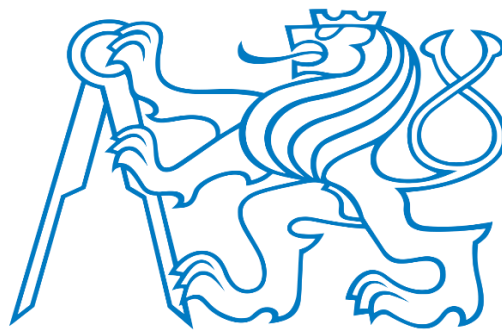


Czech Technical University in Prague
Faculty of Mechanical Engineering

Department of Automotive, Combustion Engine and Railway Engineering
Study program: Master of Automotive Engineering
Field of study: Advanced Powertrains



**Optimisation of ADAS - AD
Validation Process**

DIPLOMA THESIS

Author: Danielle FOTSO

Industrial supervisor: Mr. Freddy TIOGUIM

Academic supervisor: Ing. Václav JIROVSKÝ, Ph.D

Year: 2019



MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Fotso Fokam Danielle Audrey** Personal ID number: **481839**
Faculty / Institute: **Faculty of Mechanical Engineering**
Department / Institute: **Department of Automotive, Combustion Engine and Railway Engineering**
Study program: **Master of Automotive Engineering**
Branch of study: **Advanced Powertrains**

II. Master's thesis details

Master's thesis title in English:

Automated ADAS test development

Master's thesis title in Czech:

Automatické generování testů pro validaci systémů ADAS

Guidelines:

Improve the actual ADAS validation tool used at PSA and develop a new method for automated generation of test scenario for a defined ADAS system. All generated test cases have to cover the majority of the situations influencing ADAS functionality. Based on a set of parameters influencing the ADAS (i.e.: weather, luminosity, road structure, vehicle speed, pedestrian etc.) define relevant Markov Chains and apply them as basis for automated test scenario development. If possible, test the final results on a HIL bench. Use the most relevant software tool for development the algorithm (i.e. Python, Matlab).

Bibliography / sources:

Olivares, S.P. et al.: Virtual Stochastic Testing of Advanced Driver Assistance Systems, ISBN 978-3-319-20855-8
Bremaud, P.: Markov Chains - Gibbs Fields, Monte Carlo Simulation, and Queues, ISBN 978-1-4757-3124-8


Name and workplace of master's thesis supervisor:


Ing. Václav Jírovský, Ph.D., 16123


Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **02.04.2019** Deadline for master's thesis submission: **19.08.2019**

Assignment valid until: _____


Ing. Václav Jírovský, Ph.D.
Supervisor's signature


doc. Ing. Oldřich Vlček, Ph.D.
Head of department's signature


prof. Ing. Michael Valášek, DrSc.
Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

09/04/2019
Date of assignment receipt


Student's signature

DISCLAIMER

I, Danielle FOTSO declare that this thesis is my own work; it contains no material, which has been accepted or submitted for the award of any other degree or diploma. I also declare that all the software used to solve this thesis are legal.

This thesis contains no material previously published or written by any other person except where due reference is made in the text of the thesis.

In Paris, 16th August 2019

Danielle FOTSO

ACKNOWLEDGMENTS

This document is the fruit of the combined efforts of several persons who contributed either directly or indirectly to its elaboration. It is therefore with sincere and heartfelt gratitude that I thank:

My enterprise supervisor **Freddy TIOGUIM** (Technical ADAS service referent of PSA): it is truly a great pleasure for me to openly express my gratitude for his availability, his patience, his understanding, his guidance and high quality suggestions throughout this work.

My academic supervisor **Ing. Václav JIROVSKÝ, Ph.D**, who despite the distance has followed my work and given me orientation.

Ing. Yvana TCHANGOM, PSA employee, who helps me to develop my skills in the ADAS validation sector that was new for me. I thank her for the good follow-up of my activities and advices.

I will end, by thanking all the CVAD employees, for the very friendly working environment and their support and my beloved family in Cameroon, for the efforts they made to assist me.

ABSTRACT

Ensuring or guaranteeing the safety of autonomous systems remains one of the major challenges for car manufacturers, especially as the technologies embedded in these systems become more and more complex. As a result, test methods to validate them and thus guarantee their safety must be the most efficient and effective. It is in this order of thought that I was entrusted with the subject relating to the "**optimisation of the ADAS - AD validation process**". My work was on the automatic generation of test scenarios for the validation of ADAS functions, as well as the improvement of the tool for post-processing test results. A stochastic approach using the Markov Chain Monte Carlo algorithm has been used for scenario generation, based on a defined set of parameters that could influence the components according to a specific function and their probabilities of occurrence.

Keywords: Autonomous System Safety, ADAS Validation, Scenarios, Markov Chain Monte Carlo.

TABLE OF CONTENTS

DISCLAIMER	I
ACKNOWLEDGMENTS	II
ABSTRACT	III
TABLE OF CONTENTS	IV
LIST OF FIGURES	VII
LIST OF TABLES	VIII
GLOSSARY	IX
DEFINITIONS OF SOME TERMS [1]:	X
GENERAL INTRODUCTION	11
CHAPTER 1: CONTEXT AND PROBLEM STATEMENT	12
I PRESENTATION OF PSA	12
I.1 Different brands	12
I.1.1 Peugeot	12
I.1.2 Citroen	13
I.1.3 DS Automobile	13
I.1.4 Opel	13
I.1.5 Vauxhall.....	14
I.1.6 Free2Move.....	14
I.2 Presentation of my service	15
I.3 Roadmap of PSA for autonomous driving.....	15
II CHALLENGES OF PSA CONCERNING AUTONOMOUS VEHICLE AND ITS SAFETY	16
II.1 Standards regulating the autonomous vehicle and its safety.....	16
II.1.1 Norm ISO 26262	16
II.1.2 PAS SOTIF (Safety of the Intended Functionality)	18
II.2 Challenges related to the validation of the safety of autonomous driving.....	19
II.2.1 How does the ADAS validation is done actually at PSA	19
II.3 Problem formulation	23

II.4	Contribution and outline	23
CHAPTER 2: STATE OF ART OF AUTONOMOUS DRIVING	25	
I	DRIVER ASSISTANCE AND AUTONOMOUS DRIVING	25
I.1	What is an autonomous vehicle?.....	25
I.1.1	Environment perception layer	26
I.2	Levels of driving automation	27
II	TESTING OF AUTONOMOUS DRIVING SYSTEMS.....	29
II.1	Scenario for the test	30
II.1.1	Definitions	30
II.1.2	Source of test scenarios	31
II.2	Different testing levels	32
III	MARKOV CHAINS MONTE CARLO STOCHASTIC SIMULATION.....	34
III.1	Markov chains	34
III.1.1	Stationary distribution.....	34
III.1.2	Convergence properties.....	35
III.2	MCMC algorithms	35
III.2.1	Metropolis Hastings algorithm	35
III.2.2	Gibbs sampler	36
CHAPTER 3: METHODOLOGY OF THE WORK.....	37	
I	APPROACH FOR TEST CASES GENERATION	37
I.1	Definitions of parameters that influence an ADAS function.....	38
I.1.1	ODD taxonomy.....	38
I.1.2	Equivalence class classification.....	39
I.1.3	Ego vehicle manoeuver behaviours	41
I.2	Dependence between parameters	41
I.2.1	Likelihood of occurrence	42
I.3	MCMC method choice and application	43
I.3.1	Scenario mathematic form.....	43

I.3.2	Choice of the MCMC method and application.....	44
I.4	Convergence diagnostic	46
II	IMPROVEMENT OF THE POST PROCESSING VALIDATION TOOL NODESAT.....	47
CHAPTER 4:	PRESENTATION OF THE RESULTS AND DISCUSSION	53
I	CREATION OF INPUT DATA	53
II	SCENARIOS GENERATIONS	57
III	TEST CASES GENERATION	62
IV	DISCUSSION	64
IV.1	Limitation of the approach	64
IV.2	How our results will be used practically for the tests?.....	64
IV.3	How confident we are about the coverage of all possible requirements that the ADAS function must satisfy?.....	66
GENERAL CONCLUSION AND PERSPECTIVES		67
I	Contribution	67
II	Assignment.....	67
III	Perspectives.....	68
REFERENCES		69
APPENDICES		71
I	Appendix 1: ASIL levels [1].....	71
II	Appendix 2: Parameters classification	72
III	Appendix 3: Algorithm of method 2.....	73

LIST OF FIGURES

Figure 1: CVAD organisation chart	15
Figure 2: Roadmap of PSA for vehicle automation	16
Figure 3: Different ASIL level	17
Figure 4: Classification of type of scenario that a vehicle could face.....	18
Figure 5: Example of requirement for the camera	19
Figure 6: Validation process of the CVAD entity.....	20
Figure 7: description file of PSA	21
Figure 8: Carmaker interface.....	22
Figure 9: Validation process at CVAD	22
Figure 10: Global architecture of an autonomous vehicle	25
Figure 11: some ADAS sensors	26
Figure 12: Level of automation from the SAE J3016 standard	28
Figure 13: Procedure for test concept	29
Figure 14: Overview of the development process proposed in the ISO 26262 standard. Process steps highlighted in red may use scenarios to generate the work products.....	30
Figure 15: Illustration of a scenario representation (a) and a scene (b)	31
Figure 16: Generic V-Model.....	32
Figure 17: Post for a HIL validation at PSA	33
Figure 18: Methodology principle.....	37
Figure 19: ODD classification framework with top-Level categories and immediate subcategories	38
Figure 20: (a) Rural road; (b) Barrier and temporary cones	39
Figure 21: (a) Limited visibility Heavy rain; (b) Sun glare.	40
Figure 22: (a) Heavy traffic; (b) Speed limit	40
Figure 23: (a) animal on the road; (b) signage	40
Figure 24: Ego vehicle overtaking and changing lane	41
Figure 25: Dependence between luminosity and moment of the day	42
Figure 26: Example of probabilities of occurrence for the function LKA.....	43
Figure 27: Interface GUI of the PSA post-processing tool	47
Figure 28: Process for the post processing.....	48
Figure 29: The choice of signals to display in the report	50

Figure 30: Chart of the signal CAN_FD3.11eh.CVM.SEC_DIST_VHL_L_LINE_EXT during the simulation time	51
Figure 31: Example of unreadable drawing graph in yEd software.....	54
Figure 32: File for the input parameters.....	54
Figure 33: Verification reminder	55
Figure 34: Excel DataFrame	56
Figure 35: Python DataFrame: example for the parameter Tarmack.....	56
Figure 36: Python code interface	57
Figure 37: Case of sampling without dependence	58
Figure 38: Case of sampling with dependence.....	59
Figure 39: Example of output scenario of method 1	60
Figure 40: Example of output scenarios for method 2	60
Figure 41: Probability tab in the output Excel file	61
Figure 42: Example of parameters drawn in carmaker	62
Figure 43: Example of testrun.....	63
Figure 44: Signals for the LDW function.....	65

LIST OF TABLES

Table 1: Minimum kilometres for the validation of an ADAS	17
Table 2: Comparison between the two methods	61
Table 3: Example of scenario	63

GLOSSARY

ACC - Adaptive Cruise Control

AEB - Automatic Emergency Braking

ASIL - Automotive Safety Integrity Levels

BSD - Blind Spot Detection

CVAD - Components Design & Validation for Autonomous Driving

CVM - Camera Video Multifunction

DIL - Prototype-In-the-Loop

ECU - Electronic Control Unit

HIL - Hardware-In-the-Loop

ISO - International Organization for Standardization

LDW - Lane Departure Warning

LKA - Lane Keeping Assist

MCMC - Markov Chain Monte Carlo

MIL - Model-In-the-Loop

SIL - Software-In-the-Loop

SLI - Speed Limit Information

SOTIF - Safety of the Intended Functionality

UML - Unified Modeling Language

VIL - Vehicle-In-the-Loop

DEFINITIONS OF SOME TERMS [1]:

Requirement: is a condition or ability needed by a user to resolve a problem or an objective, which must be held by a system or component to satisfy a contract, standard, specification or other formally imposed document.

Requirement specification: a document that specifies, ideally completely, accurately and verifiable, the requirements, designs, behaviours and other characteristics of a component or system, and often the procedures for determining whether these stipulations have been met.

Validation: is the process of evaluating a system, when its development is completed, to ensure that it is conform to its specification. In other words, it aims to answer the following question: "Does the system do well what it is supposed to do (as specified by the requirements)?"

Verification: is the process of determining whether the elements produced during a given phase of system development, fulfil the established requirements during the previous phase. In other words, it aims to answer the next question "Was the system built correctly (as we had specified)?"

GENERAL INTRODUCTION

ADAS (Advanced Driver Assistance System) are vehicle control systems that help to improve driving comfort and safety by assisting the driver in recognizing and reacting to potentially dangerous traffic situations. The development of these systems is evolving at high speed and they integrate more and more vehicles.

In an automotive context oriented towards a full automation of vehicles, combined with standards such as Euro NCAP and ISO 26262 that submit vehicles to security crash tests and require minimum risk of failures and high level of road testing to reliably validate their safety, the problem of ADAS-AD validation is becoming a very important issue. For these reasons, car manufacturers as PSA need efficient methods and tools to ensure that the ADAS functions built into their vehicles operate at the required level of safety and reliability.

The objective of this thesis is to present a method for the optimisation of ADAS – AD validation process based on a stochastic approach using MCMC (Markov Chain Monte Carlo) algorithm, precisely the Gibbs sampler to generate scenarios for an ADAS function validation.

To achieve this, this document will be structured as follows:

First, we will present PSA's issues regarding the validation of ADAS systems; this will encompass the description of the current validation process developed within my internship entity and the problems they encounter. Then, we will make a state of the art of autonomous vehicles and test methods. We will continue by developing the methodology of our work based on Markov chains for the automatic generation of test cases for the validation of a given ADAS function. We will finish by presenting the different results obtained and the limits related to our resolution approach.

This research work will end up with some perspectives regarding a potential improvement of this approach, using mutation algorithms to widen the field of scenarios that we can obtain. And a possible approach that may improve the requirements coverage, by integrating within the MCMC algorithm another mathematic algorithm of decision, which will verify a minimum level of coverage by a scenario before choosing to generate it.

Chapter 1:

CONTEXT AND PROBLEM STATEMENT

I PRESENTATION OF PSA

Group PSA (formerly known as PSA Peugeot Citroën from 1991 to 2016) is a French multinational manufacturer of automobiles and motorcycles sold under the Peugeot, Citroën, DS, Opel, Vauxhall and Free2Move brands.

Europe's number-two automobile manufacturer, it has a revenue of 74 billion of Euro and 3.9 million of vehicles sold worldwide in 2018. It is present in 160 countries and has 211,000 employees worldwide.

I.1 Different brands

I.1.1 Peugeot



Peugeot is the only brand to deploy a complete mobility offer with private and commercial vehicles, scooters, bicycles and a wide range of services.

Present in 160 countries with 10,000 points of contact, Peugeot combines high standards and emotion. In 2010, the year of its 200 years, with 2.14 million of vehicles sold, Peugeot has confirmed its position as the first French car brand in the world. With the success of the 3008, 5008 and RCZ, the brand's upmarket strategy is successful and continues in 2011 with 508. Its environmental efficiency is embodied with the e-HDi technology, Peugeot i0n, while in 2011, the launch of the world's first diesel hybrid, 3008 HYbrid4.

I.1.2 Citroen



Dynamic and creative, the brand celebrates its centenary this year 2019.

Present in more than 90 countries with more than 10,000 sales and after-sales services, the brand sold nearly 1.1 million vehicles in 2017. In addition to design and comfort, "Optimistic", "Human" and "Smart" summarize its services that want to be innovative and connected through new technologies embedded in its vehicles, inspired by and for its customers.

I.1.3 DS Automobile



DS is a French brand born in Paris with the ambition of embodying French luxury in its cars. In 2017, the brand lunch the new SUV DS 7 CROSSBACK.

Its strategy is based on three pillars: exceptional cars, a multi-channel distribution networks and a unique and personalized customer experience with the "Only You, the DS experience" program.

I.1.4 Opel



Opel is one of the largest European car manufacturers and was founded by Adam Opel in Rüsselsheim, Germany, in 1862. The company started building automobiles in 1899. Together with its British sister brand Vauxhall, the company is represented in more than 60 countries around the globe selling over one million vehicles in 2018.

I.1.5 Vauxhall



Founded in 1857, Vauxhall began building cars in 1903. In 1925, it was bought by General Motors and in 2017 by the PSA Group. Most of his models, such as the Rock Adams, are derived from Opel cars, modified and adapted exclusively for the British market and roads. The brand holds the 2nd place on the British market for many years.

I.1.6 Free2Move



Created in 2016, Free2Move embodies PSA's ambition to become the preferred global mobility provider. It aims to offer customers, the most complete set of mobility solutions and especially the most practical, closer to their needs. This application offers car sharing, bike, scooter in more than 15 European cities from Germany to Sweden via Denmark, Italy, Portugal, France ... etc.

I.2 Presentation of my service

My end of course internship took place within the CVAD (Components Design & Validation for Autonomous Driving) entity linked to the Quality and Engineering Department by the flowchart presented in the figure 1 below.

