Framework for Safety in Autonomous Vehicles

Matthieu Carré[*,**] Ernesto Exposito[*]
Javier Ibañez-Guzmán[*,**]

[*] Univ Pau
Pays Adour, E2S UPPA, LIUPPA, EA3000, Anglet, 64600, France
matthieu.carre@univ-pau.fr
ernesto.exposito@univ-pau.fr
[**]Renault S.A.S, 1 av. du Golf, Guyancourt, 78288, France
javier.ibanez-guzman@renault.com

Abstract. The integration of the Safety dimension has been a critical requirement when developing and deploying Autonomous Vehicles (AV). While much progress has been achieved within the past years, most work has centred on providing vehicles with the ability to navigate autonomously. Safety has emerged as the major challenge. This paper proposes a reference architecture that incorporates the notion of self-safety into existing AV architectures. This architecture consists in a multi-layered control loop aimed at managing self-adaptation processes in order to ensure safety at run-time.

1 Introduction

The development and deployment of Autonomous Vehicles (AV) is a very challenging endeavour from a safety perspective. Vehicles must navigate through multiple situations preventing any potential harm and without disturbing traffic flow in order to be accepted by the society. Safe driving under full computer control also requires to interact and operate around with different entities within complex road networks and to appropriately address their different behaviours.

While much progress has been achieved within the past years, most work has centred on providing vehicles with the ability to navigate autonomously. Safety has emerged as the major challenge, not only on the vehicle behavioural side to address edge-cases (i.e. navigate safely) but also to manage malfunctions or external disturbances (i.e. fault tolerant).

Current work in the safety domain has proposed relevant approaches for the analysis, refinement, integration and enforcement of AV safety. Considering safety as a dynamic control problem as proposed by Leveson et al. (2015) and Leveson and Thomas (2018) shows promising applications and several interesting results have been presented in Lefèvre et al. (2014), Raste et al. (2015), Bagschik et al. (2017b) and Cook et al. (2018). Complementary research investigates the trade-off with the traditional failure analysis for hazards coverage as in Sulaman et al. (2019) and Ford (2018) while others address the compatibility of the approach with the current AV standards as in Abdulkhaleq et al. (2017), Sabaliauskaite et al. (2018) and Vernacchia (2018). However, most of these works converges on the difficulty to provide a

scalable integration and enforcement of AV safety based on the application of maturing safety analysis methodology to identify safety constraints. Moreover, addressing safety management and assurance at run-time using adaptive behaviours as in Törngren et al. (2018), Kane et al. (2015), Trapp and Schneider (2014), Cheng et al. (2014) and Amorim et al. (2018) has been shown to require a complex combination of AV system non-functional properties including observability, traceability, reconfigurability and scalability as presented in Cuer et al. (2018) and Stoica et al. (2017).

The main contribution presented in this article is a reference architecture that incorporates the notion of self-safety into an existing AV architectures. This architecture consists in two layers that manage self- adaptation processes to ensure safety at run-time. The first layer manages directly the components of the extended autonomous vehicle architecture with a collection of dependable processes. These processes guarantee the satisfaction of the requirements specified by each of the concurrent safety constraints. The second layer manages these dependable processes and guarantee their activation, management and deactivation based on dynamic conditions observed on the context (i.e. reconfiguration based on road conditions or to avoid conflicts between safety constants).

This paper is structured as follows: Section 2 presents the requirements for a run-time adaptive architecture for safety assurance that is observable, traceable, reconfigurable and flexible in its way of managing the safety constraints that can be enforced both in system design and during run-time operations. Section 3 presents the reference architecture and section 4 details the implementation guidelines of the reference architecture applied to AV. Finally, the conclusions and perspectives of this work are presented.

2 Requirement Analysis

Our approach aims at proposing a reference architecture intended to provide safety assurance at run- time as the result of active adaptation processes. These processes are dependable and intend to guarantee both internal or external safety constraints of the AV. In order to cope with the set of NFP required for AV safe systems (i.e. observability, traceability, reconfigurability and scalability), we propose to put emphasis on the following design principles:

— Consider safety as a control problem guiding the design of an effective control architecture able to react or to reduce adverse events. System Safety considers the enforcement of safety by both the elimination of the hazards by design or the management of the hazards by control. Enforcing safety in the vehicle on the operational contexts requires to identify the constraints that can assess safety at run-time through monitoring, diagnosis or a full close adaptation loop;

— Enforce vehicle safe behaviours upon the different contexts the vehicle can operate in. The system architecture needs to offer ways to satisfy the multiple safety constraints by adapting its design or by ensuring it by run-time assurance functionalities. As the safety constraints can be highly contextualized (i.e. assess in a specific scenario or use case), the architectural model has to be able to capture and store this information;

— Facilitate the integration of system functions following a black-box approach, exposing only services and interface specifications (e.g., input and output of messages). Functions allocated to the components where safety is monitored or assessed can be discovered and integrated using components-oriented architecture. For example, the in-

tegration of components allowing observations, processing, and analysis and behaviour prediction (e.g., neural network, machine learning or Markov process) would just result from the discovery and plugin of the well-adapted component within the architecture. The only preliminary requirement is for the component to be registered and specified in the knowledge base. This registry should include attributes such as required resources, expected output, exhibited component properties upon safety;

