Autonomous Driving System : Model Based Safety Analysis

Mohamed Tlig, Mathilde Machin, Romain Kerneis, Emmanuel Arbaretier, Linda Zhao, Florent Meurville, Jean van Frank

Autonomous Driving System : Model Based Safety Analysis

Mohamed TLIG*, Mathilde MACHIN†, Romain KERNEIS*, Emmanuel ARBARETIER†,
Linda ZHAO‡, Florent MEURVILLE§, Jean VAN FRANK*

* *IRT SystemX – dept. Autonomous Transport* – Paris-Saclay, FRANCE
Email: firstname.lastname@irt-systemx.fr
† *APSYS – AIRBUS – dept. Modeling and tools* – Toulouse, FRANCE
Email: firstname.lastname@apsys-airbus.com
‡ *SECTOR-GROUP – dept. Functional Safety* – Villebon-sur-Yvette, FRANCE
Email: firstname.lastname@sector-group.net
§ *VALEO– dept. Functional Safety and Modeling* – Créteil, FRANCE
Email: firstname.lastname@valeo.com

***Abstract*—The desire to introduce autonomous vehicles by 2020 on the roads represents a real technological challenge. It requires breaks with traditional design, security and validation processes to achieve a safe system.**

As part of the SVA[1] (Simulation of Autonomous Vehicle Safety) project, we present the process under development at the Institute for Technological Research SystemX[2], in order to optimally address the limitations of existing methods.

The objective is to provide designers methods and tools to support safety considerations during the design and the validation phases of autonomous vehicles functions.

In this paper, we apply a Model Based Safety Analysis methodology (MBSA) to an Advanced Driver-Assistance System (ADAS), using a modular numerical simulation platform. We describe the different activities carried out during each stage and define the associated objectives. Experimental simulation results are presented, showing the advantages of such approach.

***Index Terms*—Automated Driving Systems, Functional Safety, ISO 26262, MBSA, AltaRica, Autonomous critical systems**

I. Introduction

Studying, arguing and guarantying some level of dependability and safety in autonomous cars remains an open question. Automotive standard ISO 26262 [1] deals mainly with internal faults (random hardware failures and systematic software faults) but gives no clue to manage problems linked to interpretation of sensed data and interaction with environment which are key points of autonomy. Moreover, the severity of feared events is assessed through controllability by human driver, which is no more relevant in autonomous cars. That's why we propose to adapt a method from aeronautics called MBSA (Model Based Safety Analysis) to tackle the problem of environment perception and interaction.

MBSA requires to build a system model with hight abstraction level and enables to generate all minimal paths (called cuts by analogy with failure trees) to a given feared event. Our general approach will consist in using MBSA to generate potentially critical scenarios, i.e., set or sequences of internal failures linked to interpretation or interaction problems. Since the model is high-level, it may not contain all critical scenarios of the given system. Nevertheless, we claim that this method is more efficient and gives more guarantee than simulating a finite set of scenarios, in a detailed model (typically a physical simulator), where the set of possible scenarios is infinite. Every scenario generated by MBSA represents a "class" of scenarios, as simulated in a physical simulator. After scenarios generation, we validate them in a physical simulator, defining the precise parameter bounds of the critical scenario class.

To illustrate this method, we apply it to the autonomy function TJC (Traffic Jam Chauffeur), in charge of driving the car in a traffic jam situation while following a vehicle ahead at maximum speed of 70 km/h on highway.

After giving some background in Section II on the fields of functional safety and autonomous driving systems in general, we describe in Section III the model we propose. Then, Section IV presents the results of this approach, both for cuts and sequences. Finally, we discuss the perspectives of this work and conclude.

II. Background

We address in this paper the general problem of autonomous driving systems (ADS). The SAE J3016 [2] taxonomy describes the full range levels of autonomy, where we consider level 3 and higher ADS. This kind of systems are still new and the feedbacks do not identify all of their failure modes. Moreover, safety analyzes, such as Preliminary Hazard Analysis, do not provide an exhaustive list of possible events leading to failure.

