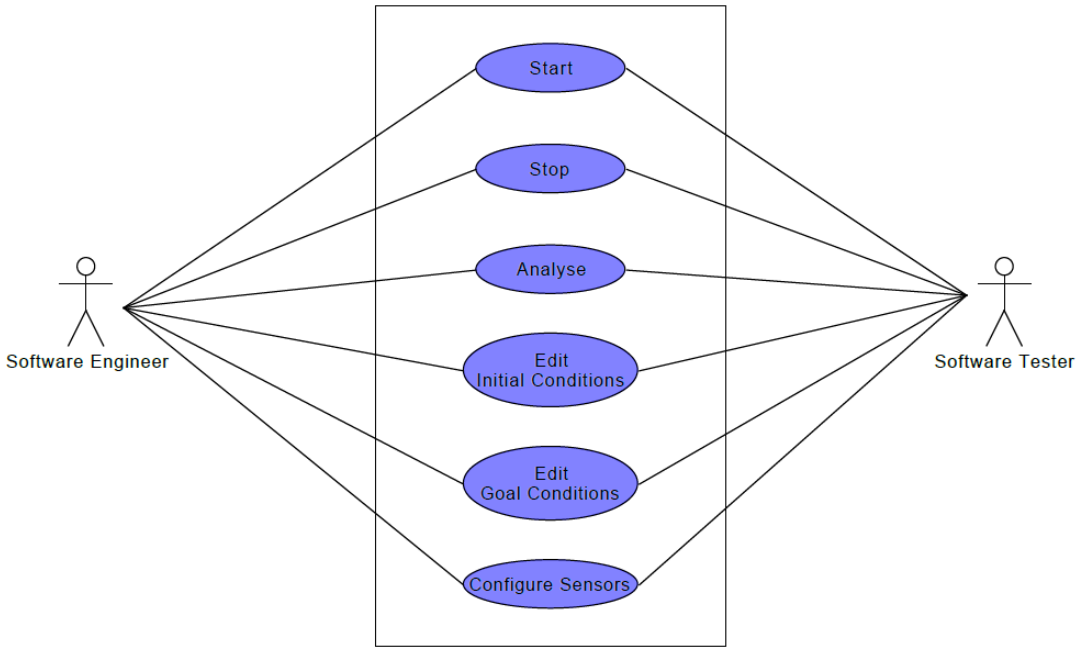


Figure 4.1: Business Use Case Diagram.

4.1 System Design & Requirements

With regard to the V-Model, software design is one of the first steps in a project and was done before the implementation of the software. Below the procedure is explained on the basis of a general overview of each component.

4.1.1 Business Use Case Diagram

In order to identify all needed features, a Business Use Case diagram, see figure **??** was created to understand the needed functionality. The tables below describes the use cases in more detail. This Business Use Case diagram was the start and of course has changed in the course of work.

4.1.2 Requirement Analysis

Based on these use cases, an requirement analysis can be performed. The goal of these analysis is to understand the requirements of the customer to the system to be developed. Next with the help

Table 4.1: Use Case Description for the activity Run

Run	
Description	Starts the simulation.
Actors	• Software engineer • Software tester
Preconditions	Simulation not running.
Basic Flow of events	Start Simulation using the initial parameters
Result	Simulation is started

Table 4.2: Use Case Description for the activity Stop

Stop	
Description	Stops the simulation.
Actors	• Software engineer • Software tester • Software Developer
Preconditions	Simulation is still running
Basic Flow of events	Press Abort Button
Result	Simulation is stopped

Table 4.3: Use Case Description for the activity Analyse

Analyse	
Description	Analysing the simulation.
Actors	• Software engineer • Software tester
Preconditions	Simulation is running
Basic Flow of events	With help of the Debug Window on the right side of the window the user will be able to analyse the simulation
Result	

Table 4.4: Use Case Description for the activity Edit Initial Conditions

Edit Initial Conditions	
Description	Change the Initial state and it's parameters
Actors	• Software engineer • Software tester
Preconditions	Simulation not running.
Basic Flow of events	Open the Configuration windows and edit the initial conditions.
Result	Changed Starting Condition.

Table 4.5: Use Case Description for the activity Edit Goal Conditions

Edit Goal Conditions	
Description	Change the hazard level limit value to detect critical situations
Actors	• Software engineer • Software tester
Preconditions	Simulation not running.
Basic Flow of events	Open the Configuration window and edit the goal conditions.
Result	Changed Hazard Condition.

Table 4.6: Use Case Description for the activity Configure Sensors

Configure Sensors	
Description	Enable or disable object classes to take these into account of the hazard level or not.
Actors	• Software engineer • Software tester
Preconditions	Simulation not running.
Basic Flow of events	Open the Configuration window and edit the object sensor settings.
Result	Object sensor effects changed.

of the pattern of [21] the most important requirements are listed. Clear defined requirements are important to get an basic understanding of it and with these it is possible to develop the first test cases for testing the software.

Functional requirements

4.1.1. **Road traffic variations:** The software must be able to simulate different road traffic variations based on Unreal Engine 4.

4.1.2. **Interaction:** The software must be able to interact with the ADAS/AD system of AVL List GmBH.

4.1.3. **Control:** The software is required to simulate a car controlled by ADAS/AD system of AVL List GmBH.

4.1.4. **Start:** The user shall be able to start the simulation with one click.

4.1.5. **Stop:** The user shall be able to stop the simulation with one click at any time during simulation is running.

4.1.6. **Transparency:** The user shall see the current hazard levels in the right part of the window.

4.1.7. **Sensors:** The software is required to provide simulated sensor values for ADAS/AD components, when requested from ADAS/AD system.

4.1.8. **Actors:** The software is responsible for executing the corresponding commands, see section 4.1.2, given by the ADAS/AD system.

4.1.9. **Traceability:** The user should be able to reconstruct the simulation found critical situations with the help of a logging protocol.

4.1.10. **Notifications:** The user shall be notified in the event of an accident.

4.1.11. **Performance:** The software shall be able to work with a environment with minimum requirements defined in section 4.1.2.

Actor Commands

4.1.1. **Throttle:** The software is required to provide an interface for receiving the throttle value.

4.1.2. **Braking:** The software is required to provide an interface for receiving the braking value.

4.1.3. **Steering:** The software is required to provide an interface for receiving the steer value.

Performance requirements

4.1.1. **CPU:** The software will be able to work with Intel Core i5 3470 @ 3.2GHz (4 CPUs) / AMD X8 FX-8350 @ 4GHz (8 CPUs) or higher.

4.1.2. **RAM:** The software will be able to work with 8 GB or more.

4.1.3. **OS:** The software will be able to work with Windows 10 64 Bit

4.1.4. **Video card:** The software will be able to work with NVIDIA GTX 660 2GB / AMD HD 7870 2GB or better

4.1.5. **Sound card** The software will be able to work with DirectX 10 compatible

4.1.6. **Disk space** The software will be able to work with minimum 72 GB free disk space

4.1.3 Prototype of Graphical Interface of Launcher

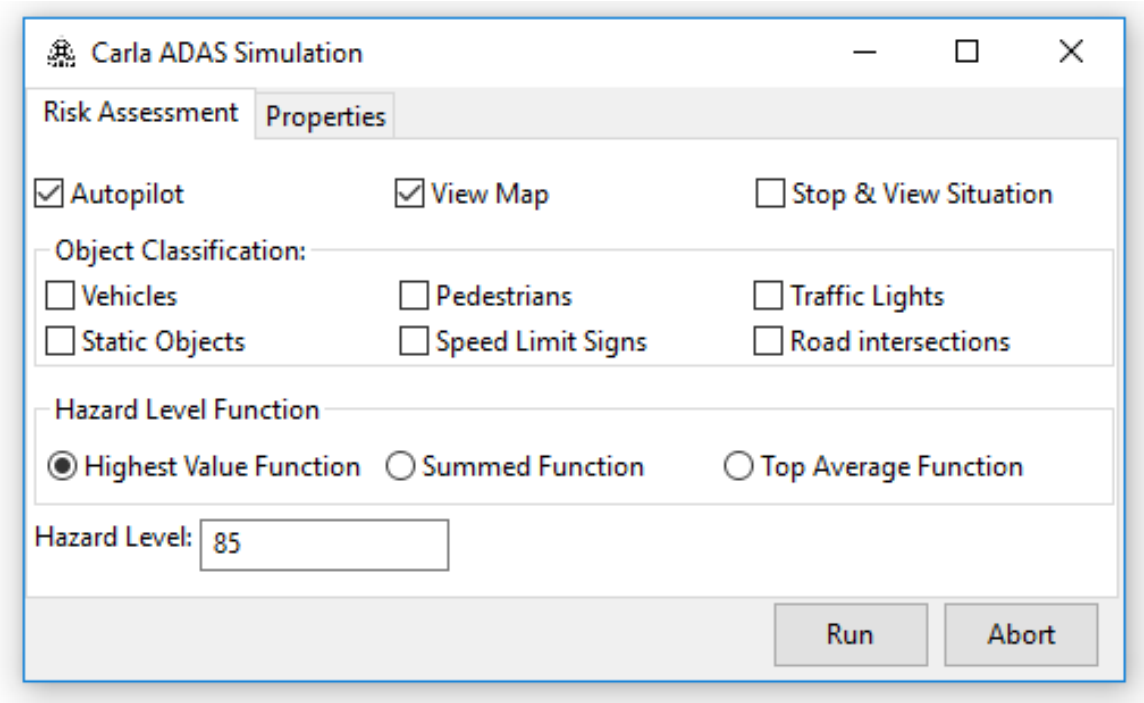
Before starting to develop, the prototype of the graphical User Interface of the launcher has to be defined. This interface should be be simple and intuitive. Figure 4.2 represents the prototype of the graphical user interface. Figure 4.2b shows the configuration properties like resolution, start point or number of agents. The other figure 4.2a demonstrates the risk assessment parameters. With help of these parameters the observed object can be enabled or disabled, the hazard level properties can be set and if wished a autopilot can enabled.