— Provide built-in and bolt-on monitoring, diagnostics and adaptation processes for individual AV mechanisms that can be composed and orchestrated at run-time in the overall design. For the sake of observability and traceability, we want the system to be able to perform using both built-in or bolt-on monitoring, diagnostics and adaptation processes. While bolt-on processes can be connected to an existing system without requiring significant modification of interfaces to the target system, built-in processes may impose design requirements. They require the system to offer an extensible and pluggable architecture where new components can be easily added with their respective logic (e.g. goals and values for the decision method). The composition and orchestration of the processes contribute to bring end-to-end visibility across the different services and to provide deep visibility into each service's performance and logic;

— Specify and store the adaptation processes expert knowledge in distinct knowledge bases following distinct roles: system architecture, interfaces, goals and operational context measured values. The presence of built-in processes imposes to have a specification model of the architecture that defines the components, their behaviours, functions and their respective logic that is perfectly aligned with the safety management knowledge-oriented bases. For example, those knowledge bases result of the model of the architecture representing the system (i.e. model-based representing and tracing the behaviours and constraints of the system to each function and physical entities) or the representation of the operational environment (i.e. observable state of the environment used for self-awareness);

This section has identified the safety requirements assurance guiding the design and implementation of an AV architectural framework. In the next section, we introduce our reference architecture that comply with those requirements by detailing and addressing each system requirement.

3 Reference Architecture

Based on the previous requirements analysis, this section proposes a reference architecture suited to enhance AV system with two levels of adaptation in order to guarantee safety constraints. First, we present the step by step construction of the reference architecture. Finally, we detail the main components of the resulting reference architecture.

3.1 Architectural Design Process

Our main base architecture is an existing Renault's vehicular architecture called ADCC (Autonomous Driving Commuter Car) that provides autonomous driving capabilities to a vehicle. We have based our study on this architecture since it is indispensable to start from a real cyber-physical implementation of an autonomous vehicle, allowing to consider the real

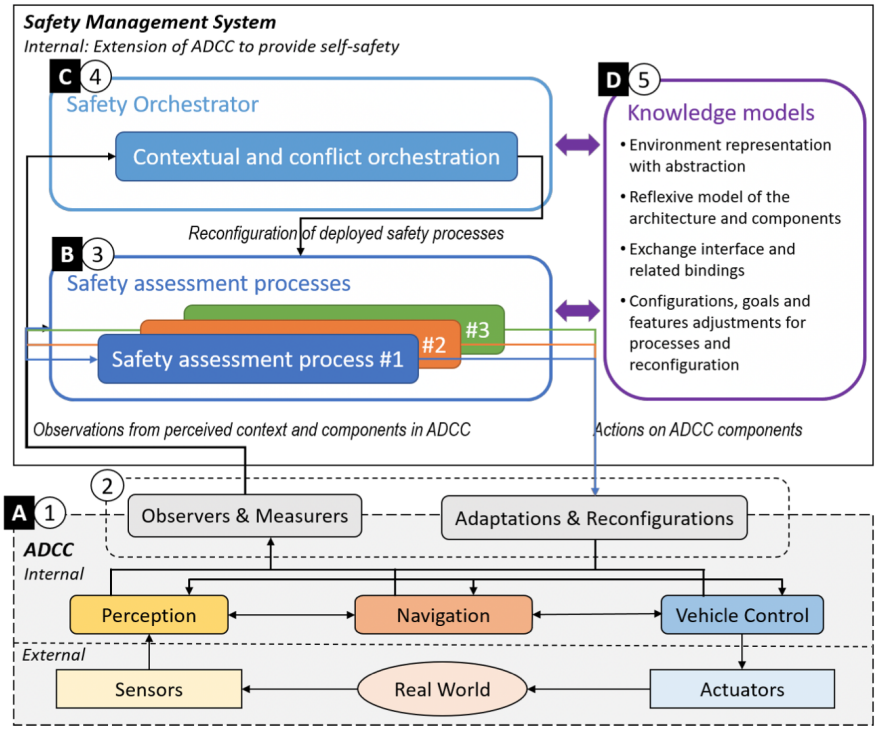

FIG. 1 – *Reference architecture involving several levels of adaptations*

restrictions and capacities offered and to evaluate its potential extensions. We are mainly interested in enhancing this architecture in order to guarantee safe decision functionalities and to guarantee appropriate scaling to the range of safety concerns we aim to consider.

On this basis, we propose a system enhancement aimed at adding an additional component intended to manage safety by integrating external expertise to the existing ADCC architecture. Figure 1 illustrates the reference architecture and details the perception, navigation and vehicle control mechanisms that are connected to the vehicle world via the sensors and actuators entities (section 1). We call this component extension the Safety Management System (SMS)

In order to manage safety, the system needs to be monitored by observing the components and dataflow from the existing ADCC architecture. For this matter, we propose to enhance ADCC with an interface of observers and measurers depicted in section 2.

As we want to perform specific adaptations or reconfiguration of components of ADCC, we also add the corresponding reconfiguration interface in section 2. The component reconfiguration can then be performed by a control loop using those two interfaces with the first as data input and the second as action output. The processes involved in each loop ensure the compliance with a safety constraint and express the links between the observations and the reconfiguration actions as illustrated by section 3. In order to reduce the complexity, we propose to specialize each process per safety constraint so there will exist as much control loop

process as safety constraints. The architecture results in several parallel processes that share similar sources of observations and components to reconfigure.

