Autonomic Framework For Safety Management In The Autonomous Vehicle

Matthieu Carre

THÈSE

UNIVERSITE DE PAU ET DES PAYS DE L'ADOUR

École doctorale ED 211

Présentée et soutenue le vendredi 13 décembre 2019

par **Matthieu CARRÉ**

pour obtenir le grade de docteur
de l'Université de Pau et des Pays de l'Adour
Spécialité : Informatique

AUTONOMIC FRAMEWORK FOR SAFETY MANAGEMENT IN THE AUTONOMOUS VEHICLE

MEMBRES DU JURY

RAPPORTEURS

• Myriam LAMOLLE	Professeur / Laboratoire LIASD, Université Paris 8
• Christophe CHASSOT	Professeur / Laboratoire LAAS-CNRS, INSA Toulouse

EXAMINATEURS

• Sylvie DESPRES	Professeur / Laboratoire LIMICS, Université Paris 13
• Khalil DRIRA	Directeur de Recherche / Équipe SARA, Laboratoire LAAS-CNRS

DIRECTEURS

• Ernesto EXPOSITO	Professeur / Laboratoire LIUPPA, Université de Pau et des Pays de l'Adour
• Javier IBANEZ-GUZMAN	Docteur / Véhicules intelligents, Division Recherche de Renault

UNIVERSITÉ DE PAU ET DES PAYS DE L'ADOUR

Abstract

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

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

This thesis addresses the research question of how safe autonomy is formulated and managed in the literature. We review safety mitigation mechanisms at run-time employing adaptive behaviours. We identify that AV systems require a handful combination of observability, traceability, reconfigurability and flexibility. Based on these non-functional properties, we propose a framework that incorporates the notion of self-safety into existing AVs a manageable and scalable manner. The framework defines our methodology to represent the safety argumentation as constraints and our reference architecture that involve two layers that operate self-adaptation mechanisms to ensure safety. The first layer is closer to the autonomous vehicle and consists of a collection of dependable processes. They specify requirements and are coupled to control loops to manage the assurance of safety closely. The second layer reconfigures the workflow of the previous layer according the association between the constraints and the requirements. The control loops operate with respect upon the context according to the context-dependence restrictions as well as the state of the AV functions. We also detail the constituent parts and application of the framework, namely, with knowledge representation, abstractions, templates and the mechanisms that connect the control-loops as composable and agnostic microservices. This novel formulation is applied to a use case relating pedestrians, thus describing how the proposed safety approach can be implemented and tested. Results analysis and discussion on the perspectives are included.

Résumé

Titre français

Cadre autonomique pour la gestion de la surêté pour générer et réduire les cas d'usage sécuritaires du véhicule autonome

Abstract en français

Le développement et le déploiement de véhicules autonomes (VA) demandent des efforts très particulier du point de vue de la sûreté. Ces véhicules sont des systèmes complexes caractérisés comme critiques face aux enjeux de sécurité routière qu'ils représentent. En effet, ils doivent être capable de naviguer dans de multiples situations complexes tout en évitant les dangers potentiels, et cela, sans perturber la circulation pour être bien acceptés par la société. La conduite en toute sécurité sous le contrôle total d'un ordinateur nécessite également d'interagir et d'opérer autour de différentes entités au sein de réseaux routiers complexes et d'aborder leurs différents comportements de manière appropriée.

Bien que beaucoup de progrès aient été réalisés au cours des dernières années, le travail s'est concentré sur la capacité des véhicules à naviguer de façon autonome. Ce type de sûreté, nommé "safety" en anglais, est apparue comme le défi majeur, non seulement pour gérer les dysfonctionnements ou les perturbations internes ou externes, mais aussi sur le plan comportemental du véhicule pour traiter les cas de bord ou potentiels inconnus dit "edge cases".

Cette thèse aborde la question de la recherche qui porte sur la façon dont l'autonomie sûre, dite "safe autonomy", est formulée dans la littérature et devrait être appliquée. Nous examinons les mécanismes de contrôle de la sûreté qui interviennent au moment de l'exécution et usent de comportements adaptatifs. Nous identifions que les systèmes de conduite autonome nécessitent une combinaison des propriétés non-fonctionnelles d'observabilité, de traçabilité, de reconfigurabilité et de flexibilité. Sur la base de ces propriétés, nous proposons un cadre qui intègre la notion d'auto-sûreté ou "self-safety" de manière gérable et évolutive dans les VA existants. Ce cadre définit d'une part notre méthodologie pour représenter l'argumentation de la sûreté comme des contraintes et d'autre part notre architecture de référence impliquant deux couches d'adaptation. Celles-ci opèrent des mécanismes d'auto-adaptation pour assurer cette sûreté. La première couche est plus proche du véhicule autonome et consiste en un ensemble de processus vérifiant le respect des spécifications et contraintes. Ils précisent les exigences et sont couplés à des boucles de contrôle pour gérer étroitement l'assurance de la sûreté. La deuxième couche reconfigure le flux de travail de la couche précédente en fonction des couples formés par des contraintes et des exigences. Ces boucles de contrôle opèrent selon le contexte en fonction de ces restrictions contextuelles ainsi que de l'état des fonctions du véhicule. Nous détaillons également les éléments constitutifs et l'application du cadre, à savoir la représentation des connaissances, les abstractions, les modèles et les mécanismes qui relient les boucles de contrôle en tant que microservices composables et agnostiques. Cette nouvelle formulation est appliquée à un cas d'utilisation concernant des piétons, décrivant ainsi comment notre approche proposée pour une autonomie sûre peut être mise en œuvre et testée. L'analyse des résultats et une discussion sur les perspectives sont incluses.

Acknowledgements

The present document, resulting of five years of research, would not have been possible without the help, encouragements and support of many persons. I thank all of them and I present them all my gratitude.

I would like to warmly thank my thesis supervisor, Prof. Ernesto Exposito (Laboratory LIUPPA, Université de Pau et des Pays de l'Adour), for his guidance and advice during my research.

The same applies to my supervisor at Renault, Mr. Javier Ibañez-Guzman (Research Division of Renault, Intelligent Vehicles), who was able to give me valuable advice and an enlightened point of view on the complex subject of the autonomous vehicle, as well as to have shared his knowledge and expertise in this field.

Profuse thanks go equally to Professor S. Despres (Laboratory LIMICS, Université Paris 13), Professor M. Lamolle (Laboratory LIASD, Université Paris 8), Professor C. Chassot (Laboratory LAAS-CNRS, INSA Toulouse) and Research Director K. Drira (SARA division, Laboratory LAAS-CNRS) for their participation to my thesis committee as jury member. I would like to thank again Prof. M. Lamolle and Prof. C. Chassot for the insights, the comments and the effort of their efficient reviewing of the complete thesis.

I would also like to thank Renault for supporting the financing and administrative changes of this CIFRE thesis and for trusting me to carry out research on a very innovative and disruptive subject in the context of behavioral safety.

All my colleagues from the university, Renault, the review comitee and the service companies that I was able to interact or work with during my thesis at the University of Anglet and at the Guyancourt Technocentre.

Special thanks to the doctoral students Mathieu Barbier, Yrvann Emzivat and Edouard Capellier with whom I was able to have long and very productive discussions about the autonomous vehicle and the coordination between our different sectors of expertise.

Finally, I wanted to thank my family for supporting me during these hard years of thesis.

Contents

List of Tables

List of Figures

Nomenclature

4D/RCS 4-dimensional real-time control system

ADS Automated Driving System

AFIS Association Française d'Ingénierie Système

ASIL Automotive Safety Integrity Level

AV Autonomous Vehicles

CPS Cyber-Physical Systems

DARPA Defense Advanced Research Projects Agency

DDT Dynamic Driving Tasks

FSC Functional Safety Concept

GSN Goal Structuring Notation

INCOSE International Council On Systems Engineering.