With help of this prototype and above defined requirements the development of the software can be started.

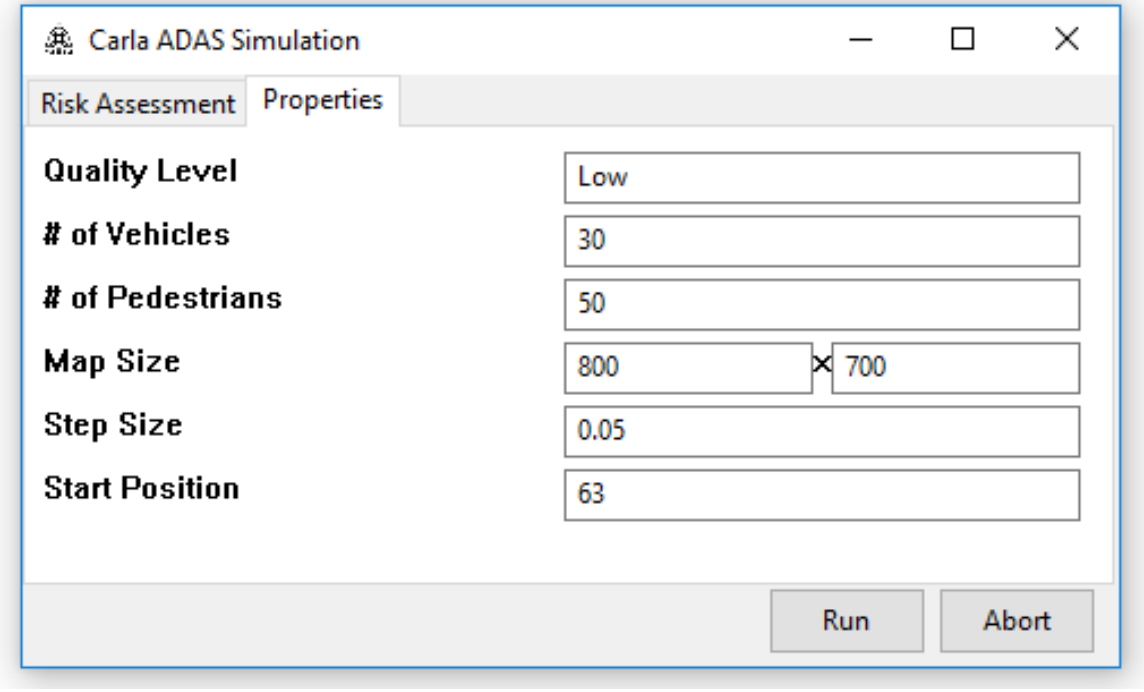
4.2 Environments

There are a variety of simulation environments in the world of car simulation, so the following lines list and compare the most important simulators. The table 4.7 illustrates some of the main characteristics of the different analysed simulators.

The left column of the table is classified in the specified requirements. The idea of using a simulator based on a gaming engine was created because of the slight expansion of the world. A gaming engine like the Unreal Engine 4 is a very effective graphical engine, which provides a physically



(a) Risk Assesment



(b) Properties

Figure 4.2: Graphical User Interface of the Launcher

realistic three dimensional world. In addition, there is a large community which is developing and extend new maps, for example urban environment. An application interface is important for implementing algorithms to use ADAS and find critical situations in the simulation environment is clearly, which makes a communication possible.
The next requirement is the three dimensional virtualization to simulate optimal or realistic sensor signals, like a depth camera or LIDAR. Every accuracy of a sensor depends on several factors like fog, rain, brightness etc. For this reason the next requirement is a controllable weather.
Another important point is the virtualization of agents. To test ADAS it's important to have a realistic scenario. So it's important to have other vehicles and pedestrians on the road. It would be optimal if it's able to take control over pedestrians.
Other requirements listed in the table are a "nice to have" and are also important to develop a risk assessment for ADAS based on a simulation environment. An autopilot simplifies the work of testing the risk assessment by driving an optimally driven car. The next "nice to have" is that the simulator supports more operating systems, like Linux and Windows as well as work as and "Client-Server" application for better usage of resources. There are many additional features, which are important for an optimal simulation environment. However, a list of these points would go beyond the scope of this thesis.

What is interesting about the data in this table is that the commercial simulator environments are often not based on a gaming engine.
Virtual Test Drive (VTD), developed by Vires ([28]), is a simulation tool chain for road traffic and provides sample maps with the possibility to extend it. All data can be exported and imported by using the OpenDrive [28] or the OpenCRG [27] format. With the help of a few mouse clicks it's possible to configure the virtual world and maybe define pedestrians or add other events.
Similar to VTD, more simulators, like monoDrive [27], rFPro [27]), Tass PreScan [27] exist with or without configuration possibilities.
Very often games have a large environment, but coding interfaces are often not available or only very limited. Grand Theft Auto V [27] for example does not have an officially supported interface (open for modifications), however, using it as a training environment for self driving cars is not desired.
The most interesting environments are at the end of the table. The AirSim Car Simulator, [26], developed from Microsoft, based on the Unreal Engine 4 (UE4) have the advantage of an easy

customization environment without further restrictions. For this reason and the real physics, it is possible to develop real behaviour sensors and actors simulation. Similar to AirSim there is Udacity [26], which is based on the Unity Engine. Both simulators have the disadvantage of not having agent interaction implemented.

The most interesting simulator is CARLA: An Open Urban Driving Simulator [11] based on UE4 as open source and developed to support Autonomous urban driving systems. CARLA provides a variety of environment settings, including weather and time of day within an urban world. The driving simulator is structured as a client-server system. With the help of the server the scene will be rendered and simulated. In the server application pedestrian and car agents are integrated. It receives commands from the client api and returns sensor values, position and so on. The Client API is used for interaction and is implemented via Python and sockets. A more detailed account of CARLA is discussed in the following section.

4.3 CARLA

CARLA [11] is an open source simulation environment developed for learning, testing and verification of advanced driver assistance systems. CARLA: An Open Urban Driving Simulator was developed from a cooperation between Intel Labs, Toyota Research Institute and the Computer Vision Center of Barcelona.

As already mentioned above, the simulator is constructed with a client-server architecture. This allows separation between the simulated environment and the software being developed. Due to the modular architecture it's possible to develop separately without having knowledge of progress or changes from the other one.
The server application is written in C++ with the help of some external libraries. The world is modelled using the Unreal Engine 4. Two standard worlds are included in the CARLA download package. However, the worlds can be expanded or changed as required, or a completely new one can be designed. Additionally, there is a big community which are providing different worlds partly free of charge and sometimes with costs. CARLA also provides digital assets of vehicles, objects, buildings, vegetation objects and many others to create a realistic scenario.
By using the C++ language some intelligence has been added to the world. Agents, like pedestrians,

	VTD	Tass PreScan	rFPro	AutoVi-Sim	GTA V	AirSim Car	CARLA	Udacity
Gaming Engine					√	√	√	√
API	√	√	√	√	√	√	√	√
Freeware						√	√	√
3D Virtualization	√	√	√	√	√	√	√	√
Ecosystem	√	√	√	√	√		√	√
Weather Control	√	√	√	√		√	√	√
Autopilot		√	√				√	
Agents	√	√	√	√	√		√	
Controllable Agents	√		√	√			√	
Map Customization	√	√		√		√	√	√
Linux system	√	√	√			√	√	√
Windows System	√	√	√	√	√	√	√	√
Server/Client Support	√	√	√			√	√	

Table 4.7: Analysed Simulators: Used ecosystem, interface, price and agents.

Figure 4.3: Scene in a street of town 2 from CARLA Simulator with four different weather conditions [11].

motorbikes and cars are implemented using an autopilot moving around the world. The weather conditions provided by the unreal engine are connected and many different weather scenarios are implemented. Furthermore, traffic signs, speed limit signs, traffic lights and other traffic important features were covered with the intelligence. In figure 4.3 the effects of the different weather conditions and other CARLA features are shown.