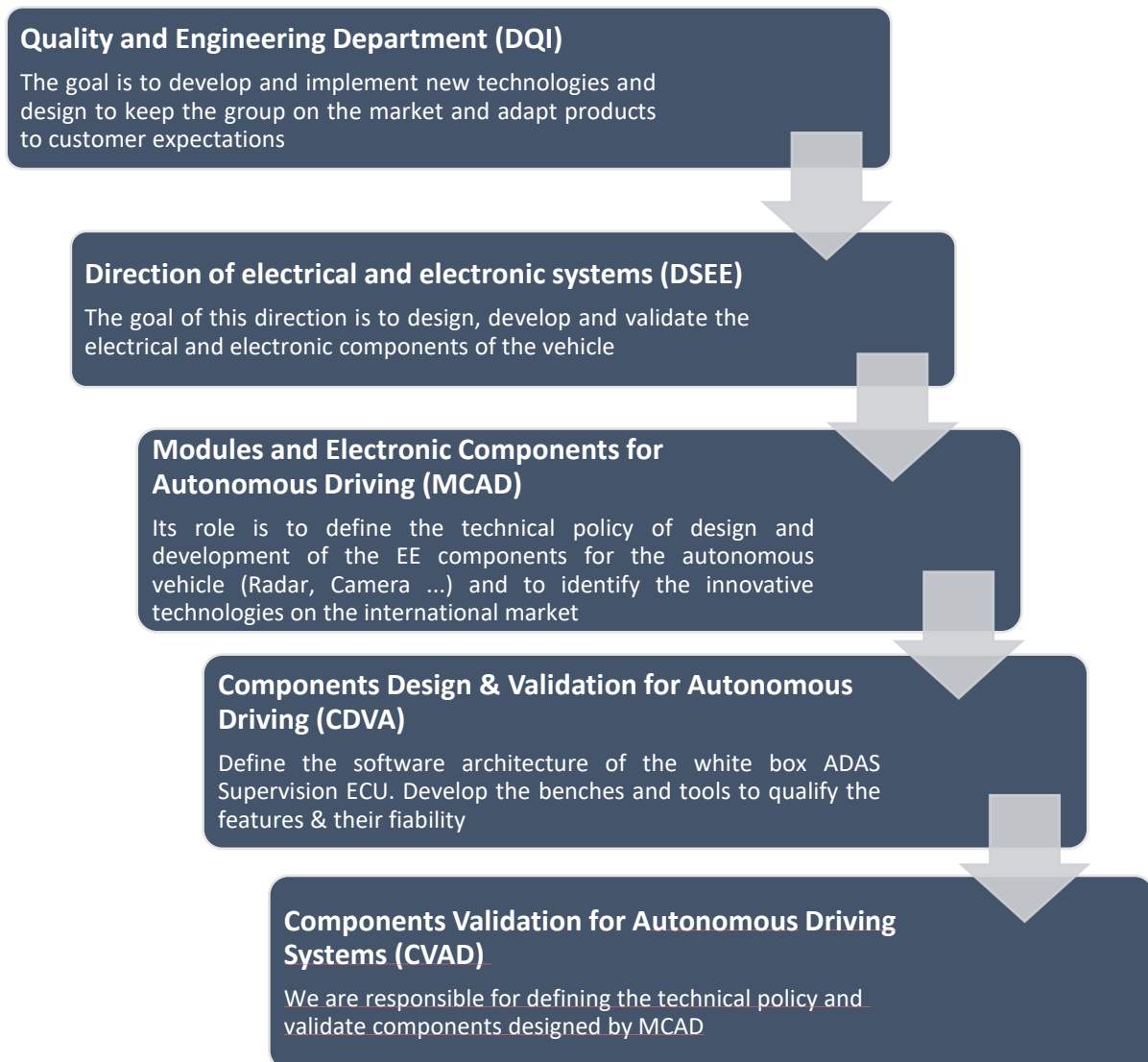


Figure 1: CVAD organisation chart [Source: PSA document]

I.3 Roadmap of PSA for autonomous driving

PSA has chosen to limit its vehicle automation to level 4 as shown in the following Figure 2.

Levels 5 is set aside for the moment because it seems to be not profitable. Only the integration of sensors for full autonomous vehicle make it very costly, and then limiting its purchasing capacity.

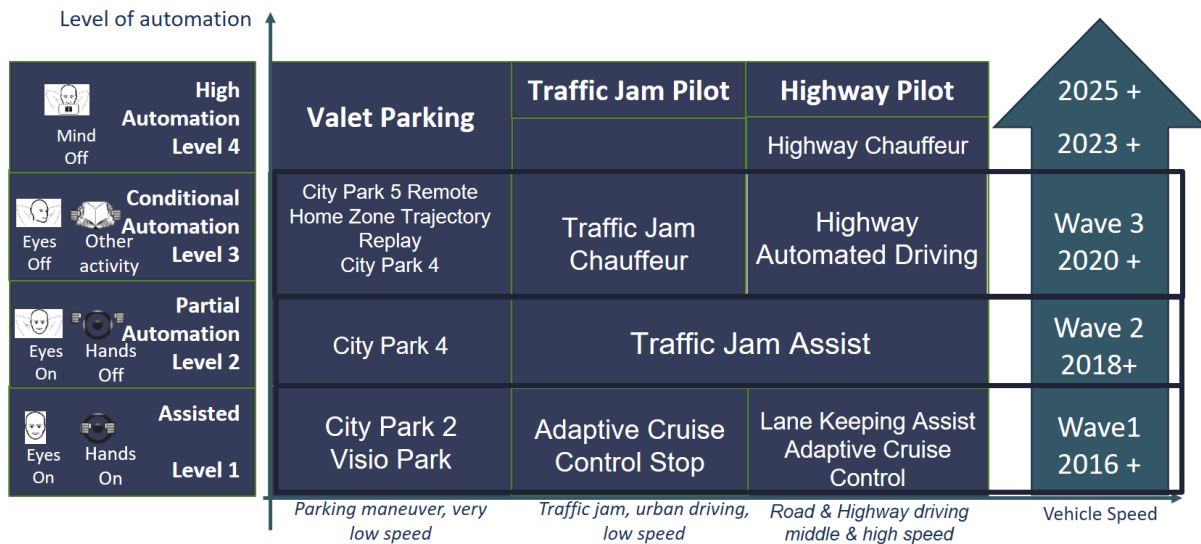


Figure 2: Roadmap of PSA for vehicle automation [Source: PSA document]

Actually, the PSA vehicles integrating the ADAS systems on the market are in level 2 and 1. We can mention: DS 7 CROSSBACK, DS 3 CROSSBACK and PEUGEOT 508 and 508 SW. Vehicles of level 3 are the projects on which they are working actually.

II CHALLENGES OF PSA CONCERNING AUTONOMOUS VEHICLE AND ITS SAFETY

The safety of road vehicles during their operation phase is of paramount concern for the road vehicles industry. Standards have been established to frame and define requirements in order to achieve this safety objective. We have the ISO 26262 standard, which deals with the functional safety of the autonomous vehicle and ISO 21448 or SOTIF that complements it.

II.1 Standards regulating the autonomous vehicle and its safety

II.1.1 Norm ISO 26262

ISO 26262 is a risk-based safety standard that derives from IEC 61508. It applied to electric or electronic vehicle systems and deals with internal hardware and software failures of the entire development process [2].

This standard uses Automotive Safety Integrity Levels (ASIL) to offer a measure of the risk associated with a subsystem. These levels go from A to D (cf. figure 3), A being the lowest level of integrity and D the highest, that is, the most demanding with the most requirements. The parameters of risk, severity, probability of exposure, and controllability determine the ASIL (cf. appendix 1).

The risk can then be defined as a function of a frequency of occurrence of hazardous events, the ability to avoid specific harm or damage through timely reactions of the persons involved (controllability) and the potential severity of the resulting harm or damage”; [3]



Figure 3: Different ASIL level

Depending on the level of ASIL, minimum-testing kilometers to reliably validate an ADAS system with a confidence of 70% on the conformity, has been defined by the ISO 26262 standard. The table 1 presents these kilometer values.

Table 1: Minimum kilometres for the validation of an ADAS [Source: PSA document]

ASIL	Probability of safety goal violation/hour	Minimum kilometers without safety-related incident (Confidence level 70 %)
A	$<10^{-7}/h$	$4,8.10^8 km$
B	$<10^{-8}/h$	$4,8.10^9 km$
C	$<10^{-8}/h$	$4,8.10^9 km$
D	$<10^{-9}/h$	$4,8.10^{10} km$

However, this standard does not deal with systematic faults related to the bad interpretation of the external environment systems, or the combination of this with internal faults (hardware and software). This is illustrated in the chapter Scope part 1 of the standard:

Scope ISO 26262 part 1: “This document addresses possible hazards caused by malfunctioning behaviour of safety-related E/E systems, including interaction of these systems. It does not address hazards related to electric shock, fire, smoke, heat, radiation, toxicity, flammability,

reactivity, corrosion, release of energy and similar hazards, unless directly caused by malfunctioning behaviour of safety-related E/E systems.” [4]

II.1.2 PAS SOTIF (Safety of the Intended Functionality)

SOTIF is Safety of the Intended Functionality. It is the shorthand for the new ISO/PAS 21448 standard. This standard focuses on the safety of driving scenarios that take into account the vehicle environment and the potential imperfections of sensors and algorithms.

Example of a SOTIF situation:

At the sight of a drawing on the roadway, the vehicle triggers the emergency braking. However, the situation was not dangerous but the ADAS system reacted due to a false detection, which is not a hardware or software failure of the system.

The PAS SOTIF is based on the idea that a system will be faced with a wide variety of life situations in its operational life, and that the behaviour it has in each of those situations must be free from an unreasonable level of risk (risk judged unacceptable in a certain context according to valid societal moral concepts) [5]. The scenarios faced by the system during its operational life are classified as follow:

- **Known / unknown:** the situation have already be identified or not.
- **Hazardous / Safe:** the vehicle will behave safely or we do not know yet its behaviour.

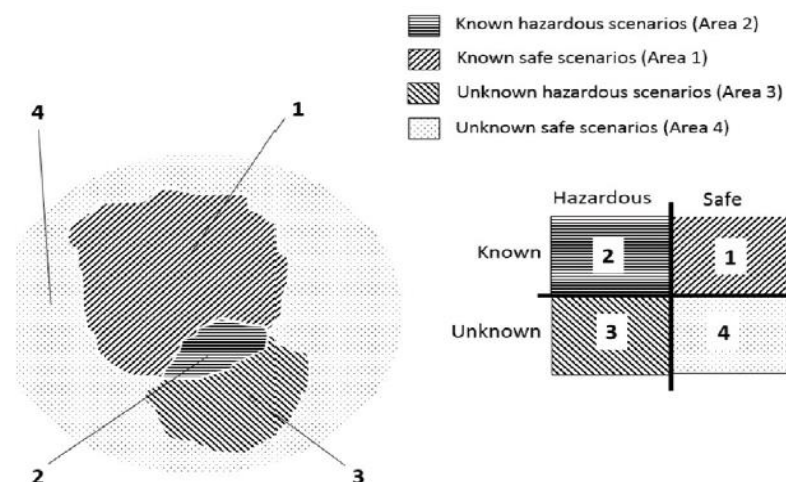


Figure 4: Classification of type of scenario that a vehicle could face [5]

The purpose of the SOTIF activities is to reduce the known hazardous scenarios by improving the system (Area 2), and to show through verification and validation that the residual risk due to potentially hazardous unknown scenarios is sufficiently low (Area 3).

II.2 Challenges related to the validation of the safety of autonomous driving

Validation is a process that consists in submitting a system to a set of tests (on road test or simulation test) during its development, in order to determine if it meets the requirements set out in its environment of use. To reliably validate autonomous driving function the norm ISO 26262 demands a very large amount of testing kilometres approximately 10^8 km [6].

II.2.1 How does the ADAS validation is done actually at PSA

Since my internship took place in the CVAD entity, I will describe in this section, its validation process.

As defined above, the validation of autonomous systems is intended to reassure that they satisfy the different requirements related to their design and use. As a result, the validations done at PSA have for entry requirements in textual form or in model form. The figure 5 presents an example of requirement for the camera.

N° Exigence(v) Requirement no. (v)	Libellé de l'exigence/Description of the requirement	Exigence amount(v)/Input requirement (v)
GEN-VHL- CD_CVM4_0058(2)	<p>The CVM shall detect walking and standing pedestrians (adult or child above 0,8m), in skateboard, in rollers...</p> <p>The detection shall not be disturbed by clothing (color and shape) or accessories they may be wearing (umbrella, hat, hood, cane, handbag, portfolio, music instrument, caddie ...).</p> <p>The supplier will define any limitation.</p>	GEN-VHL-TFD-FT4-PVE-AAS.002 R_4153.1

Figure 5: Example of requirement for the camera [Source: PSA document]

The figure 6 below presents the different steps for the validation process at CVAD.

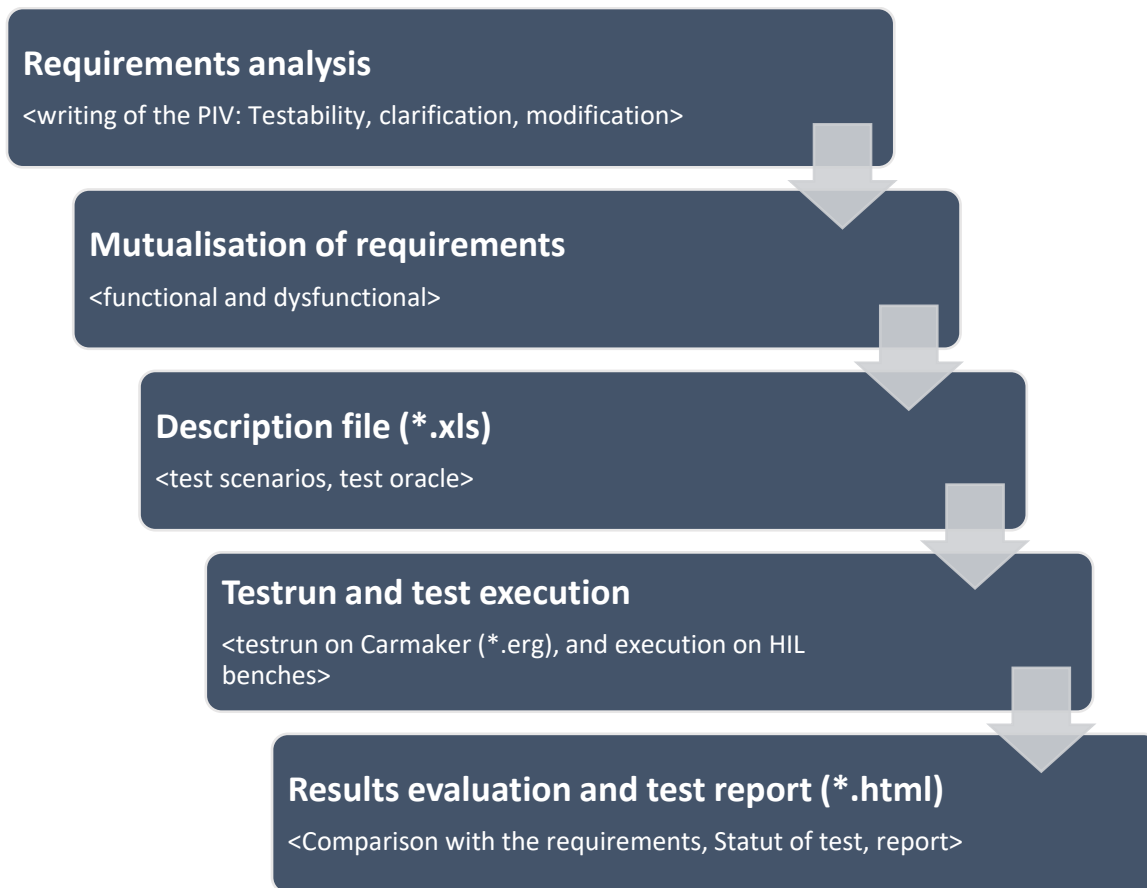


Figure 6: Validation process of the CVAD entity

- **Requirements analysis:** it consists on checking the testability, the applicability of the different requirements, depending on the capacity of the validation means available and the clarity of understanding the requirements by the validation officer. This is done in a document called PIV (Integration Validation Plan) at PSA. For example, a requirement to test the vehicle in snowy weather will be considered untestable on HIL bench.
- **Mutualisation of requirements:** this step consists on grouping together requirements that can be test in the same scenario. This to reduce the number of scenarios and then the time for the validation.
- **Description file:** an Excel file of description of the test or test plan is written, it contains for each of the requirements;
 - a description of the scenarios under which these requirements will be validated;
 - the expectation of the test;
 - the signal results expected (mathematical expression to check in order to give the test status);

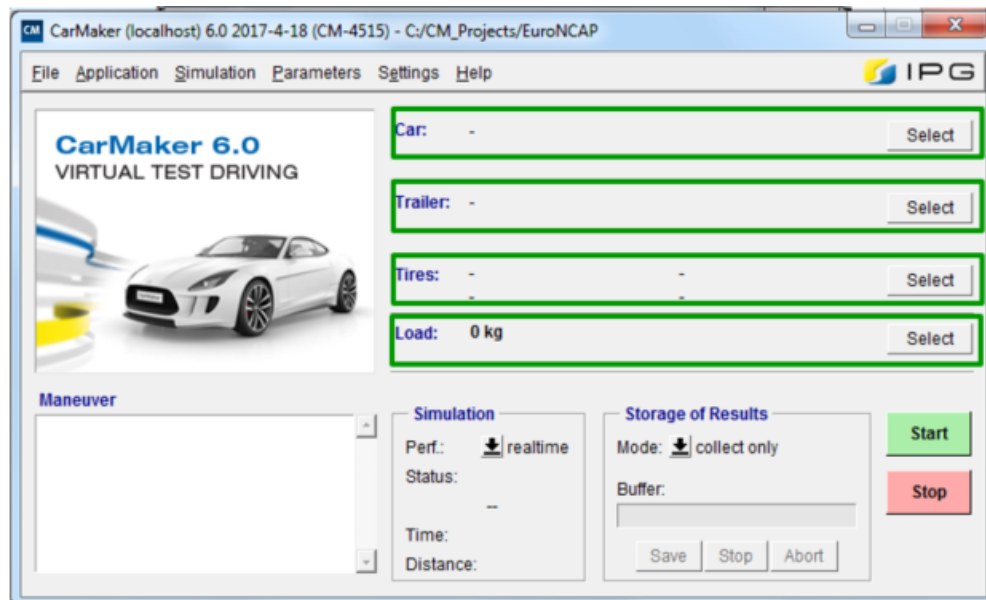
- the moments of the various manoeuvres of the ego vehicle;

Testrun description	Expectations	Signals	Reference Time	Verification Time
Vehicule speed is 80 kph Vehicule has a linear trajectory in the right driving side and on the middle lane (3 lanes). The testrun is a straight including car on the left lane: - 250 meters: car driving in the left driving side with velocity 20 kph	The CVM provides AEB attributes {GEN-VHL-CD-CVM4_1013(1)}	$0 \leq \text{CAN_P1.15ch.CVM.AEB_AVAILABILITY}(tDistance)$ & $\text{CAN_P1.15ch.CVM.AEB_AVAILABILITY}(tDistance) \leq 1$	tDistance	$\text{find}(\text{Car.Distance} \leq 400 \ \& \ \text{Car.Distance} \geq 50)$

Figure 7: description file of PSA [Source: PSA document]

- **Testrun and test execution:** for each textual scenario, a testrun is created in the simulation environment. In the case of PSA, the simulation environment tool used is CarMaker from IPG for HIL bench validations.

To create a new scenario in CarMaker, we first have to select the ego car and its different characteristics (tires, load...). After, we create the environment (the road, the different accessories on the road, the sceneries) and we add the traffic (other vehicles, roadway users...). We finish by adding the different manoeuvres of the ego vehicle and run the simulation test. The output of CarMaker simulation is an .erg file.



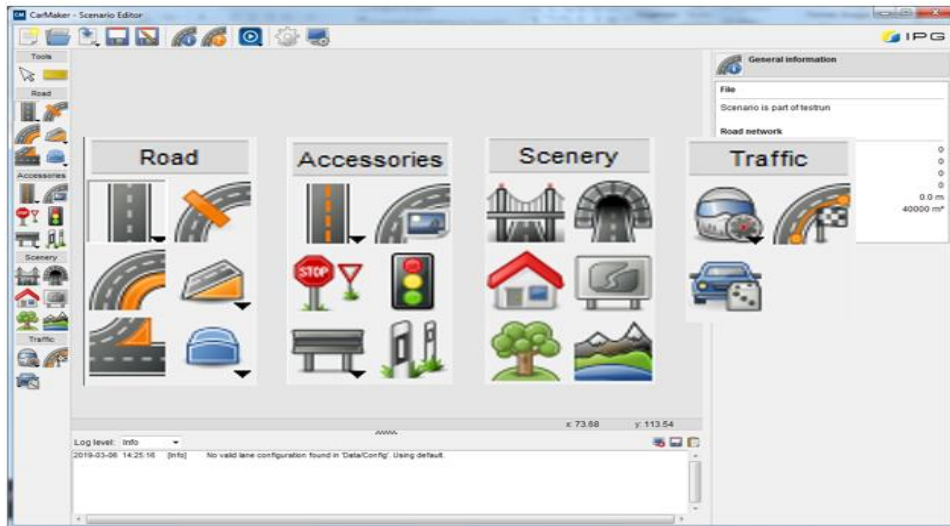


Figure 8: Carmaker interface

- **Result evaluation and report:** Finally, the post processing of the results is done under a Matlab tool, called NODESAT Post-Processing, developed by PSA. This tool makes a comparison between the expectations (which are generally mathematical formula, as we can see in the columns signal on Figure 7) defined in the description file and the results obtained after the CarMaker simulation. A "TRUE" or "FALSE" status is assigned to the test if it validate or not the requirements.