ISO International Organization for Standardization

Knowledge Bases Knowledge Bases

MBSE Model-Based System Engineering

MRC Minimal Risk Condition

MRM Minimum Risk Maneuver

NFP Non-Functional Properties

NHTSA National Highway Traffic Safety Administration

ODD Operational Design Domain

OEDR Object and Event Detection and Response

PATH California Partners for Advanced Transportation Technology

ROS Robotic Operating System

SAE Society of Automotive Engineers

SMS Safety Management System

UC Use Cases

VV Validation and Verification

CHAPTER 1

Introduction

"Remind yourself that overconfidence is a slow and insidious killer"

- Narrator, *after winning a battle*, *Darkest Dungeon*

Contents

1.1 Background

The transportation systems have represented for human civilizations a key success to explore, expand, exploit and conquer. The extensive and advanced road systems of the Roman empire, the transport networks built by the Inca civilization Qhapaq Ñan and the steam-powered railway systems have contributed to create mighty empires and revolutionize the mobility of people and goods. More recently, the twentieth century has risen the adoption of automotive vehicles to be successful for personal travel. It is only for two hundred years that humans perform this task.

Although driving seems simple, it is not an intuitive process. maneuvers and rules need to be learned and trained, and social legislation imposes to pass a theoretical and

practical exam in order to be allowed to drive and share the roadway. Acquired knowledge and experience involve the driver to deal with a large diversity of sources of information guiding him/her to perform actions (e.g. braking, curve without vision) and make decisions in a limited time based on the anticipation of situations from the vehicle surroundings including environment (e.g. road conditions, weather, etc) and other road users (e.g. pedestrians, trucks, etc).

Humans manage to drive by being intelligent enough to handle the processing of much heterogeneous information and the complexity of performances. However, accidents continue to happen due to human errors. Statistics from the American agency of National Highway Traffic Safety Administration (NHSTA) have demonstrated that most major accidents are involving human in most of the driving crashes [137]. Main reasons can be appointed to the complexity to perceive, understand and act accordingly upon a complex and dynamic driving context. Indeed, road networks are complex environments where different entities move at times in an erratic manner, and it is difficult to infer their intentions. These result in thus hazardous conditions [92].

Over the years, car manufacturers have proposed lots of improvements to simplify the driving operations by offering protective features, driver assistance or even partial driving delegation in order to avoid, prevent and mitigate accidents. Within the past years, road vehicles have become more automated with the use of advanced electric and electronic systems and automotive driver assistance systems (ADAS). On this behalf, Autonomous Vehicles (AV) proposes a paradigm shift in transportation technology to improve driver efficiency through increased levels of autonomy. AV aims to provide a safer, greener and more efficient personal mobility services to a larger market, including for instance, the elderly and impaired driving.

Since 2014, the technical report of Society of Automotive Engineer (SAE) [41] has provided a scope, a taxonomy and several recommendations that are now widely adopted in the automotive field in the form of driving automation specifications and technical requirements. The document clarifies the vocabulary to adopt in the field of vehicles offering driving automation. Both concepts of “automated” and “autonomous” are popularly used for such vehicles and are sometimes claimed to be synonymous. Although we acknowledge that “automated” vehicles may represent robotic systems performing driving automation based on sole algorithms, we prefer to use the term “autonomous”. Indeed, we consider the autonomy in driving systems implies adding consciousness capability in order to adapt the driving process dynamically based on contextual information and in-built knowledge and experience.

Further, the report partitions the driving automation into several Dynamic Driving Tasks (DDT) such as lateral control, horizontal control, monitoring the driving environment, object and event response execution, maneuver planning and enhancing conspicuity (e.g. lightning or gesture). The SAE identifies six levels of automation, each having a specific set of DDT and fallback behaviours. These describe the capabilities that the automated system needs to offer and also determine the attribution of the responsibility to the Driver and/or the System.

At level 0, the vehicle does not provide any sort of automation of the driving task. This level includes even the vehicles that are equipped with Active safety systems (e.g. electronic stability program, intelligent speed adaptation).

At levels 1-2, the system respectively performs either one or both lateral vehicle motion control (i.e. steering) and longitudinal vehicle motion control (i.e. acceleration and deceleration). Adaptive cruise control or automatic lane centering are suitable examples. The driver manually chooses when to engage or disengage the system. The fact that no

preconditions are required for activation means the system does not require to have any knowledge about the context of usage for its activation. The system only fulfils its purpose when the driver determines whether, where and when to enable or disable the system.

At level 3, the automation system has preconditions for its engagement that only permit the use of it on the operational domain. The concept of "Operational Design Domain" (ODD) captures those restrictions as a set of traffic conditions, road types, weather, geography, time of the day and others environmental features. Thus, the automation system requires to be able to recognize the environment in which it can perform to determine the possibility of engagement and disengagement. Disengagement process intervenes before encountering the limits of the ODD at an appropriate time. At this level, the system fallback is performed manually by the driver but at the request of the system itself or by the decision of the driver.

At levels 4-5, all DDT and fallback are operated by the system designated as "Automated Driving System" (ADS). Compared to level 4, level 5 is no longer limited by the ODD for specific driving operational conditions, and thus vehicle at level 5 may drive in all possible known and unknown operational conditions. The driver also has the possibility to interfere with the system and can request control back according to the system usage. Such operations involve the system to perform in real-time self-diagnosis and self-adaptation without any human intervention. The driver could intervene on the system and request control back according to the system state. In order to perform self-diagnosis and self-adaptation in real-time, the system needs to have specific sources of knowledge and reasoning capabilities. Even if the levels of automation are mutually exclusive, the scope and granularity of the sources of knowledge become wider at each new level. Moreover, to automate driving decision, it is also required to know and understand how the system works: how actions are performed (dataflow), the system's abilities are expressed and how its capabilities and skills performances are evaluated in order to carry on safe operations.

However, driving can be regarded as a collection of complex tasks where decisions are sometimes made using only partial information. Thus, if vehicles are to be automated and to be successfully deployed, they have to address the complexities of their workspace whilst relying on incomplete information from their environment that might be delayed or difficult to understand.

Nevertheless, AV have been the centre of interest for many years. They have been moving from operating in constrained environments to public roads. However, until today, none of the automated vehicle of SAE level 4 or 5 has been publicly released for retail. Recently, only ramping up of open-road testing programs has been publicly released by the actors in the AVs field (e.g. Waymo previously Google, Uber, Baidu, GM, Toyota, Renault-Nissan, Aurora, ...). Consequently, we only assist in a transition from prototypes to the production of AVs intended to gain maturity and experience in the field to discover and assess the ADS, before starting the real industrialization.

The broad diversity of functions and configurations required for driving operations has primarily limited the development and deployment of automated vehicle solutions. In particular, ensuring safety requires to characterize, understand, model and manage both the safety of the driving tasks and the appropriate behaviours to adopt in case of failures. Moreover, the representation of safety and its integration can be very complex and challenging. Furthermore, some non-functional properties that need to be guaranteed (e.g. safety, reliability, security, performance) can only emerge in real-time operations. In addition, another source of complexity comes from the large diversity of external and internal actors involved in the system life-cycle alongside to the large panel of internal

and external vehicle services that need to be offered. Those external and internal actors are referring to both humans (i.e. users, designers, experts and stakeholders in all the system life-cycle) and the systems involved (i.e. collaborative systems, sub-systems, components, functions, capabilities, etc.).

A generic framework is required in order to address the large complexity involve in guaranteeing an AV's acceptable level of safety. Moreover, this framework should include various requirements and system capabilities. These can be defined by attributes such as traceability, observability, uncertainty, adaptivity, and flexibility.

Recent research works integrate AV capabilities and non-functional safety properties throughout the system architecture analysis, design, development and operation. They propose frameworks including several patterns, methods and models to address the referred attributes.