The next big topic in the server application are the integrated cameras and sensors. Four different types of sensors are implemented from scratch: The Scene final Camera, Depth map Camera, Semantic Segmentation Camera and the Ray-cast based Lidar Camera. All these camera interfaces using different properties for example its possible to add some noise, or change the range of the camera. As well as these parameters the cameras and sensors can be located on any area of the vehicle and connected with whose location to also change the position when the vehicle is moving.

The most important external library is the Protobuf library from Google. It is used for the communication between the server and the client. To do so, the measurements and sensors and cameras value were structured with the help of XML and defined in the library and sent by using the transmission control protocol (TCP). This library provides a smaller and faster serial communication between the two applications. Through the TCP communication it's possible to run the server application

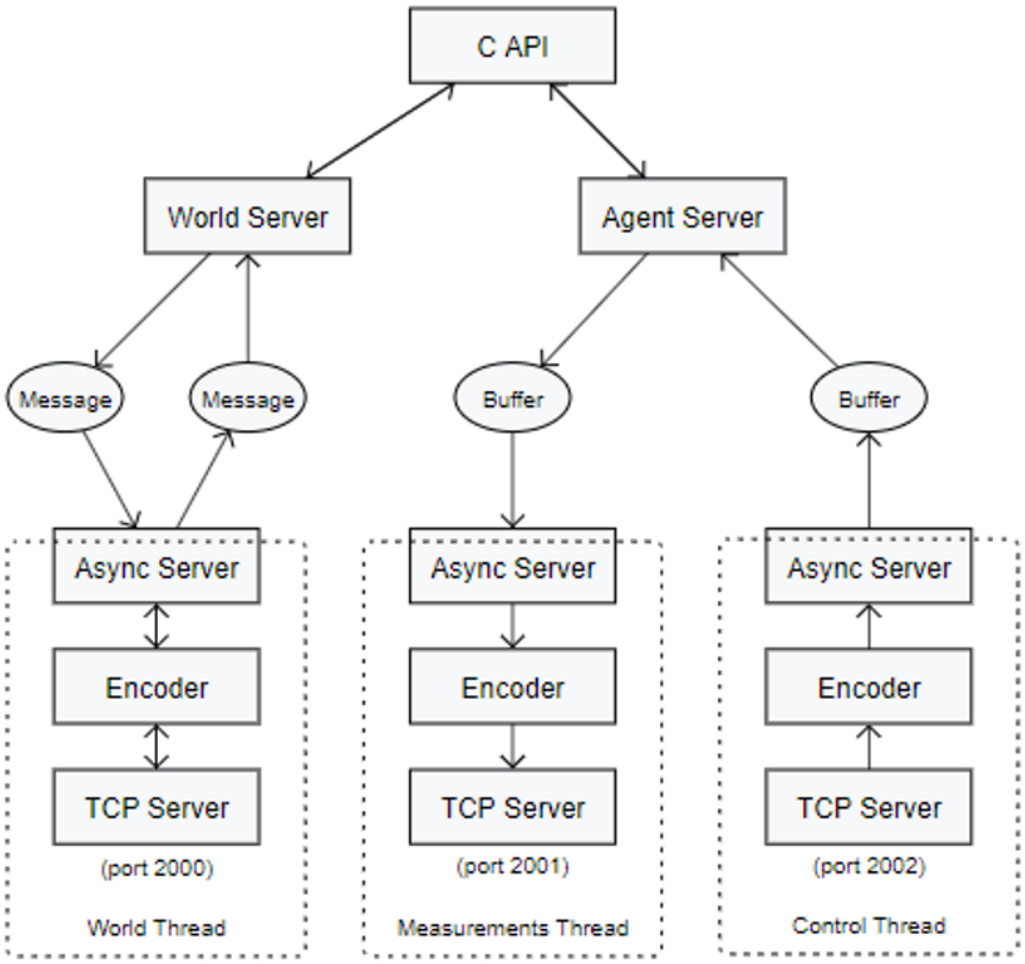


Figure 4.4: Concept of the communication between client and server [10].

on another computer or even Server and work with their resources. As can be seen in figure 4.4, the communication is divided between two different servers, a world server and a server for the agent interaction. The server for the world is a bidirectional one with the standard port 2000. The agent server uses the following two ports. The first one is used to provide the measurements to the client application. The second port is the so called ”Control Thread” which is used to receive the control commands from the client application. The code written in the C language is on the top of the tree is. This controls and manages the whole communication and interacts with the Unreal Engine, which presents the simulated scenario.

The client application has no predefined language. The most important functionality for the client

is the Protobuf library again, because of the communication between server and client. CARLA prepared some example clients in the Python language. Due to this fact Python is the recommended language and is well accepted. CARLA provides a client library in Python, including utilities for sensor signal, transforming images, sending and receiving messages from the server like agent controls or player informations. After creating a client object from this library it's easy to establish a connection between the client application and the server. For example it's possible to define the number of vehicle or pedestrian agents, or define the weather conditions. In addition, the various sensors can be defined and positioned on the vehicle. If the client is started, it can be communicated using the selected TCP ports. So, it's possible to get the measurement values for each time step, process it and send control commands to the server again.

Of course it is possible to extend the functionality, sensors, communication between the client and server, and so on due to the used open source code.

4.4 Software Architectural Design

Figure 4.5 gives a short overview of the software architecture. As mentioned above, CARLA is divided in a server application and client application. In this section the architecture of the client will described in more detail. CARLA provides a Python library to create a TCP connection to receive the sent server informations, highlighted with the dark brown color.
The main class of the client, as the name indicates, is the client. The Client is the central node point and create a connection to the server with the help of the CARLA library. In the risk interpreter the whole intelligence of the risk assessment is implemented by using the risk assessment methods and the algorithms to find the time to collision, described in the section 3.2.
Each object class uses different methods to find the time to collision. By separating these methods in own classes, it is possible to connect each object with specified time to collision methods. Section 4.5 describes the different usage of these algorithms.
When the interpreter detects a critical situation, the client should be able to save the whole scene. The class recording provides this functionality. Once a critical situation is detected, the positions of the agents in the near past will saved in a list, as well as the positions of the player vehicle. Additionally this class saves the sensor data of the enabled camera sensors.

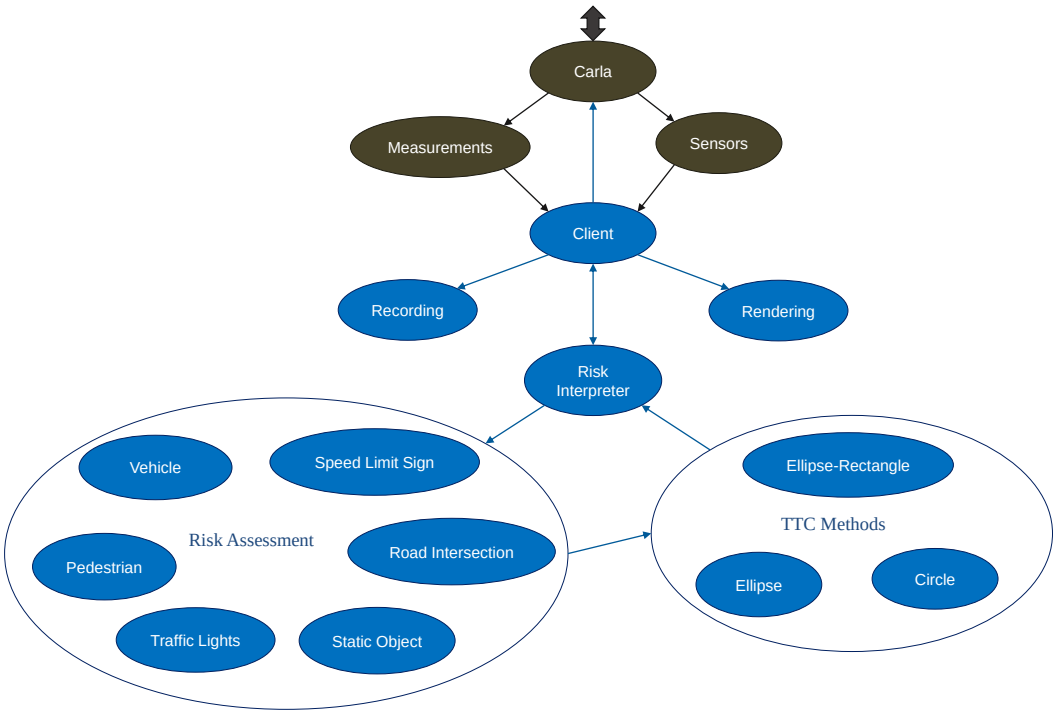


Figure 4.5: Software Architecture of the Client Interface.

Last but not least the rendering is important to give a graphical impact to the users. The rendered video shows the vehicle, it's actions and the environment in specified field of view range.

4.5 Implemented Methods for Risk Assessment

4.5.1 Used algorithms for different object classes

Not every method is suitable for every object class. In this thesis all methods are used. In the case of a detected object in the near range the ellipse method is used, because static objects, like vegetations or fences, must not have a defined shape to overlap them with a circle or rectangle. Testing every detecting point is the only way to find a time to collision.