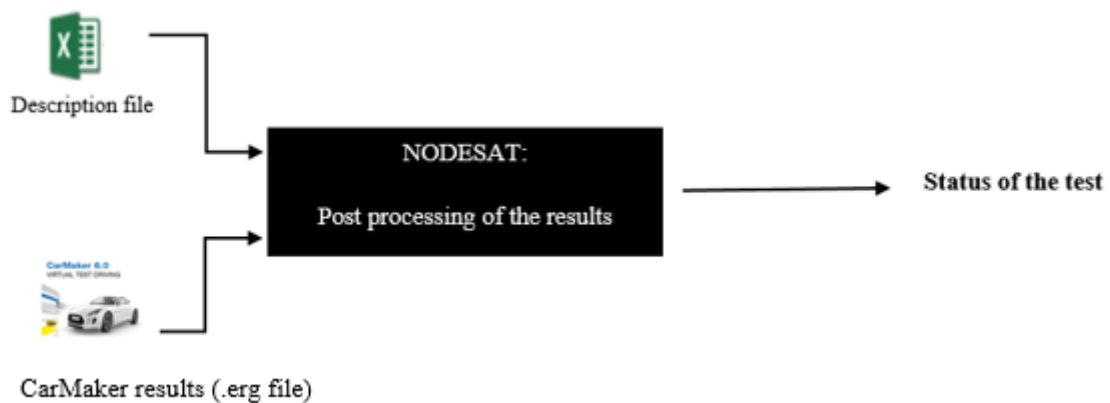


Figure 9: Validation process at CVAD

This validation process has certain limits such as validation times. For a given requirement, there is need for at least one scenario and we estimate at about half a day the completion time of a single scenario. This means, for several requirements (e.g 50) of one ADAS function, the manual creation of scenario will be estimated to months of work.

II.3 Problem formulation

The challenge of assuring safety for the autonomous vehicle has given rise to the following questions:

- How to efficiently validate the autonomous vehicle functions? How to reduce costs and validation time?
- How to reduce the risks due to potentially dangerous scenarios that are unknown, disregarded or otherwise be forgotten by the human creating the scenario? How to validate the SOTIF?
- How will we be confident on the requirements coverage by the generated scenarios?

II.4 Contribution and outline

This thesis provides a solution to each of the issues raised above.

First, to improve the validation process and reduce the time taken, we propose to automatically generate virtual driving scenarios, which is actually done manually at PSA.

Regarding the reduction of risks due to unknown potentially dangerous scenarios, we will list all the parameters likely to influence the ADAS functions and we will generate automatically the scenarios under the base of these parameters.

The degree of confidence of the requirements coverage will be ensured by generating all the possible combinatory of all the parameters. That will be done using the Markov Chain Monte Carlo algorithms, more precisely the Gibbs sampling.

However, if this number of parameters is high, as is generally the case, the combinatory will be very large and there will be the risk of a combinatorial explosion. Therefore, another contribution of our work will be to avoid this combinatorial explosion.

In addition to the major problem presented above, another objective of my work will be to improve the post-processing tool NODESAT Post-Processing which is responsible of defining a status on the tests that are made within my work department.

Our development will thereby focus on the following main axes:

- Chapter 2 summarizes the state of the art on autonomous vehicles, the different level of automation, and the different test methods that are done to validate the vehicle. Also a summary of Monte Carlo by Markov chain methods is presented: what is a Markov chain, what are its characteristics.
- Chapter 3 develops the methodology of our work: the choice of parameters influencing the ADAS, the classification of these parameters, the probabilities attribution, the sampling principle used, as well as the method used to rule on the achievement of the convergence to avoid the combinatorial explosion.
- Chapter 4 presents the different results obtained. In addition, a discussion on this results is done as well as a presentation of the limits of the work.

Chapter 2:

STATE OF ART OF AUTONOMOUS DRIVING

I DRIVER ASSISTANCE AND AUTONOMOUS DRIVING

I.1 What is an autonomous vehicle?

An Autonomous Vehicle is a vehicle capable of driving without human involvement in real traffic conditions.

In autonomous driving, the goal giving by a human is to reach a destination while respecting the road safety rules. To do so, the autonomous vehicle needs to perceive its environment, analyse the information, plan what should be done and take actions by converting the plan into a set of control inputs [7].

The figure 10 below represents how an autonomous vehicle works.

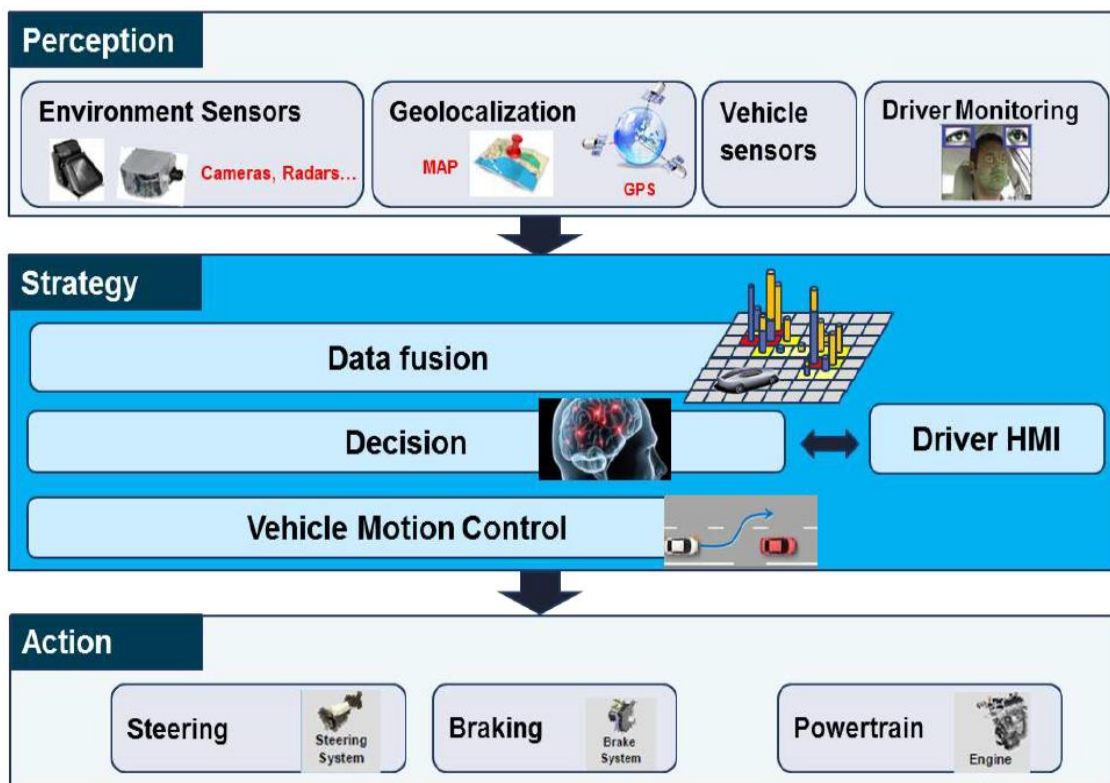


Figure 10: Global architecture of an autonomous vehicle [Source: PSA document]

The sensors (cameras, lidars, radars...) and the navigation systems, provide information about the environment and the vehicle itself to the perception layer. The goal of this perception layer is to extract relevant information that comes from:

- The vehicle itself: like position, speed, wheel angle variation, time to collision, etc....
- The road: road signs, road marking, number of lanes, obstacles on the road, etc....

Then, all this information is sent to the planning modules that decide the trajectory and the general behaviour of the vehicle. The decisions are converted in control inputs for the car making sure the planned trajectory is followed as accurately and as smoothly as possible [8].

I.1.1 Environment perception layer

The perception layer is a function that provides the vehicle with crucial information on the driving environment, including free drivable areas, surrounding obstacles, velocities even predictions of their future states [9].

An autonomous vehicle acquires knowledge of its surrounding in two stages. The first stage consists of scanning the road ahead to detect possible changes in driving conditions (traffic lights and signs, pedestrian crossing, and barriers, among others). The second stage relates to the perception of other vehicles [8].

This section presents some sensors that make up the perception systems of autonomous vehicles.

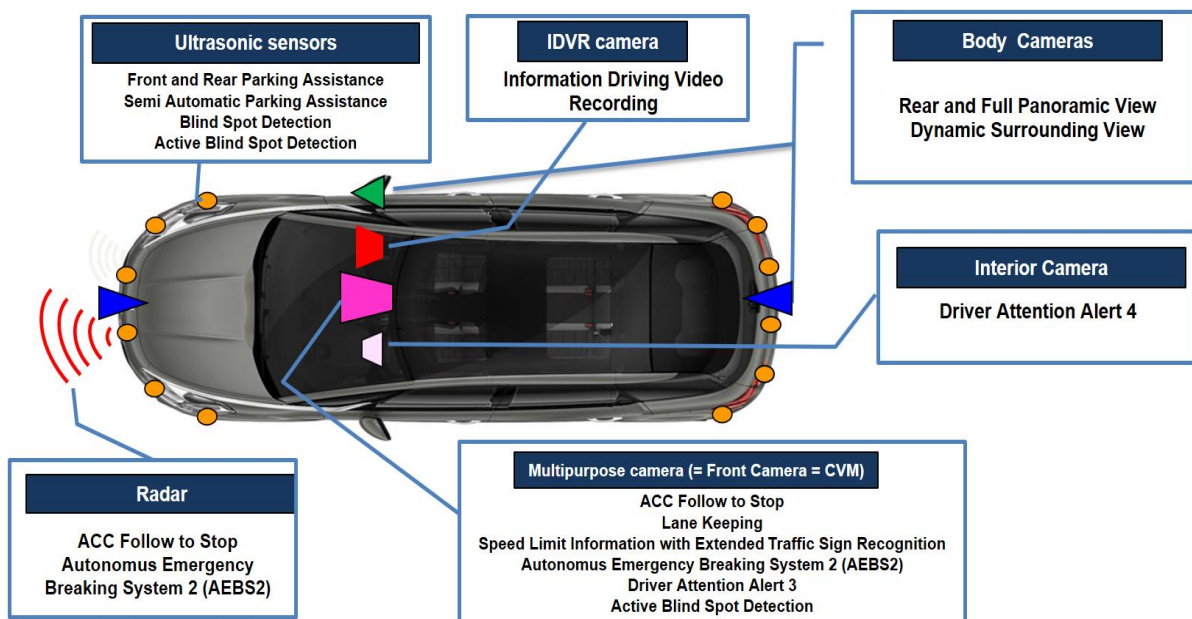


Figure 11: some ADAS sensors [Source : PSA document]

- **Radar:** is an acronym for RAdio Detection And Ranging, its works in wavelengths in order of millimetres. For autonomous vehicles, radar systems work at frequencies of 24/77/79 GHz [8]. Its principle consists of measuring the distance between the emitter and the object, by calculating the time of flight of the emitted signal and the received echo. In addition, it can measure velocity or angle of that object. Radars cover applications at the front of the vehicle such as ACC (Adaptive Cruise Control).
- **Camera:** is today a key sensor for embedded vision applications. At the front, rear or on the sides, it allows the detection and recognition of objects on the road to meet the needs of automation systems such as lighting, speed limitation and automatic emergency braking.

Sometimes, for more precision on what sensors perceive, there is data fusion between the different data of sensors, before the decision is taken.

1.2 Levels of driving automation

Automation levels were established as a classification system for self-driving cars in January 2014, this is done by the Society of Automotive Engineers (SAE) [10]. These levels are defined based on four criteria [7]:

- First, who is performing the driving action - human or machine (“**hands on**”/”**hands off**”).
- Next, who is monitoring the driving environment - human or machine (“**eyes on**”/”**eyes off**”).
- Then, who is reacting when the system falls out of its operating domain - human or machine (“**mind on**”/”**mind off**”).
- At last, can the vehicle be autonomous everywhere, or is it restrained to some use cases?

The figure 12 presents the different levels of automation for a vehicle.

SAE Level	Name	Control of steering and acceleration	Monitors driving and environment	Fallback responsible	Capability of system
1	Driver Assistance	Human driver and system	Human driver	Human driver	n/a
2	Partial Automation	System	Human driver	Human driver	Limited scope
3	Conditional Automation	System	System	Human driver	Limited scope
4	High Automation	System	System	System	Limited scope
5	Full Automation	System	System	System	Full scope

Figure 12: Level of automation from the SAE J3016 standard [11]

We distinguish [7]:

- **Level 0** is a classical car. Human drivers do everything.
- **Level 1** ("hands on"): here, the control of the vehicle is shared between the machine and the driver.
- **Level 2** ("hands off"): the longitudinal and the lateral control tasks are performed by the vehicle. However, in this level, like in the previous one, the driver has to constantly monitor the road and take back the control of the vehicle if anything happens.
- **Level 3** ("eyes off"), the driver can safely turn their attention away from the driving tasks. The vehicle will handle situations that call for an immediate response, like emergency braking. The driver must still be prepared to take the control within some limited time, specified by the manufacturer, when called upon by the vehicle to do so. Presently, there is no only one Audi A8 Level 3 Car in market.
- **Level 4** ("mind off"), as level 3, but no driver attention is required for safety, e.g. the driver may safely go to sleep or leave the driver's seat. Self-driving is supported only in limited spatial areas or under special circumstances, as traffic jams.
- **Level 5**, no human intervention is required at all.

II TESTING OF AUTONOMOUS DRIVING SYSTEMS

Testing is the process of planning, preparing and executing or exercising a system or system component to verify that it satisfies specified requirements, to detect errors, and to create confidence in the system behaviour.

A test concept comprises the reception and analysis of the requirements, the test case generation, the test execution, and the test evaluation as shown in Figure 13.

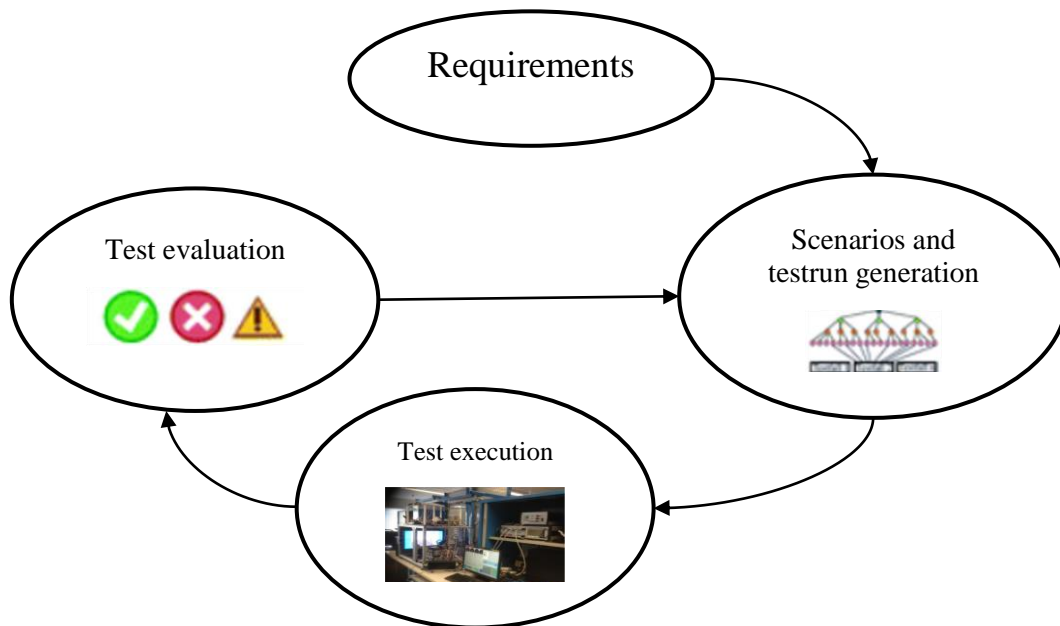


Figure 13: Procedure for test concept [12]

The analysis of the requirements and the scenarios generation should be performed during the design phase, so that the testrun to be carried out are already defined for the verification and validation [12]. The test case generation consists on creating scenario under the execution test should be done. The execution is done in different levels following the V cycle and the evaluation consists on comparing the result signals of the execution test to entry requirements and to give a status (if the test is good or not).

The requirements to assess a test as safety are [12]:

- **Representative:** the test must encompass the minimum degree of reality required and had to cover the defined requirement;
- **Reproducible:** for example if we detect an error from some tests, we have to be able to redo the same scenario of those tests if there is the need;
- **Economical:** it must be ensured that the test execution is prepared and carried out at the lowest cost possible;

- **In good time:** the earlier in the development process a product can be tested, the fewer the development steps that need to be repeated in the case of an error.

II.1 Scenario for the test

The ISO 26262 standard from 2018 represents the state of the art for developing vehicle guidance systems with regard to functional safety. An overview of the development process proposed in the ISO 26262 standard is shown in Figure 14. The process steps that may use scenarios to generate the demanded work products are highlighted in red [13].

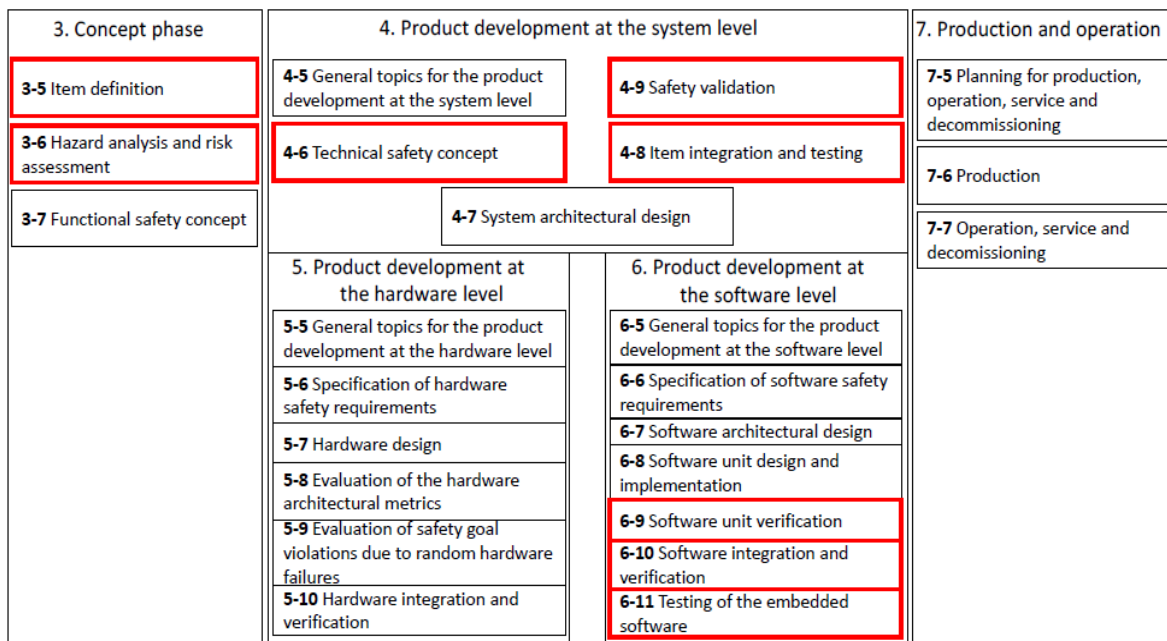


Figure 14: Overview of the development process proposed in the ISO 26262 standard. Process steps highlighted in red may use scenarios to generate the work products [12].