According to Ramaswamy et al. [117], adopting a system's perspective provides a systematic means to specify the different aspects of software architecture development and their interplay as part of a framework. Within this context, to assess a "safe by design" architecture, we need to systematically integrate: the complexity of the scene, system capability and safety concerns (i.e. context-dependent capabilities and requirement traceability); uncertainty management which emerges from a partially known environment (uncertainty propagation); monitoring and assessment of quality requirements contributing to an holistic view of AV safety (observability and adaptivity): reusability and maintenance of the system for changes, extensions and replacement (i.e. to favour flexibility with the management of the operational competencies of AV to future changes).

In this thesis, we propose a framework aimed at integrating AV capabilities and modelling synergies, and able to address non-functional safety properties of AV systems. Our reference framework architecture is designed to support run-time monitoring capabilities and to implement actions intended to satisfy AV safety constraints. This is based on two system perspectives: For the developer, the various components can be specified with knowledge able to represent the attributes and mechanisms to perform self-management services (i.e. discovery, composition, orchestration, deployment, etc.). For the system integrators, the framework offers an extensible, flexible and pluggable architecture where new components can be added, and specialized stereotypes can be used to be dynamically integrated. The approach seeks to integrate the referred features into a single framework where the safety properties can be managed and guaranteed.

The objectives of this chapter are: to provide the context to this research and introduce a typical functional architecture for autonomous vehicles (section 1.2). To introduce safety as applied to autonomous vehicles including the links related their deploying which define it (section 1.3). To provide the rationale and formulate the research questions (Section 1.4) and define the thesis objectives (Section 1.5).

1.2 Context

This section defines vehicle autonomy, the need for safety in AV, the complexity of the working environment and the need for system engineering.

Vehicle Autonomy

Full computer-control vehicles have the potential to provide accessibility, improve road safety, and optimize the productivity of the driving task. Autonomous vehicles should

enhance people's mobility, not able to drive [71].

The introduction of machine intelligence into these systems has broadened their scope of work. Not only they are able to offer more functionalities, but it has increased their capacity to scale, adapt and perform efficiently and effectively. With the progress of Artificial Intelligence (AI), automated driving technology can contribute even more and propose new services within transport systems.

Definition of Autonomy and Challenges

Autonomy of a system refers to its ability to make decisions independently and self-sufficiently [41, 77, 106]. It can also include the capacity of self-governance [134].

Abeywickrama et al. [5] have proposed a relevant decomposition of autonomy into four attributes: Reflexivity, Evolvability, Model-based system engineering and Uncertainty. Reflexivity implies the system has access to the knowledge of its components, current status, capabilities, limits, boundaries and inter-dependencies with other systems. The system can be considered as self-aware as it knows itself.

Evolvability represents the ability of the system to evolve in dynamic situations where new services, new requirements and alternative configurations can emerge. Thus, it can perform self-organizing and self-optimizing as evolutions of the system to provide more context relevancy.

Model-Based System Engineering (MBSE) helps to represent the system as models maintaining system requirements, design, analysis, verification and validation activities throughout the life-cycle of the system. MBSE provides fundamental guidelines to design and develop complex systems in particular to include autonomy capabilities. Uncertainty is usually mitigated by robust solutions at design-time in traditional approaches. However, they reveal to be impractical in highly dynamic environments. Run-time models have been adopted to cope with uncertainty achieving both reflexivity and evolvability.

The referred attributes help to understand what is the autonomy of the system in terms of needs, requirements, processes and complexity of the overall design. When applied to vehicles, the autonomy required by the system to delegate partially or entirely the driving tasks implies a totally different design approach that needs to integrate safety, reliability and security dimensions.

Autonomy for vehicles may require a more global design approach as the considered system does not only include the vehicle (e.g. cooperative as involving communication with infrastructure or with other vehicles). Indeed, a distributed system design approach needs to be adapted in order to consider not only the vehicle itself but also the other systems interacting with.

For the clarity purposes, we assess the description of the automated driving system and vehicles equipped following the taxonomy in Recommended Practice published by SAE International [41]. The suitable term to designate the system operating the driving automation is the "Automated Driving System (ADS)" and the vehicle is referred as "ADS-operated vehicle" or "automated vehicle". Cooperative systems are the naming for a system that includes the vehicle and communications with outside entities (infrastructure, services and any other road actors). However, for the sake of consistency in this thesis, we prefer to designate AV as "autonomous vehicles" to purposely target the vehicles that can perform its actions independently and self-sufficiently, whereas all "automated" vehicles do not. Consequently, the scope of the thesis will include these ADS addressing the fact to the vehicle to perform its actions independently and self-sufficiently from an autonomy perspective (i.e. reflexivity, evolvability, model-based system and uncertainty).

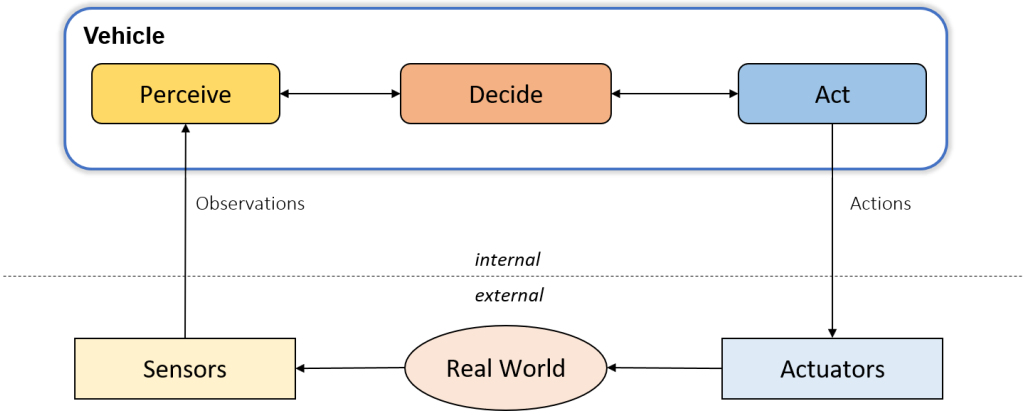


Figure 1.1 Representation of a computer-controlled vehicle

Autonomy and Highly Automated Vehicles

Driving is a complex task; a machine is being controlled in an almost random environment where multiple entities converge and where the environment in terms of man-made infrastructure and the weather can make the whole task more difficult. Driving is governed by road geometry, the behaviour of the entities (e.g. other vehicles, power two wheels, pedestrians, etc.), the weather, traffic rules, etc. Numerous hazardous situations arise due to multiple disturbances, including the unexpected behaviour of entities, limited performance of the systems in use (e.g. lights at night), dramatic changes in weather conditions, etc. A significant source of concern is the interaction that exists between the different entities and the influence that their behaviour will have on the behaviour of the other entities. Statistics have shown that most vehicle accidents occur due to human error [137]. The origins are multiple including the lack of situation understanding and the ability to anticipate what other entities might do, though lately distraction due to the interaction with other devices (e.g. mobile phones) is emerging as a big issue.

When a computer rather than a driver controls the vehicle, the controlling system needs to sense, reason and act, as shown in Figure 1.1. Sensing is done through a perception system that builds a digital representation of the vehicle's immediate environment. Reasoning can be partitioned into situation understanding of the digital representation and the relationship of the vehicle with its environment and decision-making, which is the product of such understanding and that it governs the vehicle behaviour. Finally, the results of the reasoning are executed by acting on the vehicle.