In contrast to the static objects, the circle and the ellipse-rectangle methods are used to detect TTC for vehicles. Both techniques have advantages. We can use the simple and fast circle method to limit the time period of a collision and by using these restrictions the ellipse-rectangle method can be used. Figure 4.6 shows a combined method to find a TTC including both methods. For calculation

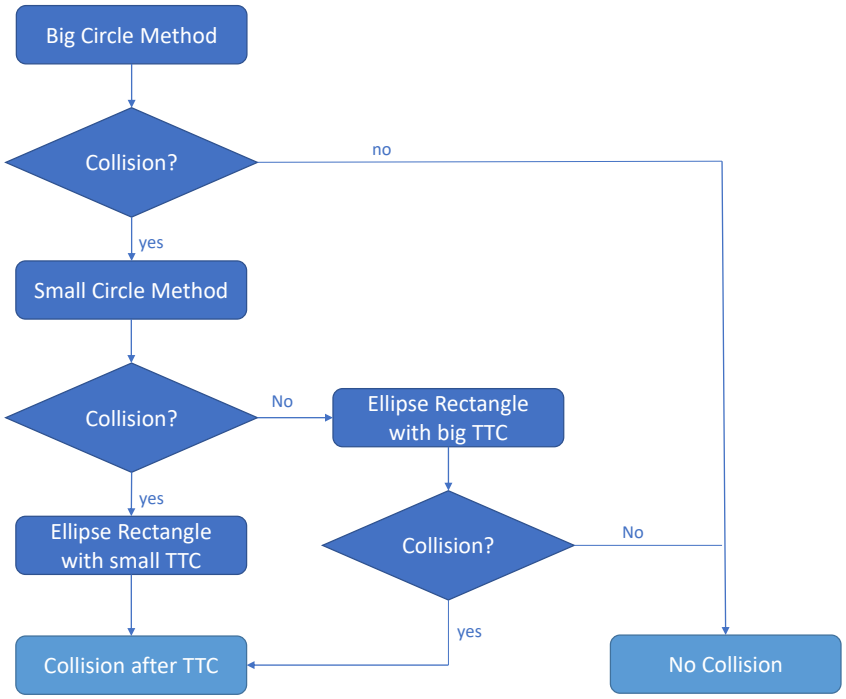


Figure 4.6: Combined method to find time to collision including Circle and ellipse-rectangle method.

the process is divided into several steps. First with the help of the circle method, a circle with a big radius will check if there is a collision in range or not. By using a small circle it's able to limit the time in most cases. So if there is a collision found with the circle method with a small radius the ellipse-rectangle method can be used and is limited within a collision time between the time of the big and the small circle method. This helps to speed up the algorithm and a valid TTC will be detected for sure. If there is no intersection with the small circle method the ellipse-rectangle method should be used in a wide range of time possibilities up to the time of the circle method with the big radius. It's possible to not find a collision, because the shapes are smaller than in the first detected big circle method. The result of the combined method is very exact and quite fast.

The used TTC detection methods for speed limit signs and traffic lights are the circle method and a line intersection calculation. First the circle method is used to filter out unaffected signs or lights. It is assumed that each sign or light approximately in the right angle to the car is shown. Due to this fact and by calculating the path of the car, a line intersection of path and a line in a right angle to the signs or lights will detect the next sign and it's TTC.

4.5.2 Various Accelerations

The vehicle is simulated with an ideal trajectory, known velocity and the acceleration of the own vehicle. Because static objects always stay in their place, we also know their velocities and accelerations, they are zero all the time. It is more difficult to define the accelerations of dynamic objects. We assume that we know the speed through ideal sensors. It is not possible to define a with 100% certainty correct acceleration. For this reason in this thesis a probability distribution is used. A Gaussian bell curve is used to define the forward acceleration with mean value 0 and a Gaussian bell curve for the rotation with mean value 0 again. These curves support to evaluate the occurrence of the change in the behaviour of the other vehicle or pedestrians. It is for example not very likely that a car drives on the opposite lane, but more likely that this car may slow down and keep the direction.

4.5.3 Risk Interpretation

By using these approaches all values for determining the current hazard value are given. In section 3.2 the risk assessment methods are presented. Each single risk value will be determined depending on the classified object.

By application of equation 4.1 the risk value for each detected static point in the near surrounding will be determined and the point with the highest risk will be used for determining the hazard level. The baseline time $t_{baseline}$ is equal to the stopping distance of the vehicle multiplied by a free selectable factor.

$$risk_{static} = 100 * (1 - \frac{TTC}{t_{baseline}}) \tag{4.1}$$

To determine the risk value of dynamic objects the TTC is important as well as the used acceleration for collision finding. In the thesis, the accelerations for each vehicle is varying and the acceleration with the highest risk value will be chosen and reused later in the risk assessment of the situation.

Hybrid object classes uses the TTC as an risk factor again. However, for each class, like speed limit sign a proper function will be used. For example calculating the risk value for speed limit sign will

be calculated applying equation 4.2. The first part will be determining the risk, that driving too fast in the next speed restriction and the second is determining the risk of driving to fast in the current location. The maximum value is defined with 100.

$$\begin{aligned} risk_{speedlimitsigns} =& (Velocity - Velocity_{SpeedLimit}) * (1 - \frac{TTC}{t_{baseline}}) + \\ & (Velocity - Velocity_{SpeedLimitNext}) * 3.6 \end{aligned} \tag{4.2}$$

Calculating the risk value of a traffic light, see 4.3, depends on the TTC time again and a factor T. The factor T is zero if the state of the traffic light is green, 50 if the state is yellow and T is 100 if the state of the traffic light is red. Thus, there is no danger from a green traffic light.

$$risk_{trafficlight} = T * (1 - \frac{TTC}{t_{baseline}}) \tag{4.3}$$

Now, the hazard level can be calculated with one of the provided methods using the determined risk values for each object. In the following chapter the advantages and disadvantages will discussed in more detail.

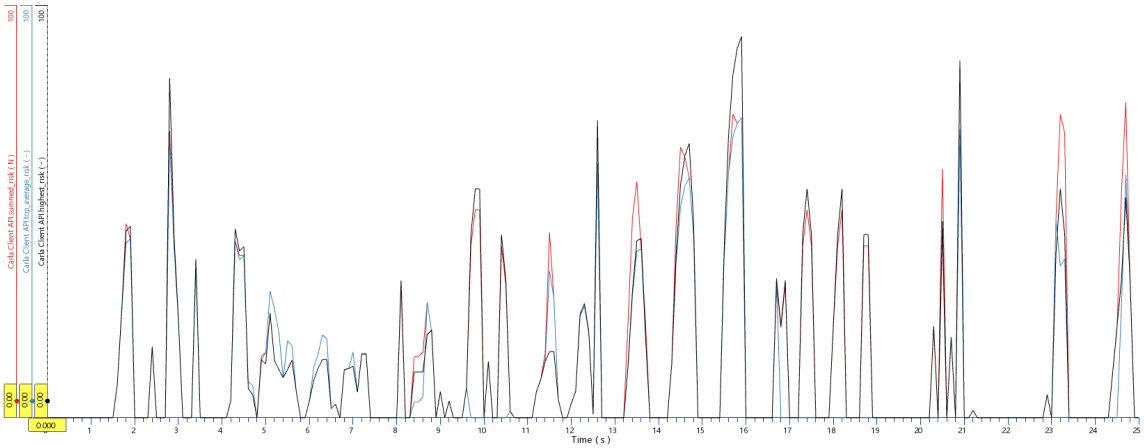
4.6 Critical Situations

With the help of this hazard level it's possible to define a critical situation. There are different types of critical situations. The "Highest Value"-function finds the situations, where exactly one dangerous object is, quite often. It's easy to define a good threshold for this function. In contrast, the top average function can be used to find a more general dangerous situation. It's possible to have a critical situation due to the heavy traffic density. With the summed function or top average function it's able to find it, but it's difficult to define a threshold.

Figure 4.7c demonstrates the output value of the different hazard level functions using the example controller. Due to the fact that the controller intelligence is the not well programmed the curve goes up and down, because of the bad dimensioned throttle values. The red, blue and grey coloured curves and bars in the figure 4.7 illustrates the main hazards level function, the "Summed Function

(a) Small ”highest risk”-value with bigger ”summed risk”-value.

(b) High risk to collide with pedestrians.



(c) Measured values of the hazard level functions within a test-run.

Figure 4.7: Measurements of test run with example controller.

(red)”, the ”Top Average Function (blue) and the ”Highest Value Function (grey)”. As shown in figure 4.7a it’s possible to have a situation with a high risk without having one potentially risky object. Figure 4.7b shows a critical situation with a high risk to collide with the pedestrians. More details of these situations are explained in the chapter below.

4.6.1 Software in the Loop - Model.CONNECT

Figure 4.8 shows the software in the Loop concept of this thesis. With the help of the tool Model.CONNECT from AVL List GmBH [3] and the including ICOS interface, a software in the loop can be provided. On the left side of the figure the Python client interface is modelled.

