Reliability Engineering and System Safety 174 (2018) 29–40

Contents lists available at ScienceDirect

Reliability Engineering and System Safety

journal homepage: www.elsevier.com/locate/ress

A formal framework for the safe design of the Autonomous Driving supervision

Romain Cuer[a,b,*], Laurent Piétrac[a], Eric Niel[a], Saidou Diallo[b], Nicoleta Minoiu-Enache[b], Christophe Dang-Van-Nhan[b]

[a] Université de Lyon, CNRS, INSA-Lyon, AMPERE, F-69621 Villeurbanne, France
[b] Renault S.A.S., 1 avenue du Golf, 78280 Guyancourt, France

ARTICLE INFO

Keywords:
Autonomous vehicle
Systems engineering
Safety analysis
Requirements analysis
Design systems
Discrete-event dynamic systems
Redundancy control

ABSTRACT

The autonomous vehicle is meant to drive by itself, without any driver intervention (for the levels 4 and 5 of automated driving, according to the National Highway Traffic Safety Administration(NHTSA)). This car includes a new function, called Autonomous Driving (AD) function, in charge of driving the vehicle when it is authorized. This function may be in different states (basically active or inactive), that shall be managed by a sub-function, named supervision. The main focus of this work is to ensure that the supervision of a function, performed by a safety critical embedded automotive control system (controlled systems are not considered), respects functional and safety requirements. Usually two processes are involved in the system design: the systems engineering process and the safety one. The first process defines the functional requirements on the function while the safety one specifies redundant sub-functions (realizing together the function) allowing to ensure a continuous service under failure. Since two different aspects of the system are specified, it is a major challenge to make all requirements consistent, from the outset of the design process. In this paper, a method is precisely proposed to address this issue. A progressive reinforcement of the treated requirements is achieved by means of formal state models. In fact, the proposed approach permits to build state models from requirements initially expressed in natural language. Potential ambiguities, incompletenesses or undertones in requirements are in this way gradually deleted. The enrichment of conventional formal verification of control properties with safety requirements constitutes the main originality of the deployed method and contributes to solve inconsistencies between functional and safety verification processes. In addition, the application of the method to the design of AD function supervision highlights its efficiency in an industrial context.

1. Introduction

The autonomous vehicle causes a break in the automotive embedded systems design mainly because it is no more possible to count on the driver reaction in order to keep always the vehicle safe. One of the main arguments in favor of autonomous driving is actually the potential huge reduction of crashes, precisely by eliminating common drivers mistakes [1,2]. This paradigm change will deeply impact the design process. But the autonomous vehicle design must also take into account the constraints of the existing and be built on the know-how, considering the high time-to-market pressure [3,4]. The autonomous vehicle design can be carried out following the usual automotive engineering process. Indeed, the AD system, in charge of the new function Autonomous Driving, is integrated in a specific type of vehicle, already equipped with several ADAS (Advanced Driver Assistance Systems such as Adaptive Cruise Controller or Automatic Parking). This approach is then consistent with the introduction of other ADAS functions. However, if these systems were already safety critical [5,6], the challenge is higher for the AD system because the driver is no more the ultimate safety barrier [7]. In addition, prove the AD system safety only by validation tests appears almost impossible [8]. It is consequently crucial to ensure safe design of the AD system. Moreover, parallel processes of systems engineering and safety are difficult to integrate, given that the differences in terms of planning, constraints, objectives and work teams, as recently highlighted in [9]. Taofifenua [10] also emphases this issue and illustrates it in Fig. 1.

On related fields, like aerospace and railways, this topic is also central. Specific methods (such as Safety driven design methodology [11]) and software environment, like SCADE [12], are implemented in

* Corresponding author.
E-mail address: romain.cuer@insa-lyon.fr (R. Cuer).

https://doi.org/10.1016/j.ress.2018.01.014
Received 19 May 2017; Received in revised form 14 November 2017; Accepted 20 January 2018
Available online 2 February 2018
0951-8320/

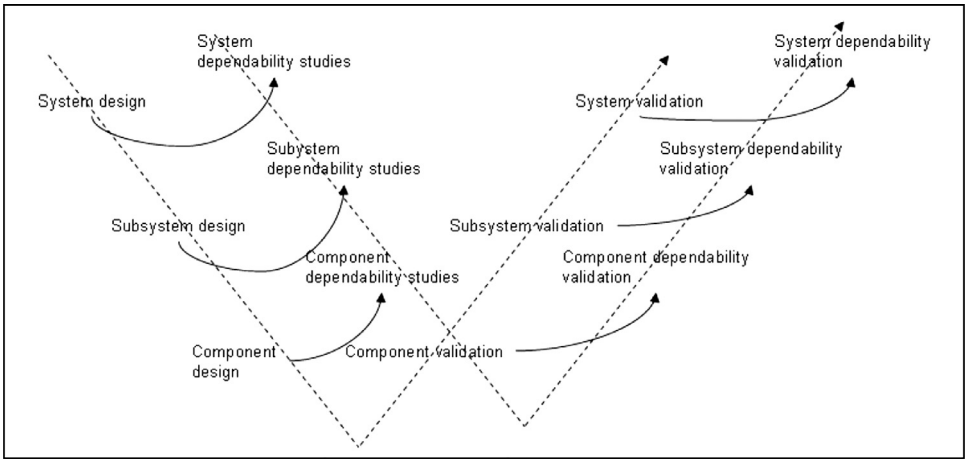

Fig. 1. Integration of safety approach in systems engineering process [10].