In addition, the different safety constraints are possibly conflicting and may intervene only in a specific context. To manage the concurrent safety constraints to guarantee, we propose a macro process to manage the context and conflicts (see section 4).

Finally, to ensure the system to be evolutive, traceable, have its components embeddable and interoperable, we propose to follow a common semantic model (see section 5). It enables each of the interfaces to communicate and favour a granular design of the knowledge (i.e. expert knowledge on the observations, reconfiguration, decisions, safety constraints and potential conflicts).

3.2 Components Description

1. ADCC interfaces to the SMS. In the scope of our study, we map sensor information to the perceived objects by the ADS (i.e. reading vehicle's sensor information), the intention (i.e. reading vehicle's planned manoeuvres), operations of the ADS (i.e. reading vehicle's trajectory) and other component status (e.g. vehicle profile). Reconfiguration information encompasses the recommendation of policies, manoeuvres or imposed trajectory for the ADS to adopt (i.e. affecting the configuration of ADS functions at different levels).
2. Safety Assessment Processes and Behavioural Safety Assurance. The first level of the reference architecture hosts the parallel dependable processes that enforces the different safety constraints at run-time. Each safety assessment process is allocated to the monitoring and the assurance of a specific safety constraint. The process itself consists in a composition of functions operating either monitoring, diagnostic from identified symptoms, planning or reconfiguration to adapt how the ADS is behaving. The specifications are based on the constraint requirements and the restrictions of its operating context. For example, the minimum distance between a pedestrian and the vehicle shall be at 5 meters in urban areas at 30km/h.
3. Safety Orchestrator and Context-Dependence of Safety Assurance. The second level of adaptation is built above the first level as a macro process to reconfigure the deployed safety assessment processes upon the observed context. This reconfiguration operates according to the context, the conflicts between constraints and the available resources. For example, the minimum distance between a pedestrian and the vehicle shall be at 12 meters in urban areas at 50km/h, and 25 when the vehicle position becomes too uncertain (i.e. localization may not work correctly). This orchestration results in structural adaptation of the deployed processes to fit the actual context and ensure relevant and safe configurations.
4. Knowledge Models. The two presented levels of adaptations are designed as generic processes that operates appropriately with their operative information. In our approach, we propose to make the process agnostic. They can obtain the required knowledge from a shared knowledge base. It will be only at run-time, when deployed, that they will may only acquire the knowledge to operate. In this reference architecture, we explicitly store all required information regarding the context, the deployment rules of constraints such

as context-dependence and restrictions, the configuration of the system, and process operations into models that are accessible through a shared knowledge base.

4 Reference implementation

To achieve observability, traceability and flexibility of the reconfigurable architecture, this section presents a requirement analysis for the architecture implementation motivated by the composition of existing solutions for a flexible, composable and observable architectural approach.

First, we detail how the successive adoption of the Autonomic Computing paradigm, the microservice architectural style and the knowledge representations based on semantic models may contribute to design a self-managed system with the expected attributes. Next paragraphs present the consecutive collection, composition, allocation and orchestration schemes based on the combination of those structural and behavioural concepts. In our case, the methodology consists of decomposing the safety constraints into structured and manageable safety assurance processes and their respective functions. A more detailed implementation view of the architecture is also provided to illustrate the results of the decomposition and application on safety in a MBSE tool. Finally, details how the knowledge of the self-managed system should be structured and illustrates its applications to the AV.

The system's attributes of observability, reconfigurability, traceability and flexibility have been identified as required for our self-managed system to appropriately tackle the different challenges of AV safety. This section identifies the potential solutions to satisfy these requirements based on Autonomic Computing, microservices and semantic knowledge representation approaches.

4.1 Autonomic Computing

The Autonomic Computing paradigm proposed in Kephart et al. (2006) provides a hierarchical organization between components to perform relevant adaptations via so-called MAPE-K autonomic loops. Those loops are constituted by a chain of components providing the separated functions of Monitoring, Analysing, Planning, and Executing (MAPE) operating around a shared knowledge base. Each of the MAPE-K loops offers a specific reconfiguration that can be implemented within a discipline, i.e. coordinates the same type of adaptation (e.g. self-configuring, self-healing, self-optimizing and self-protecting). It may also address across different disciplines as they coordinate a mixture of the self-* capabilities. The management of a MAPE-K loop by a higher-level loop is defined as autonomic orchestration. It contributes to building hierarchical decisions that are made possible thanks to the genericity and composition of the MAPE-K loops.

The Autonomic Computing paradigm is well-suited for self-managing architectures where components or resources need to be reconfigured based on the monitored environment conditions and guided by policies and goals. The adoption a hierarchical structure imposes the reconfigurability with different levels of specialized decisions that can manage other components or be managed. AC provides such decomposed structure for decisions (MAPE) and the possibility of orchestration to develop self- adaptive systems.

The design of autonomic processes based on the four MAPE functions and the knowledge base contributes to simplify the complexity by defining loops with a specific concern making it more manageable over the time and easily observable. Besides, each MAPE function's inputs and outputs can be observed, logged and replayed if necessary.