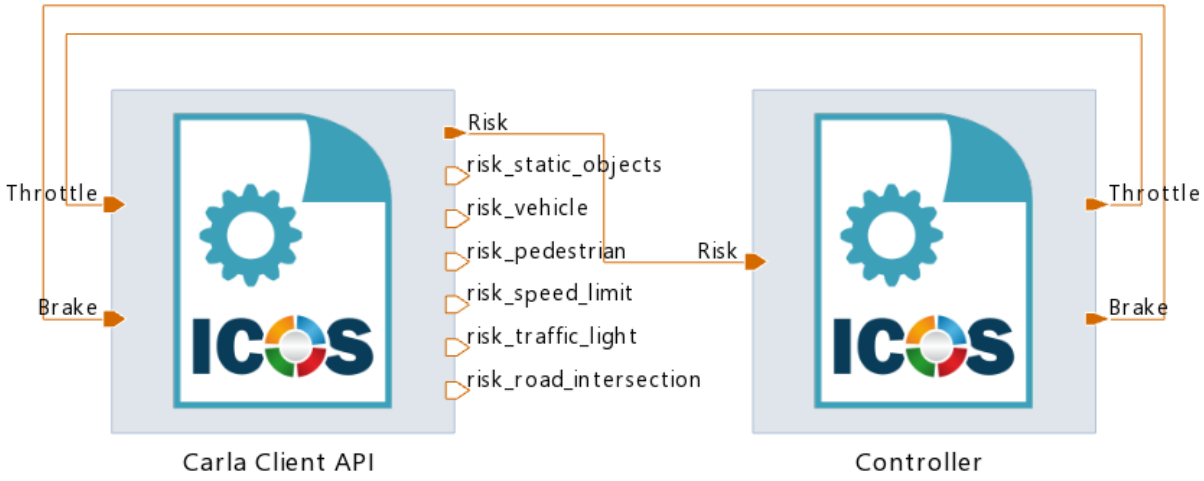


Figure 4.8: Software in the Loop Concept using AVLs Model.CONNECT and its ICOS interface.

In this python application the communication with the CARLA server is managed and the risk assessment is done. The Controller shows an example ADAS application, which takes control over the throttle and the brake. In this example the controller gives a throttle value of 0.8/1 if there is no risk, 0.4/1 if there is a hazard level higher than 30 and 0/1 if the risk is higher than 50. In the next section the results of using a controller like this will be shown and the different risk assessment methods will be presented.

5 Data analysis and Results

A summary of the main findings and results, together with the hazard level functions, is provided in this chapter. As mentioned in the previous chapters, in section 5.1 the three different cost functions are faced together. The structure and the results will be explained in more detail. A more detailed account of ASIL is given in section 5.2. In the section 5.2 the interrelation of critical situations, test cases and ASIL will be argued. The last part of this chapter describes in greater detail the problems and findings within modelling and simulation.

5.1 Comparison of Hazard Level Functions

Figure 5.1 compares the hazard level functions based on various situations. The most interesting aspect of the hazard level bars on the right of the figure are the different level values.

What stands out in a classic hazard situation of figure 5.1a is that the "Highest Value"-function is very high because of the potential colliding car highlighted in the blue circle. Also the level of the summed function is very high. In contrast, the top-average level is relatively low. The reason for it is, that there is only the other vehicle above the average risk limit and therefore included in the hazard level of the top-average function. The risk values of the static objects, the speeding and other risk values are ignored. In this situation all functions has detected the critical situation and a value higher than 75.

Next figure 5.1b on the right shows a similar situation with a crossing pedestrian. The highest value with the red highlighted circle pedestrian, is low in contrast to the other levels of the hazard functions. The levels of the other functions are higher, because of the car behind the pedestrian,

the static objects, like the street lamp and in addition for the summed value function the other pedestrians.

Turning now to the figure 5.1c the value of the summed function is again high. In this situation the level is too high because there is no real hazard. The reason for this is the high number of pedestrians in the near range of the vehicle. Every pedestrian is at a low risk, which adds up to a high value. Due to this fact the value of the summed function is too high. In comparison the top-average function has a much lower risk value, because of the filtering. Most of the pedestrians are too far away and their risk is too low to be included in the hazard level. Again the pedestrian within the red circle is the one with the highest risk value.

Figure 5.1d demonstrates that the highest value function does not detect every hazard function in an early stage. In this situation the crossing pedestrian has a risk value of about 55. The problem in this situation is the high amount of risky objects. This high number of other road users will make it difficult to prevent the accident. The only way is to slow down or endanger others. With the help of the top-average function this critical situation will be detected again as well as the summed function.

To confirm this experiences many test trial with defined parameters were simulated. Each parameter set was simulated 50 times for five minutes to receive a balanced data set.

In table 5.1, the average hazard level values over these 50 test runs are provided. The left two columns represents the defined number of vehicles and pedestrians. In order to get better results, the ”stop time” (time where velocity of the player is zero) of the was filtered out. The reason for this is that there is no danger in this time by the own driving behaviour. This filtered data set was used to calculate the average velocity of the players car. In the right half of the table the average and the maximum hazard level of the three different function types are described. Closer inspection of the table shows the average velocity decreases with an increasing number of vehicles. The autopilot of CARLA is driving with 10 km/h slower than the speed limit. Due to that reason the average velocity of 7 km/h is quite okay and the decreasing speed is because of the higher traffic density. Figure 5.2a demonstrates the the rising average hazard level when the number of vehicles gets higher. The maximum hazard level of the ”Summed Value” and the ”Top Average” function is nearly constant. In contrast to it, the hazard level of the ”Highest Value” function is falling with a higher number of vehicles. This result is somewhat counterintuitive. The reason is that the player

(a) Critical situation detected by all hazard level functions.

(b) Critical situation with a high summed function.

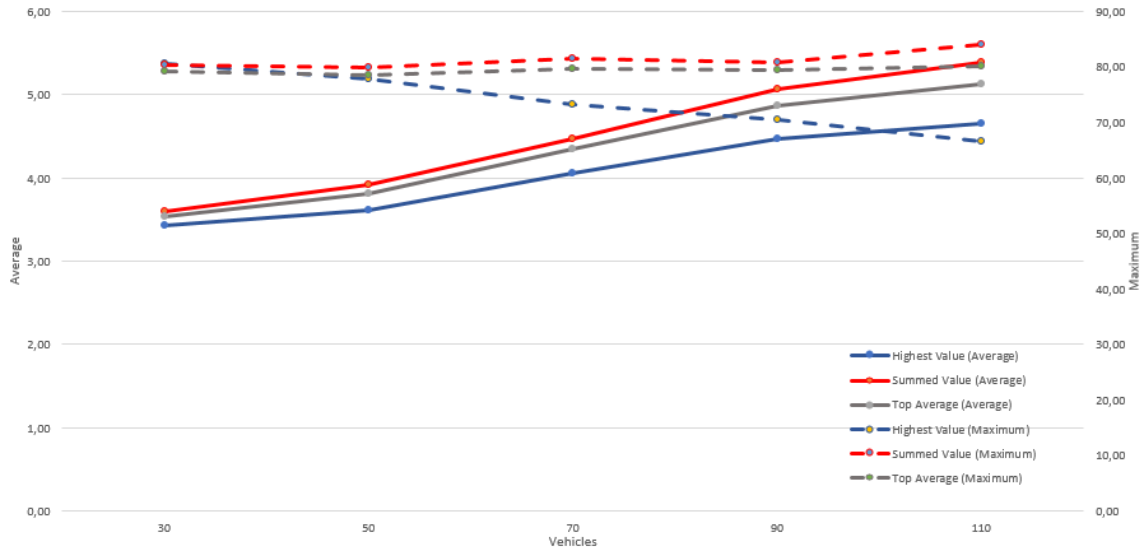
(c) No critical situation detected by the a high summed function.

(d) Critical situation not detected by the highest value function.

