Assessing Safety of an Automated Vehicle Through Model-driven Analysis and Simulation

Morayo Adedjouma, Fabien Gaudin, Philippe Fiani

Assessing Safety of an Automated Vehicle Through Model-driven Analysis and Simulation

Morayo Adedjouma
Université Paris-Saclay, CEA, List
F-91120, Palaiseau, France
morayo.adedjouma@cea.fr

Fabien Gaudin
Sherpa Engineering, R&D department
92000, Nanterre, France
f.gaudin@sherpa-eng.com

Philippe Fiani
Sherpa Engineering, R&D department
92000, Nanterre, France
p.fiani@sherpa-eng.com

***Abstract*—Validating the safety of automated systems is a highly complex task that cannot be done effectively through one validation methodology alone. As a result, current trends recommend adopting a multi-pillar approach for the validation of such systems. In this paper, we share our experience in applying a combined safety approach for the safety evaluation of an automated vehicle. The evaluation approach couples Model-Driven Engineering paradigm and simulation for a detailed assessment of critical scenarios. Based on a system model, we perform analytical safety analysis to identify the critical failures that may lead to undesired events. The analytical analysis is complemented by extensive simulation experiments to assess finer the impact of the identified malfunctions. The overall approach builds upon a tool chain consisting of Physistem as a modeling framework, Papyrus-Sophia for dysfunctional analysis support, and Phisim as a simulation environment. We report on the experiment results and discuss the advantages and limitations that the proposed approach brings for the evaluation of safety-critical automated systems.**

***Index Terms*—safety validation, automated system, model-driven engineering, simulation**

I. Introduction

The work reported in this paper was developed in the context of an industrial study to operate a fleet of autonomous shuttles to transport daily about 8000 people on a private company site large of 220 ha while guaranteeing operational safety inspired by public transport recommendations. Prior to enabling long-term operation, the operating limitations of the shuttles must be studied. The absence of a normative and regulatory framework as well as the absence of dedicated and applicable operating safety requirements on the subject of the autonomous shuttle delays the emergence of methodology and processes aimed at demonstrating the safety of such an autonomous transportation system. However, it is well-recognized that validating the safety of automated systems is a highly complex task that cannot be done effectively through one validation methodology alone. As a result, current trends recommend adopting a multi-pillar approach for the validation of such systems [1]. In this paper, we reflect on the application of a safety approach that couples Model-Driven Engineering (MDE) paradigm and simulation for evaluating the safety of an automated system.

Based on a model-based design of the system, we perform analytical safety analysis to identify the potential failure chains that may occur within the system and that may lead to undesired events. The analytical analysis results were then used to define advanced critical scenarios to assess through extensive simulation experiments. The latter simulations aim to validate finer and deeper the impact of the identified failures in real operating conditions and define appropriate safety measures for the safe operation of an autonomous shuttle. The overall approach builds upon a tool chain consisting of Physistem as a modeling framework, Papyrus-Sophia for dysfunctional analysis support, and Phisim as a simulation environment.

The rest of the paper is organized as follows. Section II presents the autonomous shuttle specification and depicts its architectural modeling. Section III presents our approach for safety validation of the automated vehicle and the accompanying tool support. Section IV presents the application of the model-based safety analysis method to the case study and the limitations of such analytical analysis. Section V reports on the simulation experiments. Section VI summarizes the lessons learned on applying the proposed approach to the autonomous vehicle case and the benefits that this approach brings for the safety validation of automated systems.

II. Case Study

A. System specification

The target of our study was a real industry-academy experimentation of a fleet of autonomous shuttle deployment on a sensitive site of 220 ha [2]. An autonomous shuttle has some particularities that increase the severity of harm and damages in case of an accident and make the autonomous shuttle deployment critical. Among these particularities, one can cite its usage scenario dimensions (high weight, high number of passengers, etc.), the traveling conditions (passengers in standing position, no use of handholds by passengers, etc.), the prominent presence of fragile users (elderly people, children, handicapped persons, pregnant women, etc.), the limited safety mechanisms (lack of seat belts, absence of airbags, limited or unreachable handholds, etc.). Besides, it is assumed that the vehicle should achieve the highest level of autonomy according to SAE J3016 [3] (L5: Full Automation); so, the human system interactions (driver, passengers) in case of emergency are very limited.

For our safety study, the system operating scenarios have to relate to functional safety and have an interest in being simulated for further validation. We focus then on specific scenarios