We have seen that the four MAPE functions and the associated knowledge base compose the decision process. Consequently, the chain of commands between the components is made explicit by the respective definition of their roles, their specified I/O, their goals and policies that tune their functions, and their operations that implement a specific set of techniques, methods, and algorithms. The whole definition and structure of each function contribute to facilitating the traceability in the system during design and run-time.

In addition, the MAPE functions may not have only one operation or implementation possible. Trades off at a given time may have imposed specific definitions allowing only a certain spectrum of mechanisms to be used. However, future design may replace how the function is operated by displacing the component in the decision process. The replacement of the components of a MAPE-K loop or of the whole loop is possible as long as the functions and roles are maintained, and I/O and knowledge are appropriately updated. Having replaceable components for each function promotes maintainable and appropriate evolutions of the architecture over the time in the development iterations or during its operation with the selection of the appropriate process.

The hierarchical structure offered by the AC also complies with requirement extensions as we can add new interfaced managed resources or new MAPE loops to perform a new specific process or supervise existing ones.

The Autonomic Computing paradigm contributes to cover the structural range of our expected attributes as it specifies a structure of decision for self-adaptive systems and also contributes to the behaviour implementation, management and traceability (modelled and handled).

4.2 Microservices

The microservice architectural style as described in Fowler and Lewis (2014) is applied in the IT domain to reduce coupling and break down monoliths in web-service architectures improving thus their scalability. It enforces a different approach to implement the capabilities, functions and features. Each component is designed to do only one job; "Do one thing and do it well"'. This type of usage on cyber-physical systems is reflected in their structure and design enhancing the loose coupling and high cohesion of its services. Additional knowledge is necessarily required to describe how components should connect, how the capabilities and features are associated, how the microservices can be deployed (i.e. semantic of the applications requirements), the properties we want to ensure (e.g. QoS, safety) and the manner how microservices can be orchestrated (i.e. goals and policies). Microservices also claims to contribute to improve the scalability of software architectures.

The adoption of microservice architectural style appears to be an appropriate solution to ease the scaling and flexibility of the architecture of a self-adaptive system in AV with regard to the diversity of functions involved and dynamic complex operating environment. The microservices contributes to cover the structural range of our expected attributes as it specifies the structure of intercommunication between components and allocation as microservices.

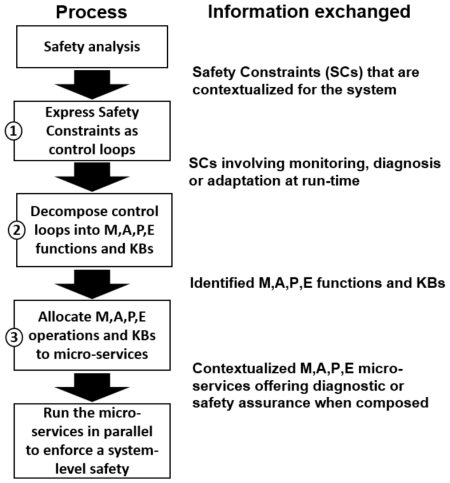

FIG. 2 – *Process of integration of the safety constraints in the system*

4.3 Knowledge Representation and Semantic Model

Model-Driven Engineering (MDE) have contributed to facilitate the development of architecture for complex systems including self-adaptive systems by addressing their representation problems (i.e. flexibility, scalability, and traceability) as presented in Cuer et al. (2018). In MDE, the use of abstract models of the systems separated from the systematic implementation are not only used for documentation but also as the vector of the architecture refinement (e.g. understand, design, develop and maintain a system architecture). Therefore, Model-Driven Architecture (MDA) approach insists on the separation between the system and its implementation with the objectives to guarantee the evolutivity (i.e. being interoperable and reusable), flexibility (i.e. being portable, extendable), and traceability (i.e. containing the system specification and capabilities) of the resulting system architecture.

Ontology-Driven Architecture (ODA) promotes the use of semantic models or ontologies to represent the abstract models of the system of MDA. They contribute to define domain vocabulary, the specification and the capabilities of the represented system. Ontologies provide expression for queryable semantic relationship between the different existing concepts and instances, and provide consistency checking and validation capabilities for the model. The adoption of an ODA promotes a higher level of observability and traceability in the system architecture with machine and human- readable and queryable concepts.

In addition, the adoption of ODA in self-adaptive system contributes to represent and to make accessible knowledge necessary for the system reconfigurability as proposed in Exposito et al. (2009), Diop et al. (2012), Exposito (2013) and Koh-Dzul et al. (2013). In our perspectives, it rigorously and consistently uses models for engineering feedback loops as it captures the adaptation mechanisms and exchanged information.