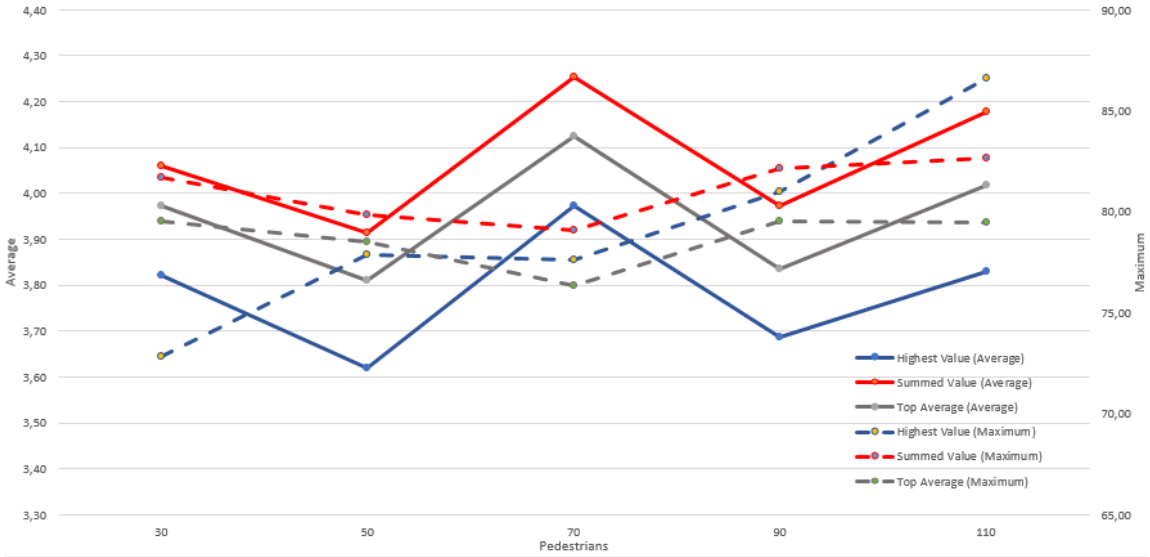
Figure 5.1: Hazard level measurements with highest value (grey), top-average (blue) and summed (red) function.

vehicle drives automatically slower on a danger situation. As a result, the car reaches speeds less frequently, which leads to a high risk situation.
In the sense of this, an increasing number of pedestrians causes no traffic jam behaviour. Figure 5.2b represents the average hazard level depending on the number of pedestrians, which is nearly constant. Interestingly, the level of the ”Summed Value” function is not increasing. As expected, the average hazard level of this function would have had to increase with the number of pedestrians, because of the case described above, see figure 5.1c. After further analysis, it was found that a situation like this rarely occurred in the simulation test runs. The pedestrians are randomly generated and also move randomly, although on the pavement but rarely together in groups. The rising maximum value of the ”Highest Value” function was expected because of the more often crossing pedestrians, which causes a high risk value.
For the sake of completeness figure 5.2c represents the hazard level of the different function types with various additional throttle. 50 pedestrians and 50 vehicle agents are set in these runs. In contrast to a modified number of agents the throttle value of the autopilot was changed. When the player vehicle is moving 10%, 20% or 30% was added to the throttle value of the autopilot in this test runs. The braking system and when the autopilot stops or the throttle value is about zero, no additional throttle was added. With the help of this settings the behaviour of a bad car driver could be simulated. The average hazard level of all function types are very high, and increasing with the additional throttle value. The results data is able in table 5.2c.
In all figures the hazard level of the ”Top Average” function is lower than the ”Summed Value” function. This is in fact of the filtered low risk objects. Unfortunately, due to the lack of grouped of pedestrians, it was not possible to prove the strength of the ”Top Average” function, but you can see that this function is constantly delivering good results.

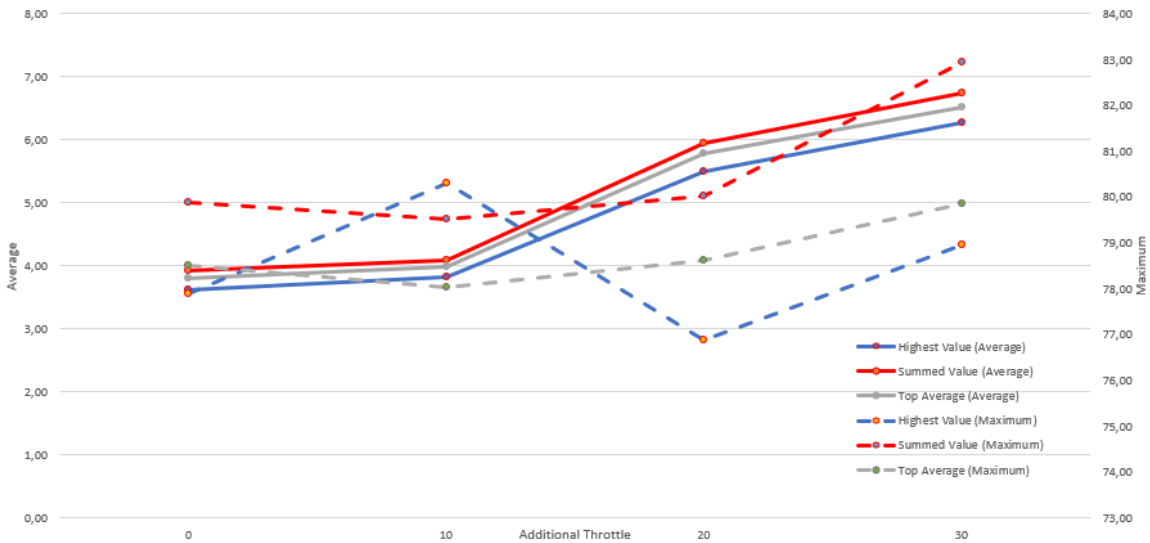
Table 5.3 provides the summary statistics for the different test runs and found critical situations. The table is separated in the three classes of hazard level functions. Next the number of critical situations depending on the number of agents and the defined level presented. Each row of the table represents the average values of 50 test runs. The calculation of the hazard level is done after 0.05 seconds. Due to this fact one critical situation listed in the columns often counts more than one. The single most striking observation to emerge from the data comparison was that the number of critical situation are decreasing with a higher number of agents. As discussed above, this is due to the traffic density and of course, a small factor is the greater computational effort delayed or less



(a) Various number of Vehicles.



(b) Various number of Vehicles.



(c) Various additional throttle values during driving.

Figure 5.2: Average and maximum hazard level values depending on the number of agents and type of hazard level function.

Table 5.1: Average and maximum hazard level values depending on the number of agents and type of hazard level function.

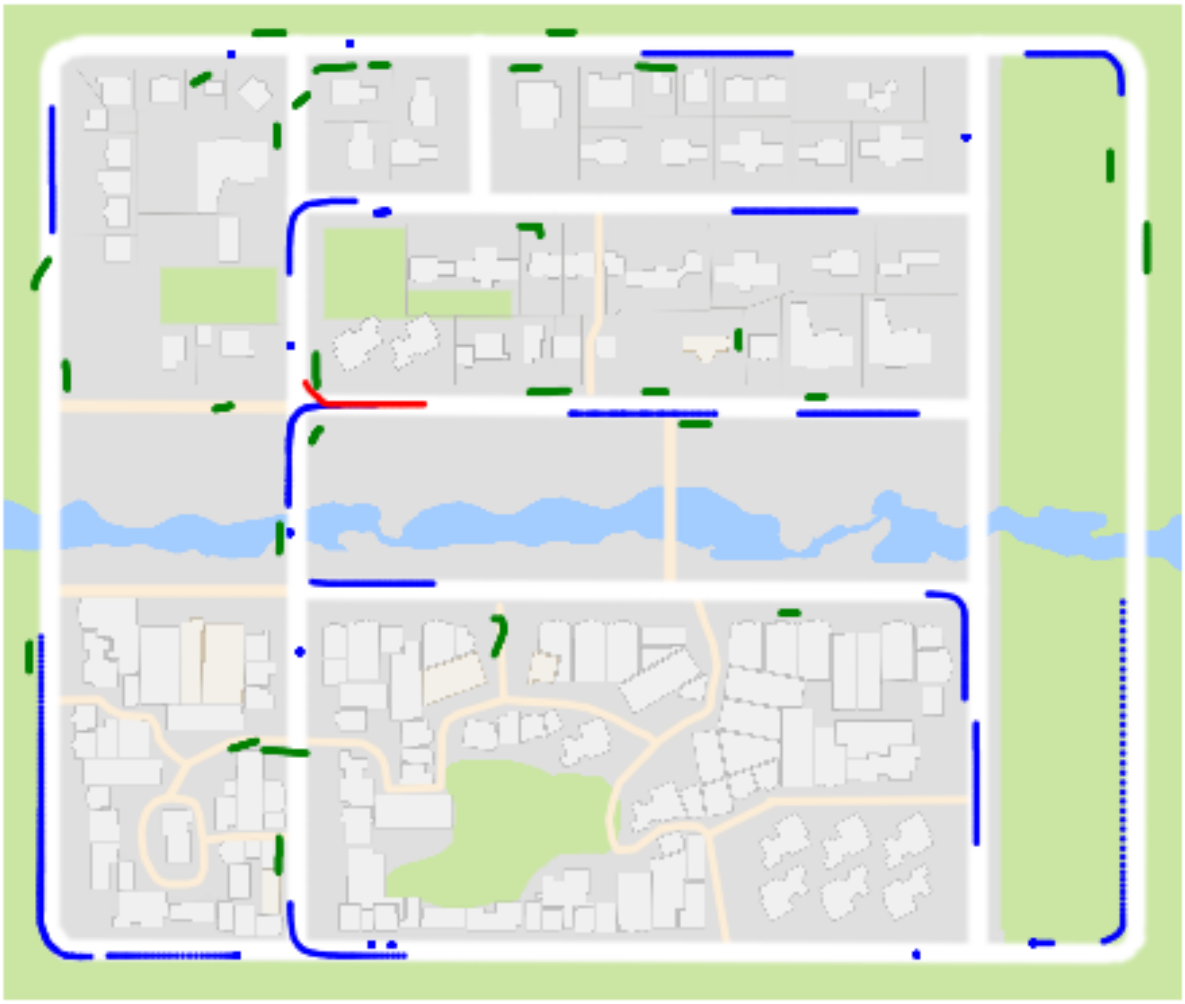
Vehicle	Pedestrian	Velocity	Highest		Summed		Top Average	
#	#	∅ km/h	∅	Max	∅	Max	∅	Max
30	50	7.02	3.43	80.56	3.59	80.29	3.54	79.18
50	50	6.57	3.62	77.89	3.91	79.87	3.81	78.50
70	50	6.52	4.05	73.17	4.48	81.44	4.34	79.59
90	50	6.69	4.47	70.45	5.07	80.75	4.87	79.53
110	50	6.23	4.66	66.65	5.39	84.08	5.13	80.16
50	30	6.54	3.82	72.84	4.06	81.73	3.97	79.56
50	70	6.54	3.97	77.65	4.25	79.07	4.12	76.33
50	90	6.57	3.69	80.98	3.97	82.15	3.83	79.57
50	110	6.44	3.83	86.60	4.18	82.67	4.02	79.46

computation steps.