All these problems lead car manufacturers and their engineers to work on the development of new approaches that deal with the shortcomings of current methods. As example, we cite the PAS 21448 [3] which is an initiative to handle new sensor problems, i.e., malfunctioning behaviors in the absence of faults that are not addressed by the standard ISO 26262 [1].

Recent researches aim to improve safety by quantifying the uncertainties of the component outputs by propagation. Most of these approaches are based on Bayesian tools [4] and their variants, e.i., Deep Learning [5]. In the following, we suggest a formal method inspired from aeronautic[3] [6].

III. Modeling autonomy function

The modeling objective is to represent the TJC and all linked elements (e.g., forwarding vehicles) in a MBSA language so as to generate some critical scenarios. In particular, we look for scenarios

This research work has been carried out in the framework of IRT SystemX and therefore granted with public funds within the scope of the French Programme "Investissements d'Avenir". Industrial partners involved are: Apsys-Airbus, Sector Group, Valeo, Continental, Renault Group, PSA Group, AV Simulation, Assystem, All4Tec, Optis. Academic partners involved are: CEA, LSV Lab, Paris-Saclay University, LNE Lab.

[1]https://www.irt-systemx.fr/en/project/sva/

[2]https://www.irt-systemx.fr/en/

[3]This method was used to study the safety of the hydraulic system of the Airbus A320 aircraft family

with problems in the sensing process such as dazzling or erroneous pattern and image recognition.

A. Tools used

To model the TJC, we use Simfia, developed by Apsys-Airbus, based on the formal language AltaRica [7], originally designed in LaBRI[4] and offering a user-friendly interface.

AltaRica model is a set of bricks. Each brick has input and output connectors. Its state is defined by state variable and event, an event changes state variable values and thus defines a state machine. An event "failure" makes the variable "s" change from "nominal" to "failed". By logical expression, output connector "o" is set to the value "ko". AltaRica enables also to define brick hierarchy: a composite brick contains other bricks.

Starting from an observation point (typically a feared situation), AltaRica compiler generates cuts, i.e., set of events that result in the feared situation. AltaRica modeling is very abstract and enables to generate all cuts whereas a physical model is much more detailed but does not allow to generate all paths resulting in a given situation.

AltaRica compiler can also generate sequences, i.e., set of events with a given occurrence order. For example, for the model shown in Figure 1, with a simple failure in each brick:

- Cuts are {C}, {D} and {A,B};
- Sequences are (C), (D), (A,B) and (B,A).

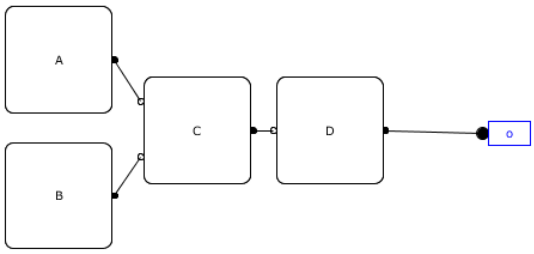

Fig. 1: Model example with 4 bricks

B. Model structure

In order to model the ADAS, we take into account the autonomous vehicle and its environment (e.g., other vehicles, weather...). This is original and it will increase the complexity of the calculations. However, in our case we are looking for feared scenarios, thus the environment representation is needed. Model structure, as illustrated by Figure 2, is as follows:

- Environment: all relevant environment elements for TJC working,
- Perception: sensors of the autonomous vehicle (also called *ego vehicle* in the following) used by TJC,
- Fusion: merging same class of informations seen by different sensors (e.g., obstacles, front vehicle...),
- Control: deciding of a vehicle motion behavior,
- Activation conditions of the TJC.

The feared situation we study is **the collision with another vehicle**. It is considered as reached when the ego vehicle and another vehicle are on the same square (discrete position).