A major source of difficulty is uncertainty associated with the environment due to the limits of the system itself. We are not sure with certainty what entities might do, prediction on their behaviour might be wrong or misled (e.g. an indicator shows the vehicle turning left, whilst the vehicle will go straight). Artificial sensors have multiple limits, in terms of resolution, fields of view, etc. Further, when inferring information from their raw data, there is much uncertainty associated to it, that is it is difficult to estimate precisely the type of entity and distance in front of the vehicle. Despite major advanced in machine perception due to the rapid adoption of machine learning (ML) methods like Deep Learning, the perception remains a challenge. The uncertainty associated with it means that there is an incomplete representation of the environment, upon which decisions depend.

From a safety perspective, the uncertainty associated with the result of machine-controlled systems is a significant issue.

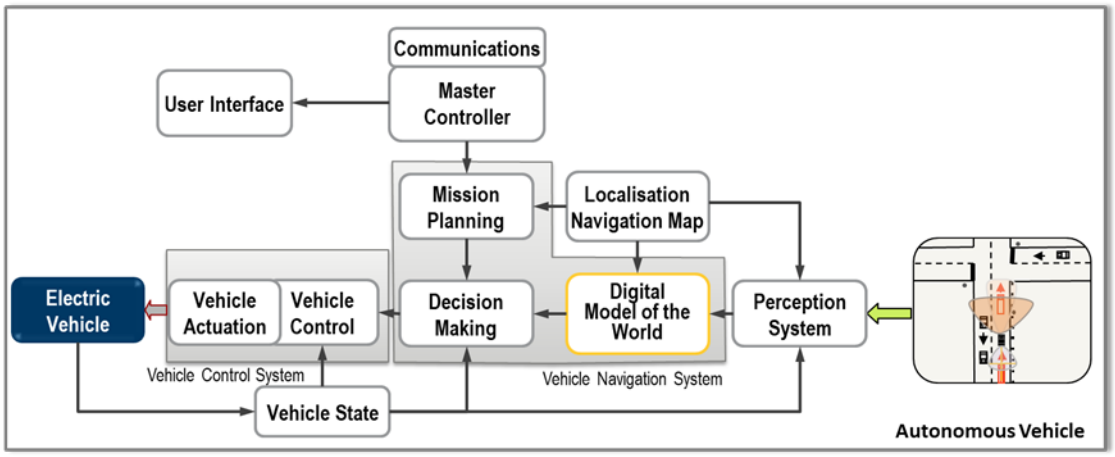


Figure 1.2 Functional Architecture of Renault's research AV prototypes (Courtesy of J.Ibanez-Guzman, Renault).

Functional Architecture of an Autonomous Vehicle

There are different vehicle architectures. However, all provide different elements of the sense-reason-act automation paradigm. The functional architecture described in this thesis is based on the one developed by Renault as part of their autonomous research vehicles. Figure 1.2 shows a block diagram representation of this functional architecture.

The different functional components of the reference architecture shown are described to provide the baseline that is to be used for our analysis in this research.

Perception System Perception provides information about the vicinity of the vehicle environment such as static and dynamic elements, their states, dynamics and behaviours (e.g. current actions, intent). AV systems comprise sensors used to capture in general radiant energy in the scene (i.e. exteroceptive sensors), complex algorithms are used to detect, locate and recognise a potential feature of interest (e.g. a pedestrian) [71]. The perception system consists of a combination of different proprioceptive and exteroceptive physical sensors. For example, ADCC is equipped with physical sensors such as scanning lasers (LIDAR), video camera with stereo vision and RADAR. The different sources of information are then fused using sophisticated algorithms to obtain a list of the different entities classified and annotated.

Perception is a complex process that is limited by the physics of the sensors used, which leads to undefined areas, uncertainty in the measurements, and delays [71].

Localisation System and Map The localisation function estimates the vehicle pose as its position and orientation for an absolute or relative reference coordinate frame or determining the whereabouts of the vehicle [71]. Usual localisation systems use Global Navigation Satellite Systems (GNSS) such as Global Positioning System (GPS) or Galileo. As it based on weak radio signals from constellations of GNSS satellites, the location estimations are usually subject to errors due to noise and physical disturbances (e.g. urban environments, atmospheric effects with ionospheric and tropospheric delays). The combination with such systems as perception allows performing sensor and map tracking that provides higher integrity measures. The map stores information regarding the road

network. It is used to determine the best trajectories the vehicle should follow. High-Definition maps store lane level information. Their geometries are very accurate.

Vehicle Navigation System The function is to understand whether or not the vehicle can traverse the immediate environment, relies on perceiving the presence of obstacles along the desired path [71]. It includes a series of algorithms used for mission planning, decision making, situation understanding, and behaviour generation to determine the path that the vehicle should follow. There are three types of decisions: Strategic, Tactical and Operational. Each of them contributes to tackle with granularity the specific tasks of the driving automation and generate behaviours to accomplish the mission.

The digital model of the world is built from the perception system and the map. This world model is used to perform for some situation understanding and initiate the pipeline to decision making. Situation understanding consists in a machine-understandable model of the world that reflects the scene of the road elements and the vehicle's current state (e.g. vehicle motion as position, speed, acceleration, yaw rate, functional capabilities and degradations, etc.).

Based on this representation, the decision making determines the set of appropriate behaviours to adopt, generates possible trajectories with the help of the map and sends the final trajectory to the vehicle control system.

Vehicle Control System Control ensures that the vehicle follows the desired trajectory by commanding the vehicle's actuators. The actuators are generally commanding the steering mechanism or propulsion system of the vehicle platform. The trajectory to follow sent by the navigation system consists of several nodes where the expected heading and speed are specified. Constant communication in the form of a close-loop ensures that the vehicle is as desired by the navigation system, and if not, provides proper trajectory corrections.

Master Controller System The master controller system consists of a supervisor that monitors and controls the state of the system and its individual components and health. It maintains the coherence of the underlying systems and ensures that these navigate safely or fail-safe [71]. It consists of monitoring mechanisms and processes that are triggered in case of performance degradation or malfunction of any of the systems.

Communications Gateway This system contributes to making the vehicle-centric architecture communicates with the external world, such as other vehicles and the infrastructure. The shared information can take different forms like road usage and states, scene occlusion in intersections, prediction of potential safety threats and other external data sources access. These source of information are integrated into the world model of the vehicle, which gives more consistency to future decisions.

Autonomous Vehicles, Operational Complexity

Driving even in nominal conditions is already a complicated task, that requires cognitive and physical abilities, only possible after some training and evaluation by skilled authorities (the driving test). In this section, a preliminary insight into the complexity of operations is presented from a computer-controlled vehicle perspective. The purpose is

to infer some of the issues that these systems must address even when disturbances such as aggressive drivers, harsh weather conditions, etc. are ignored.

Negotiation. Driving a vehicle implies some negotiation between the subject vehicle and other entities; instances occur in which drivers negotiate with other drivers in a very subtle manner. At a road intersection, for example, drivers look each other, observe attitudes and then decide. For a computer to effect such behaviour is challenging [132].

Interaction. They are different levels of interaction, for this purpose, for this it is necessary for the subject vehicle to communicate with other entities and to understand any signs that should be enabling inference of their intentions [69].

Expectations. As the environment is observed, there are certain expectations from other entities to be predictable; these are dependent on the context and the manner in which the subject vehicle is perceived. That is, decisions are taken based most of the time in such expectations. However, if there is a difference between expectations and the actual behaviour hazards arise. For example, a vehicle is expected to slow down as it arrives at a yield sign intersection, it if does not a hazard situation will emerge [91].

Incomplete Perceived World. Current sensors are far from perfect, their layout on the vehicles is constrained by the vehicle geometry itself, safety, style configurations etc. These result in incomplete representations of the world; further, occlusion will occur; thus, entities will be hidden or difficult to classify. For a computer-controlled vehicle, the unavailability of a full representation of the world makes the understanding and decision-making tasks more complex [46].