II.1.1 Definitions

According to S. Ulbrich and al [14]:

A scene describes a snapshot of the environment including the scenery and dynamic elements, as well as all actors and observers self-representations, and the relationships among those entities [14].

Example of scene illustration on Figure 15.

A scenario describes the temporal development between several scenes in a sequence of scenes. Every scenario starts with an initial scene. Actions & events as well as goals & values may be

specified to characterize this temporal development in a scenario. Other than a scene, a scenario spans a certain amount of time [14].

Scenes in a scenario are linked by actions and events.

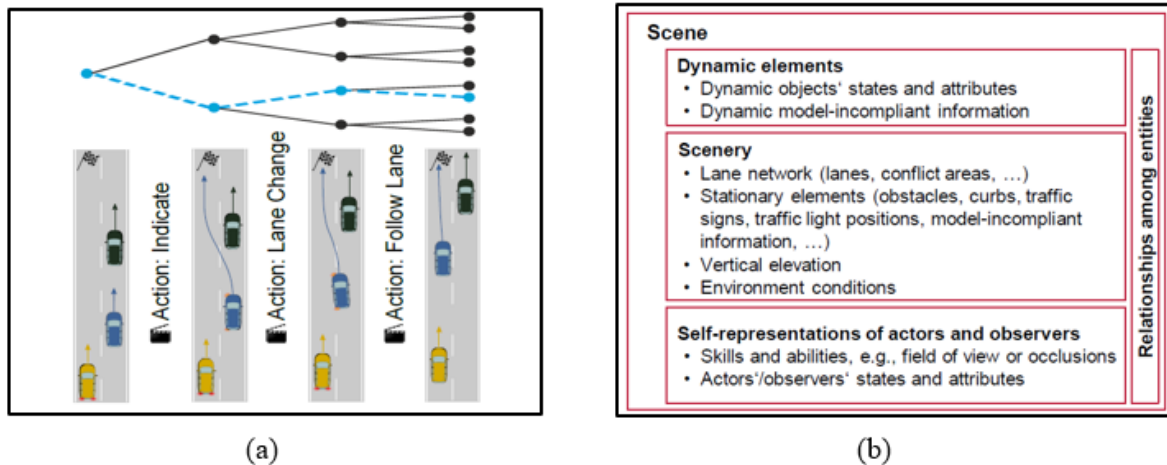


Figure 15: Illustration of a scenario representation (a) and a scene (b) [14]

Scenarios have to fulfil the following requirements to be used during the testing phase of the ISO 26262 standard [13]:

- Scenarios shall be modelled via concrete state values to ensure their reproducibility and to enable test methods to execute the scenario.
- Scenarios shall not include any inconsistencies.
- Scenarios shall be represented in an efficient machine readable way to ensure an automated test execution.

II.1.2 Source of test scenarios

Test scenarios can be extracted from [15]:

- **Requirement specification:** are the best source, they list the different types of scenarios to be expected and test cases are derived from them.
- **Problem understanding for creating parameterizable scenarios:** the autonomous driving engineers could use their expertise to create a potential scenario that the system should support.
- **Gathered data:** like Accident databases. As it contains particularly dangerous and difficult scenarios, the data from these accident databases is relevant for test cases.

II.2 Different testing levels

The test levels are defined according to the different step of the V-cycle as defined in the system engineering process.

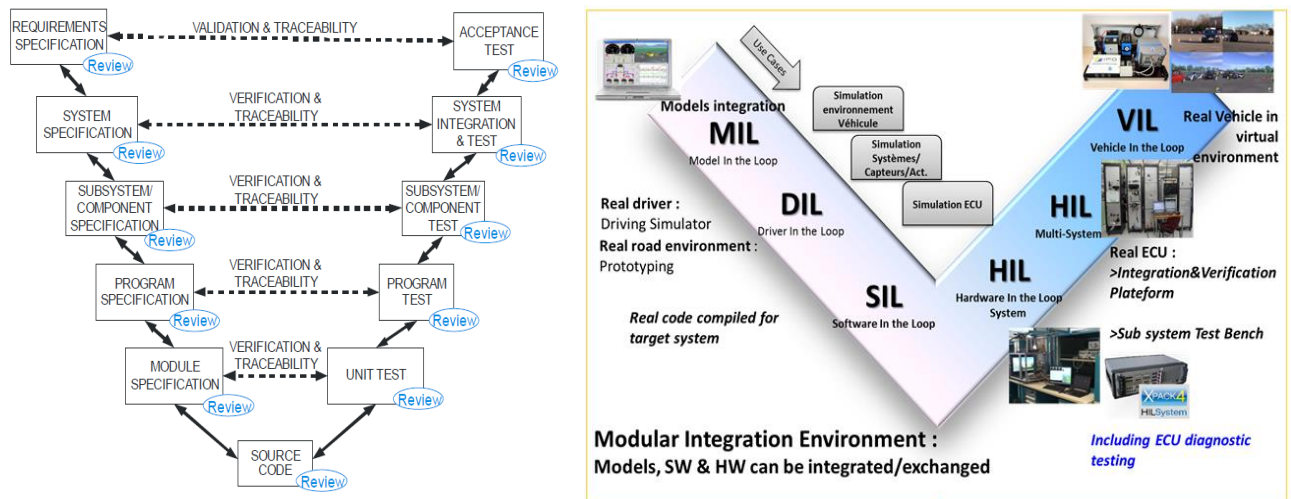


Figure 16: Generic V-Model [source: PSA document]

The left side of the V works its way from requirements through design to implementation. The right side of the V iteratively verifies and validates larger and larger parts of the system as it moves up from small components to a system-level evaluation [16].

The different methods of testing from the first step to the last are:

- **MIL (Model-In-The-Loop):** is the first level of testing. Based on the specification requirements, the designers who are in this stage of development create different models. Those models can be for example, Matlab Simulink models or UML models. MIL testing involves checking on these models in a simulation, making different corrections necessary and choosing some models from among all, that will be called the specifying models and which can move down the chain for the next test steps.
- **DIL (Driver-In-The-Loop or Prototype-In-The-Loop):** consist on integrating the models resulting from the MIL in a prototype car. In the case of PSA, they have some prototype vehicles called labcar that they use for those tests. The goal of this phase of test is to ensure that the different models connected together comply with the functional requirements.

- **SIL (Software-In-the-Loop):** The developers, who are at this stage of testing, receive both specifying models from the upper level and textual requirements. From the previous models, a C code is generated and from the requirements, they write manually the code. The goal of the SIL test is to ensure that the codes created here are still conform to the requirements. These tests are done under virtual environment and are usually simulated in accelerated time compared to reality.
- **HIL (Hardware-In-the-Loop):** at this level, the software validated in SIL is integrated in the hardware (sensor, calculator ...). This hardware is thus integrated on the bench with CAN buses, and the tests are carried out. This level of testing is done in real time. For a validation using a camera for example, a scenario is played in the simulation environment and is displayed on a screen. The camera, which is connected to the ECU with different CAN, films the screen, and send the different signals via the CAN buses to the PC for the analysis.

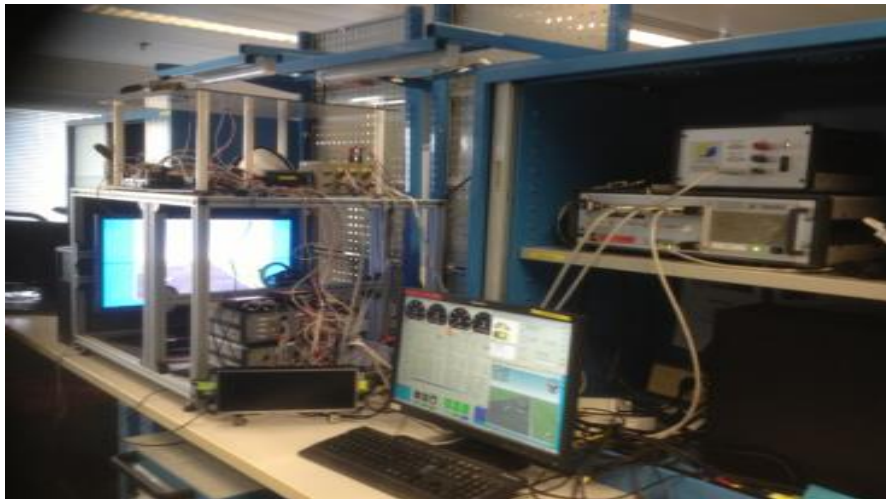


Figure 17: Post for a HIL validation at PSA

- **HIL sub-system:** the goal of this step is to assure the good communication between all the different components, sensors and calculators, which have been tested separately. These different elements are connected so that they must be on a real vehicle and validation tests of good functioning are done together.
- **VIL (Vehicle-In-the-Loop) or real test-driving:** consists of integrating all hardware sensors and calculators recovered from HIL validation and integrated on a real vehicle.

Then this vehicle will be put in real driving situation for the tests. This test method presents a huge risk, especially for the driver of the vehicle in a case of accident.

Apart from the VIL test that is done in a real world and HIL and DIL that incorporate some level of physical hardware to provide real data inputs for some parts of the system, the other tests level are done totally in a virtual simulation environment. This means that the complete driving scenario for the test is modelling with a software; the driver, sensor, traffic, realistic vehicle dynamic are all include in the environment. In contrast with real test-driving, virtual environment simulation is safe.

III MARKOV CHAINS MONTE CARLO STOCHASTIC SIMULATION

The MCMC methods are sampling algorithms that construct a Markov chain according to a probability distribution π , which is the stationary distribution of the created chain.

III.1 Markov chains

A Markov chain is a sequence of random variables $X_1 \dots X_2$ in a denumerable set Ω ; such that, the conditional distribution of X_{n+1} in the future depends only of the present state X_n . For discrete-time Markov chain, we have:

$$P(X_{n+1} = x | X_1 = x_1, X_2 = x_2, \dots, X_n = x_n) = P(X_{n+1} = x | X_n = x_n) \quad [17]$$

The matrix $P = (p_{ij})$, with: $p_{ij} = P(X_{n+1} = x_j | X_n = x_i)$ is the transition matrix of the chain. It says to be stochastic if it satisfies the following two conditions:

$$p_{ij} > 0 \quad \text{and} \quad \sum_j p_{ij} = 1$$

III.1.1 Stationary distribution

If we denote by $\pi_n(x) = P(X_n = x)$ the probability that the chain is in the state x at time n , if the following limit exists and is unique,

$$\lim_{n \rightarrow +\infty} \pi_n(x) = \pi(x) \quad \text{For any state } x,$$

That is, the chain at each state x follows the same distribution π . Thus, π is the **stationary distribution** of the chain.

This means that when iterating this distribution with the Markov chain, it remains constant.

Under certain conditions during the construction of the chain, it will have to converge towards its stationary distribution that is unique [18].

III.1.2 Convergence properties

If in a finite space Ω , an irreducible and aperiodic Markov chain is iterated from any initial distribution π_0 , then the sequence of π_n of X_n will converge to the stationary distribution π of the chain.

- **Irreducible:** A Markov chain is irreducible if it is possible to go from a state i to a state j with a non-zero probability in a finite number of steps n : $p_{ij}^n > 0$.
- **Aperiodic:** A Markov chain is aperiodic if there is no value $k \geq 0$ such that the transition matrix P satisfies $P^k = I$ where I is the identity matrix.

III.2 MCMC algorithms

There are different kinds of MCMC algorithms. Among them, the most commonly used are the Metropolis Hastings algorithm and the Gibbs sampler. It is those two methods that we going to present in the following section.

III.2.1 Metropolis Hastings algorithm

This algorithm is based on the principle of acceptance-rejection. π is the distribution we are trying to simulate and that we know. For that, we will define a distribution of proposition $q(x, x')$ according to which we will sample. The algorithm is defined as follows [18]:

Algorithm1: Metropolis Hastings algorithm

1. Select X_0
2. For $t = 1, 2, \dots, n$
 Generate Z_t with the probability $q(X|X_{t-1})$
3. Calculate the acceptance ratio:

$$\alpha(X_{t-1}, Z_t) = \min\left\{1, \frac{\pi(Z_t)q(Z_t|X_{t-1})}{\pi(X_{t-1})q(X_{t-1}|Z_t)}\right\}$$

4. Accept:

$$X_t = \begin{cases} Z_t & \text{if accepted} \\ X_{t-1} & \text{if rejected} \end{cases}$$

5. Next iteration, and go back to step 2.
-

III.2.2 Gibbs sampler

Assume we want to simulate a random vector $X = (X^1, X^2, \dots, X^N)$ with distribution $\pi(X)$. Further, suppose all full conditional distributions $P(X_i|X_{[-i]})$, where $X_{[-i]} = \{X_j : j \neq i\}$, are available in the sense that a sample can be drawn from these distributions. The Gibbs sampler updates components X_i of X with a sample from the distribution $P(X_i|X_{[-i]})$ conditioned on the current states of the other components [18].

Algorithm2: Gibbs sampler

1. Select $X_0 = (X_0^1, X_0^2, \dots, X_0^N)$
 2. For $t = 1, 2, \dots, N$
Generate $X_t^1 \leftarrow P(X^1|X_{t-1}^2, X_{t-1}^3, \dots, X_{t-1}^N)$
 $X_t^2 \leftarrow P(X^2|X_t^1, X_{t-1}^3, \dots, X_{t-1}^N)$
 \vdots
 $X_t^N \leftarrow P(X^N|X_t^1, X_t^2, \dots, X_t^{N-1})$
 3. Repeat step 2 until reaching equilibrium.
-

The stationary distribution in this case is the joint probability $\pi(X)$ given by:

$$\pi(X) = P(X^1, X^2, \dots, X^N). \quad [19]$$

Chapter 3:

METHODOLOGY OF THE WORK

I APPROACH FOR TEST CASES GENERATION

This part describes an approach that use the statistical method of MCMC algorithms presented above to generate automatically the combinatorial of test scenarios for the validation of an ADAS function. The ADAS operation (correct or incorrect decision) depends on the environment it observes and on the behaviour of the ego vehicle, this approach will consist in:

- Identify exhaustively these parameters that likely to influence the ADAS function;
- Classify these parameters in equivalence class to avoid incompatible parameters in the same scenario (day and night together for example) and assign likelihood of occurrence to each class;
- Generate by a Monte Carlo method a set of test cases corresponding to all the possible cases, the criterion of stopping the generation being defined by reaching the chain convergence;
- Remove redundant or duplicate scenarios.

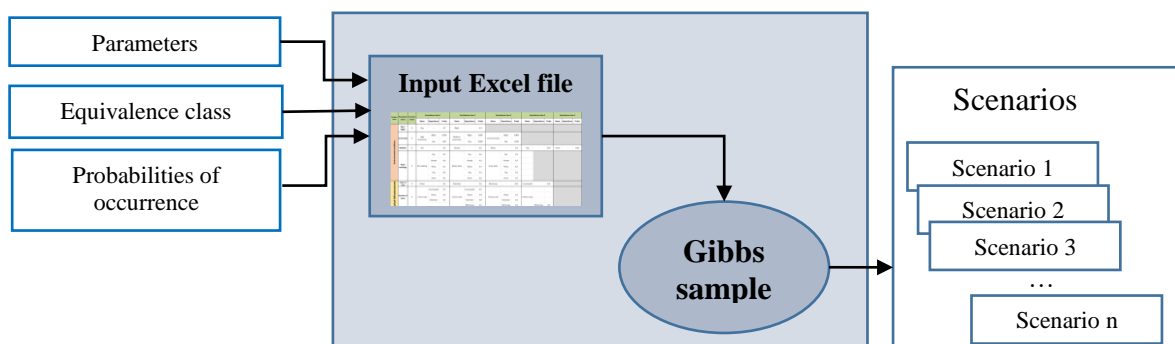


Figure 18: Methodology principle

I.1 Definitions of parameters that influence an ADAS function

The parameters that influence the ADAS function are defined based on the ODD (Operational Design Domain) taxonomy and Ego vehicle behaviours presented in the technical report of the National Highway Traffic Safety Administration (NHTSA) in September 2018 [20].

I.1.1 ODD taxonomy

An ODD describes the specific operating domains in which an ADAS feature is designed to function with respect to roadway types, speed range, lighting conditions (day and/or night), weather conditions, and other operations constraints. ODD will likely vary for each ADAS feature.

To identify the different attribute that define the ODD taxonomy the NHTSA used a lot of information as press releases, NHTSA pre-crash scenario analysis, technical journals, videos and conference proceedings.

The ODD taxonomy includes the following top-level categories: Physical Infrastructure, Operational Constraints, Objects, Connectivity, Environmental Conditions, and Zones.

The figure 19 presents the hierarchical ODD taxonomy.

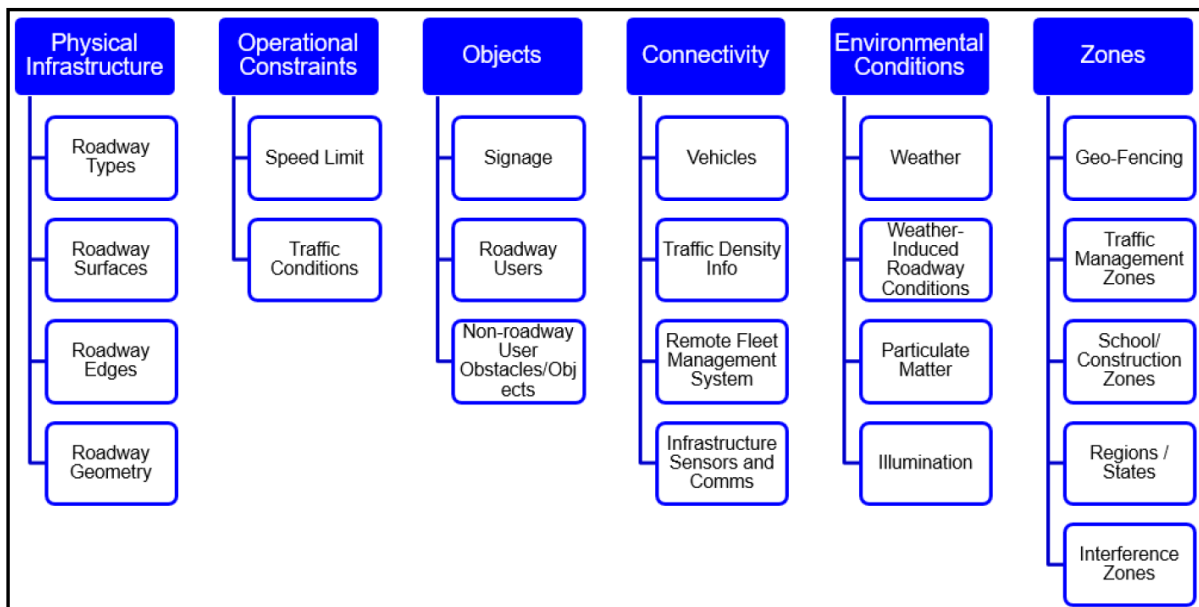


Figure 19: ODD classification framework with top-Level categories and immediate subcategories [20]

The different sublevels in each top level or category will be called Parameters.