[4]http://www.labri.fr

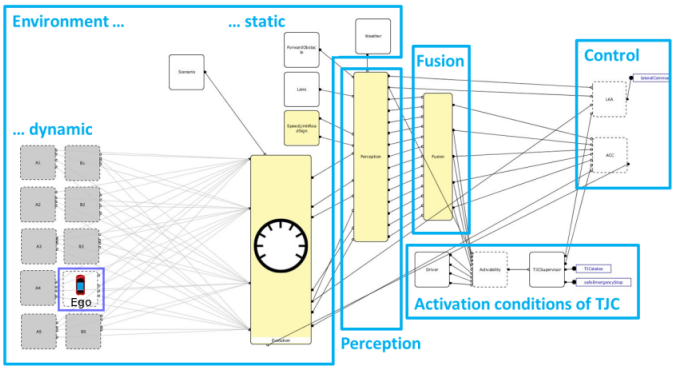

Fig. 2: Proposed model overview

1) Environment: The first element of the model is constituted by the environment elements used by TJC, in the limit range of the sensors. Static environment encompasses marking lanes, weather, road signs and, possibly, forwarding obstacles. As we considered dynamic environment, we refer to traffic. Up to 2 other cars can be present in the environment of ego vehicle. This part will be detailed in Section III-D.

2) Perception: Perception part contains environment sensors and specific intern sensors (which measure the speed and the yaw rate of the ego vehicle). Environment sensors are one front camera and one front radar. They are said to be "smart" as they are not returning picture frame (e.g., for the camera) but presence and distance of an object identified to be a vehicle (car, truck...).

Both radar and camera sense obstacles and forwarding vehicles. Additionally, camera detects marking lanes and road signs.

3) Fusion: As both camera and radar return information about forwarding vehicles and obstacles, information has to be merged to provide to TJC control one world representation. In classical embedded systems, fusion is usually a selection of one information source, by priority, vote or more complex logics. For autonomous systems, fusion algorithms take into account confidence levels. We model in this work 4 levels : level 0 is no confidence, level 1 is low confidence, level 2 is medium confidence and level 3 is confidence high. As a consequence, we have chosen to follow the classical conservative safety approach : if radar and camera disagree on an object, the most critical (dangerous) is preferred. For example, if camera "sees" the front vehicle far and radar senses it close, fusion states that front vehicle is close. In the same way, if camera "sees" an obstacle on the road which is not sensed by radar, fusion states that there is an obstacle.

As fusion algorithms are not properly modeled, we have chosen that modeled fusion cannot contain failures. In other words, we consider fusion as validated by another method and we claim to assess only environment and sensing problems.

4) Activation conditions of the TJC: The TJC can be activated by the driver when some conditions are fulfilled. For example, the driver must have hands on the steering wheel and seatbelt fasten. When one of these conditions is no longer fulfilled, the TJC sends a take over request to the driver. If the driver fails to take over (e.i., after few seconds), the TJC triggers an emergency braking.

5) Control: Similarly, control part cannot fail. The model contains two control modules:

- ACC (Adaptive Cruise Control) controls longitudinal speed,
- LKA (Lane Keeping Assist) controls lateral trajectory.

Control part embeds decision logics. Those logics are quite complex to design and debug because they shall result in a steady behavior of the ego vehicle, i.e.:

- when there is no change in environment or system state, the command shall remain steady,
- when a change in environment or system state occurs, control shall react and put the system back in a steady state.

For example, if distance to front vehicle reduces, ACC gives command to brake. Actuators are not considered as they are similar to existing cars and that their technology is well mastered.

C. Modeling perception

Modeling environment and its perception is a key feature to assess TJC safety. In the following part, we present some modeling artifacts we have used, that are quite specific to autonomy and generic enough to be reused for other autonomous systems.

1) Data type for environment elements: Environment elements are numerous and very different from one another. To model them, we propose three data types:

a) Object without parameter: An object is an element of the environment that may (or may not) be present (e.g. an obstacle on the road). A real object has plenty of parameters: size, position, color, speed... Nevertheless, none of these parameters is modeled. Either because they don't play any role in perception, thus any role in the TJC decisions, or because too much parameters can impact the system and instead of separate failure modes on each parameter, we prefer considering an overall error in object perception. Failure modes considered are:

- omission: the object is present but not detected,
- commission: there is no object but an object is detected,
- erroneous: the present object is detected with some wrong parameters.

b) Flow: A flow is a data that is always present. For example, ego speed or light level always exist. Failure modes considered are omission and erroneous.

c) Object with parameter: To model object parameters, we use flows. Fusion for an object with parameter is done in two steps: 1) fusion of object presence, 2) if presence is stated, fusion of each parameter. Failure modes for object presence detection are omission and commission only.