The knowledge representation with semantic models contributes to cover the structural

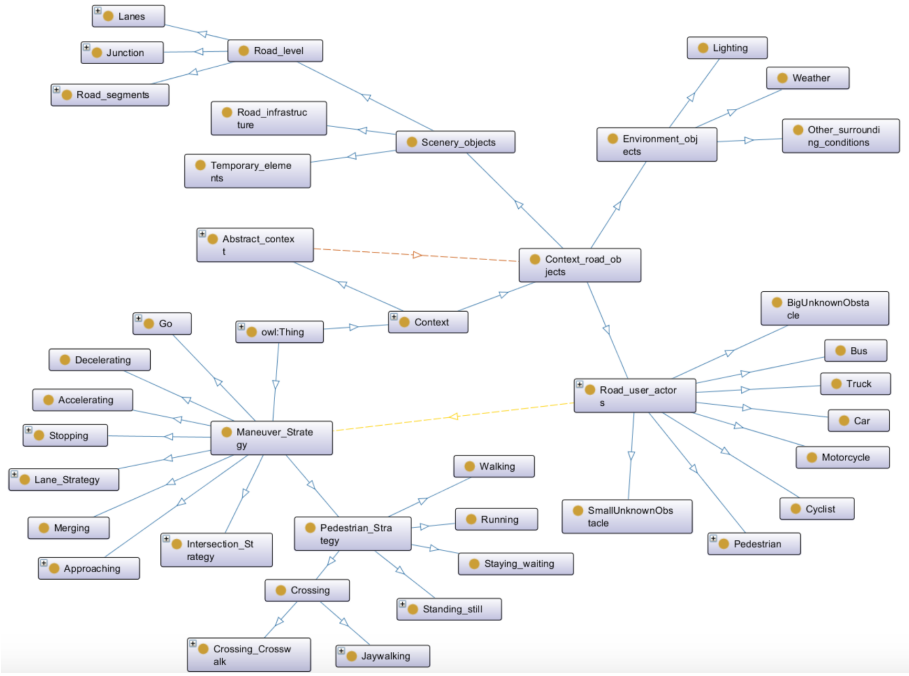

FIG. 3 – *Ontology representation of the environment of the vehicle with road entities*

range of our expected attributes regarding the capture of knowledge and how it can be documented and structured.

Figure 2 summarizes the step-by-step methodology followed to build our reference implementation architecture, guided by the safety analysis constraints, controlled by autonomic and ontology-driven processes loops based on composable and orchestrable microservices. Due to space limitations, in this paper we will only present the ontology-driven knowledge based allowing to guarantee the safety constraints.

5 Specification of the safety-oriented knowledge base

To clearly identify the scope for each source of knowledge required to implement to knowledge based of our reference architecture, we adopt the abstraction of models proposed by Aßmann et al. (2014) in the model@run.time architecture. The following paragraphs present the particular purposes of those models, and their coverage, and the identified operating knowledge in our approach.

5.1 Ontology representing the safety symptoms of the environment of the vehicle and the ADS

The Ontology representing the safety symptoms of the environment of the vehicle is presented in Figure 3. This ontology considers entities such as dynamic objects (e.g. pedestrian, car, bus), static objects (e.g. stationary obstacles, traffic lights), the road geometry (e.g. lane, inter- section, crosswalk), their interactions and also possible manoeuvres (e.g. crossing activity from the pedestrian. Current perceived actions (i.e. manoeuvres) of the entities and their respective interrelations are described using object properties. Additional information regarding the entities can be captured using the data properties (e.g. position, speed, heading, id, age) to integrate some relevant readings or correlations from sensor.

Instead of allocating only one manoeuvre to a road user, we propose to extend the possible cardinality of the relation. The understanding of the behaviours of the other road entity is a fundamental key for automated driving. It is a matter of considering the actions (i.e. what it is currently performing), the intentions (i.e. know what it will perform next) and the expectations (i.e. know what will be the next actions) of each respective entity. They contribute semantically enhance the scene. We see the solution for their representations as twofold.

Firstly, we satisfy the fact that manoeuvres are not all atomic, some can result from the composition of others and some are interchangeable. Such claims can be identified to a commonly occurring problem in software engineering that can be address by the Design Pattern Strategy firstly introduced by the Gang-Of-Four in Gamma (1995). As a way to configure a class with one of many behaviours, it aims to lets the algorithm vary independently from clients that use it. Thus, the term Strategy in Manoeuvre Strategy refers to the application of the Design Pattern and not robotic usage. In application, the Pedestrian Strategy concept contributes to semantically identify and represent the main manoeuvres, moves or interactions the pedestrian can perform. In our scope, we consider that the pedestrian can perform Crossing, Crossing_Crosswalk, Jaywalking, Running, Standing_still, Staying_waiting and Walking.

Secondly, the identification of the perceived maneuvers are subject to uncertainties due to observability restriction raised in Törngren et al. (2018) for example. In fact, the uncertainty reflects the capacity of the observation to be false or partial. Thus, methods as Dempster-Shafer theory and Bayesian method are commonly used to perform the reasoning in an uncertain world for safety monitoring according. The confidence is largely used to express a weighting on how much we are sure of a specific sampling. In order to allow more than one cardinality for road-user/maneuvers relation, we choose to create a Strategy instance for each manoeuvre and attach the confidence of the observed result as a data property. The link between road user and possible manoeuvres are then represented by the has behaviour semantic relation. The existence of Strategy individual associated to a specific road user or the value of the confidence can be used in our framework to create symptoms of specific scenes or situations.