Table 5.4 represents similar simulation runs using different additional throttle values with fixed number of agents, 50 pedestrians and 50 vehicles. As expected increases the number of found critical situations with the increasing forward acceleration of the player vehicle.

5.2 In a Loop Testing and relation to ASIL

As explained in section 3.2, it is clear that a critical situation is a good test case. The results from the section above, show that a well defined hazard level function is the first step to a well defined test case. With the help of a buffer and the sensor signal the last 50 time steps of the simulation are saved. When a triggered value reaches a value, for example if the top-average level is higher than 75, then the buffer will be saved. Figure 5.3a provides a short overview of the last 50 positions of the player position, the vehicles and pedestrians before the critical situation starts. The figures below 5.3b and 5.3c shows the rgb camera pictures 15 time steps before and when the critical situation was detected. In Addition all relevant information, like risk values of each object class or player position will saved.



(a) Last 50 positions of vehicles and pedestrians before the critical situation.

(b) 15 time steps before critical situation was detected.

(c) Detected critical situation (Collision with waiting vehicle & too fast).

Figure 5.3: Saved sensor data and measurements of a simulation run.

Table 5.2: Average and maximum hazard level values depending on various additional throttle values and type of hazard level function.

Additional Throttle	Velocity	Highest		Summed		Top Average	
%	∅	∅	Max	∅	Max	∅	Max
10	6.74	3.81	80.30	4.08	79.52	3.98	78.02
20	7.14	5.50	76.89	5.94	80.01	5.78	78.62
30	7.05	6.26	78.95	6.73	82.94	6.51	79.86
40	7.27	4.91	78.13	5.31	79.74	5.15	78.91

By using this informations it's easy to create a test case. All relevant information to rerun the same situation exists or can be calculated, for example velocity and throttle value can be readjusted without much costs. Now every imaginable test case can be adjusted. Thus, the ASIL requirements are met and tested. If, the severity S of the ASIL will be included in the weights of the hazard level function, for example weight an accident with a pedestrian with 0.7 and a collision with a static object only with 0.2, then it can be roughly looked for critical situations on the basis of ASIL. The calculation with the help of the hazard levels corresponds to the probability of an accident and the weights corresponds the severity.

As was pointed out in the previous chapter testing advanced driver-assistance systems is possible with the software in the loop method, discussed in section 4.6.1. The loop is designed using Model.CONNECT, like the example shown in figure 4.8. In loops like this the interaction between vehicle controller and environment can simulated quite well. Figure 4.7 shows the results of a loop with an example controller. Closer inspection of the figure shows the step wise increasing and decreasing hazard levels. The reason for this behaviour is the bad controller which gives more throttle when less risk exists and visa versa. Normally the risk level should only be used for the purpose of controlling and not serve as an aid to the controller. A future controller should only use sensor measurements and calculate an independent control mechanism.

Table 5.3: Number of found critical situations depending on the level threshold, number of agents and hazard level function.

Function Type	Vehicle	Pedestrian	# Critical Situation				
-	#	#	>40	>50	>60	>70	>80
Highest Value	30	50	38.52	22.90	13.70	7.78	1.78
Highest Value	50	50	26.90	13.84	7.26	4.06	1.44
Highest Value	70	50	24.30	10.92	4.64	2.46	0.72
Highest Value	90	50	21.26	8.56	3.82	1.78	0.30
Highest Value	110	50	24.10	8.78	3.44	2.14	1.36
Highest Value	50	30	32.04	15.84	7.74	3.84	0.90
Highest Value	50	70	28.21	13.26	6.03	3.06	1.53
Highest Value	50	90	24.48	11.66	6.30	3.70	1.72
Highest Value	50	110	17.93	6.63	1.93	0.93	0.30
Summed Value	30	50	45.02	27.36	17.76	11.02	2.14
Summed Value	50	50	36.08	21.58	12.48	7.28	1.70
Summed Value	70	50	35.10	20.54	12.22	6.76	2.04
Summed Value	90	50	32.78	19.60	11.18	5.94	1.80
Summed Value	110	50	39.98	22.56	12.48	6.88	2.12
Summed Value	50	30	39.92	23.20	12.70	7.24	1.70
Summed Value	50	70	36.38	20.71	11.26	6.12	1.26
Summed Value	50	90	33.04	18.50	11.12	6.56	1.70
Summed Value	50	110	28.93	17.22	9.59	5.04	2.00
Top Average	30	50	43.74	25.80	16.76	10.58	2.02
Top Average	50	50	34.12	19.60	11.42	6.56	1.36
Top Average	70	50	32.50	18.14	10.82	5.70	1.46
Top Average	90	50	29.64	16.76	9.56	5.16	1.56
Top Average	110	50	25.63	13.89	7.52	3.78	1.33
Top Average	50	30	37.84	21.00	11.56	6.54	1.34
Top Average	50	70	33.53	17.50	9.35	5.15	0.74
Top Average	50	90	29.94	16.06	9.46	5.82	1.18
Top Average	50	110	34.28	19.60	13.02	8.22	1.86

Table 5.4: Number of found critical situations depending on the level threshold, various additional throttle value and hazard level function.

Function Type	Additional Throttle	# Critical Situation				
-	%	>40	>50	>60	>70	>80
Highest Value	10	28.50	15.00	7.52	4.04	1.30
Highest Value	20	37.14	17.66	7.34	3.68	1.10
Highest Value	30	40.02	18.78	9.16	4.66	1.50
Highest Value	40	34.90	18.06	8.44	5.32	2.78
Summed Value	10	37.20	21.20	12.22	7.46	1.94
Summed Value	20	50.48	27.56	11.88	6.54	1.08
Summed Value	30	53.30	30.14	15.26	8.34	2.28
Summed Value	40	46.08	26.18	12.64	7.94	1.84
Top Average	10	34.98	19.12	11.04	6.72	1.52
Top Average	20	46.84	25.16	10.38	6.04	0.92
Top Average	30	48.36	26.52	13.28	7.28	1.64
Top Average	40	42.76	23.66	11.58	7.22	1.64

6 Conclusion

In the introduction the importance of the safety factor in the development of self driving car was motivated. It shows how necessary a basic understanding of functional safety of assistance system components is. With the help of the functional safety standards and the described V-Model the thesis gives enough background information to understand the requirement specified by the ISO 26262 to develop a safe assistance system for vehicles. In addition to the safety requirements, the advantages of a simulation environment are described. The aim of the presented research to the V-Model and the simulation was to examine the modelling of a test environment for ADAS. The discussed modelling process provides a guideline for the development of such a test system based on a simulation. In the practical part of the work, the V-Model was applied and a system was created based on this modelling process. It was developed step by step according to this model. First, it was important to define the requirements to filter out the correct environment using the defined validation options. Gradually, it has become clear that this approach has greatly simplified the development of the test system. The simulation environment based on CARLA is working quite fine for the purpose of this thesis.

The initial objective of the project was to identify critical situations to support the development of advanced driver-assistance systems.
In this study critical situations were found with so called hazard level functions. The previous chapters have shown some differences between the different hazard level functions. Three different cost functions are provided to define the hazard level. First the ”Highest Value”-function selects the object with the highest risk value. Contrary to exceptions, this study did not find a significant scenario where the hazard level of the ”Highest value”-function is totally wrong. The most obvious finding to emerge from the results is that the risk value is too low if there are many traffic participants. For example when there is a crowd of people going along the sidewalk, the risk should often be higher than with the function, because the probability of unexpected behaviour of at least

one of the people is very high.
In contrast the ”Summed Value”-function sums up all the risk values of all objects. This leads as expected to huge value differences. By applying equation 3.6 it is possible to limit these values between 0 and 100. However, the definition of the parameter B and the hazard limit value is difficult to define. Previous case for example leads to a high value because of the sum of many traffic participants. Therefore, the value quickly becomes too high. On the other hand if the parameter is defined for situations like this, the hazard value for other situations, like a crossing pedestrian is too low. Due to this fact using this function should only be used carefully in special situations.
The difference between the levels of the ”Summed Value” and the ”Top Average”-functions was significant. By filtering out the smallest risks, the value gap between the situations becomes smaller. This in turn leads to a simpler definition of the factor B of 3.6. Therefore, the examples above can handle both. A group of people along a sidewalk can be better processed. Only the most dangerous people from this group will be handled and this leads to a good result. When evaluating the situation of the crossing pedestrian the ”Top Average”-function will lead to a value which is good again.
Taken together, these results suggest that the ”Top Average”-function is working quite well in all situations.