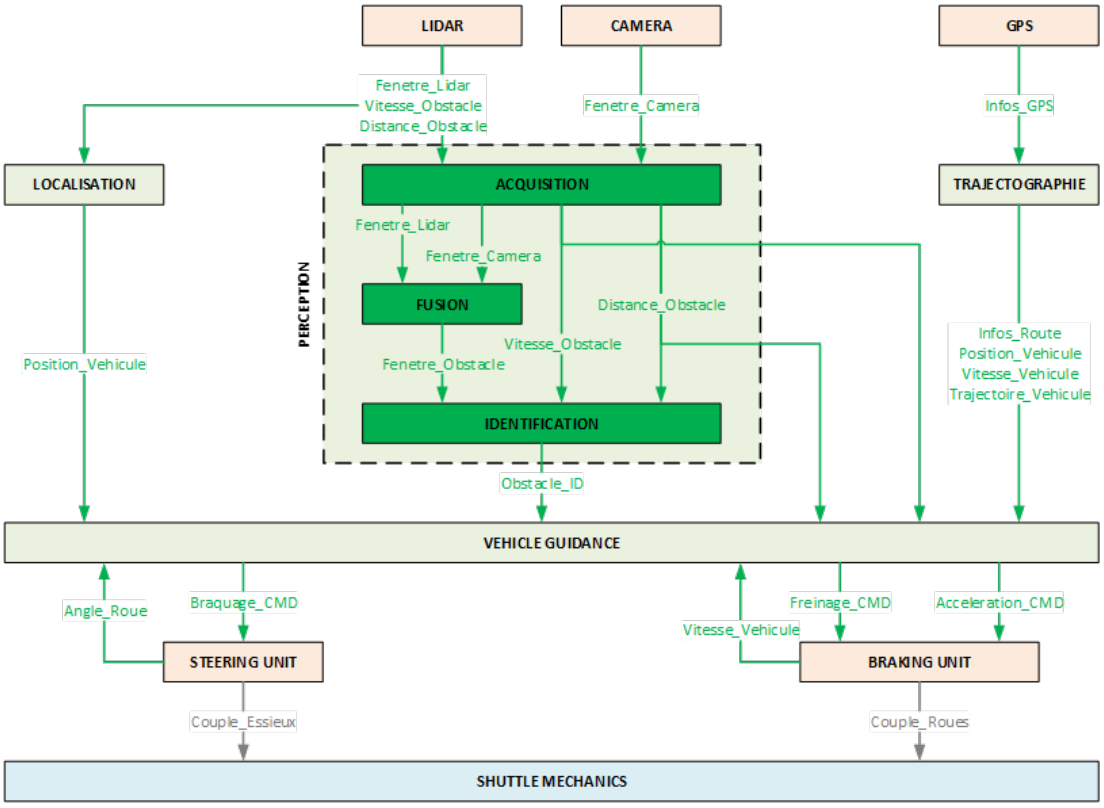

Fig. 1: Simplified functional architecture of an autonomous shuttle

that may lead to a collision or a trajectory deviation. The aim is therefore to find the faults propagation paths within the system architecture that may lead to such an undesired event. The selected scenarios involve the *Perception*, the *Localisation*, the *Trajectography*, the *Navigation*, *i.e.*, $Vehicle\_Guidance$, and the *Motion control* functions of the system. Depending on the relative position of the shuttle (localization), the objects detected (perception), the configuration of the road (trajectography), the vehicle guidance function controls these 3 degrees of freedom: The acceleration, the braking, and the steering, i.e., the orientation of the wheels of the shuttle.

B. System Architecture

Figure 1 presents an excerpt of the system architecture. The system architecture illustrates the logical functioning of the system independently of how its implementation is carried out. For simplicity and clarity, the depicted architecture comprises 3 sensors, but one may consider that each of them represents a set of sensors:

- GPS provides the position of the shuttle
- CAMERA provides the objects detected in the scene including their type
- LIDAR provides the objects detected in the scene including their characteristics: envelope, speed, position of the obstacle

The *Perception* relies upon LIDAR and Camera sensors, and an ML/DL data fusion component [4] which aims to increase accuracy by integrating different, sometimes redundant data sources. These components structure a model of the surrounding vehicle's environment to enable object detection as well as their type (obstacle, fence, pedestrian, cyclist, car, etc.); infer the position, dynamics, and relative status of such objects (standing, approaching, crossing, etc). The *Localisation* relies upon LIDAR information as well. Both Camera and LIDAR sensors enable also the detection of the shuttle dynamics: speed, angular momentum, acceleration, etc. The *Trajectography* component builds upon a relative GPS allowing the vehicle to support satellite exchanges conveying local and global positioning data. The Trajectography component mainly enables cartography localization, trajectory searching, and itinerary following. The $Vehicle\_Guidance$ is an AI-based component that interprets the scene and makes a decision according to the *Perception*, *Localisation*, *Trajectography* state, and additional safety and AI-based requirements. This component computes the vehicle dynamics, e.g., speed, acceleration, and momentum, thus settling the model of the system itself. The computed vehicle dynamics are afterward sent to the $Braking\_Unit$ and $Steering\_Unit$ components for vehicle motion control actions, i.e., to move, to steer, to brake, etc.

III. Safety validation framework for automated systems

Our safety validation approach adopts a joint model-driven analysis methodology and testing through simulation that is aimed at ensuring the functional safety of the system while enabling the specification of the safety criteria and thresholds for guaranteeing safe operation. The methodology relies on the PhiSystem tool for the modeling of the system [5], the Papyrus-Sophia tool for the safety analysis, and Phisim tool for the simulation [5].

First, we develop UML-based models of the shuttle architecture in Physistem. PhiSystem is an industry-ready tool based on the Papyrus [6] framework and the SysML standard. PhiSystem leverages modeling capabilities to implement a top-down approach for the design of CPS. The tool accounts for multiple viewpoints to enable the definition of requirements, system missions, as well as architecture (functional and physical) models. It provides a SysML model library of components to represent the functional and physical units that enable to specify the full multi-physics modelling of autonomous systems based on the Bond-Graph method. Within the tool, we define the shuttle architecture models at the system level, and at the hardware and software level. This allows the definition of finer-grained system properties and characteristics within the modeling.

Second, we use the Sophia to support safety analysis and assessment. Sophia is a model-based toolset integrated with the Eclipse Papyrus framework [7]. Sophia uses Papyrus extension mechanisms to support safety and reliability analyses like Hazard Analysis and Risk Assessment (HARA), Failure Mode and Effect Analysis (FMEA), Fault Tree Analysis (FTA), etc. Sophia allows for conducting functional safety analyses based on system models defined in UML, SysML, or derived languages. Since both Physistem and Sophia rely on Eclipse, Papyrus, and UML-based languages, it was possible to integrate them into a unified framework. We were then able to automatically apply Sophia's methods on top of the design models produced in Phisystem, with the benefit that using the same model for design and safety analysis avoids misinterpretations and reduces design time and cost.