1.1.2 Equivalence class classification

An equivalence class consists of a group of elements, which has the same impact on the system. For each parameter of the ODD taxonomy, we are going to classified it in different groups of equivalence class. The reason why we do this classification in equivalence class is to avoid in the same scenario incompatible elements, we separate them at this stage so that we cannot choose them together when we will do the generation.

We will only present some classification examples of the main categories. A table resuming the different categories, their parameters and their equivalence class is presented in appendix 2.

1.1.2.1 Physical infrastructure

- Roadway types: we divided it into highway, rural, urban, countryside, suburban.
- Roadway characteristics: number of lane, slot, straight, curve...
- Roadway surfaces: Asphalt, concrete, gravel, grass...
- Roadway configuration: roundabout, gyratory, straightaways, Y-intersection...
- Roadway edges: concrete barriers, cones...

The figure 20 illustrates an example of physical infrastructure.



Figure 20: (a) Rural road; (b) Barrier and temporary cones¹

1.1.2.2 Environmental conditions

The environment can affect visibility, sensor fidelity, vehicle manoeuvrability and communication systems, figure 21 presents some examples. It is divided in:

- Weather: rain, snow, cloudy, fog, clear...
- Road masking: standing water, flooded roadways, ice roads, snow on road...
- Illumination: high luminosity, low luminosity, medium luminosity...
- Moment of the day: night, day, dawn, dusk...

¹ <https://www.google.fr/pictures>



Figure 21: (a) Limited visibility Heavy rain; (b) Sun glare¹

1.1.2.3 Operational constraints

These include elements such as dynamic changes in speed limits, traffic characteristics. For example, an autonomous vehicle entering a school zone is subjected to lower speed limits and must respond appropriately to ensure the safety of its passengers and other road users.

- Traffic conditions: minimal traffic, normal traffic and heavy traffic
- Speed limitation
- Sign plate: motorway entrance and exit, home zone, Stop...
- Traffic light

Figure 22 presents some examples.



Figure 22: (a) Heavy traffic; (b) Speed limit¹

1.1.2.4 Objects

We classified objects in:

- Roadway users: pedestrians, cyclists, vehicles (trucks, buses, cars, motorcycles...)
- Non-roadway user: animals, perturbation target, signage, debris...



Figure 23: (a) animal on the road; (b) signage¹

These objects could have a dynamic like for example:

- The vehicles can decelerate, change line, parking, entering roadway...
- Pedestrian or cyclist can crossing road, walking or riding on sidewalk...

I.1.3 Ego vehicle manoeuvre behaviours

The ego vehicle is the vehicle having ADAS function integrated that we want to test. It can have many operational manoeuvres that influence the ADAS function.

We can list: lane switching, accelerating/decelerating, overtaking, parking...



Figure 24: Ego vehicle overtaking and changing lane [20]

I.2 Dependence between parameters

Some equivalence classes depend on others. For example, illumination depends on the moment of the day; road topology depends on the type of road.

For each equivalence class, we going to search if it depends or not on another class. In the case of dependence, we will put them together. We can see an example of dependence in the figure 32 of the input Excel file presented in the next chapter.

We need to know all the dependences before the attribution of probability of occurrence to equivalence class, because in the case an element depends on another, its probability of occurrence will be the conditional probability of occurrence knowing the value of what it depends.

The figure 25 illustrates and example of dependence.

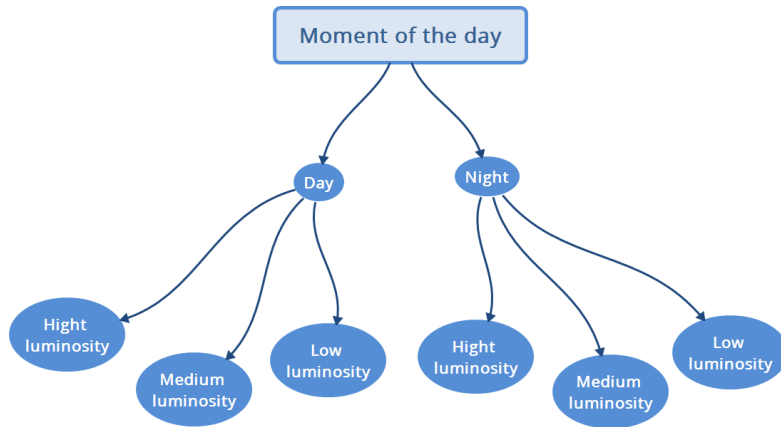


Figure 25: Dependence between luminosity and moment of the day

1.2.1 Likelihood of occurrence

For each equivalence class we must assign a probability of occurrence.

The criteria for assigning those probabilities is based on the ADAS function that we want to test. For example, the parameter presence of a pedestrian on the road is more likely to influence the AEB function than the LKA (Lane Keeping Assist) function, thus the probability in the case of AEB will be greater than in the case of LKA. These probabilities must be as close as possible of reality. As a result, people who have to assign them must have good experience in the automotive sector: good knowledge on automotive environment and ADAS.

In figure 26, we present an example of probabilities of occurrence of some parameters related to the LKA function (given by the ADAS responsible). The Day have a greater probability than the night for this function because during the day the sensors have a better visibility to detect line and then achieved the function. The other probabilities are conditional probabilities because there is a dependence. In the Day, we have in majority High luminosity than in the Night. For example in this figure, the probability of occurrence of High luminosity knowing that is the Day is equal to 0.87, while its probability knowing that is the Night is equal to 0.044.

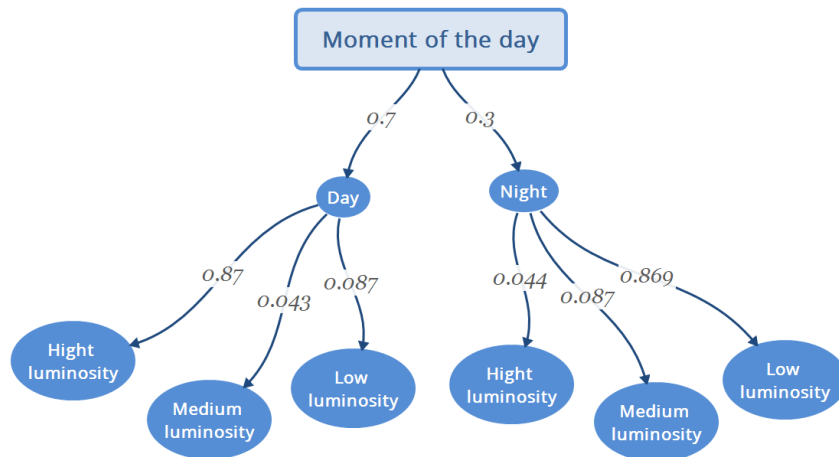


Figure 26: Example of probabilities of occurrence for the function LKA

We can notice that for each parameter the sum of the probabilities of its equivalence class has to be equal to one, in order to be coherent with mathematics.

These probabilities will be used in Markov algorithms for the automatic generation of scenarios for the test.

I.3 MCMC method choice and application

I.3.1 Scenario mathematic form

The goal of using MCMC algorithms is to generate scenarios, in the space parameter likely to influence the ADAS function. Thus, a generated scenario will consist of a set of influent parameters.

We will define the space $\Omega = \{X_s\} s \in S = \{1, 2, \dots, N\}$ composed of the different influent parameters for a given ADAS function (N being the maximum number of parameters).

Thus, in mathematical form a scenario can be written as a vector $X = (X_1, X_2, \dots, X_N)$;

$X_i, \{i=1 \dots N\}$ being an influent parameter taking its values in the set of equivalence classes related to him.

For example, if $X_i = \mathbf{Weather}$, its possible value will be in the set $\{\mathbf{Snow}, \mathbf{Rain}, \mathbf{Cloudy}, \mathbf{Fog}, \mathbf{Dry}\}$.

I.3.2 Choice of the MCMC method and application

As we assume scenario as a vector $X = (X_1, X_2, \dots, X_N)$, it could be consider as multivariate random variable. According to [17], the Gibbs sampler algorithm is more appropriate to these cases of variables than Metropolis Hastings algorithm. Also, as we can have all the full conditional probabilities (probabilities between equivalence class); given by the PSA ADAS Expert, the Gibbs sampler algorithm could be implement [5].

The algorithm used in our work is based on algorithm 2 of chapter 2. We will just a little revised it for a better use in our case of study.

The algorithm 2 is a periodic Gibbs sampler; this means that, at each iteration all parameters of a scenario vector is being sampled. Thus, taken a very long time if we have many parameters. To reduce time taken we propose two approaches:

- **Method 1:** At each iteration, sampling only the parameters of a given category that will be choose randomly.
- **Method 2:** At each iteration, sampling only one parameter randomly choose in a randomly chosen category.

Only the algorithm of the first proposal is presented here, for the second, confer to appendix 3.

If we note by C the set of top-level category (cf. section I.1.1 of this chapter) we have:

$C = \{\text{Physical Infrastructure, Operational Constraints, Objects, Environmental Conditions, Ego vehicle behaviour}\};$

We note M its number of elements.

A category C_i is composed of parameters: $C_i = \{X_{i1}, X_{i2}, \dots, X_{ip}\}$ “p” being the number of parameters in category C_i ; X_{ip} is a parameter that is also a set as described in *section I.3.1 of this chapter*.

We then have the following algorithm:

Algorithm3: Gibbs sampler method 1

1. At initial step, select $X^0 = (C_1^0, C_2^0, \dots, C_M^0)$, where $C_j^0 = \{X_{j1}^0, X_{j2}^0, \dots, X_{jp}^0\}$ and a probability vector $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_M)$ applied on the categories
 2. For $t = 1, 2, \dots, N$
 - a. Randomly choose C_j , $j \in \{1, 2, \dots, M\}$ with the probability α_j
 - b. Generate
$$X_{j1}^t \leftarrow P(X_{j1}^t | X_{j2}^{t-1}, X_{j3}^{t-1}, \dots, X_{jp}^{t-1})$$
$$X_{j2}^t \leftarrow P(X_{j2}^t | X_{j1}^t, X_{j3}^{t-1}, \dots, X_{jp}^{t-1})$$
$$\vdots$$
$$X_{jp}^t \leftarrow P(X_{jp}^t | X_{j1}^t, X_{j2}^t, \dots, X_{jp-1}^t)$$
 3. Repeat step 2 until reaching equilibrium.
-

We can notice that in this algorithm at the initial step we had to select an initial scenario vector and attribute a probability vector to the categories. We will explain how practically we will do that:

- **Defining the initial scenario vector $X^0 = (C_1^0, C_2^0, \dots, C_M^0)$:** the initial vector do not really influence the generated sequence, since with any starting point as the sequence becoming long, it will converge to a stationary distribution. It is only the first generated scenarios that are strongly correlated to the initial vector.

In the case of our work, since we want to resolve the SOTIF problematic, by generating scenario that are disregarded or otherwise be forgotten by the human who generate the scenario, we will choose as parameters for the initial vector, those having a minimal probability of occurrence.

- **Probability vector of categories $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_M)$:** it will be calculated under the base of parameters present in each category divided by the total number of parameters. Thus, the sum of the probability of this vector will be equal to 1.

$$\sum_{i=1}^M \alpha_i = 1$$

- The randomly choice of a category is done here concretely using the **mathematic uniform law** with discrete values.

- At each iteration, each parameter of the chosen category is sampler using the conditional distribution of this parameter knowing the values of all other parameters of this category.

I.4 Convergence diagnostic

After presenting the sampling algorithm that allows us to generate our scenarios, we might ask ourselves how many times will we have to go through the iterative loop before stopping, how long should we run the MCMC chain? Hence the notion of convergence that defines the stopping point.

To detect convergence, we going to use an approach of German & Rubin (1992) [20] that consists of generating multiple sequences and compare their variances. The different steps of this approach are:

- Generate independently $m \geq 2$ sequences, each of length $2n$ with different starting points.

In the case of our study, we going to generate $m = 4$ sequences with length $2n = 1000$ scenarios and different starting points that we choose as:

- A scenario with parameters the least probable
 - A scenario with the most probable parameters
 - A scenario with a mix of most and least probable parameters
 - A scenario with parameters chosen randomly
- To diminish the effect of the starting points, we going to discard the first n scenarios and focus our attention to the second n scenarios of each sequence. For those important scenarios we going to calculate the **variance between chain B and the mean within chain variance W**, given by the following equations:

$$B = \frac{1}{m-1} \sum_{j=1}^m (\bar{\pi}_j - \bar{\pi})^2 \quad \text{and} \quad W = \frac{1}{m} \sum_{j=1}^m \frac{1}{n-1} \sum_{i=1}^n (\pi_{i,j} - \bar{\pi}_j)^2 \quad [21]$$

With:
$$\bar{\pi}_j = \frac{1}{n} \sum_{i=1}^n \pi_{i,j} \quad \text{and} \quad \bar{\pi} = \frac{1}{m} \sum_{j=1}^m \bar{\pi}_j$$

- Then we going to calculate the estimate variance of the target distribution π , given by the following equation:

$$\sigma^2 = \frac{n-1}{n} W + \frac{1}{n} B \quad [21]$$

σ^2 overestimate the variance when chains are not reached convergence.

The convergence is reached when the potential scale reduction factor $\sqrt{R} = \sqrt{\sigma^2/W}$ tends to 1 when $n \rightarrow \infty$.

Concretely we going to increase the length of the sequence and redo those calculations until we obtain the convergence.

II IMPROVEMENT OF THE POST PROCESSING VALIDATION TOOL NODESAT

For the post processing of its test results, in order to know if they are conform to the requirements, PSA has designed and developed on Matlab a tool called NODESAT Post-Processing. As described in the chapter 1, this tool takes in entry the description file that contains the different scenarios and the expected results for each of them; he takes also as input the .erg files, which are the result files obtained after the execution on the simulation environment Carmaker. The goal is to compare both results in a defined time.

The figure 27 presents the graphical interface of the tool.

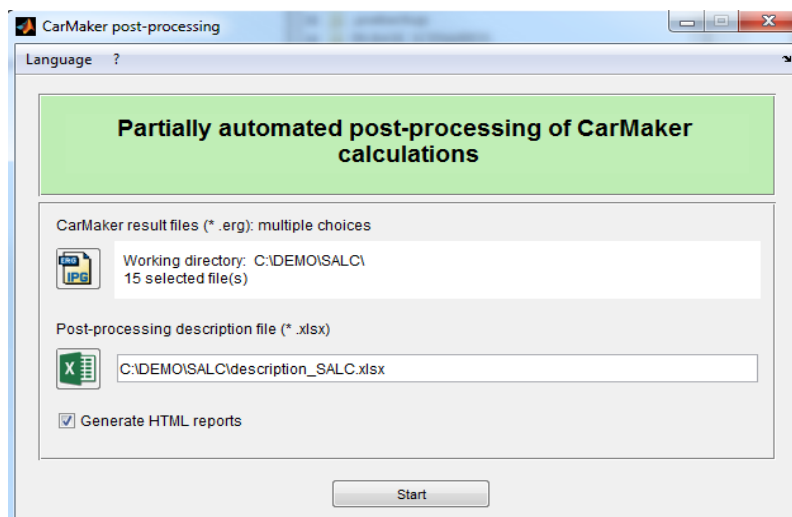


Figure 27: Interface GUI of the PSA post-processing tool

The next figure presents the detailed process of the post processing of results.

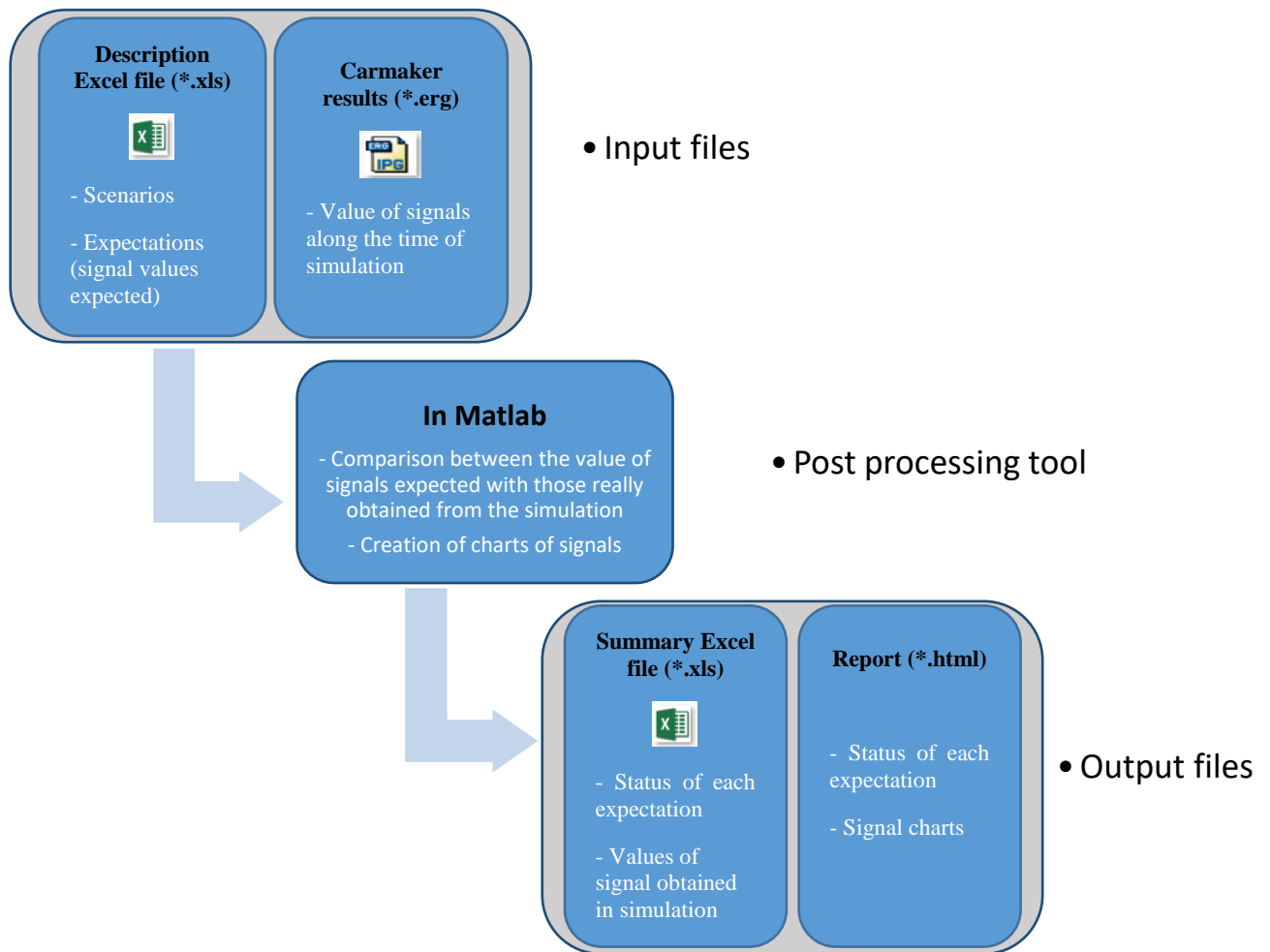


Figure 28: Process for the post processing

At the beginning of my internship at PSA, this tool briefly described above, was presenting some defects. So, one of my objectives was its improvement. To achieve that, I proceed as follow:

- **Collect all the defects of the tools:** I organized a meeting with the tool users (ADAS validation officers), during which I asked them some questions about problems that they encounter with the tool, in order for me to well understand each of it. I resumed all the defects in a document called QIA (Inter-Active Questioning) at PSA. This latter will help me for a traceability and a better organization of my work. The major defects that I collected are resumed as follow:
 - Suppression of the summary Excel generation file and directly integrate in the description file, the status of each expectation;
 - Enable the possibility for the user to choose which signal charts he would like to display in his html report.