2) Modeling sensing: In classical embedded systems, most sensing errors are caused by hardware internal failures of sensors. However, for autonomous cars, perception errors are mainly due to interpretation of sensed data (e.g., erroneous pattern recognition) or interaction with environment (e.g., dazzling by sun, snow or white truck). In these cases, the camera can identify correctly a road sign on the roadside while missing an obstacle on the road. Therefore, we separate these perception functions in order to intercept separately perception errors. Each function can fail independently from others, as it is transient in real world, and it is modeled by a repairable failure. We model also the hardware failure of the sensors, causing the loss of all functions, so as to be able to assess the sensor architecture and the possible redundancies.

D. Modeling control consequences and dynamic environment

We propose here a method that takes into account dynamic traffic.

1) Other vehicles on the road: On top of the ego vehicle, we model two other vehicles that freely change their speed and lane. To avoid unrealistic scenarios, some restrictions are added to their behavior:

- they can change lane only on a free space, if and only if their speed is different than the nearest vehicle speed on the other lane (higher or lower, respectively after or before EGO), and,
- changing speed higher or lower depends on respectively the front or rear vehicle speed.

Due to these restrictions, which imply dependencies between vehicles, modeling more than two vehicles would make the model very complicated and complex. That is why we limit up to 2 the number of extra vehicles in this study.

2) Loop ACC command to speed value: As the traffic is dynamic and the situation depends on ego vehicle behavior, we create a loop/connection between the ACC command and the ego speed. Thus, when a change occurs in environment or system state and a new command is provided by ACC, an event is triggered to update the ego speed value. It enables to check the logics of the ACC and to visualize the different motion steps.

3) Road situation: Finally, to have graphical representation of the road situation, we add display bricks. As shown by Figure3, these bricks are a sliding scene around the ego vehicle, moving at the same speed.

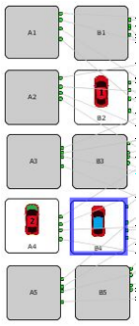

Fig. 3: In blue the EGO vehicle displayed in a possible scene

This display enables (i) debugging, (ii) validating by scenario the model representativeness and (iii) demonstrating to a non-expert audience.

IV. Results

In this section, we present the process we set up to extract information from the model. With Simfia, we can generate cuts or sequences resulting in the feared situation "collision". Cuts are a set of events, where occurrence order does not matter whereas a sequence is ordered. We use both generations to have two levels of interpretation. The number of events in a cut or sequence is called "order". The more events there is in a cut, the less this cut is probable, and so hazardous.

A. Cut analysis

After 60 hours of computation, we get cuts of order 4 (typical order for industrial studies is order 3) shown in Table I.

Ordre	Generated by Simfia	After post-treatment
1	1	1
2	29	6
3	153	19
4	283	13
Total	466	39

TABLE I: Cuts results

To be able to exploit these results we apply three steps of post-treatment:

- Filtering cuts that encompass a "take over request", to get only cuts where TJC is not able to detect its own failure;
- Removing events of position change decided by ACC, which is system evolution;
- Minimizing cuts, i.e., removing cuts that are already included in other cuts, due to system evolution informations present in cuts.

Thereafter, we obtain 39 cuts, as shown in Table I and we classified them in 6 categories :

1) Erroneous yaw rate (order 1);
2) Erroneous detection of marking lane *and* loss of yaw rate (order 2);
3) Erroneous longitudinal speed *and* slowing of the front vehicle(order 2);
4) Erroneous distance to front vehicle both on radar *and* camera (order 2);
5) Erroneous distance to front vehicle on one sensor *and* loss of distance to front vehicle on the other sensor (order 2);
6) Commission (inadvertent detection) on lane detection by camera *and* end of marking lane *and* loss of yaw rate (order 3).