In the third step, we use the Phisim tool to test through simulation several critical scenarios issued from the Sophia safety analysis results, to further validate the safety of the automated system. Phisim is also a mature simulation-based framework. PhiSim is a Simulink-based package that allows the building of complex cyber-physical systems using sets of reusable library elements, including ADAS (Advanced Driver Assistance System) models for drivers, cars, and their environment. The Massif framework[1] is used as the bridge connecting PhiSystem and PhiSim. Massif supports the representation of Simulink models and libraries (such as PhiSim) in the Eclipse Modeling Framework (EMF). It also features a dedicated API that enables Java RMI-based communication with a running Matlab instance. PhiSystem provides a SysML model library of components to represent the functional and physical units at the system level. It uses the UML profile mechanism to extend SysML to model concepts in the CPS domain. Through specific stereotype attributes, every Physistem component is linked to a corresponding PhiSim executable model, which is a Simulink block in the PhiSim blockset at the simulation level. Automated tools process the Phisystem models and generate corresponding Phisim models - without any manual intervention required - in two sequential steps. First, a model-to-model transformation processes the PhiSystem model and produces an intermediate (Massif) model representing the Simulink/PhiSim equivalent model in EMF format. Second, the intermediate model is processed by a model-to-text transformation that produces a set of Matlab scripts with construction commands. Once executed, these scripts create the .slx model which simulation engineers can work with. Model transformation and scripts execution are automatically performed via the Java RMI framework, which is perceived by designers as a single-step. To address model synchronization issues, even PhiSystem and Phisim features round-trip engineering capabilities to automatically maintain consistency between their models, by incrementally updating one model to reflect changes made to the other model. So, Physistem and Phisim models can evolve concurrently.

[1]Massif- MATLAB Simulink Integration Framework for Eclipse: https://viatra.github.io/massif/

IV. Model-based Safety analysis

The model-based safety analysis of our system is conducted based on an architecture model as depicted in Figure 1. As seen in the figure, the following naming convention rules were followed:

- The name of the output ports is noted $[Signalname]\_[Destination]$
- The name of the input ports is noted $[Source]\_[Signalname]$
- In the case of the same signal used twice by the same function, we add $\_[Number]$ to differentiate the ports' name

Only the perception component with its subsequent components are analyzed in detail, the other components were considered as black box.

For the analysis, we made the following assumptions:

- The vehicle is moving in a straight line
- Any detected obstacle is on his predefined route
- The vehicle mechanics are not faulty, i.e., we will not look into the failures of the hydraulic or mechanical circuits. We are mainly interested in potential failures in the algorithms, vehicle sensors, and control/command actuators.

A. Model safety annotation

Sophia defines analytical expressions associated with the system components to model their failure behavior and the possible propagation of failures through the system architecture. The analytical expressions are dysfunctional equations that show how a component failure can be caused by internal

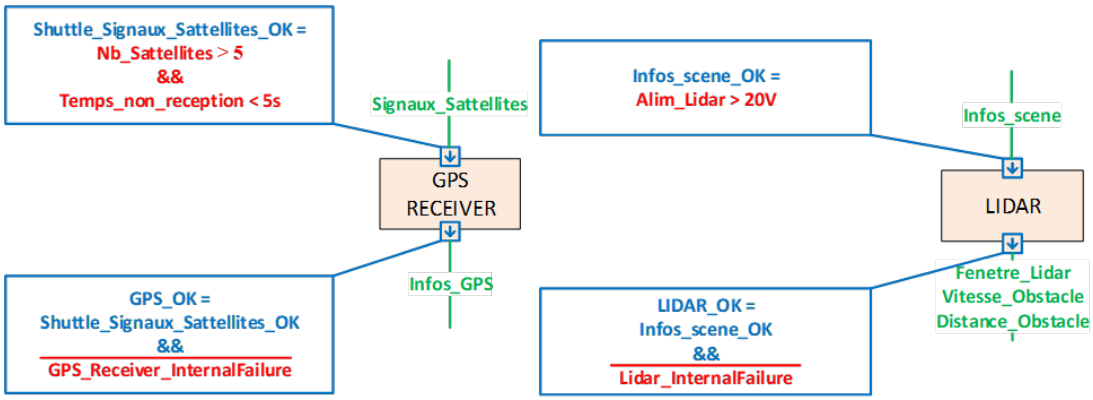

Fig. 2: GPS and LIDAR Dysfunctional equations

TABLE I: Fusion component dysfunctional equation true table

Fenetre_Camera	0	0	1	1	1
Fenetre_LIDAR	0	1	0	1	1
Coherence_Fenetre	-	-	-	0	1
Fusion	**0**	**1**	**1**	**0**	**1**

failures of the component and/or possible deviations in the component inputs. These equations are written in the output ports of the components and therefore propagate via the connectors to the other connected ports. Then the overall failure behavior of the system is automatically computed from the output deviation equations of the individual components.

For the failure modes identification, we focus on analyzing two functional requirements of our autonomous shuttle. The first requirement is to adopt a safe reaction in the presence of an obstacle on its trajectory, e.g., decelerating, or braking. The second requirement is to keep its trajectory.

1) sensors dysfunctional equations: Figure 2 presents the dysfunctional equations for the GPS and the LIDAR components. A GPS failure occurs if it catches less than 5 satellite signals for more than 5 seconds. In addition, an internal GPS failure is considered, i.e., a physical defect. A camera failure occurs when the is not powered or because of an internal failure. Similarly, the LIDAR fails when it is not powered or because of an internal failure as well.

2) Perception dysfunctional equations: The perception component comprises the Acquisition, Fusion, and Identification subcomponents. All the components can experience internal failures. We add in the acquisition what we call a "range check". For any data provided by the Camera or LIDAR outside a defined range, the data is considered invalid leading to the Acquisition failure.