This insight has shown that ensuring safety for an autonomous vehicle is difficult, uncertainty and task complexity are major issues even in nominal conditions. If disturbances are added, the complexity is much higher, hence the need for safety-related mechanisms that could warrant the safe operation and social acceptability of autonomous vehicles. We have also seen that the functional architecture of an autonomous vehicle is also complex, the whole implies that a systematic approach based on systems engineering principles is needed to address such levels of complexity [22].

1.3 Safety in Autonomous Vehicles

The scope of the thesis is the safety of autonomous vehicles looking into means to warrant the safe behaviour of these vehicles when operating in nominal conditions (i.e. there are no faults within the system nor disturbances in their operating environment) as well as when undergoing either internal malfunctions and external disturbances. The latter include those related to the operational environment and those created by the behaviour of entities interacting with the autonomous vehicle.

This section defines some terms related to safety, provides an insight into safety as a multidisciplinary endeavour and includes an analysis of an accident involving a highly autonomous vehicle.

Definitions

To provide coherence about safety and its processes, we propose to define some terms first. This is based on Koopman's vision [84] and in adequacy with ISO 26262 [1]. The following definitions are applied throughout the thesis:

- **Risk**, a combined measure of the probability and consequence of a mishap that could result in a loss event.
- **Safety**, absence of unreasonable risk.
- **Safety goal**, top-level safety requirement because of the hazard analysis and risk assessment.
- **Safety validation**, demonstrating that system-level safety requirements (safety goals) are sufficient to assure an acceptable level of safety and have been achieved.
- **Safety arguments (or safety case)**, argument that the safety goals for an item are complete and satisfied by evidence compiled from work products of the safety activities during development.

Safety as a Complex Endeavour

Ensuring safety is a core and strategic challenge in the competitive field of autonomous vehicles. Philip Koopman in [79] underlines that autonomous vehicles will not be perfect, but instead, we need to demonstrate that the degree of safety is sufficiently high to guarantee that the human is out of the driving loop. Based on decades of long experience in the field, he states that "even understanding what 'safe' really means for autonomous vehicles is not so simple".

The vehicle is not only required to take drive safely upon the contextual environment (e.g. obey traffic laws) but also to adopt safe behaviours (e.g. deal with road hazards in construction sites). It needs to operate safely even in case of a fault (e.g. localisation revealed to have errors, some sensor stopped working, erratic maneuvers of others).

Autonomous vehicle safety is concerned not only with ensuring the vehicle operates safely under nominal conditions as well as when failures or external disturbances occur, but it also needs to address cross-disciplinary concerns in a coordinated and interdisciplinary manner [79] as shown in Figure 1.3. The concerns are (1) the field of robotics to pursue more resilient machine learning to provide low failure rate and errors in perception, navigation, localisation and decision in order to become better at managing edge cases and adversarial situations; (2) The update of the safety engineering processes to create an end-to-end design and validation process that addresses both these safety concerns [79]; (3) Appropriate software architecture and software development processes to have a full understanding and formalisation of the nature of the autonomous system, its goals and life-cycle; (4) To design tests and validate the driving functions and safety with safe real-world prototype deployments efficiently; (5) To promote self-awareness in each individual AV so that they can detect and react in case of an attack in a safe manner (e.g. information received from the external world gets compromised); (6) To master the different means the AV have to communicate and cooperate with people so that it behaves in a way that is comprehensible to humans (e.g. risks of human supervisor inattention, customer trust in the product safety, AV interaction and negotiation road users that can be ill-behaved or unpredictable); (7) To improve the reliability, sufficient redundancy and

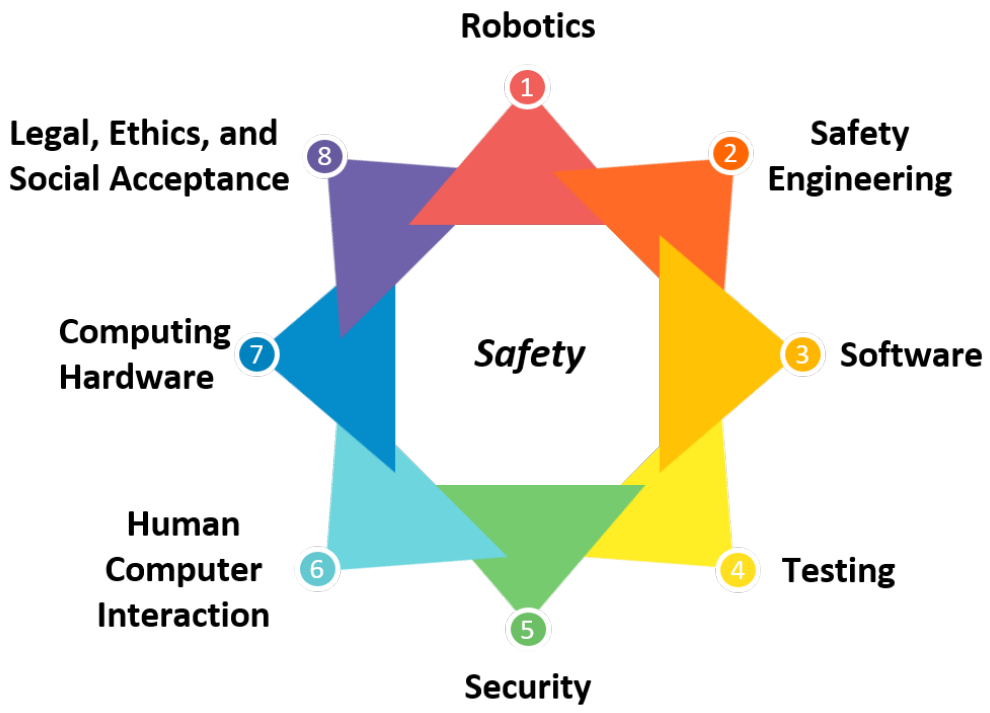


Figure 1.3 Multi-disciplinary and Inter-disciplinary areas needed to ensure safety. After Fig. 1 in [79].

fault tolerance in the ultra-low-cost hardware of AV with safe failure behaviours; (8) To cope with the legal implications of safety as it involves legislation compliance to be addressed, including liability in case of accidents. Another concern is social acceptance: the tolerance to such accidents by society is low [29] as their media coverage can be excessive; the headlines are well known for the fatal accident incurred by one of Uber's vehicle in 2018 [72]. This is in contrast to what it occurred during the early days of the aviation industry, where accidents were part of the development process.

Considering safety for ADS-operated vehicle results in examining the harm the system can cause to the environment (e.g. through malfunction, dysfunction, fault, capability limitation) or the environment can cause on the vehicle (e.g. with obstacles, aggressive behaviour of other users, unexpected events, etc.).

A body of knowledge and processes for safety already exists and has been documented from all fields over the years, it enables safety-critical computer-based systems such as trains, aircraft or automobiles to be deployed. They have contributed to the creation of the main guidelines for the automotive safety domain: ISO26262:2011 [1], and SAE J3016_201806 [41].

The high levels of automation envisaged imply that drives are outside the vehicle control loop at all times. Within the scope of this thesis, only ADS-operated vehicles of level 4-5 are considered. Consequently, the vehicle is required to rely on system autonomy to be completely responsible for vehicle safety.

Compared to other domains, vehicles can potentially achieve an ultimate safe sate quickly. Also called "safe stop" or Minimum Risk Maneuver (MRM), that is the vehicle is pulled to the side of the road. This MRM is more difficult for other safety-critical systems such as aircraft, rerouting them in case of safety concerns might be insufficient.