With the help of the implemented ”Software in the Loop” concept, it is possible to run the complete functionality of a real vehicle, including all assistance components in the simulation. The developed system is able to run a nearly realistic simulation of a urban environment with the ADAS vehicle as the main actor. Today, the vehicle is using a near perfect autopilot, provided by CARLA, to find critical situations, which can replaced with ADAS technology if wanted. Already, the risk assessment methods were able to detect a set of critical situations, which were recorded and maybe will saved in the future for developing new or better assistant system for vehicles.

The investigation of finding test cases to fulfil the ASIL recommended testing parts is doing quite well. With the software in the loop method the simulation is able to test ADAS until the requirements are reached. However, these test cases alone are not sufficient to meet all the requirements based on ISO 26262 of the system. They can only be used as an accessory or for finding additional cases for unit or integration testing. If the debate is to be moved forward, the adjustment of the found test cases has to be developed. To achieve this, the agent must be controlled by the saved trajectories and started at a specific point. At some point, it may be possible to replace certain classic tests with

this method and raise risk assessment based on a simulation environment to the level of safety that fulfil the requirements of ISO 26262.

Despite these promising results, questions remain. An issue that was not addressed in this study was the inaccuracies of the sensor measurements as well as the positions. This is an important issue for future research and with the help of the CARLA simulator it should possible to solve this and simulate more realistic sensor data. In future investigations, it might be possible to use ”not ideal”-driving agents instead of using a probability distribution within hazard level functions. Whether this approach also produces good or even better results has to be checked on the basis of further studies.

Although this study focuses on real time risk assessment. It can be used for machine learning as well. On the one hand the hazard level can be included in learning a complete computational intelligence to drive an autonomous vehicle. On the other hand the properties and parameters, like time to collision can be used as input parameters for a computational intelligence which defines again a hazard value, for example for a warning signal to the driver if there is a critical situation.

Bibliography

[1] National Highway Traffic Safety Administration. *Automated Vehicles for Safety*. 2018. URL: https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety (cit. on p. 2).

[2] Jerry Albright et al. "Marketplace of change: Automobile insurance in the era of autonomous vehicles." In: October (2015), p. 52. URL: https://assets.kpmg.com/content/dam/kpmg/pdf/2016/06/id-market-place-of-change-automobile-insurance-in-the-era-of-autonomous-vehicles.pdf (cit. on p. 17).

[3] AVL List GmBH. *AVL - Development, testing & simulation of powertrain systems - avl.com*. 2012. URL: www.avl.com (visited on 08/22/2018) (cit. on pp. 38, 58).

[4] Rahul C. Basole and Douglas A. Bodner. *Modeling and Simulation in the Systems Engineering Life Cycle*. 2015, p. 369. ISBN: 9781447156338. DOI: 10.1007/978-1-4471-5634-5_10. URL: http://link.springer.com/10.1007/978-1-4471-5634-5%7B%5C_%7D10 (cit. on p. 34).

[5] Mayer R Benjamin P, Patki M. "Using ontologies for simulation modeling." In: *Proceedings of the winter simulation* (2006) (cit. on p. 36).

[6] R Bott. *Guide to Modeling and Simulation of Systems of Systems*. 1. 2014, pp. 1–5. ISBN: 9780874216561. DOI: 10.1007/s13398-014-0173-7.2. arXiv: arXiv:1011.1669v3 (cit. on p. 34).

[7] A. Cimatti et al. "From Informal Requirements to Property-Driven Formal Validation." In: *Formal Methods for Industrial Critical Systems* (2009), pp. 166–181 (cit. on p. 19).

[8] R Cimperman. *UAT Defined: A Guide to Practical User Acceptance Testing (Digital Short Cut)*. 2006 (cit. on p. 25).

[9] Martin Horn Daniel Watzenig. *Automated Driving: Safer and More Efficient Future Driving*. 2016 (cit. on p. 18).

[10] Alexey Dosovitskiy et al. *CARLA - Documentation*. URL: https://carla.readthedocs.io/en/stable/carla%7B%5C_%7Dserver/ (visited on 09/04/2018) (cit. on p. 52).

[11] Alexey Dosovitskiy et al. "CARLA: An Open Urban Driving Simulator." In: CoRL (2017), pp. 1–16. ISSN: 1938-7228. arXiv: 1711.03938. URL: http://arxiv.org/abs/1711.03938 (cit. on pp. 49, 51).

[12] John C Hayward. "Near-miss determination through use of a scale of danger." In: *Highway Res. Rec.* 384 (1972), pp. 22–34. DOI: TTSC7115 (cit. on p. 26).

[13] Jia Hou, George F. List, and Xiucheng Guo. "New algorithms for computing the time-to-collision in freeway traffic simulation models." In: *Computational Intelligence and Neuroscience* 2014 (2014). ISSN: 16875273. DOI: 10.1155/2014/761047 (cit. on pp. 28, 30).

[14] International Electrotechnical Commission. *IEC 61508: Functional safety of electrical/electronic/programmabl electronic safety-related systems*. 2010 (cit. on p. 9).

[15] International Organization for Standardization. *ISO 26262: Road vehicles – Functional safety*. 2011 (cit. on pp. 11, 13–15).

[16] Sofia Kockum et al. "Volvo Trucks Safety Report 2017." In: (2017). URL: http://www.volvogroup.com/content/dam/volvo/volvo-group/markets/global/en-en/about-us/traffic-safety/Safety-report-2017-0505.pdf (cit. on pp. 2, 9).

[17] G. Birta Louis and Gilbert Arbez. *Modelling and simulation*. Vol. 98. 2-3. 1997, pp. 245–247. ISBN: 0750308435. DOI: 10.1016/S0304-3800(96)01916-3. URL: http://linkinghub.elsevier.com/retrieve/pii/S0304380096019163 (cit. on p. 34).

[18] Balci Osman. "A life cycle for modeling and simulation." In: *Simul Trans Soc Model Simul Int* (2012), pp. 870–883 (cit. on p. 35).

[19] Reactis. "Achieving ISO 26262 Compliance with Reactis : ISO 26262 Tasks Sup ... Achieving ISO 26262 Compliance with Reactis : ISO 26262 Tasks Sup ..." In: (2014), pp. 1–6 (cit. on p. 23).

[20] Stephan Höppner Reinhard Höhn. *Das V-Modell XT: Grundlagen, Methodik und Anwendungen*. 2008 (cit. on p. 17).

[21] Chris Rupp and SOPHISTen. *Requirements- Engineering und -Management*. 2014 (cit. on p. 45).

[22] SAE International. *ARP4754: Aerospace Recommended Practice (ARP) - Guidelines For Development Of Civil Aircraft and Systems*. 2010 (cit. on p. 15).

[23] SAE International. "U.S. Department of Transportation's New Policy on Automated Vehicles Adopts SAE International's Levels of Automation for Defining Driving Automation in On-Road Motor Vehicles." In: *SAE international* (2016), p. 1. DOI: P141661. URL: https://www.sae.org/news/3544/ (cit. on p. 2).

[24] R. G. Sargent. *An Assessment Procedure and a Set of Criteria for Use in the Evaluation of Computerized Models and Computer-Based Modeling Tools*. Tech. rep. 1981 (cit. on p. 36).

[25] Robert G. Sargent. "Verification and validation of simulation models." In: *Proceedings of the 2011 Winter Simulation Conference* (2011), pp. 2194–2205. ISSN: 08917736. DOI: 10.1109/WSC.2011.6148117. arXiv: arXiv:1105.4823v1 (cit. on pp. 35, 37).

[26] Shital Shah et al. "AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles." In: (2017), pp. 1–14. ISSN: 1938-7228. DOI: 10.1007/978-3-319-67361-5_40. arXiv: 1705.05065. URL: http://arxiv.org/abs/1705.05065 (cit. on pp. 48, 49).

[27] VIRES Simulationstechnologie GmbH. *OpenCRG*. URL: www.opencrg.org (visited on 09/04/2018) (cit. on p. 48).

[28] VIRES Simulationstechnologie GmbH. *Virtual Test Drive - Complete tool-chain for driving simulation applications*. URL: http://www.mscsoftware.com/product/virtual-test-drive (visited on 09/04/2018) (cit. on p. 48).

[29] Fredrik Warg and Viacheslav Izosimov. "Computer Safety, Reliability, and Security." In: 9923.September 2016 (2016). DOI: 10.1007/978-3-319-45480-1. URL: http://link.springer.com/10.1007/978-3-319-45480-1 (cit. on pp. 20, 21).

[30] Elizabeth T. Whitaker. "Agent-Based Simulation." In: *Modeling and Simulation in the Systems Engineering Life Cycle, Simulation Foundations, Methods and Applications* (2015) (cit. on p. 39).