The output from the fusion is valid, i.e., not in a failed mode, whenever we had available at least the LIDAR `Fenetre__Lidar` data or both the LIDAR `Fenetre__Lidar` and camera `Fenetre__Camera` data with some consistency between the two outputs (see Figure 3). The `Coherence__Fenetre` variable represents the consistency check between the envelope of the obstacle determined

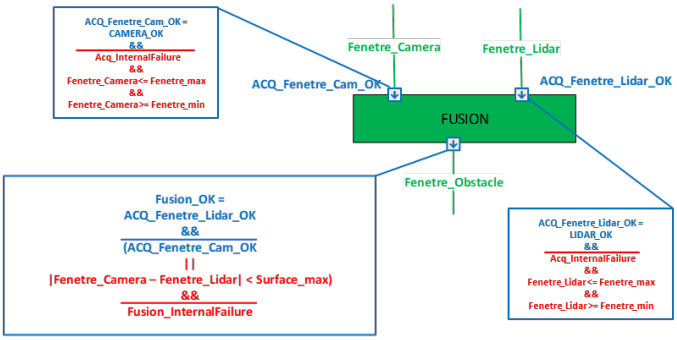

Fig. 3: Fusion component dysfunctional equation

by the LIDAR and the ones determined by the Camera. There is consistency if the surface difference between the two envelopes is less than the maximum threshold. Table I presents the True table that allows us to determine the `Coherence__Fenetre` value.

The Identification component satisfies the equation: Identification_OK = Fusion_OK && ACQ_DistanceObs_OK.

Hence, the Identification is not faulty if and only if we had the Fusion output available (therefore consistent envelope of the obstacle) and the distance and velocity of the obstacle from the Acquisition component, which makes it possible to determine its shape and its size.

3) Localisation and Trajectography dysfunctional equations: For simplicity, one manages to locate if the Lidar works properly and if there is no internal fault in the localisation component, e.g., software bug, or defective computer. In the same way, The trajectography relies essentially on the GPS and therefore on the correct functioning of the latter.

4) Vehicle guidance dysfunctional equation: The validity of the steering, braking, and acceleration commands are deduced from the Localisation, Trajectrography, and Perception dysfunctional equations, as shown in Figure 4.

B. Analysis results

The model-driven safety analysis makes it possible: 1/ to capitalize on libraries of failure modes, Causes, effects, and

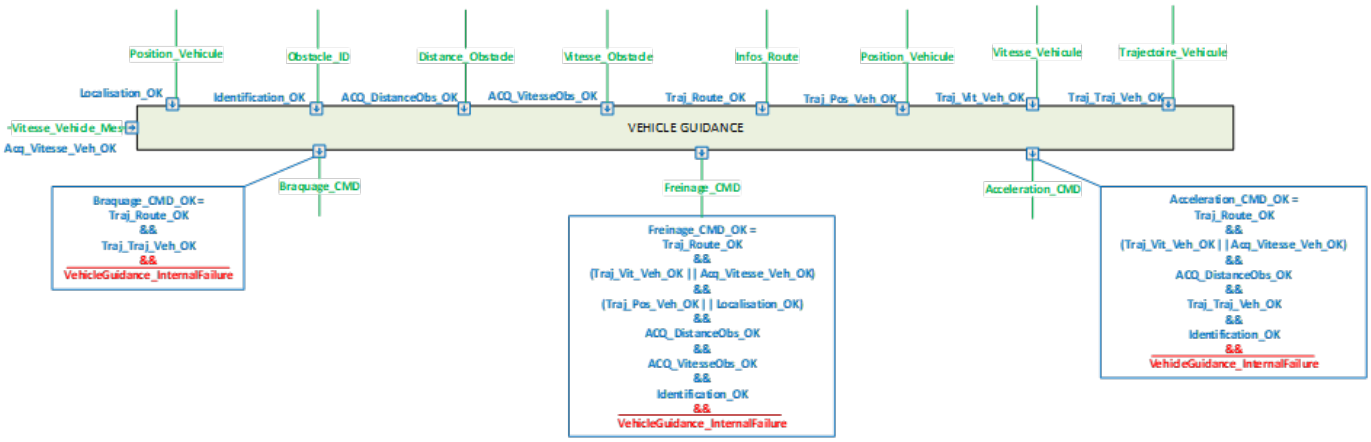

Fig. 4: Vehicle guidance component dysfunctional equation

risk reduction measures, 2/ to link automatically these safety elements to the system architecture, and 3/ to automate the criticality assessment. As an excerpt of the safety results, we report below on how the sensors' failures impact the safety of the system.

The loss of the camera can be created by a loss of power supply. This induces the output $CAMERA\_OK = 0$. Through failure propagation, the $ACQ\_Window\_Cam\_OK$ signal becomes therefore faulty. However, at the system level, the Camera failure does not affect the shuttle mission because its output is considered together with the Lidar data - which is assumed correct - in the failure propagation.

Lidar loss can result from power loss. This induces the output signal $LIDAR\_OK = 0$. The following signals therefore become faulty: $ACQ\_Window\_Lidar\_OK$, $ACQ\_SpeedObs\_OK$, $ACQ\_DistanceObs\_OK$, $Fusion\_OK$, $Localisation\_OK$, $Identification\_OK$. At the system level, the Lidar loss does not affect steering control but does affect brake and throttle control.

GPS loss can be caused by loss of signals for more than 5 seconds or less than 5 satellites detected. This induces $GPS\_OK = 0$. All the following GPS-dependent signals therefore become faulty: $Traj\_Route\_OK$, $Traj\_Pos\_Veh\_OK$, $Traj\_Vit\_Veh\_OK$, $Traj\_Traj\_Veh\_OK$.

With a GPS loss, the shuttle is no longer able to correctly calculate steering, braking, and acceleration commands.

C. Limitations

The main limitation of the model-based safety analysis method is the nature of the dysfunctional equations. They are only composed of boolean and numerical operators. Thus, they rely on the unique assumption that the system and the components are correctly functioning or not functioning. However, in the real world, the system can experience more complex breakdowns, e.g., intermittent, erratic, etc. failure) with more or less significant degradations, of variable durations. In addition, the analysis does not consider multiple cascading failures, e.g., both Camera and Lidar being lost simultaneously, nor other systems functionalities that may compensate, to some extent, some other failures occurrence within the system. For example, in case of a GPS loss, one could consider that the position of the vehicle and its speed can be estimated for some duration via the inertial measurement unit (IMU) and a precise map until the GPS information is available back. This is typically a benefit that the simulation can bring into the safety validation. In simulation, not only we can assess corner and critical scenarios that will be difficult to perform in a real environment, but it helps also confirm design-based safety analysis results as well as evaluate scenario variants, e.g., where the signal from a sensor can be valid only for 75% of the time, to determine the maximum interruption time of a signal before generating a malfunction. In addition, the simulator makes it possible to assess the effect of risk reduction measures, for example, to enable to simulate more or less strong braking, with potentially a delay compared to the expected braking time.