- Integrate an acceptable error margin in the comparison between the values of the signals expected and those obtained with the simulation, for the assignment of test status.
- **Proposing solutions for the correction and improvement of the tool:** to better apprehend the different problems, I first tried to understand the entry files of the defect tool.
 - The description file: I wrote by myself a description test file for the ADAS function SLI (panel detection) of the camera and Gateway (communication function between the trams) of the radar. This helped me to carry out the limitation of the document, and to propose solution for its improvement.
 - Carmaker result: I simulated on the HIL bench, the scenarios defined in the above description file and I got the different result files. This helped me to perceive and understand the signals composing the .erg file.
 - I launched the post processing of my results on the Nodesat tool, in order for me to see how the summary file and the html report are presented.

I then proposed solutions:

- I made up a new template for the description file. Compared to the latest one, here there is the possibility for each testrun that we want to simulate, to choose the relevant signals that we will like to display in the html report. The figure 29 presents the tab of the Excel file where this maneuver has to be done. For each testrun of column A, he choose the signals that he want in the next columns.

A	B	C	D
Variation names (noms des fichiers variations)	Carmaker signals to display in the report (multiple values separated by semicolons ";" in the same cell displays the values on the same axis)	Additional calculations (XXX = Expression)	Units (for addition/ calculation)
Perception_Video_Object_Info_AEB	car.v (vitesse véhicule);car.Distance;Car.alHori;Car.alHori;		
	car.v (vitesse véhicule);car.Distance;		
	CAN_P1.15ch.CVM.CAM_OBJ_CLASS;	<input type="checkbox"/> car.v (vitesse véhicule)	
		<input checked="" type="checkbox"/> car.Distance	
Perception_Video_Object_Info_Oncoming_Target_Car		<input checked="" type="checkbox"/> CAN_P1.15ch.CVM.CAM_OBJ_CLASS	
		<input type="checkbox"/> Car.alHori	
		<input type="checkbox"/> CAN_P1.15ch.CVM.CAM_OBJ_RELAT_VELO_LONGI	

Figure 29: The choice of signals to display in the report

Macro VBA has been implemented to facilitate the use of this document.

In addition, I added the possibility for the user to define the error margin he can permit for the expectation checking. In the following figure, we can see the influence of the error margin that the user filled, and the status obtained after the post processing of test results with the tool I improved by Matlab coding using mathematic algorithm.

When the error margin is 25% the test is True, while when the error permitted is 5% the test is False.

G	H	J	K	L
Signals	Reference Time	Verification Time	Error margin	Status
-10 <= CAN_FD3.11eh.CVM.SEC_DIST_VHL_R_LINE_EXT(tDistance2) & CAN_FD3.11eh.CVM.SEC_DIST_VHL_R_LINE_EXT(tDistance2) < 0 & 4.889 < CAN_FD3.11eh.CVM.SEC_DIST_VHL_L_LINE_EXT(tDistance2) & CAN_FD3.11eh.CVM.SEC_DIST_VHL_L_LINE_EXT(tDistance2) <= 10	tDistance2	find(Car.Distance>=950 & Car.Distance<=1100)	5,00%	FALSE

G	H	J	K	L
Signals	Reference Time	Verification Time	Error margin	Status
-10 <= CAN_FD3.11eh.CVM.SEC_DIST_VHL_R_LINE_EXT(tDistance2) & CAN_FD3.11eh.CVM.SEC_DIST_VHL_R_LINE_EXT(tDistance2) < 0 & 4.889 < CAN_FD3.11eh.CVM.SEC_DIST_VHL_L_LINE_EXT(tDistance2) & CAN_FD3.11eh.CVM.SEC_DIST_VHL_L_LINE_EXT(tDistance2) <= 10	tDistance2	find(Car.Distance>=950 & Car.Distance<=1100)	25,00%	TRUE

Concretely, in the code what I do, is to first find the corresponding time value using the simulation result; for this example with the verification time equation, I found:

tDistance2 = [42.85 – 49.60]; Car.Distance is the distance driven by the ego car.

<u>tDistance2</u>	find(Car.Distance>=950 & Car.Distance<=1100)	42.85 -> 49.60
-------------------	--	----------------

Then, in this corresponding time, I calculate the percent of index that did not verify the signal expectation and I compared this result with the error margin imposed by the user. If the percent that I calculated is less than what defined by the user then the test is True, otherwise the test is False. When the user leave the cell of margin error empty, I considered in the status checking that no margin error is permitted.

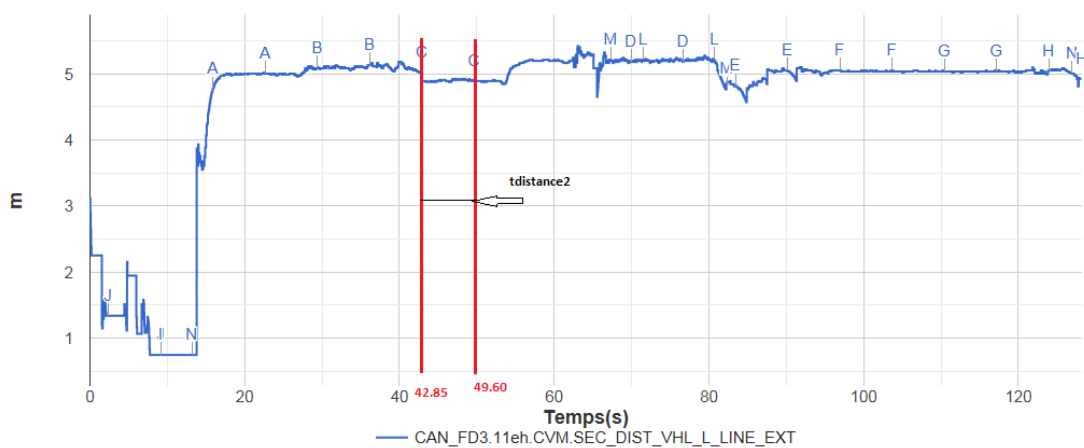


Figure 30: Chart of the signal CAN_FD3.11eh.CVM.SEC_DIST_VHL_L_LINE_EXT during the simulation time

At the output of the tool, I generate also a .log file where I resumed all the error of warning obtained during the post-processing.

To clarify this margin error explanation, I will give some examples of cases where the error can be allowed and where it cannot.

- A case of a scenario where the ego car should detect a motorcycle that is driven before him. The detection signal need to be equal to 1 (an example of value) to state that it well detect the motorcycle. In the detection time, it is not permitted that the detection signal takes another value (example a value stating that it is detecting a pedestrian while it should detect a motorcycle), so no margin error is permitted in this case.
- A case where an error margin could be permitted is for example at a beginning of a scenario of detection; the time taken by the camera for its initialization before starting the detection. We could state an error margin that will correspond to that time where the

detection signal will be not equal to its expected value, because the camera will not yet initialized.

Therefore, the user will not define the error margin with hazard.

I will conclude this section by saying that the global objective was achieved. I made up the tool using the same GUI interface like the previous one, and it will now be the new post processing tool for the ADAS validation in the CVAD entity of PSA. Moreover, the description template that I presented will be the new model for writing their test plan.

Chapter 4:

PRESENTATION OF THE RESULTS AND DISCUSSION

I CREATION OF INPUT DATA

As presented in the methodology, the first step of our work consists on choosing the parameters that influence the ADAS function, then classify them into equivalence class and assign to them probabilities of occurrence.

The choice of parameters and their classification have been done using the ODD taxonomy presented in the methodology and by reading the requirement documents for some function to know which parameters influence their behaviour. After, I thought about how I will present these parameters, so that it will be easy for the user to complete the necessary probabilities and for me to easily manipulate them in the programming tool.

My first idea was to present them in graphs, drawing each category in form of graph where there will be a link between an equivalence class and its dependence. I started drawing the categories using the software yEd Graph Editor, but I found many difficulties:

- When a category have many equivalence class, it is difficult to draw and to read the graph;
- If there is another category that the user will want to add, there will be a need for him to know how to use the tool in order for him to draw the graph;
- The connection between the yEd tool with my programming tool python, took me a lot of time to implement.

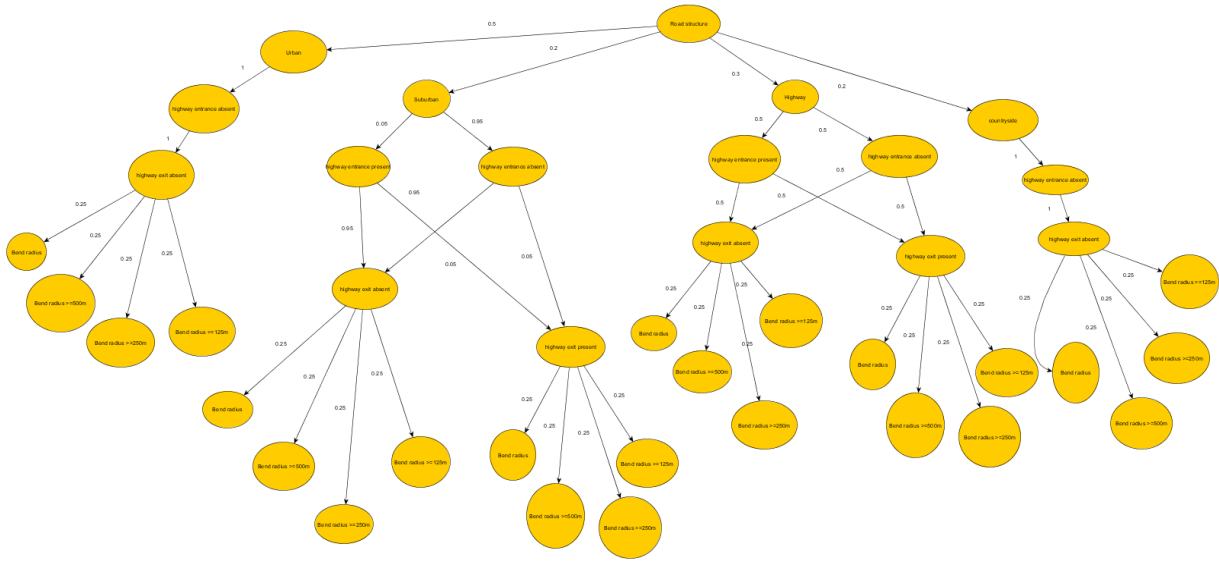


Figure 31: Example of unreadable drawing graph in yEd software

Therefore, we decided to change the form for the presentation of the parameters.

We built up an Excel file where we write all these parameters in a “**user-friendly**” form, so that the user could easily complete the different probabilities needed in the Gibbs algorithm for the automatic generation.

The Excel file is presented as follow:

Category name	Parameter name	Function name	Equivalence class 1			Equivalence class 2			Equivalence class 3			Equivalence class 4			Equivalence class 5			
			Name	Dependance	Proba	Name	Dependance	Proba	Name	Dependance	Proba	Name	Dependance	Proba	Name	Dependance	Proba	
Environmental conditions	Day / Night	X	Day	-	0,7	Night	-	0,3										
	Luminosity	X	High luminosity	Night		0,044	Medium luminosity	Night		0,087	Low luminosity	Night		0,869				
				Day		0,87		Day		0,087		Day		0,043				
	Weather	X	Dry	-	0,5	Cloudy	-	0,2	Rainy	-	0,2	Fog	-	0,07	Snow	-	0,03	
	Road maskings	X	No masking	Dry		0,7	Water slabs	Dry		0,2	Snow slabs	Dry		0,2				
				Cloudy		0,6		Cloudy		0,2		Cloudy		0,2				
Rainy					0,1	Rainy			0,7	Rainy			0,2					
Fog					0,6	Fog			0,3	Fog			0,1					
Snow					0,1	Snow			0,2	Snow			0,7					
Physical infrastructures	Type of road	X	Urban	-	0,5	Suburban	-	0,2	Motorway	-	0,15	Countryside	-	0,15				
	Number of lanes	X	1 lane road	Countryside		0,5	2 lanes road	Countryside		0,5	3 lanes road			4 lanes road				
Urban					0,4	Urban			0,4	Urban			0,2					
Suburban					0,3	Suburban			0,6	Suburban			0,1					
						Motorway			0,3	Motorway			0,3		Motorway		0,4	

Figure 32: File for the input parameters

The first column is the category names. It is followed by the names of the parameters in each of those categories. The user fills the third column; he had to write the X symbol into the cell when the corresponding parameter influence the ADAS function on which he is working. In the case this parameter does not influence his function, he leaves the cell empty. The next columns are the different equivalence class for each parameter. It is divided into three parts: the name of the equivalence class, its dependence (if there is nothing we put the symbol -) and then the probabilities that have to be completed by the user.

The sum of the probabilities following a line should be equal to 1 in order to be coherent with mathematics, as explained above. I built up a macro VBA to remind it to the user when he fills the file.

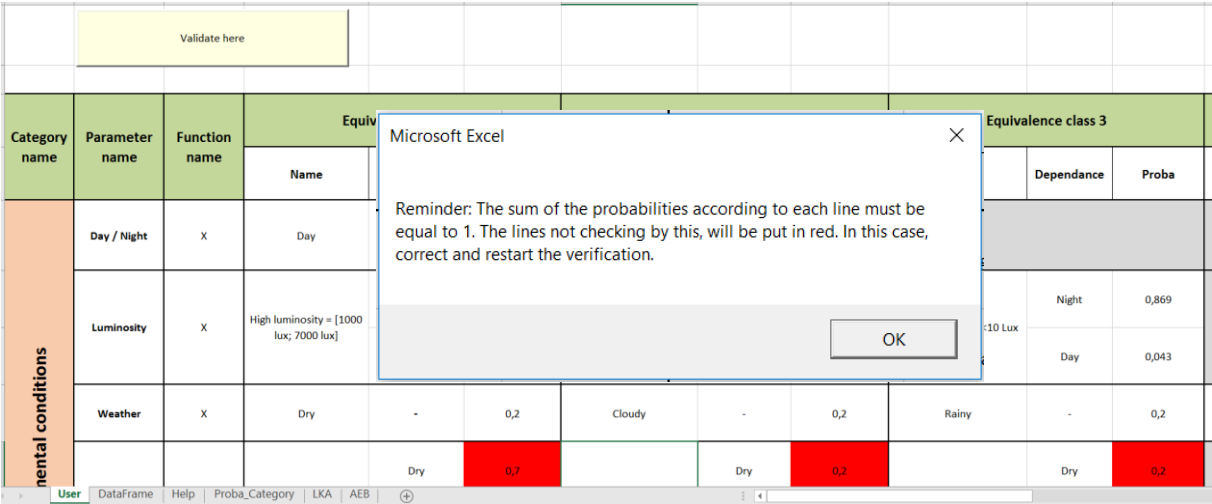


Figure 33: Verification reminder

For the connection between this Excel form and my programming tool, I added a tab in the Excel file. In this tab, I copy automatically using a macro VBA, the information from this first tab in the second one, but in the form easy to use and manipulate in the programming tool. This easy form is the DataFrame form used in python and other programming tool.

Then, is from this second tab, that the programming tool takes information to generate scenarios.

Category_Name	Parameter_Name	Function_Name	Equivalence_Class_Name	Dependance	Probability
Environmental conditions	Day / Night	X	Day	-	0,5
Environmental conditions	Day / Night	X	Night	-	0,5
Environmental conditions	Luminosity	X	High luminosity = [1000 lux; 7000 lux]	Night	0,044
Environmental conditions	Luminosity	X	High luminosity = [1000 lux; 7000 lux]	Day	0,87
Environmental conditions	Luminosity	X	Medium luminosity = [10 lux ; 1000 lux]	Night	0,087
Environmental conditions	Luminosity	X	Medium luminosity = [10 lux ; 1000 lux]	Day	0,087
Environmental conditions	Luminosity	X	Low luminosity = <10 Lux	Night	0,869
Environmental conditions	Luminosity	X	Low luminosity = <10 Lux	Day	0,043
Environmental conditions	Weather	X	Dry	-	0,2
Environmental conditions	Weather	X	Cloudy	-	0,2
Environmental conditions	Weather	X	Rainy	-	0,2
Environmental conditions	Weather	X	Fog	-	0,2
Environmental conditions	Weather	X	Snow	-	0,2
Environmental conditions	Road maskings	X	No masking	Dry	0,6

Figure 34: Excel DataFrame

In the next figure, we can see how python converts this Excel DataFrame tab exactly as the same manner. Therefore, it is easy to link each parameter to its dependance and probability.

Index	Function_Name	ivalence_Class_Na	Dependance	Probability
Tarmack	X	Guard rail	Suburban	0.5
Tarmack	X	Guard rail	Motorway	0.2
Tarmack	X	Guard rail	Urban	0.15
Tarmack	X	Guard rail	Countryside	0.3
Tarmack	X	Guard Rail (double)	Suburban	0.5
Tarmack	X	Guard Rail (double)	Motorway	0.2
Tarmack	X	Guard Rail (double)	Urban	0.15
Tarmack	X	Guard Rail (double)	Countryside	0.3
Tarmack	X	Jersey Barrier (wall)	Motorway	0.2
Tarmack	X	Jersey Barrier (wall)	Urban	0.05
Tarmack	X	Jersey Barrier (wall)	Countryside	0.05

Figure 35: Python DataFrame: example for the parameter Tarmack

The fact of having imposed to the user to select with a symbol X his relevant parameters, help us in the code to delete lines of the unselected cells, in order to work only on the important parameters.

We made available to the user a document that explain the Excel file, what are the different columns and how to complete it.