The idea behind this context ontology is to keep a representation of the world that is both human and machine readable. Hence, we have the possibility to store all sort of observations and results regarding the vehicle external context. However, only relevant symptoms from the monitoring (i.e. aggregations and correlations from sensor information processing and ADS information) aims to be stored within the ontology to serve as a base for inference. In fact, ontologies show limitations in processing a large amount of information (e.g. number of axioms in the T-Box, A-Box for the types, individuals, relations and equivalence rules) in

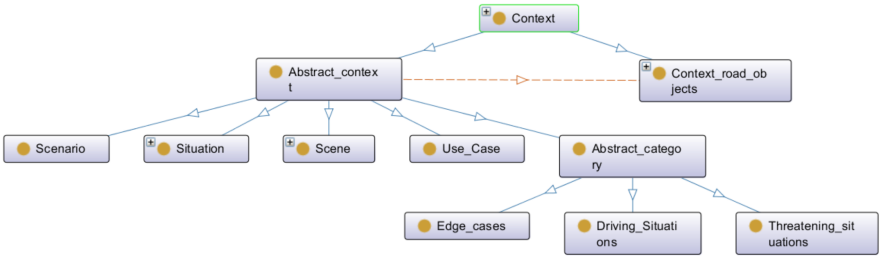

FIG. 4 – *Ontology representation of the use cases for situations encounter using com- position of road entities*

a short time. Certain ontology features can also impact the reasoner performances and might causing unexpected reasoning results according the analysis presented in ISO-26262 (2018).

5.2 Ontology for the representation of the current context of the vehicle as symptoms

Context abstractions such as use cases or situations are also captured as concepts within the same ontology. They aim to offer a higher-level representation of the context is to facilitate scene understanding, scene tagging and identification as presented in Armand et al. (2013), Armand (2016), Bagschik et al. (2017a), Geyer et al. (2014), Zhao et al. (2015) and Geng et al. (2017). We can envisage them as patterns that match the different parts we intend to study like the concepts introduced in Ulbrich et al. (2015). Figure 4 illustrates the different forms of context abstractions we propose to capture based on the following specific representation goals and prerequisites.

A Scene corresponds to a snapshot of the scenery and the self-representation of the dynamic elements (i.e. it can encompass the state, intention or expectation of each of the road objects).

A Situation is an extended representation of the perceived scene where some information are selected (i.e. only consider relevant entities for defined driving functions and subjective restricted observation) and some are semantically enhanced (e.g. added information as relation or property) fitting with the current objectives of the ego vehicle (e.g. goals and values for realizing Yielding to pedestrian safety goal).

A Scenario corresponds to a sequence of scenes using Maneuvers (i.e. from actions or events) as transitions with at least an initial scene. This concept helps representing context abstractions where temporal development is needed.

A use case captures the guidance of one or several scenarios where a functional range (e.g. roadway) is specified and a desired behaviour (e.g. yield to pedestrian) are involved. This concept helps covering the definitions of use cases from ISO-26262 (2018).

Finally, we have included a last category for the gathering of the edge cases or threatening situations in which the system needs to perform a specific strategy or meet specific objectives.

Figure 5 shows the classes of situations covering the Pedestrian crossing the road on crosswalk use case within the black selection. The image is a screenshot of the class hierarchy

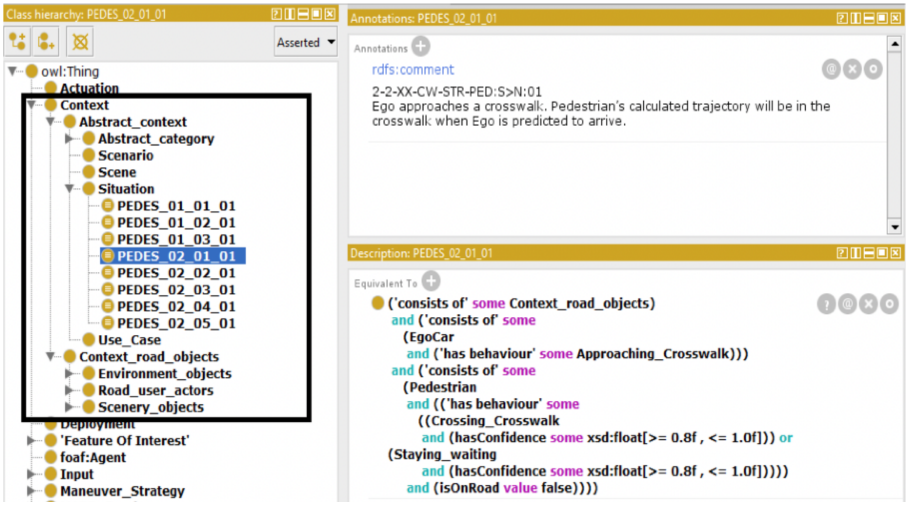

FIG. 5 – *Ontology representation of the use cases for situations encounter using composition of road entities*

of our ontology on the Protégé tool. Each displayed PEDES_XX_XX_XX axiom captures the different entities and describes the expected relations between the different road objects (e.g. EgoCar approaching crosswalk or a Pedestrian crossing the road) that are involved. For example, the highlighted PEDES_02_01_01 situation aims to detect the pedestrian's calculated trajectory will be in the crosswalk when EgoCar is predicted to arrive to the crosswalk. An a-priori definition of the requirements to meet is described as an Equivalent To relation. Figure 5 illustrates the relation of equivalence for the PEDES_02_01_01 situation at the bottom-right panel. The abstraction aims to detect a pedestrian that may cross or have the intention to cross in the proximity of a crossroad but is currently not on the road.