V. Simulation Experiments

The goal of our simulation experiments is to further evaluate the quality of service and functional safety, based on the analytical analysis outcomes. Looking at the assumptions set in the preamble of the analytical analysis (see Section IV), many safety-relevant cases have not been considered, e.g., any mechanical failure, the obstacle speed, the route geometry, the impact of environmental factors (rain, sun, snow, etc.). In addition to being able to test degraded cases more precisely, the simulation will allow us to test cases that would take a longer time to analyze analytically, as they would have required to model in detail all functions and flows contributing to this case, all the organic components contributing to this case, and to be able to write dysfunctional equations closer to reality more complex than boolean equations. All this modeling is easily carried out in the Phisim tool which therefore gives the possibility of discovering edge effects.

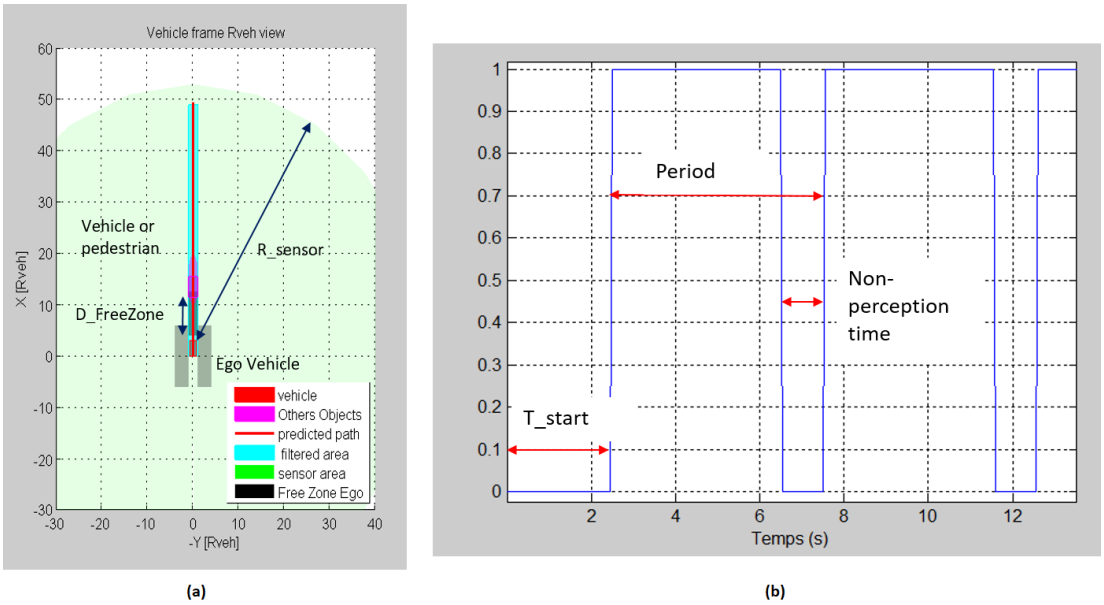

Fig. 5: (a) Perception flaw parameters definition, (b) Simulation period parameters

In our simulations, the ego vehicle satisfies the following characteristics:

- Mass: 1800 kg
- Maximum power: 90 kW
- Max torque: 220 Nm
- Max Acceleration = 2 m/s²
- Max Deceleration = -5 m/s²
- maximum speed = 15 km/h

We mainly focus the simulation experiments on perception and localisation flaws. Figure 5 (a) presents the main parameters that describe a perception flaw. R_sensor and D_FreeZone are variables that depend on the vehicle's speed. At a speed of 15 Km/h, the R_sensor = 50 m and the D_FreeZone = 12 m. In the case of the perception default, the R_Sensor= 0 m, which means that the other object is no longer detected. With this default, we consider two main accident scenarios: a collision with a pedestrian, and an ahead collision with a preceding vehicle traveling the same route as the ego. For the localisation flaw, we consider the trajectory path deviation as undesired behavior. In all scenarios, The ego vehicle starts at zero speed.

A. Pedestrian collision scenario

The simulated scenario is about a collision with a pedestrian. A pedestrian goes back and forth across the ego vehicle route. It completes its journey in 10 seconds, i.e., at a speed of 1 m/s. At the scenario start, the pedestrian is located at 180m of the ego. The pass success criteria for the scenario is that the ego vehicle does not hit the pedestrian. We assume that the pedestrian continues to walk regardless of the behavior of the vehicle, so we only consider frontal collisions between vehicle and pedestrian as a failure of the scenario.

1) Simulation configuration: The intermittent perception loss is configured as follows.

- Non-perception duration is 0.1 second, 0.3 second, 0.5 second, 1 second
- For each duration, we define the period so that this time represents a % of time failure of 10% or 20% of the simulation run (see Figure 5 (b)).
- The pedestrian's departure time is variable, from 0 to 10 seconds with a step of 0.1 second.

TABLE II: Number of Collision following the non-perception time and the perception failure duration

Duration NOK	% NOK	Nb Collision
0	0	3
0.1	20	6
0.3	20	8
0.5	20	13
1	20	20
0.1	10	5
0.3	10	8
0.5	10	13
1	10	18

Table II presents the simulation setting parameters. For a vehicle/pedestrian configuration, we define 1 out of 4 possible Perception Outage Time (NOK Time) and 1 out of 2 possible % of time without perception (%NOK). For a defined configuration (i.e., a line's table), 10 simulations are carried out by shifting the time of failure. From the 10 trials, we determine if the considered setting generates a collision (at least one collision out of the 10 trials). The Nb collision column counts the number of collisions obtained for the 100 vehicle/pedestrian configurations.