aerospace sector. Regarding railways area, the B-method [13,14] has already proven its efficiency. These methods are clearly effective in their application domains, as illustrated by the proven safety of trains and aircrafts. Nevertheless, it should be noted that the particular constraints of automotive field in terms of time-to-market pressure, extremely variable conditions (countries, regulations, driver abilities, climatic conditions…) strength constraints of the existing, costs reduction, limited available space and volume, and organization chart (formal methods are much more common for aeronautics and railways engineers than for automotive ones) [3,4,15] make the adaptation of these methods difficult. Consequently, accurate and adapted means, methods and tools, inspired by this experience, have to be proposed in the context of automotive area. The proposed approach contributes to address this issue. More particularly, it deals with the control system (realizing the designed function): the controlled systems are out of the scope of this study. Special emphasis is focused on verification of deterministic requirements specifying expected behavior in normal conditions on one hand, and safety requirements addressing redundancy and reconfiguration management in case of failures on the other hand.

The paper is structured as follows. In the Section 2, related works are presented and our position is specified. The Section 3 is dedicated to the framework proposed in this paper, that allows safely designing a function performed by an automotive embedded system. Specifically, the method deployed is centered on the behavior of the function: the main aim is to ensure that the intended function remains always in a safe state, whatever happens. This section contains the main contributions: the approach itself, that specifies a formal behavior model, correct by construction, from requirements written in natural language; and the reinforcement of the requirements, notably by highlighting ambiguities, incompleteness (in the sense of incomplete formulation of a requirement itself, not completeness of all requirements), inconsistencies or implicit early in the design process. The contributions of this study are focused on methodological aspects, by improving the current engineering processes, explicitly the design and safety ones. The Section 4 illustrates the interest of the approach by applying it to the AD system design. Lastly, the Section 5 remains the principal conclusions and contributions and outlines future works.

2. Related works

The consideration of the risks analysis at the first step of the system design process is an acknowledged problem for safety critical systems [16] and in particular for the automotive embedded systems [5,6,17,18]. More precisely, the main problem addressed concerns the impacts of safety requirements (requirements provided from safety analyses) on system design and the verification of design compliance with such requirements. It shall also be guaranteed that the system respects an expected behavior in normal conditions, determined by functional requirements. Many works already address this issue.

The most shared way to deal with this topic consists in modeling safety-critical embedded systems in a unique formal, or semi-formal, model [17–23]. It actually eases the merging between system design activities and safety ones, in terms of modeling activities. However, in the context of this work, we more specifically focus on methods aiming at verifying requirements compliance. The three main verification methods implemented [22], both in industrial domains and in the research community, are:

- *The simulation*: this widespread technique [19,24] is based on the symbolic execution of models and the realization of compliance tests corresponding to the users' needs. The symbolic execution requires an operational semantic defining in a deterministic manner the model behavior in reaction to input stimuli. The **main limit** is the completeness of the tested scenarios. The simulation gives only a presumption of correct behavior but is strongly correlated to the expertise and experience of the practitioners;
- *The theorem proving*: the verification is viewed as a theorem to prove from a set of axioms. The program and the properties shall be transformed in mathematical objects. Conclusions are inferred from the description of events or operations allowing to animate the system [25]. The **principal limits** are that the complete automation is rarely possible and the proof preparation requires the determination of elements exceeding the framework of the specifications;
- *The model checking*: automatized technique that, considering a finite state model of a system and a formal property, systematically verifies if this property is valid for this model [26]. This method is divided into three phases: the *modeling* of the intended system, the *execution* of the *model checking* algorithm, and the *analysis* of the results (property satisfied, non satisfied or saturate memory). Different tools implement this method: UPPAAL[1] [22,23,27], UPPAAL-Port[2] [18] or NuSMV[3] [28]. Among the **main limitations** of model checking, one can cite [26] that its applicability is subject to decidability issues. Thus, for infinite-state systems, model checking is, in general, not computable. Besides, it suffers from the combinatorial explosion problem. Finally, it requires expertise in finding appropriate abstractions to obtain smaller system models and to state properties in the logical formalism used.

[1] http://www.uppaal.org/.
[2] http://www.it.uu.se/research/group/darts/uppaal/port/.
[3] http://nusmv.fbk.eu/.