Based on the descriptions of the context abstractions, the reasoner can actually perform scene identification by inferring on the provided equivalence relations and observations. The context identification fulfills the role of Monitor function in the OAM. The identified context abstraction constitutes symptoms in the OAM adaptation loop. We also associate this ontology with run-time representation that abstract our system's configuration and is operable by the OAM based on the SOSA/SSN ontology Haller et al. (2018). Other more complex contexts can be described using more extended equivalences or SWRL rules. In our approach, we only capture high-level observations and do not store raw observations or raw data to keep the ontology at a run-time manageable scale and the reasoning time efficient.

6 Acknowledgment

This work has been financed by Renault and by FUI 23 under the French TORNADO research project focused on the interactions between autonomous vehicles and infrastructures

for mobility services in low-density areas. Further details of the project are available at https://www.tornado-mobility.comhttps://www.tornado-mobility.com

7 Conclusions and perspectives

In this article, we have presented an original architecture resulting from the extension of a real cyber-physical autonomous architecture and implemented based on a microservice-oriented and knowledge-based model-driven framework for designing autonomic and cognitive AV systems.

Within this framework, we combine a set of patterns to perform two levels of adaptations and their respective knowledge. We define the different models and their use in the different functions of the autonomic adaptation. The definition of such patterns and models has been motivated by the need of traceability, flexibility and composability in AV systems. We have introduced the use of ontologies in order to implement the knowledge base of our reference architecture. Regarding performance and scalability, even in this article we have not presented the evaluation of these properties, since the architecture can be potentially distributed within internal or external vehicle infrastructure, dynamic and distributed deployment of MAPE functions could considered for managing large number of processes.

Likewise, in this work we have considered basic situations where some can be considered as atomic. However, a real-world scene would more result of the composition of different abstract contexts with their own attributes variations. As an example, we can consider a perceived scene as a composition of situations involving a pedestrian crossing the road and the EgoCar followed by vehicle.

Future works will propose an extension to this knowledge-based implementation in order to integrate this kind of scenarios. Moreover, next works will address architecture verification as well as the evaluation of functional and non-functional requirements satisfaction.

References

Abdulkhaleq, A., S. Wagner, D. Lammering, H. Boehmert, and P. Blueher (2017). Using STPA in compliance with ISO 26262 for developing a safe architecture for fully automated vehicles. *CoRR abs/1703.03657*.

Amorim, T., D. Ratasich, G. Macher, A. Ruiz, D. Schneider, M. Driussi, R. Grosu, and A. Hoeller (2018). Runtime safety assurance for adaptive cyber-physical systems: ConSerts M and ontology-based runtime reconfiguration applied to an automotive case study. In *Solutions for Cyber-Physical Systems Ubiquity*, pp. 137–168. IGI Global.

Armand, A. (2016). Situation Understanding and Risk Assessment Framework for Preventive Driver Assistance. *IV'14* (2016SACLY008).

Armand, A., D. Filliat, and J. Ibañez-Guzmán (2013). Detection of Unusual Behaviours for Estimation of Context Awareness at Road Intersections. In *5th Workshop on Planning, Perception and Navigation for Intelligent Vehicles*, Proceedings of the 5th Workshop on Planning, Perception and Navigation for Intelligent Vehicles, Tokyo, Japan, pp. 313–318.

Aßmann, U., S. Götz, J.-M. Jézéquel, B. Morin, and M. Trapp (2014). *A Reference Architecture and Roadmap for Models@run.time Systems*, pp. 1–18. Cham: Springer International Publishing.

Bagschik, G., T. Menzel, and M. Maurer (2017a). Ontology based scene creation for the development of automated vehicles. *CoRR abs/1704.01006*.

Bagschik, G., T. Stolte, and M. Maurer (2017b). Safety analysis based on systems theory applied to an unmanned protective vehicle. *Procedia Engineering 179*, 61 – 71. 4th European {STAMP} Workshop 2016, {ESW} 2016, 13-15 September 2016, Zurich, Switzerland.

Cheng, B. H. C., K. I. Eder, M. Gogolla, L. Grunske, M. Litoiu, H. A. Müller, P. Pelliccione, A. Perini, N. A. Qureshi, B. Rumpe, D. Schneider, F. Trollmann, and N. M. Villegas (2014). *Using Models at Runtime to Address Assurance for Self-Adaptive Systems*, pp. 101–136. Cham: Springer International Publishing.

Cook, S. A., H.-H. Fan, K. Pennar, and P. Sundaram (2018). Building behavioral competency into stpa process models for automated driving systems.

Cuer, R., L. PiÃ©trac, E. Niel, S. Diallo, N. Minoiu-Enache, and C. Dang-Van-Nhan (2018). A formal framework for the safe design of the autonomous driving supervision. *Reliability Engineering & System Safety 174*, 29 – 40.

Diop, C., G. Dugué, C. Chassot, E. Exposito, and J. Gomez (2012). QoS-aware and autonomic-oriented multi-path TCP extensions for mobile and multimedia applications. *International Journal of Pervasive Computing and Communications 8*(4), 306–328.

Exposito, E. (2013). *Advanced Transport Protocols: Designing the Next Generation*. John Wiley & Sons.

Exposito, E., J. Gomez, and M. Lamolle (2009). Semantic and architectural framework for autonomic transport services. In *2009 Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns*, pp. 99–104.