2) Results Analysis: Figure 6 presents different results about the scenario. We are interested in some collision-related

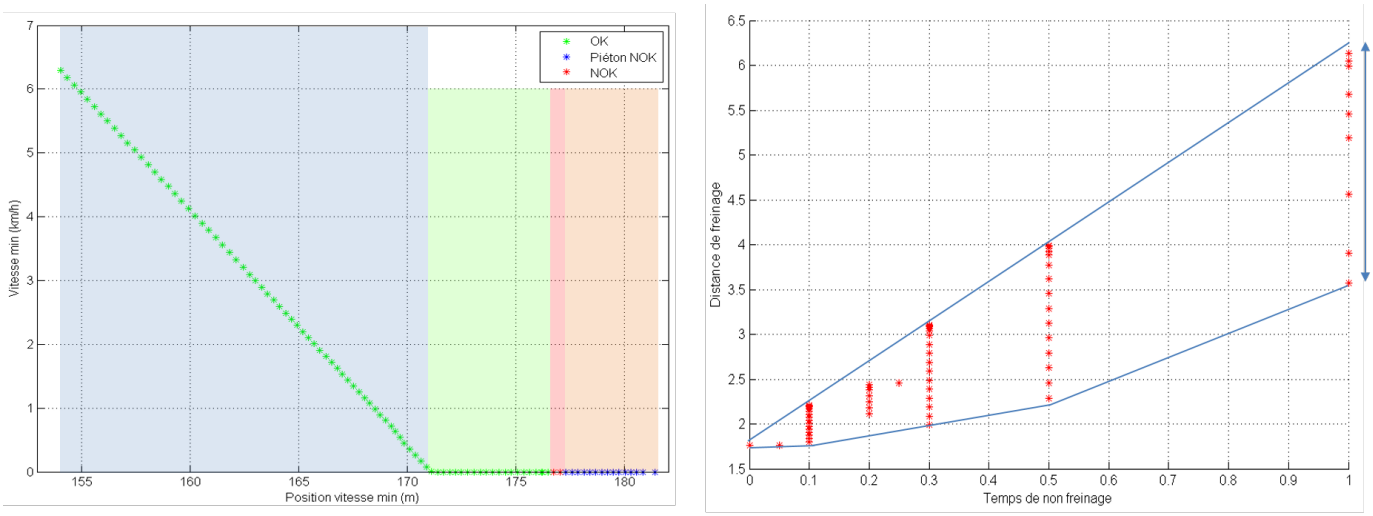

Fig. 6: Collision with pedestrian results: (a) collision occurrence as a function of the position and speed of the ego-vehicle; (b) braking distance as a function of the disturbance time during braking

indicators in the presence of an intermittent loss of perception, e.g., the time-to-collision, the vehicle speed in the event of a collision, the minimum speed during the journey, the relative position when the ego is at its minimum speed (stop position), the braking distance, and the non-braking time (during braking operation).

Figure 6 (a) presents collision with pedestrian scenario results based on the ego vehicle position and speed. In the blue zone, the vehicle decelerates so as not to knock over the pedestrian, but it does not need to stop. In the green zone, The vehicle stops without colliding with the pedestrian. In the red zone, The vehicle knocks the pedestrian over and then stops. This represents 3% of cases, i.e., 3 car/pedestrian configurations out of 100 simulation generates a collision. In the orange zone, the pedestrian enters the vehicle perception zone then the vehicle stops. We observe that the % of collision increases rapidly as a function of the time of non-perception. We also see that the frequency of failure occurrences does not influence much on the number of collisions. This leads us to reflect on the impact of non-perception on braking. Hence, when the vehicle no longer has perception, it no longer detects the pedestrian and re-accelerates during the braking phase. This phenomenon has the effect of increasing the braking distance and can cause a collision if the pedestrian is in front of the vehicle.

Figure 6 (b) shows the braking distance as a function of the disturbance time during braking. The red dots correspond to tests where there was a collision. Depending on the vehicle speed when the breakdown occurs, we obtain a +/- large braking distance: we travel more distance (at the same breakdown time) if the breakdown occurs at 12 km/h than if it occurs at 5 km/h. The minimum braking distance is 1.8 m and can go up to 6.2 m. This information can be taken into account to define safety metrics. For example, if we define that the sensors can have a maximum failure duration of 0.5 second, the vehicle must brake a maximum of 4 m ahead of the pedestrian.

Overall, the loss of perception has a very significant impact on the number of collisions, since the current detection strategy fails to avoid collisions. For this experimentation, the simulation allows complex calculations regarding the determination of the braking distance threshold in case of disturbance.

B. Ahead vehicle collision scenario

The scenario is about the ego vehicle that follows another vehicle moving at a fixed speed, at a distance of 25 meters ahead. The defined success criteria for the scenario is that the ego vehicle does not hit the car ahead during the occurrence of an intermittent loss of perception. In a case of collision, the distance between the 2 vehicles is calculated between the rear of the vehicle to follow and the front of the ego vehicle.

1) Simulation configuration: Each simulation runs for 90 seconds during which the ego travels a distance of 2000 meters. We test the scenario with different vehicle speeds: at 5, 10, and 14 km/h. We also configure the non-perception duration as: 0.1 second, 0.3 second, 0.5 second, 1 second, 2 seconds, and that the failure occurs 10, 20%, 40%, 50%, and 80% of the simulation time. For each vehicle speed, we perform 25 simulations, so 75 simulations in total.

2) Results Analysis: We are mainly interested in the following indicators: the occurrence of a collision, the minimum distance between vehicles, and the average distance between the vehicles.