II SCENARIOS GENERATIONS

To program the two algorithms described in the precedent chapter, we used Python programming tool. The code takes as input the Excel file presented above and generates as output an Excel file, whose parameters in a given line constitute a scenario.

In the figure below, we present the Graphical User Interface for our code. We name it the **MCMC Test Tool**. The user will upload its input Excel file, and choose the directory where the result will be registered. Then the generation process can be launched/started.

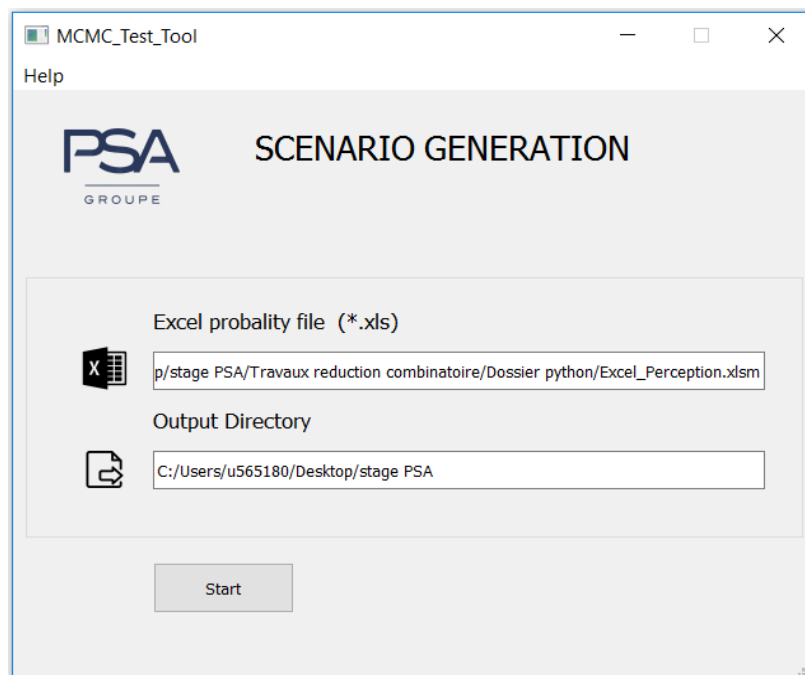


Figure 36: Python code interface

We will describe concretely how the algorithm is implemented in python, how we step from a scenario to another one.

Suppose we have an initial scenario, chosen as described in the methodology chapter.

For the next 5000 scenarios, at each iteration we will randomly choose a category with a function that we implemented called `choice_Category`. Then, each parameter of this chosen category will be sampled.

```

new_sampler = initial_sampler

for K in range(5000):

    #choice of the changing category to modify by considering the probabilities
    num_category = G.choice_Category()

    # sampling of all the parameters of this category
    for i in range(len(G.Parametres[num_category])):

```

For each parameter, we will look if there is dependence or not.

```

if (param.iat[0,2] == u'-'): #if there is no dependance
    #sampler directly on it probabilities' values
    index = G.Indice(num_category, i, new_sampler)
    [new_sampler[index],echantillon_proba_param[index]] = G.nouvel_val_param(param)

else: #if it has dependance, look for his dependance before sampling
    [index2, num_cliq_param] = G.Indice_Param (param)
    index3 = G.Indice(num_cliq_param,index2, new_sampler)
    dependance = new_sampler[index3]
    param_2 = param[param[u'Dependance']== dependance]
    index = G.Indice(num_category,i, new_sampler)
    [new_sampler[index],echantillon_proba_param[index]] = G.nouvel_val_param(param_2)

```

If there is no dependence as is the case for this selected parameter in the next figure, the algorithm will directly sample from the probability distribution of this parameter.

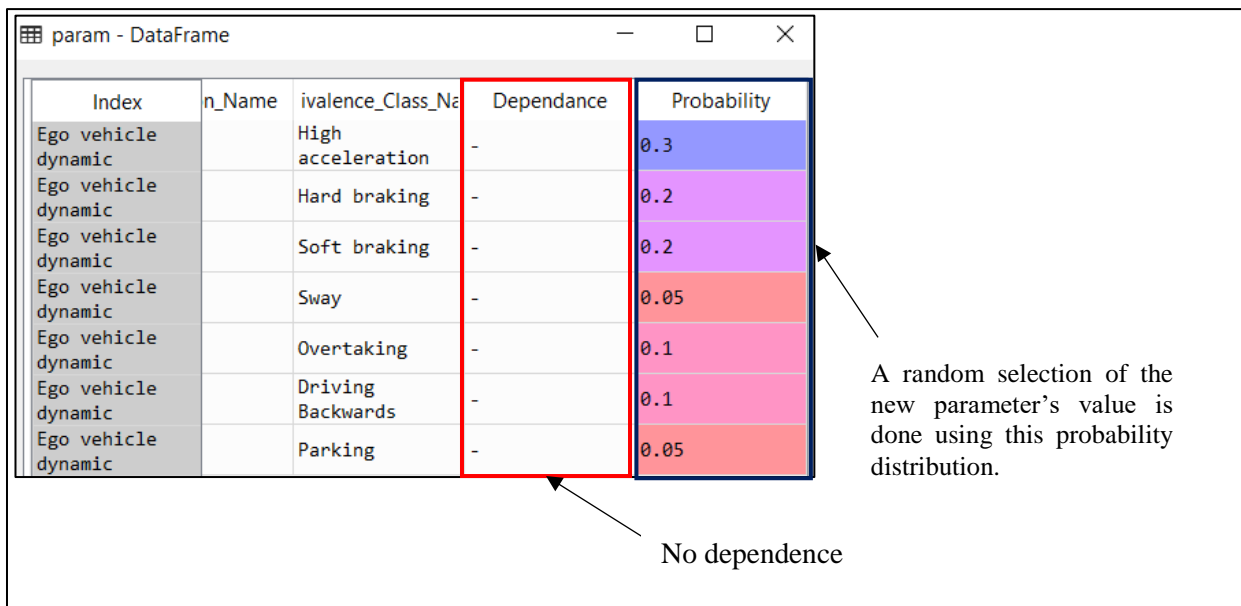


Figure 37: Case of sampling without dependence

While if there is dependence, the figure 38 below explains the different steps.

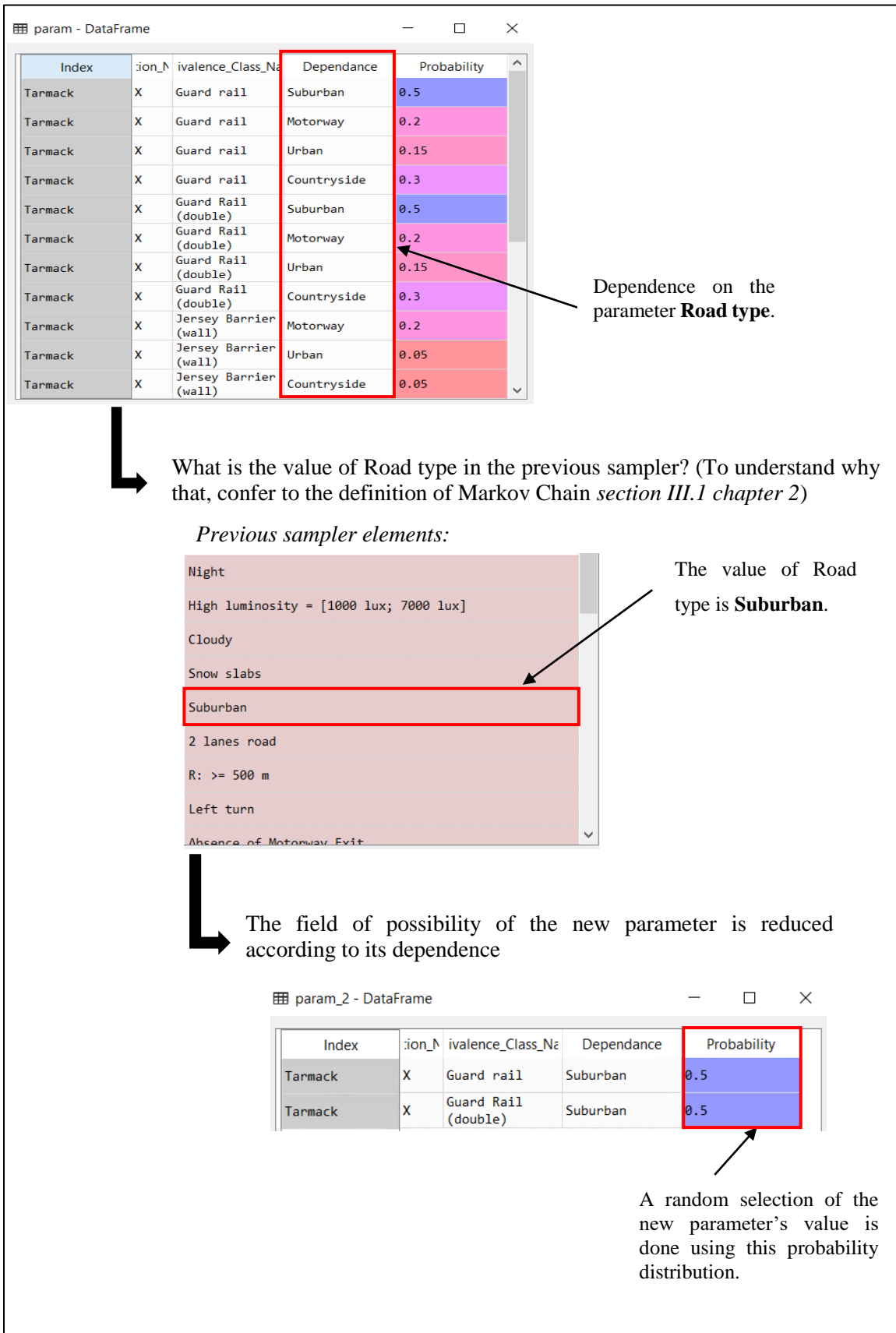


Figure 38: Case of sampling with dependence

Here in figure 39, an example of scenarios obtained in the output Excel file with method 1.

AEB function														
category 1				category 2								category 3		
Day	Low luminosity	Dry	Water slabs	Urban	3 lanes road	Straight road	Toll	Asphalt	Flat drop	Tunnel present	Work on road Present	Bridge present	Normal traffic	Stop
Day	Low luminosity	Dry	Water slabs	Urban	1 lane road	Straight road	Gyrator	Concrete	Flat drop	Tunnel absent	Work on road Present	Bridge present	Normal traffic	Stop
Day	Low luminosity	Dry	Water slabs	Motorway	3 lanes road	R: >= 250 m	Yintersection	Asphalt	Ascending drop	Tunnel present	Work on road Absent	Bridge present	Normal traffic	Stop
Day	Low luminosity	Dry	Water slabs	Motorway	3 lanes road	R: >= 250 m	Yintersection	Asphalt	Ascending drop	Tunnel present	Work on road Absent	Bridge present	Normal traffic	Traffic Sign
Day	Low luminosity	Dry	Water slabs	Motorway	2 lanes road	R: >= 250 m	Toll	Asphalt	Flat drop	Tunnel present	Work on road Present	Bridge absent	Normal traffic	Traffic Sign

category 4					category 5								
Initial lane ego vehicle = 1	0 <=Speed ego vehicle < 60km/h	High acceleration	Windshield wiper ON	Outbreak longitudinal distance = Low	0 <=Speed traffic vehicle < 60km/h	High acceleration	Shift: [0.5; 1m[Shift on the Left	Animals	Crossing road	No vehicle is parked	Trompe œil	
Initial lane ego vehicle = 1	0 <=Speed ego vehicle < 60km/h	High acceleration	Windshield wiper ON	Outbreak longitudinal distance = Low	0 <=Speed traffic vehicle < 60km/h	High acceleration	Shift: [0.5; 1m[Shift on the Left	Animals	Crossing road	No vehicle is parked	Trompe œil	
Initial lane ego vehicle = 1	0 <=Speed ego vehicle < 60km/h	High acceleration	Windshield wiper ON	Outbreak longitudinal distance = Low	0 <=Speed traffic vehicle < 60km/h	High acceleration	Shift: [0.5; 1m[Shift on the Left	Animals	Crossing road	No vehicle is parked	Trompe œil	
Initial lane ego vehicle = 1	0 <=Speed ego vehicle < 60km/h	High acceleration	Windshield wiper ON	Outbreak longitudinal distance = Low	0 <=Speed traffic vehicle < 60km/h	High acceleration	Shift: [0.5; 1m[Shift on the Left	Animals	Crossing road	No vehicle is parked	Trompe œil	
Initial lane ego vehicle = 1	0 <=Speed ego vehicle < 60km/h	High acceleration	Windshield wiper ON	Outbreak longitudinal distance = Low	0 <=Speed traffic vehicle < 60km/h	High acceleration	Shift: [0.5; 1m[Shift on the Left	Animals	Crossing road	No vehicle is parked	Trompe œil	

Figure 39: Example of output scenario of method 1

In the case of the five scenarios (five lines) present in this picture, only the category 2 was randomly chosen at each iteration of the algorithm. It is the reason why only its parameters change and those of other categories remain unchanged. The others categories have been chosen in other scenarios, but I could not present the entire file in one picture.

Unlike method 1, in method 2, only one parameter change from one scenario to another. In the figure 40, we can see the parameter vehicle dynamic of category 5, which change at the next generation.

Category 1				Category 2								Category 3		
Night	High luminosity	Snow	No masking	Motorway	2 lanes road	Straight road	Yintersection	Asphalt	Ascending drop	Tunnel present	Work on road Present	Bridge present	Minimal traffic	Stop
Night	High luminosity	Snow	No masking	Motorway	2 lanes road	Straight road	Yintersection	Asphalt	Ascending drop	Tunnel present	Work on road Present	Bridge present	Minimal traffic	Stop

Category 4					Category 5								
Initial lane ego vehicle = 1	60 km/h <=Speed ego vehicle <	Sway	Windshield wiper ON	Outbreak longitudinal distance = Far	Speed traffic vehicle >= 100km/h	Sway	Shift: [1m50;2m[Shift on the Left	Animals	Crossing road	Vehicle parked on the right side	Car on trailer	
Initial lane ego vehicle = 1	60 km/h <=Speed ego vehicle <	Sway	Windshield wiper ON	Outbreak longitudinal distance = Far	Speed traffic vehicle >= 100km/h	High acceleration	Shift: [1m50;2m[Shift on the Left	Animals	Crossing road	Vehicle parked on the right side	Car on trailer	

Figure 40: Example of output scenarios for method 2

The difference between the two methods is presented in the following table:

Table 2: Comparison between the two methods

Algorithm	Method 1	Method 2
Number of samples	2500	2500
Percent of duplicates	2,40%	41%
Running time	2min10s	21s

Method 1 generates less duplicates but take more time, while the second generates more duplicates and take less time.

For the convergence diagnostic, we generated four sequences with different initial points as described in the methodology. We started with a length of 100 scenarios and increase the length until the potential factor converge to one.

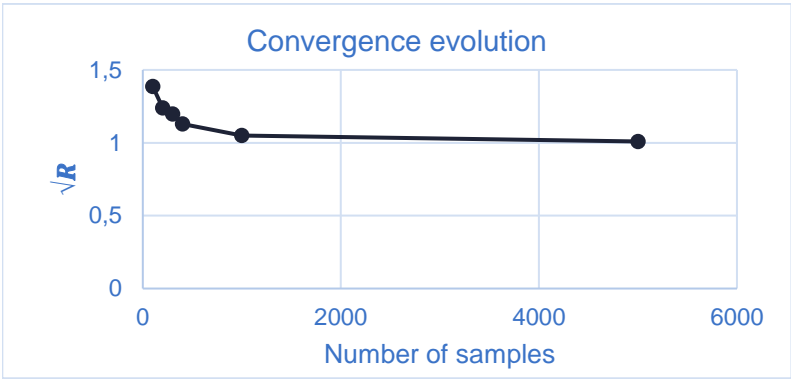
If we go back to the different formulas to calculate in order to diagnostic the convergence, we will see that there is a need of the joint probability. This value is directly obtained in the output Excel file in the tab called '**Probability**' of the scenarios generation for each sequence.

A	B	C	D	E	F	G
category 1	category 2	category 3	category 4	category 5		Joint probability π
0,00086	1,4648E-07	0,05	0,00125	0,000003		2,36206E-20
0,00086	1,9531E-07	0,05	0,00125	0,000003		3,14941E-20
0,00086	1,9531E-07	0,1	0,00125	0,000003		6,29883E-20
0,00086	1,9531E-07	0,1	0,00125	0,025725		5,40125E-16
0,00174	1,9531E-07	0,1	0,00125	0,025725		1,09281E-15
0,00174	1,9531E-07	0,1	0,00125	0,002205		9,36694E-17
0,00174	1,9531E-07	0,1	0,00125	0,00686		2,91416E-16
0,00174	1,9531E-07	0,1	0,0105	0,00686		2,44789E-15
0,00174	1,1719E-06	0,1	0,0105	0,00686		1,46874E-14
0,01738	1,1719E-06	0,1	0,0105	0,00686		1,46705E-13
0,01738	1,1719E-06	0,1	0,0105	0,000216		4,61928E-15
0,01738	1,1719E-06	0,05	0,0105	0,000216		2,30964E-15
0,01738	1,0986E-06	0,05	0,0105	0,000216		2,16529E-15
0,01738	1,0986E-07	0,05	0,0105	0,000216		2,16529E-16
0,01738	2,1973E-07	0,05	0,0105	0,000216		4,33057E-16
0,01738	2,1973E-07	0,05	0,0105	0,00147		2,9472E-15
0,01738	5,8594E-06	0,05	0,0105	0,00147		7,85919E-14
0,01738	2,9297E-07	0,05	0,0105	0,00147		3,92959E-15
0,05214	2,9297E-07	0,05	0,0105	0,00147		1,17888E-14
0,05214	2,9297E-07	0,05	0,063	0,00147		7,07327E-14
0,05214	2,9297E-07	0,05	0,063	0,0011025		5,30495E-14
0,05214	2,9297E-07	0,05	0,063	0,003675		1,76832E-13
0,05214	2,1973E-06	0,05	0,063	0,003675		1,32624E-12
0,0609	2,1973E-06	0,05	0,063	0,003675		1,54906E-12
0,0609	4,3945E-06	0,05	0,063	0,003675		3,09812E-12
0,0609	5,8594E-07	0,05	0,063	0,003675		4,13082E-13
0,0609	5,8594E-07	0,05	0,063	0,003675		4,13082E-13
0,0609	5,8594E-07	0,05	0,0525	0,003675		3,44235E-13
0,0609	5,8594E-07	0,05	0,0525	0,003675		3,44235E-13
0,0609	5,8594E-07	0,05	0,0525	0,00049		4,5898E-14
0,0609	5,8594E-07	0,05	0,0525	0,00098		9,1796E-14
0,0609	2,9297E-06	0,05	0,0525	0,00098		4,5898E-13

Figure 41: Probability tab in the output Excel file

We then used those different joint probabilities to make our calculations, and we obtained the following results:

n	\sqrt{R}
100	1,38536227
200	1,23865952
300	1,19784319
400	1,12967112
1000	1,05060458
5000	1,00914174



For 5000 generated scenarios, we assumed that the convergence has been reached as the factor \sqrt{R} tends to 1.