Since system safety cannot be ensured by an individual process or system in AVs,

Table 1.1 Decomposition of safety into five distinct areas by Waymo [152]

Name	Covered Aspect	Avoided Hazards	Applied Methods
Behavioural Safety	Ensure safe driving decisions and behaviours in both expected and unexpected scenario (system do not fail)	Unjustified, unrealistic or unsafe maneuvers is performed	Functional analysis, Testing on simulation tools and on-road driving,
Functional Safety	Ensure the vehicle to continue to operate safely even in the case of a fault (system do fail)	Fault makes the vehicle unable to perform the DDT as intended	Backup systems and redundancies in the system, handling a safe stop (i.e. a minimal risk condition)
Crash Safety	Protect the protect passengers inside the vehicle	Injury or death due to a crash	Vehicle structural design, seat-belt, airbags
Operational Safety	Ensure appropriate interactions between the vehicle interaction and the humans (HMI, features, intentions)	Misunderstanding making a feature to be misused	Interaction between our vehicles and passengers
Non-Collision Safety	Prevent and mitigate collisions	Causing harm to passengers or bystanders (e.g. doors ajar)	Analysis focus on physical collisions

different perspectives and areas need to be addressed, analyzed and validated. A recent Waymo report [152] categorizes safety into five distinct areas to cope with AV's complexity: Behavioural Safety, Functional Safety, Crash Safety, Operational Safety, and Non-Collision Safety. The decomposition addresses the different concerns through the study of the different sources of hazards (e.g. hazard while system not failing, fault management, unintended usage, etc.). Together, they serve to ensure and validate the safety of the AV. Their decomposition is listed in Table 1.1. A description of the type of hazard that the area aimed to avoid is also included as well as some methods on how to address such concerns.

The current and accepted practices on automotive safety are based on the functional safety approach defined by the ISO 26262 standard [1]. The main goals are to describe the state-of-the-art for the development of safety-relevant vehicle functions and to address the growing complexity of its systems.

As shown in Table 1.1, functional safety mainly addresses the system failures by the analysis of the possible hazards caused by malfunctioning behaviour of electrical/electronic systems. It ensures the system to be prone to the absence of hazards during the presence of malfunctions.

However, a shared belief within the safety community [4, 95, 97, 98] considers that the simple application of ISO 26262 guidelines does not mean that the resulting system is "safe". In fact, the standard does not cover the absence of hazards in the absence of malfunctions (i.e. hazards can appear while the system is not failing). This perspective prevents from identifying some knowledge gaps, under-specified requirements or invalid

safety evidence.

In addition, the established safety standard ISO26262 fails to address the safety validation for recent technologies (i.e. inductive and probabilistic approaches as neural net and machine learning) and self-adaptive systems (i.e. real-time learning, procedural behaviour construction) as they "require considering all possible system behaviours up-front in the design and validation process" according to [79, p. 3].

Furthermore, future reviewed standards, new safety approaches or recommendations such as SOTIF (Safety Of The Intended Functionality or ISO/WD PAS 21448) [2] , STPA [4, 19, 98, 128, 148], J3016 [41] and UL4600 (Standard for Safety for the Evaluation of Autonomous Products) [80, 124] show promising evolution towards the inclusion of all possible hazards and contexts to cover all scope of analysis and operations. However, current available sources are still not mature to cover all the aspects of safety to sufficiently state and validate that an automated vehicle can be safe enough and socially acceptable for High and Full Driving Automation.

Behavioural safety and Operational Design Domain. Specifying the context of the operation of a system is essential, not only for features but also for the safety analysis. Preventing component or functional failures is not enough. To prove that affirmation, let us take the example of the spoon by Leveson [94] to illustrate the importance of context in software. A butter knife is generally safe due to its rounded tips. In the context of cutting butter, the butter knife fills its functionality and perform its safety. However, in the case of a child who pushes a butter knife into an electrical outlet, it is not. Consequently, analysing context helps to identify separation between analysis of scenarios involving failures and unsafe scenarios.

The next section presents insights on the major AV accidents and failure where the automated driving system were engaged and in some at fault.

Summary of accidents involving AV

The intensification of open-road testing programs has been publicly followed by the broad diversity of actors within the AV domain. They are either IT companies (Waymo by Google, Apple, Facebook, Baidu, Tencent, Alibaba, etc.), car manufacturers (Tesla, GM, Toyota, Ford, Volvo, BMW, Renault-Nissan, Mercedes-Benz, etc.), suppliers (Bosch, Daimler, Delphi, etc.) or start-ups (Nutonomy, Aurora, Zoox, etc.).

AV road testing in public roads

AVs are currently entering a phase where extensive testing occurs in public road networks. It allows for enhancing the performance of AVs. Table 1.2 summarizes the accidents in which autonomous vehicles have been involved in recent years. It can be observed that AVs have difficulties on anticipating maneuvers by third parties and that it is very difficult for the systems to understand situations based on limited information from the perception systems. Further, it appears that there are no safety mechanisms that can anticipated malfunctions or the limits of the system when under operation.

The fragility of autonomous vehicle technologies has been demonstrated by two fatal accident within a week in 2018 [72][111]. These raised significant concerns from public authorities and other stakeholders [29]. This could lead to the adoption of stricter legislation and thus delay or derail the adoption of AVs. It is socially unacceptable for multiple prototypes with high levels of automation to be tested on public roads. Moreover, major

Table 1.2 Accidents caused by vehicles with an automated driving system engaged

Date	Involved road actors and infrastructure	Accidents description	Determined cause of accident	Echo in the scientific world
July 2015	Google Lexus SUV, second vehicle	During testing, the Lexus SUV received a rear crash. Result in minor neck strain and whiplash. [65]	Following vehicle fails to stop accordingly.	-
January 20, 2016	Tesla Model S (Autopilot), infrastructure	Model S slammed into a road sweeper on a highway. Result in driver fatality. [66]	No intervention of the driver to regain control of the vehicle. No attempt at braking was made.	Perception system far from perfect; Humans as supervisor may fail.
February 14, 2016	Google Lexus SUV, bus	Google's car crashed into a bus while left merging to avoid construction site. Result in minor fender-bender. [35]	Google's car have emitted an incorrect assumption of the bus driver behaviour. [160]	Failure to negotiate; Wrong predicted intent.
May 7, 2016	Tesla Model S (Autopilot), truck	Model S drove full speed under the truck's trailer. Result in driver fatality. [6]	Car's sensors system failed to distinguish a large white 18-wheel truck and trailer crossing the highway.	Perception system far from perfect; Robustness of sensor vision.
March 18, 2018	Uber 'refitted Volvo' vehicle, pedestrian	Uber's car hit a pedestrian with a bicycle crossing the road outside of a sanctioned crosswalk. Result in pedestrian fatality during night testing. [72]	The threshold for detection rate has been set to exclude a lot of false positive.	Testing and validation can be unsafe in public roads; Humans as safety driver may fail.
March 23, 2018	Tesla Model X (Autopilot), infrastructure	Model X collided with the concrete divider in the road. Result in driver fatality. [111]	No intervention of the driver to regain control of the vehicle despite visual and audible warnings.	Humans as supervisor may fail.
March 1, 2019	Tesla Model 3 (Autopilot), truck-tractor with a semitrailer	Model 3 drove under the tractor's semitrailer at an intersection. Result in driver fatality. [112]	Driver is suspected not monitor the system during 10-seconds or hand-detection failure. [30]	-
July 16, 2019	UDOT shuttle in Utah	The autonomous shuttle operating along a pre-planned route came to a sudden stop. Result in passenger injuries. [120]	Unexpected stop caused by false positive detection from one sensor is assumed. [113]	False positive detection from one sensor is assumed. [113]

concerns emerged regarding the safety validation process, with regards to the increased use of machine learning methods (e.g. in the perception function) as well as the ability of human drivers to act as simple supervisors on for example Level 3 ADS.

Safety practitioners face the difficult choice to deploy vehicles with imperfect technology based on the ongoing process of iterative improvement rather than conventional air-tight proofs of safety [84]. It is important to notice that even Waymo that appears to be ahead of the driving automation competition has still not deployed its fleet of level 4 automated vehicles without a human safety driver to ensure safety [87].