Figure 7 presents 3 graphs that illustrate the minimum distance between vehicles according to the time of non-perception and the relative speed between vehicles, when the ego vehicle is traveling at 14 km/h (Figure 7 (a)), 10 km/h (Figure 7 (b)) and 5 km/h (Figure 7 (c)), respectively. We

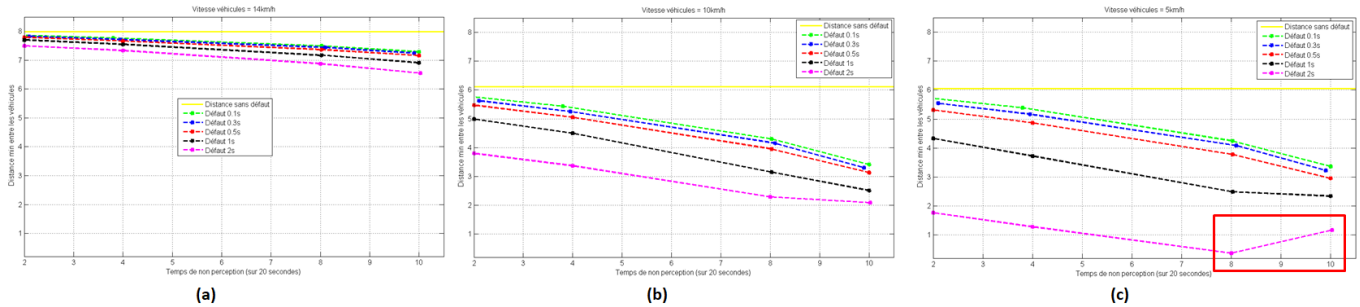

Fig. 7: Collision with ahead vehicle results

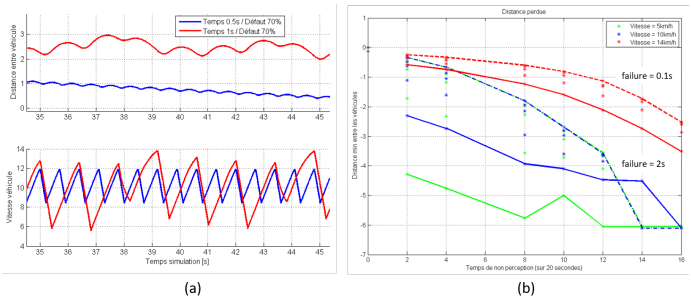

Fig. 8: Ahead vehicle collision results according to vehicle speed and failure time and duration

observe that the loss of perception leads to an increase in the speed of the vehicle to tend toward 15 km/h, its set speed. This phenomenon is a consequence of reducing the distance between vehicles. The distance tends to stabilize around an average distance. We also observe that the greater the frequency of the fault, the more the distance between the vehicles decreases. When the distance between the vehicles is small, the regulation uses emergency braking. This leads to a rapid increase in the distance between vehicles. There is not much impact between faults of 0.1 second and 0.5 second duration. If the speed difference between the vehicles is low, e.g., 1 km/h if the vehicle to follow is traveling at 14 km/h, even a very significant fault, e.g., 2 seconds of breakdown and 50% of the fault time does not have a major impact. However, while the sensor is more often faulty, the minimum and average distance are greater than the other fault (see red box in Figure 7 (c)). This is explainable as the vehicle continues to accelerate when perception is lost. When the speed exceeds 5 km/h, the distance decreases because the vehicle starts moving - as the distance between the vehicles is quite large ($>$ 3 m) - before the perception becomes back to ready status. This configuration generates a longer acceleration time and therefore a greater approach.

It is very difficult to end up with a collision in the configuration settings. The speed cycle resulting from the failure gives an average speed $>$ 10 km/h, i.e., the speed of the car to follow. The distance between the vehicles gradually decreases until a collision occurs. As shown in Figure 8, to have the first collisions, you must have a loss of perception for at least 70% of the time. With this analysis, one can define a maximum lost distance, e.g., 2 m, and deduce the type of perception defect therefore a sensor performance.

In concluding observations, the current detection strategy makes it possible to avoid the accident in very many cases. The sensors must be very faulty to cause an accident. It may therefore not be necessary to have high-performance sensors to achieve this functionality. The scenario simulation makes also it possible to identify interactions between the threshold for using emergency braking and the threshold to set the vehicle in motion.

C. *Trajectory following scenario*

The considered scenario is about the ego vehicle journey on a predefined route trajectory. During a journey of 120 seconds, the ego must experience a localization loss for 90 seconds. One would like to check the impact of this failure on the vehicle trajectory. Note that in case of localization loss, the vehicle uses an inertial unit to calculate its position based on longitudinal and lateral speeds which can also experience calculation inaccuracies leading to Longitudinal drift and lateral Drift. The simulation is considered as failed if there is a crossing line with a gap $>$ 2 m, i.e., a deviation from its trajectory of more than 1 m over 50 m.

1) Simulation configuration: At the simulation start, the vehicle is placed in the right place on the map. We configure the simulation settings with lateral and longitudinal speed gains from 0.9 to 1.1 in steps of 0.01 s; which yields 440 different simulations.

2) Results Analysis: As simulation indicators, we look into the maximum deviation from the route with an acceptable deviation threshold of 2 m from the intended path; and the maximal drift with an acceptable threshold of 1 m for 50 m, i.e., a tolerable drift of 0.02 m/m at maximum.