For the suppression of duplicates, we used a command of Excel, which delete all the duplicates in a selected range.

III TEST CASES GENERATION

In order to test my scenario results on HIL bench, I worked with another student in apprenticeship at PSA to transform my textual scenarios in testruns, which are scenarios in the simulation environment (Carmaker for PSA).

The requirement to do that job was a very good knowledge of Carmaker and of its internal functioning. Having a large number of scenarios, we design another tool to generate automatically the testruns, by taking in entry our scenarios.

Concretely, we drawn separately each parameter of a scenario in Carmaker (example in figure 42). Moreover, the tool works in grouping together the drawn parameters of a given scenario, in order to constitute its testrun.



Figure 42: Example of parameters drawn in carmaker

For example, the testrun resulting of the scenario of table 3 below is presented in figure 43.

Table 3: Example of scenario

Night	High luminosity	Snow	No masking
Urban	3 lanes road	Absence of Motorway Exit	Presence of Motorway Entrance
Guard rail	Initial lane ego vehicle = 1	Initial lane other vehicle =1	60 km/h <=Speed other vehicle < 100km/h
60 km/h <=Speed ego vehicle < 100km/h	Straight road	Windshield wiper ON	Dark coating
Slot Present	Markings partially deleted	Tunnel present	Skid Present
Work on road Present	Bridge present		

The parameters in red colour are non-possible to be generate automatically with the generation code. They can only be changed manually in the carmaker interface, so for a given testrun, we generate automatically all other elements and we complete the other elements manually.



Figure 43: Example of testrun

IV DISCUSSION

IV.1 Limitation of the approach

The approach presented in the methodology for the scenarios generation, allows to generate scenarios with only one equivalence class for a given parameter. This means that, for a given influent parameter, we cannot have more than one of its equivalence class in the generated scenario; this is because the goal of the classification in equivalence class was to avoid impossible scenarios, like Day and Night in the same scenario. Thus for the parameter “**moment of the day**” for example, we considered “**Day**” as one equivalence class (this includes all hours where the sky is clear) and “**Night**” for another one (this includes all the hours of the day where the sky is dark).

However, this classification in equivalence class presents a limitation that we will call “unitary scenario”. To clarify this statement, we are going to take an example.

Let us take the parameter « **obstacles** » which is divided in equivalence class: “**Pedestrians**”, “**Cyclists**”, “**Animals**”, “**Vehicles (cars, motorcycles, trucks, buses)**” and “**Other obstacles**”.

Then, with our approach of generation, we are not able to have in the same scenario together a pedestrian, a cyclist, an animal... However, in the reality, the autonomous vehicle can face situations with all those parameters.

Thus, our approach helps to avoid impossible scenarios, and generates only some possible case of test cases that can happen during a road driving.

IV.2 How our results will be used practically for the tests?

The goal of our work was to generate scenarios for the validation of a given ADAS function, such as SLI, LKA and AEB in HIL test level.

This means that by playing our scenarios, in the simulation environment with the different sensors and calculators connected, we could test all the requirements related to the different modules of that function (perception, decision, communication ...).

Let us take for example the function LDW (Lane Departure Warning) of the camera, which consists in alerting the driver if the vehicle crosses a line of his lane without any driver’s intention. The camera contributes to this function by finding out the marking of the lane, the

position of the host vehicle inside his lane and finally by sending an alert signal on CAN network. This signal will be used to warn the driver.

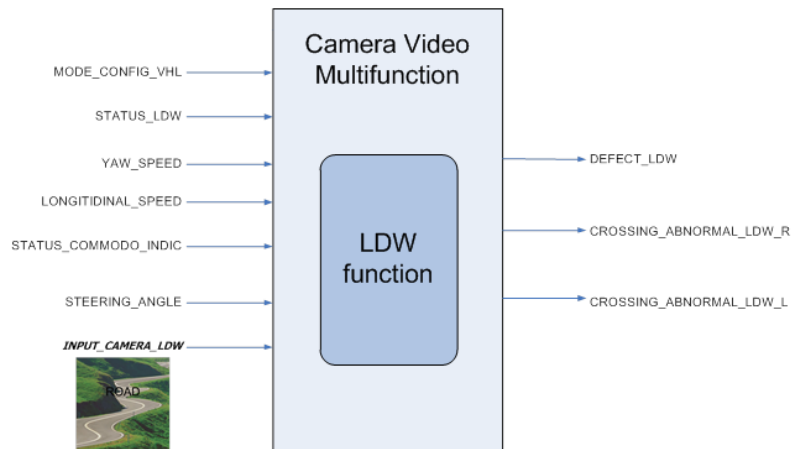
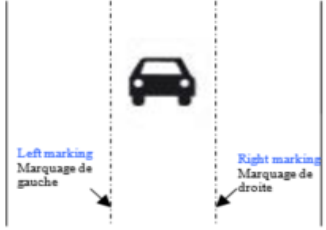


Figure 44: Signals for the LDW function [Source: PSA document]

If we take the following requirements for HIL validation of this function (Source: PSA document):

1. detection

<p>GEN-VHL-CD_CVM4_0546(1)</p>	<p>The CVM shall detect the marking in the lane where the vehicle is traveling and identifies the left and right markings :</p> 	<p>GEN-VHL-DC-AFIL-0075(5) GEN-VHL-DC-AFIL.0523(0)</p>
--------------------------------	---	--

2. fail safe mode

<p>GEN-VHL-CD_CVM4_0567(2)</p>	<p>IF <u>YAW_SPEED</u> is invalid OR lost during less than 100ms THEN Last received valid value is used for LDW function</p>	<p>GEN-VHL-DC-AFIL.0374(5)</p>
--------------------------------	--	--------------------------------

The requirement 1 for the markings detection can be tested by simulating the camera in our generated scenarios (for example scenarios with road marking). However, the second one for fail safe mode need that the signal YAW_SPEED should be command manually by the validation officer to achieve the given instruction and thus test this requirement.

So only by playing our scenario, we cannot test and validate all the requirements for a given function, thus we cannot say our scenario can validate a function, but only some module of that function, as precisely the module “**perception**”. This is because the perception module does not need another information or signals exterior to the scenario information to test the requirement.

This conclusion is only for the HIL test.

IV.3 How confident we are about the coverage of all possible requirements that the ADAS function must satisfy?

As discussed in the precedent section, only the module ‘perception’ of different functions could be test and validated on the HIL bench with our generated scenarios. Thus, to ensure ourselves for good coverage of the perception requirements by our generated scenarios, we first generated scenarios for the global perception of the camera and radar and not for a particular function as we done at the beginning.

To achieve that, I read the component design or the requirement documents related to the perception of the camera and radar of PSA. Then, I understood the different parameters that influence the perception layer in order to select them in my entry Excel file. To assign the probabilities, I discussed with the ADAS validation responsible, who told me the different probabilities of occurrence of those parameters. Then, with all these information, I generated automatically 5000 scenarios that we simulated on HIL bench, and then the results of this simulation had to be used in a software called “**Stimulus**” to evaluate the coverage level of the requirements by the generated scenarios.

This latter work could not be done in the available time, this because it needed the availability and group work with another PSA employees who have better knowledge on the Stimulus tool. These persons were not available on time unfortunately, but I hope by the end of my internship even I could not presented it in my document because it will be already submitted, that I could obtained these coverage results when those persons will be available.

GENERAL CONCLUSION AND PERSPECTIVES

I Contribution

The purpose of this thesis was the “**Optimisation of ADAS-AD validation process**”, as an endeavor to reduce the validation time and to propose an efficient method for reliably validate ADAS functions. To achieve that, we designed a tool that we called MCMC Test Tool, which takes in entry an Excel file that we already predefined so that the user will only choose his most relevant parameters and define their probabilities. Then, he will upload the Excel file in the tool and launch; our developed tool will automatically generated in less of 5min 5000 scenarios for his use.

In addition, we improved the post-processing tool of PSA, we made up a new template of the Excel description file easier than the previous for the use, and with more possibilities. The validation officer can select for each testrun that he want to simulate, the signals to display in his report for the better analysis of his results. Also, he can define a margin that will be used in the tool for determining the test status.

II Assignment

The first task was to use MCMC method for the generation of scenarios. To achieve that, we first defined what a scenario is, and analyzed the parameters that compose it. Then, we defined a list of those parameters likely to influence an ADAS function; with the use of the ODD taxonomy and of our proper knowledge acquire from the reading and studying of the requirements document for the Camera and Radar of PSA. To avoid impossible scenarios, we classified the parameters in equivalence class based on our knowledge of the parameters and our research. We continued by choosing the MCMC method, precisely the Gibbs sampler that we used; this choice has been made because there was the possibility to have the full probabilities of occurrence of the different parameters given by a person having a deep knowledge of ADAS’s operation. We found a way in which the user could easily defined the information needed, and we chose the Excel file, which is the easiest one. We implemented the algorithm in the tool and we found a method to connect the information in the Excel file with our code. We can say that the goal was globally achieved, because we succeeded to obtain results with two different algorithms that we proposed.

The second task was, if possible to test our results on HIL benches. For that, I worked with another apprentice at PSA to implement a tool in python, which will take our generated

scenarios in entry and will create a Carmaker testrun for each of them in an automatic manner. I could not do this second task by myself, because my principal task and the next that I will present, took me a lot of time and concentration, so I could not have enough time to deeply my knowledge on the simulation tool Carmaker. We succeeded on creating testruns corresponding to the number of scenarios and we launched it on HIL bench to get the .erg file. The next step, as we have a large number of .erg files, was to launch it in a software called stimulus in order to obtain the coverage and the status of each test. Unfortunately, this latter work could not be done on time.

The last task was the improvement of the post-processing tool NODESAT. We can say that this goal was globally achieved. We succeeded to implement the tool on Matlab, which answer to all their problematics; and we added a new template of the description file easy to fill and to read. We did the unitary verification test of the different block of code, and we did the global verification by launching input files and getting the expected results.

III Perspectives

One of the limitation of our approach in generating scenarios is the ‘unitary scenario’ that we explained in the *section IV.1 of the last chapter*. While we were working, we started looking for methods for its improvement, but we could not develop more our research on it. This method is the mutation of the resulting scenarios. Based on some mathematics’ algorithms, we can made a crossing of scenarios in order to obtain a more compact and realistic scenario. A lot of documentation on mutation algorithms are available; the major work will be to find how to use them to achieve the goal.

Another limitation of our work is that, to study the coverage there is a long way. First, finish our scenarios generation, then convert them into testrun, launch them on HIL benches to finally test the coverage on the stimulus software, which require another information to be done. Finally, we not got our results (on stimulus as explained above). I started thinking about it: is it not possible to directly integrate in the MCMC algorithm another algorithm that will ensure the coverage of a given requirement, before choosing which scenario could be generated? We have for example the MC/DC (Modified Condition/ Decision Coverage) method developed in [1].

REFERENCES

- [1] S. Kangoye, «Development of a verification and validation approach of automotive embedded software, based on the automatic generation of test cases,» PhD thesis, Anger University, 2017.
- [2] C. Barbara, D. Joseph and D. Rami, «Functional Safety Draft International Standard for Road Vehicles: Background, Status, and Overview,» General Motors Research and Development, 2010.
- [3] T. João, «Automating ISO 26262 Hardware Evaluation Methodologies,» Thesis University of PORTO, 2014.
- [4] I. 26262-1:2018, “Road vehicles -- Functional safety -- Part 1: Vocabulary,» December 2018. [Online]. Available: <https://www.iso.org/standard/68383.html>.
- [5] N. Becker, «The Safety of the Intended Functionality : Failures are not the only system safety problem,» *Study day, SIA ISO 26262*, p. 93, 2018.
- [6] S. Prialé, N. Rebernik, A. Eichberger et E. Stadlober, *Virtual Stochastic Testing of Advanced Driver Assistance Systems*, Springer, 2015.
- [7] B. Guillaume, «Autonomous Vehicle Overview,» MOOC, IFP School, 2018.
- [8] R. Francisca and N. Pedro, *A Systematic Review of Perception System and Simulators for Autonomous Vehicles Research*, 2019.
- [9] P. Scott, A. Hans and al, «Perception, Planning, Control, and Coordination for Autonomous Vehicles,» *Machines*, 2017.
- [10] C. Vineet, «Society of Automotive Engineers (SAE) Automation Levels for cars,» [En ligne]. Available: <https://www.automotiveelectronics.com/sae-levels-cars/>. [Accès le 17 06 2019].
- [11] D. Åsljung, *On Safety Validation of Automated Driving Systems using Extreme Value Theory*, Sweden, 2017.
- [12] I. H. Walther, «Dissertation: How Stochastic can Help to Introduce Automated Driving,» 2016.

- [13] M. Till, B. Gerrit and M. Markus, Scenarios for Development, Test and Validation of Automated Vehicles, Germany: Institute of Control Engineering, 2018.
- [14] U. Simon, M. Till and al, «Defining and Substantiating the Terms Scene, Situation, and Scenario for Automated Driving,» IEEE, 2015.
- [15] M. Pascal, «An Efficient Method for Testing Autonomous Driving Software against Nondeterministic Influences,» Technischen Universität München, 2017.
- [16] K. Philip and W. Michael, «Challenges in Autonomous Vehicle Testing and Validation,» SAE World Congress, 2016.
- [17] S. Dr. Volker, «Markov Chains and Monte–Carlo Simulation,» Ulm University Institute of Stochastics, 2010.
- [18] V. Sandrine and C. Thierry, «Modélisation et Simulation Stochastique,» Télécom Bretagne, 2014.
- [19] D. Bernard, «Simulation et modélisation,» Master course University of Rennes France.
- [20] E. Thorn, K. Shawn and C. Michelle, «A Framework for Automated Driving System Testable Cases and Scenarios,» NHTSA, Washington, DC, 2018.
- [21] G. Andrew and B. Rubin, «Inference from Iterative Simulation Using Multiple Sequences,» Department of Statistics, University of California, 1992.

APPENDICES

I Appendix 1: ASIL levels [1]

The following matrix defines the level of ASIL to be applied according to the three factors E, C and S:

		C1	C2	C3
S1	E1	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	ASIL A
	E4	QM	ASIL A	ASIL B
S2	E1	QM	QM	QM
	E2	QM	QM	ASIL A
	E3	QM	ASIL A	ASIL B
	E4	ASIL A	ASIL B	ASIL C
S3	E1	QM	QM	ASIL A
	E2	QM	ASIL A	ASIL B
	E3	ASIL A	ASIL B	ASIL C
	E4	ASIL B	ASIL C	ASIL D

S0	S1	S2	S3
No injuries	Light and moderate injuries	Severe and life-threatening injuries (survival probable)	Life-threatening injuries (survival uncertain), fatal injuries

E0	E1	E2	E3	E4
Incredible	Very low probability	Low probability	Medium probability	High probability

C0	C1	C2	C3
Controllable in general	Simply controllable	Normally controllable	Difficult to control or uncontrollable

QM: Quality Management

II Appendix 2: Parameters classification

Categories	Parameters	Equivalence class
Physical infrastructure	Type of road	Urban, Suburban, Motorway, Countryside
	Number of lanes	1, 2, 3, 4 lanes road
	Bend radius	Straight road, R: ≥ 500 m, R: ≥ 250 m, R: ≥ 125 m
	Turn direction	Right, Left turn
	Topology	Yintersection, Crossroads, Roundabouts, Gyration, Toll
	Motorway exit	Absence, presence
	Motorway entrance	Absence, presence
	Roadway surface	Asphalt, concrete
	Slot	Absence, presence
	Drop	Flat, ascending, descending
	Markings perturbation	Normal, partially deleted, work road marking
	Tunnel	Absence, presence
	Tire skid marks	Absence, presence
	Work on road	Absence, presence
	Bridge	Absence, presence
	Bumps	Cone, Beam, wave, Roadside, Mesh
Tarmack	Guard rail, Guard Rail (double), Jersey Barrier (wall), Jersey Barrier Elements, Wall, Wall (type 2)	
Environmental conditions	Day / Night	Day, Night
	Luminosity	High: [1000 lux; 7000 lux] , medium: [10 lux ; 1000 lux] , low: <10 Lux
	Weather	Dry, Cloudy, Rainy, Fog, Snow
	Road maskings	Water slabs, Snow slabs, No masking
Operational constraints	Traffic density	Minimal, Normal, Heavy traffic
	Traffic sign & Light	Traffic light, Traffic sign, Stop
Ego vehicle conditions	Initial lane ego vehicle	1, 2, 3, 4
	Speed ego vehicle	$S \geq 100\text{km/h}$, $60 \text{ km/h} \leq S < 100\text{km/h}$, $0 \leq S < 60\text{km/h}$
	Ego vehicle dynamic	High acceleration, Hard braking, Soft braking, Sway, Overtaking, Driving Backwards
	Time to collision	TTC = 1s, TTC= 2s, TTC= 3s, TTC= 4s, TTC near limits
	Windshield wiper	ON, OFF
	Wheel angle	-
	Wheel angle variation	-
	Outbreak longitudinal distance	Far, Medium, Low
Outbreak Time	Far, Medium, Low	
Roadway users	Initial lane traffic vehicle	1, 2, 3, 4
	Speed traffic vehicle	$S \geq 100\text{km/h}$, $60 \text{ km/h} \leq S < 100\text{km/h}$, $0 \leq S < 60\text{km/h}$
	Traffic vehicle dynamic	High acceleration, Hard braking, Soft braking, Sway, Overtaking, Driving Backwards

Shift	[0.5; 1m[, [1; 1m50[, [1m50;2m[
Shift direction	Shift on the Right, Shift on the Left
Obstacles	Pedestrians, Cyclists, Animals, Vehicle, Other obstacles
Obstacles manoeuver	static, Walking or riding , running or riding , Encroach the way of the ego vehicle
Obstacles relative position	Frontal, Side
Parked vehicle	No vehicle is parked, Vehicle parked on the right side, Vehicle parked on the left side, Vehicle parked on both side of the road
Pert. Target	Car on trailer, Trompe œil, No pert. Target

III Appendix 3: Algorithm of method 2

Algorithm4: Gibbs sampler method 2

1. At initial step, select $X^0 = (C_1^0, C_2^0, \dots, C_M^0)$, where $C_j^0 = \{X_{j1}^0, X_{j2}^0, \dots, X_{jp}^0\}$ and a probability vector $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_M)$ applied on the categories
 2. For $t = 1, 2, \dots, N$
 - a. Randomly choose C_j , $j \in \{1, 2, \dots, M\}$ with the probability α_j
 - b. Randomly choose X_{jp} , $p \in \{1, 2, \dots, N\}$ with an equiprobable probability
 - c. Generate $X_{jp}^t \leftarrow P(X_{j1}^t | X_{j2}^{t-1}, X_{j3}^{t-1}, \dots, X_{jp-1}^{t-1})$
 - d. Leave the other parameters unchanged
 3. Repeat step 2 until reaching equilibrium.
-

CD content:

- Input Excel file for the ‘Perception test’
- Output results of the ‘Perception test’
- MCMC Test Tool software
- Electronic Version of Diploma Thesis