Case Study

An appropriate case study to illustrate the need and the consequences of iterative AV improvement would be the crash involving a Google's vehicle and a bus in a merging scenario close to an intersection [35]. This accident was the first accident where Google's vehicle was recognized as at fault [160]. The objective of this case study is to propose an illustration for the operational complexities of the autonomous vehicles and emphasizes their refinement (e.g. behaviours restriction for unsuitable context, edge cases analysis).

Context. Initially, the car intends to turn right at the next intersection and to position itself on the right lane. The presence of sandbags from a construction area block the right way forcing the vehicle stops on its lane. The vehicle then decides to wait for merging on the left traffic to proceed to activate its left indicators. After a few vehicles have passed, it chooses to start manoeuvring by heading towards the centre of the lane at around 3 km/h. A collision occurs with the side of a passing bus travelling at 24 km/h. The investigation of this incident shows that the car had correctly detected the approaching bus, but predicted that the bus would yield because the car was ahead of it. The test driver who was monitoring the AV without intervention also reported expecting the bus to slow or stop. However, contrary to all expectations, the bus did not return. The result was only a minor fender-bender.

Likely Causes. There are several issues learnt from this collision are several; these can be summarised as follows:

Interaction. As the vehicle is waiting to marge into a slow-moving lane to its left, despite the activation of its left indicators, they collide in vehicle fails to perceive the intention of the AD vehicle.

Negotiation. There is no understanding that the acceptance of the maneuvers by each party have been made. Both vehicles took the right-of-way while the other one should have given it. This failure to negotiate is a likely cause of this crash.

Expectations. The decision made by Google's vehicle expecting the vehicle to give way, hence it committed. Furthermore, the safety driver assumes the same situation. The colliding vehicle was to stop. It was reported by Google [35] that the accident was due to the wrong assumption of the bus intent: Google's vehicle was expecting the bus to yield for it to merge.

Incomplete Perceived World. This accident reveals that misunderstandings and unknown knowledge gaps have existed. They prevent appropriate interactions that leads to wrong negotiations, inaccurate expectations and predictions to finally lead to a collision.

Implications with respect to the AD design. Interaction with other vehicles during complex maneuvers appears as necessary features for AV but definitely goes with negotiation and expectation.

The ADS needs to know the rules of how to interact with people as well as along with courtesy and common sense in driving. Two main rules can be identified as missing or inappropriately applied: "Right-of-way is given but not taken." and "Obey the law of gross tonnage". The right-of-way always needs to be confirmed by the different actors in the road. Then, buses and other large vehicles are less expect to yield to smaller types of vehicles by experience. Likely a normal driver would have yield to the bus despite the later not having right of way.

This scenario shows that assumptions and prediction on other's movement can reveal to be wrong and lead to inappropriate interactions. The perception of the intention of others road actors is also subject to uncertainty. Probabilistic analysis of intention, prediction and expectation could help reduce such collision scenario[91].

Thus, an approach sensitive to the context needs to deal with the complexity and uncertainty of the system and helps to integrate those behaviours, rules, constraints and properties in the system design. It also provides good support for simulation and validation of edge cases. For this purpose, the complexity and the dynamicity of the environment needs to be captured in relation to the driving functions and behavioral safety. However, by specifying the system more on these different views, the design of AV architecture becomes even more complex, making it more difficult to manage (e.g. alternative configurations, distinct profiles, or abstractions for some use cases). Furthermore, regulations from authorities, standards, recommendations and company policy may requires additional non-functional properties to be guarantee (e.g. security, ethic) making the whole quite heterogeneous.

1.4 Rationale and Research Questions

Rationale

From the preceding discussions on lessons learnt from accidents and current practices in AV safety, it is clear that several challenges in AV are an urgent need for better guidance in AV design and development. In order to automate the driving process, it is required to:

Perceive and understand the complexity of the environment as well as taking into account its dynamicity, the uncertainty and being able to predict future events The AV operates in changing and complex environments where a vast number of scenarios can be considered creating a combinatorial explosion. The ADS is expected to perceive, understand and react appropriately to the numerous types of actors, their different behaviours and possible variants.

Uncertainty is inherent to cyber-physical systems such as AV and spread all over the system. However, autonomous vehicles are also safety-critical systems that may potentially harm in case of errors, knowledge gaps, or unsupported edges cases. The AV system

needs abilities to assess and mitigate these uncertainties. Moreover, functional and non-functional properties initially specified may change over the years and the countries.

Master the driving process by understanding and controlling the vehicle internal functions and safety concerns Driving is a difficult operation making it a technological, automation and decision-making challenge. Safety has a simple definition, but ensuring safe driving with AV requires many cross-cutting efforts at different levels of the system. The functional architecture and behaviours need to be related to the safety domain providing enough traceability, observability and flexibility to facilitate the design of AV architectures. Others non-functional properties as security, social acceptance and ethics shall not be forgotten in the AV design.

Act and take driving decision based on perceived external and internal context while guaranteeing safety conditions AV design also faces a multidisciplinary and interdisciplinary challenge. The architecture and adaptive behaviours need to be related to the safety domain and provide safety constraints. This operation needs to happen during system design and run-time. At conception, from a system engineering perspective, the functionalities and knowledge need to be traceable and observable. At run-time, monitoring and diagnostics are required to identify potential knowledge gaps and possibly ensure an acceptable level of safety.

For these three reasons, a safe reference architecture for AV is required to provide an appropriate level of safety to AV to drive on public roads.

Research Questions

This thesis addresses the problem of ensuring the safety of an autonomous vehicle despite its operational context and disturbances it might suffer when driving autonomously. The primary research questions are:

RQ1: How to represent knowledge that links the operational environment with the vehicle components and behaviours?

RQ2: How to use this knowledge to configure the vehicle driving functions in order to ensure its safety at all times?

RQ3: How to configure the system so it responds to the safety requirements while it operates autonomously for a given context in a safe manner?

1.5 Purpose and Objectives

The purpose of this thesis is to provide a framework that enables behavioural safety in a manageable and scalable manner. The approach is based on the active adaptation of a safety system in response to perceived contextual changes in order to guarantee internal and external AV safety constraints in terms of observability, traceability, reconfigurability and scalability.

For this purpose, the following objectives are defined:

- To examine in detail the safety implications related to AD functions and the methods used in different domains to ensure the safe operation of critical systems.

- To model all of the AV behaviours and structures of functions with a systematic approach that connects how the AV system reactions to the different internal or external events whilst it operates
- To formulate a control mechanism that configures the functions determining the behaviour of the AV to maintain the safety constraints, whatever changes in the vehicle evolving context occurs.
- To include a hierarchical coordination function (i.e orchestration) to ensure a reduction in complexity and management of the different available mechanisms.
- To demonstrate the application of the proposed safety approach via a use case study.

1.6 Contributions

The significant research contributions in this thesis are:

1. A survey of existing self-adaptive and reconfigurable critical-systems that performs Quality of Service over a dimension such as safety
2. An approach to model AV functionalities with an MBSE methodology based on ADCC architecture to systematically integrates the behavioural safety requirements as constraints.
3. A reconfigurable reference architecture that copes with uncertainty by operating flexible and composable mechanisms of safety assurance satisfying the required attributes of observability, traceability, reconfigurability, flexibility). Implementation of dynamic reconfigurations in a microservice-based software architecture and application are proposed.
4. Experiments of the framework applied to safety-critical scenarios of AV involving a pedestrian.

1.7 Thesis Structure

The rest of this thesis is organized as follows.

Chapter 2 presents the background and related works. We investigate the principles, challenges and requirements of more flexible autonomy. We detail the safety of the autonomous vehicle, understand its limits and survey the tracks being explored in the literature to facilitate their integration into the design of the AV. Finally, we detail and survey self-adaptive or self-managed systems that perform reconfiguration upon their context and qualities such as safety.