Figure 9 presents some results of the simulation runs. As seen in Figure 9 (a), the yaw rate error generates a drift in the trajectory. The error in the longitudinal speed generates an anticipation of turns. The failure impact is greater during right-angle turns. This is due to the angle of curvature (greater yaw rate) but also to the fact that it happens after a long time of failure which enables accumulated speed errors. while in straight and curve lines - that appears at the beginning of the journey - the problem has just occurred so the errors have not

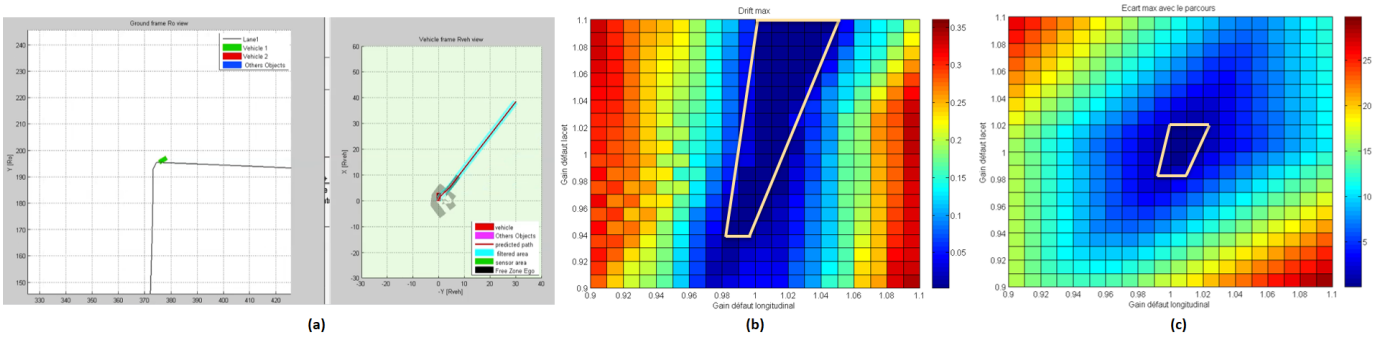

Fig. 9: Trajectory deviation from Localisation Loss: Deviation and drift as a function of lateral and longitudinal drift

accumulated too much and generate a low drift. Figure 9 (b) let us observe that for a drift error $< 0{,}02$, an error on longitudinal speed has more impact than on lateral speed. Figure 9 (c) lets us observe that there is very little configuration left to avoid exceeding a 2m gap within the journey. The impact on the gap is very similar to a fault on the longitudinal or lateral speed for such a long fault zone.

Overall, such experiments make it possible to better understand the behavior of the vehicle according to the quality of the inertial unit. The failure time duration and the angles of curvature are also to be taken into account, i.e., the integration of errors and higher yaw rate. The longitudinal error will tend to anticipate or delay taking a bend into account. The lateral error will tend to cause the vehicle to deviate from its lane with little influence in turns.

VI. Lessons Learned

This paper presents our experience in applying model-based safety analysis methodology and extensive simulations to validate the performance and safety of an automated vehicle. The validation approach is built upon a seamlessly integrated tool chain of 3 different industry-ready frameworks for system modeling, safety modeling and analysis, and simulation: Phisystem, Papyrus-Sophia, and Phisim. While the model-based safety method helps us identify critical failure propagation paths in the system design, we have seen that, solely, it is not sufficient to validate precisely the quality of service of the system as it shows limitations in addressing complex failures, combinations of failures, or the dynamic evolution of the failures within the system.

Simulations are initially seen as a means of completing the later validation and testing. In addition to enable testing cases that would not be possible to do in a real case, e.g., running over a pedestrian; or expensive to analyze analytically, as they would have required to model in detail all functions, organic components and flows within the system, and to write dysfunctional equations closer to reality more complex than boolean equations, it also enable to assess more precisely complex and continuous-based scenarios. Hence, for example, while the analytical analysis lets us identify that a localization loss, i.e., a GPS loss, is highly critical because the shuttle will no longer be able to correctly calculate steering, braking, and acceleration commands, the simulation experiments let us estimate that this failure may not be so critical - depending on the loss duration period - since the shuttle can rely on the IMU for the trajectory following. Similarly, we identified during the safety analysis that the perception function failure (both Camera and Lidar) is a critical scenario that may lead to collisions. During the simulation, we notice However that it is the time of non-perception that increases rapidly the number of collisions rather than the frequency of the failure occurrences. So, simulation also proves to be especially useful for better framing the specifications by integrating dynamic aspects that are not easy to anticipate in analytical reasoning. Besides, simulation implicitly takes into account some external factors that are hard to qualify/quantify in the analytical analysis, but that may influence indirectly the system's safety and performance, e.g., weather conditions, route layout, objects speed, etc. It further enables the identification of trends and influences on the overall behavior of the vehicle on a large number of configurations to adjust safety metrics, e.g., minimum time to collision, minimum safety distance, etc. depending on the type of disturbance and the operational context.

References

[1] U. W. P. on Automated/Autonomous and C. V. (GRVA), "New assessment/test method for automated driving (natm) guidelines for validating automated driving system (ads) – amendments to ece/trans/wp.29/2022/58," https://unece.org/sites/default/files/2022-05/WP.29-187-08e.pdf, 2022.

[2] M. Adedjouma *et al.*, "Representative safety assessment of autonomous vehicle for public transportation," in *ISORC 2018 Proceedings*, 2018, pp. 124–129.

[3] Society of Automotive Engineers International, *Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems*. SAE, 2014. [Online]. Available: https://www.sae.org/standards/content/j3016$_2$01401/

[4] Rachel C. King and Emma Villeneuve and Ruth J. White and R. Simon Sherratt and William Holderbaum and William S. Harwin, "Application of data fusion techniques and technologies for wearable health monitoring," *Medical Engineering & Physics*, vol. 42, pp. 1 – 12, 2017.

[5] M. Morelli[1], P. Fiani, A. Cuccuru[1], and S. Gerard[1], "PhiSystem: a tooled methodology for design and validation of ADAS," in *ERTS 2018*, ser. 9th European Congress on Embedded Real Time Software and Systems (ERTS 2018), Toulouse, France, Jan. 2018. [Online]. Available: https://hal.science/hal-02156190

[6] Eclipse. (Accessed: 2018) Papyrus modelling tool. [Online]. Available: https://www.eclipse.org/papyrus/

[7] M. Adedjouma and N. Yakymets, “A framework for model-based dependability analysis of cyber-physical systems,” inpress 19th IEEE International Symposium on High Assurance Systems Engineering (HASE) 2019, Hangzhou, China, January 3-5, 2019, 2019.