From these results, yaw rate and longitudinal speed of ego vehicle are critical elements (because they appear in order 1 cuts). The reliability of ESP (Electronic Stability Program) that provides them to TJC shall be secured. Second, sensing distance from front vehicle is important (order 2). One can note that sensing front vehicle speed does not appear in cuts. Indeed, with low distance to front vehicle, the ACC always commands to bake, no matter the speed. Third, marking lane detection is highlighted. When no lane is detected, TJC sends a take over request. Nevertheless, an erroneous detection or an inadvertent detection inhibit error detection. Finally, note that the thirteen order 4 cuts (not present in the proposed classification) are in fact due to a weather condition combined to one of our six categories.

Sequence analysis: After sequences computation, we check that each cut (previously generated) is present with each of its permutation in the sequence set. For example, for the cut A,B, we look for the sequences (A,B) and (B,A). Thus, we found that occurrence order matters for the following scenarios:

- Erroneous detection of marking lane *and* loss of yaw rate (order 2);
- Commission (inadvertent detection) on lane detection by camera *and* end of marking lane *and* loss of yaw rate (order 3).

Indeed, loss of yaw rate results directly in a take over request, whereas an erroneous detection of marking lane followed by loss of yaw rate is not detected. In the same way, commission must occur before the end of marking lane so as not to be detected.

B. Simulations with SCANeR Studio

Afer the extraction of critical scenarios from our MBSA approach, we then simulate them in order to verify the feared event (collission in this case). In this work, we used SCANeR Studio[5] which is a software designed for industrials to address the specific needs of dynamic simulation.

As shown in Figure 4, this makes it easier to demonstrate some complex failures thanks to the SCANeR's visualization module.

V. Conclusion

In this work, we present a model based safety analysis for autonomous vehicles. We use AltaRica language and Simfia software to model this complex system. The application concerns a Traffic Jam Chauffeur developed and used for testing. We try, through this approach, to improve the model, to bring it closer to reality and not to make it more complex, so as not to "blow up" the calculation time. The autonomous vehicle is represented in an environment with two other vehicles, two lanes, obstacles and road signs. A separate display of the scene is set up and it allows an easy interpretation of the states of the vehicles. Thus, we can easily debug and present the model. We then generate the sequences and cuts of the hazardous event "collision" and analyze the results by identifying the critical elements of the system. Finally, some sequence examples are simulated for demonstration purposes. The followed approach is illustrated in Figure 5.

Future work includes continuing the experimental study to analyze the collisions that took place after the take over request. The other perspective that motivates our research is to model other situations (not only highway scenarios) and other hazardous events (feared situations).

[5]http://www.avsimulation.fr

Fig. 4: SCANeR Studio visualization module when simulating

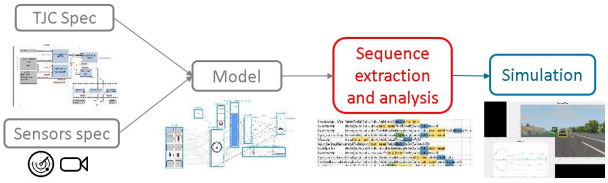

Fig. 5: The MBSA approach

References

[1] ISO 26262, "Road Vehicle – Functional Safety, Working Group TC22 SC32," International Organization for Standardization, Standard, 2011.
[2] SAE J3016, "Taxonomy and Definitions for Terms Related to Driving Automation Systems for on-Road Motor Vehicle," On-Road Automated Driving (Orad) Committee, SAE International, 2016.
[3] ISO/WD PAS 21448, "Road vehicles – Safety of the intended functionality," International Organization for Standardization, Standard, Under development.
[4] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.
[5] M. Rowan, G. Yarin, K. Alex, v. d. W. Mark, S. Amar, C. Roberto, and W. Adrian, "Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2017, pp. 4745–4753.
[6] P. Bieber, C. Castel, and C. Seguin, "Combination of fault tree analysis and model checking for safety assessment of complex system," in *Proceedings of the 4th European Dependable Computing Conference on Dependable Computing*, ser. EDCC-4. Springer-Verlag, 2002, pp. 19–31.
[7] A. Arnold, G. Point, A. Griffault, and A. Rauzy, "The AltaRica formalism for describing concurrent systems," *Fundamenta Informaticae*, vol. 40, no. 2, 3, pp. 109–124, 1999.