Ford (2018). A matter of trust fords approach to developing self-driving vehicles. techreport, Ford.

Fowler, M. and J. Lewis (2014). Microservices: a definition of this new architectural term. *ThoughtWorks. http://martinfowler.com/articles/microservices.html [last accessed on July 06, 2016]*.

Gamma, E. (1995). *Design patterns: elements of reusable object-oriented software*. Pearson Education India.

Geng, X., H. Liang, B. Yu, P. Zhao, L. He, and R. Huang (2017). A scenario-adaptive driving behavior prediction approach to urban autonomous driving. *Applied Sciences 7*, 426.

Geyer, S., M. Baltzer, B. Franz, S. Hakuli, M. Kauer, M. Kienle, S. Meier, T. Weissgerber, K. Bengler, R. Bruder, F. Flemisch, and H. Winner (2014). Concept and development of a unified ontology for generating test and use-case catalogues for assisted and automated vehicle guidance. *IET Intelligent Transport Systems 8*(3), 183–189.

Haller, A., K. Janowicz, S. J. Cox, M. Lefrançois, K. Taylor, D. Le Phuoc, J. Lieberman, R. García-Castro, R. Atkinson, and C. Stadler (2018). The modular ssn ontology: A joint w3c and ogc standard specifying the semantics of sensors, observations, sampling, and

actuation. *Semantic Web Pre-press*(Pre-press), 1–24.

ISO-26262 (2018). ISO 26262 - Road vehicles - Functional safety.

Kane, A., O. Chowdhury, A. Datta, and P. Koopman (2015). A case study on runtime monitoring of an autonomous research vehicle (arv) system. In E. Bartocci and R. Majumdar (Eds.), *Runtime Verification*, Cham, pp. 102–117. Springer International Publishing.

Kephart, J., D. Chess, C. Boutilier, R. Das, and W. E. Walsh (2006). An architectural blueprint for autonomic computing. *IBM White paper*.

Koh-Dzul, R., M. Vargas-Santiago, C. Diop, E. Exposito, and F. Moo-Mena (2013). A smart diagnostic model for an autonomic service bus based on a probabilistic reasoning approach. In *2013 IEEE 10th International Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing*, pp. 416–421.

Lefèvre, S., D. Vasquez, and C. Laugier (2014). A survey on motion prediction and risk assessment for intelligent vehicles. *ROBOMECH Journal 1*(1), 1.

Leveson, N. G. and J. P. Thomas (2018). *STPA Handbook*. MIT Partnership for a Systems Approach to Safety (PSAS.

Leveson, N. G., J. P. Thomas, and MIT (2015). *STPA Primer*. MIT Partnership for a Systems Approach to Safety (PSAS).

Raste, T., H. B. Ali, and A. Houry (2015). Fallback strategy for automated driving using stpa. In *3rd European STAMP Workshop*.

Sabaliauskaite, G., L. S. Liew, and J. Cui (2018). Integrating autonomous vehicle safety and security analysis using stpa method and the six-step model. *International Journal on Advances in Security 11*, 160–169.

Stoica, I., D. Song, R. A. Popa, D. A. Patterson, M. W. Mahoney, R. H. Katz, A. D. Joseph, M. Jordan, J. M. Hellerstein, J. Gonzalez, K. Goldberg, A. Ghodsi, D. E. Culler, and P. Abbeel (2017). A berkeley View of Systems Challenges for ai. Technical Report UCB/EECS-2017-159, EECS Department, University of California, Berkeley.

Sulaman, S. M., A. Beer, M. Felderer, and M. Höst (2019). Comparison of the fmea and stpa safety analysis methods–a case study. *Software Quality Journal 27*(1), 349–387.

Törngren, M., X. Zhang, N. Mohan, M. Becker, X. Tao, D. Chen, and J. Westman (2018). Architecting safety supervisors for high levels of automated driving. In *the 21st IEEE Internal Conference on Intelligent Transportation Systems*.

Trapp, M. and D. Schneider (2014). *Safety Assurance of Open Adaptive Systems – A Survey*, pp. 279–318. Cham: Springer International Publishing.

Ulbrich, S., T. Menzel, A. Reschka, F. Schuldt, and M. Maurer (2015). Defining and substantiating the terms scene, situation, and scenario for automated driving. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pp. 982–988.

Vernacchia, M. A. (2018). Gm presentation for introducing stamp/stpa tools into standards. MIT STAMP Workshop.

Zhao, L., R. Ichise, T. Yoshikawa, T. Naito, T. Kakinami, and Y. Sasaki (2015). Ontology-based decision making on uncontrolled intersections and narrow roads. In *2015 IEEE*

Intelligent Vehicles Symposium (IV), pp. 83–88. IEEE.

Résumé

L'intégration de la dimension "safety" est une exigence essentielle lors du développement et du déploiement des véhicules autonomes (VA). Si de nombreux efforts ont été réalisés au cours des dernières années, la plupart des travaux se sont concentrés sur la capacité des véhicules à naviguer de façon autonome. La "safety" est devenue le principal défi. Cet article propose une architecture de référence qui intègre la notion de self-safety dans les architectures AV existantes. Cette architecture consiste en une boucle de régulation multicouche destinée à gérer les processus d'auto-adaptation afin d'assurer la sûreté en temps réel.