Chapter 3 details the use of an MBSE approach to understand and model the functional and safety (non-functional) requirements for a level 4 AV. The use of different methodologies contributes to understanding the requirements from the stakeholders and modelling the ADS points of view.

Chapter 4 presents the proposed reference architecture that incorporates the notion of self-safety into an existing AV. We detail how it answers to the identified needs for a reconfigurable, flexible, traceable, and observable for the safety management system. Our

proposal is an extension of the ADCC architecture that operates at two levels of adaptations to facilitate the management of the safety assurance processes while guaranteeing the context-dependence and possible conflicts of in their specifications. The resulting framework is developed including its theoretical supporting formulation.

Chapter 5 describes our implementation of the reference architecture within the ADCC architecture using techniques and methods for software development, knowledge representation, scenario construction, environment simulation and safety metrics. The approach is evaluated against a safe use case involving a pedestrian crossing the road in a simple scenario.

Chapter 6 presents the summary of the major findings, design, trade-off issues, future work and conclusions.

CHAPTER 2

Autonomous Vehicles and Safety

"Slow down and enjoy life. It's not only the scenery you miss by going too fast – you also miss the sense of where you are going and why."

- Eddie Cantor,

Contents

2.1 Introduction

Autonomous Vehicles (AV) belong to the paradigm shift shaping future transportation technology. Not only they already improve driver efficiency through increased levels of autonomy, but they also aim to ultimately provide safer, greener and more efficient personal mobility services. Level 4 and Level 5 AV are designed to operate without human intervention. That is, people and goods can be transported in an autonomous manner. This imposes major functional and non-functional heterogeneous operational requirements.

Safety is critical for AV since guaranteeing safe operation and interaction should lead to their acceptance by society. Higher levels of autonomy require system self-awareness and learning in the form of machine-understandable knowledge, mechanisms and actions to operate in a safe manner. While conventional approaches to AV safety show limitation addressing these, new safety approaches are emerging to provide better perception robustness and testing, broader hazard coverage, relevant safety metrics, etc.

System engineering addresses the complexity of these systems by modelling, coordinating and integrating via different views with functional chains as well as cross-cutting approaches to manage non-functional qualities like safety. The complexity of designing and maintaining AV architectures can be managed through a systematic approach that identifies and puts in relation the different operational needs, functions, components,

hardware and qualities. Achieving such traceable effort comes in pair with making the system architecture more observable, flexible and reconfigurable for it to be maintained over time. Evolvable architectures meet such expectations and are currently in our scope of the study for their application in AV for a variety of qualities, including safety.

This chapter initially examines the implications of autonomy in complex systems, in particular, the one related to autonomous vehicles. It is followed by a study of safety in autonomous vehicles by defining first safety related terms, the different approaches applied to ensure safety for AVs. Major issues are identified, trends recognized, solutions presented and key non-functional properties introduced that support the problem formulation of Autonomous Vehicle Safety.

2.2 Autonomous Vehicles and Autonomy

The section presents the definition of autonomy as a system, its components, requirements and challenges as applied to an autonomous vehicle. The manner how autonomy is represented and implemented in full computer-controlled vehicles is included.

What autonomy means in systems, what it consists of, its mandatory requirements and challenges. Then, we detail how autonomy is represented and implemented in full computer-contolled vehicles.

Autonomy in Complex Systems

This section provides the definition of autonomy, details how it can be achieved in complex systems and describes the requirements needs to function within their operating conditions (e.g. per adaptations). This is completed by determining what is necessary to address the challenges encountered by systems having a high degree of autonomy. The review is made through the framework of scalable autonomous Cyber-Physical Systems (CPS). The tenet is that CPS represent the integration of embedded computing into physical phenomena like for Autonomous Vehicles.

Defining Autonomy

The Oxford dictionary defines ‘autonomy’ as “the ability to govern or control oneself”. It has a close relation to the concept of self-governance that refers to states able to manage themselves. The performance of self-control or self-discipline is described as the ability to take your own decisions without being controlled by anyone else (Cambridge Dictionary). Indeed, autonomous systems refer to devices that have the ability to operate without external control (i.e. human control), or more precisely, without persistent control.

In the robotics domain, Alami et al. [7] suggest that autonomy is highly associated with the capability of adaptation of a system by carrying out actions, refining or altering its ordered tasks or its own behaviour upon the current operating context and its goals. They also point that autonomy of a system could be evaluated through “the robot’s effectiveness and robustness in carrying out tasks in different and ill-known environments” or called survivability as a non-functional quality requirement by [52].

J. Sifakis [134] advocates that “autonomy is understood as the capacity of an agent (service or system) to achieve a set of coordinated goals by its own means (without human intervention) by adapting to environment variations”. Where a goal refers to the

importance of achieving a desired target regarding some behaviour, datum, characteristic, interface, or constraint [52]. It is above the level of a policy, any strategic decision that establishes a desired goal, and not sufficiently formalized to be verifiable. A requirement is the mandatory, externally observable, verifiable (e.g., testable), and validatable behaviour, datum, characteristic, or interface [52]. J. Sifakis [134] identifies three aspects to evaluate system autonomy: (1) the autonomy of decision, representing how the system can choose among possible goals; (2) the autonomy of operations, planned to achieve the goals, and finally (3) the autonomy of adaptation, portraying how the system may learn and evolve over time. These previous definitions insist on the ability to perform decisions that are based on the perception of world, the decision process to select an appropriate behaviour or set of changes to operate, and finally the performance of the chosen actions. The author advocates that the complexity of system autonomy can be captured by the interplay of three factors: complexity of the environment, complexity of the mission and non-intervention of humans operators.

In biology, homeostasis involves self-regulatory human mechanisms that contribute to maintain optimal conditions and stability for the living organism features (e.g. body temperature, blood sugar level, breathing, etc.) despite the change of environment, diet, or level of activity. The different processes regulate the features by themselves without specific attention of the human. They can also deliberately be overridden by the human at any moment (e.g. change the rate or volume of breathing). This type of autonomy with removed consciousness of the task and with possible control is known as an autonomic process. Some organs act to regulate locally particular values in the system based on some threshold, watchdogs and detection of anomaly or intrusion. For example, the pancreas regulates blood sugar level with insulin secretion. This specific mitigation action is based on a local specialized knowledge. However, the knowledge of the behaviour to adopt does not appear to be always centralized. Indeed, the detection of anomaly or event is monitored by systems like organs, glands or specialized cells.

The human body internal system and the capability of humans to sense, decide and act are a great illustration of self-adaptation mechanisms upon different situation acknowledged or not. The human body is a complex system of systems as different autonomic systems take care of most of the body functions without consciousness of their task orchestration (e.g. autonomic nervous system). Kephart et al. [77] matured a paradigm in analogy to human biology called Autonomic Computing to address complexity for complex computing systems that are heterogeneous, fast and dynamic, and increasingly scalable and interoperable. The main application was to make IT systems able to have autonomic properties and independently take care of their optimization and regular maintenance in order to reduce the system administrator's workload.

Designing autonomy is difficult in a single step, For this purpose, Kephart et al. [77] propose five degrees of autonomicity to represent the maturity of self-management in the IT systems as illustrated in Figure 2.1.

Level 1 (Basic) corresponds to manual analysis and problem solving from the system reports. Manual actions from skilled staff are required to configure, optimize, heal and protect the system.

Level 2 (Managed) proposes centralized tools such as a management software to facilitate the task automation on a specific resource. Staff still performs the analysis and takes appropriate actions. System awareness is more significant.

Level 3 (Predictive) involves the system in monitoring the environment, performing cross-resource correlation analysis and providing guidance like recommendation actions. Staff has to approve and initiate the recommended actions. Decision making is improved

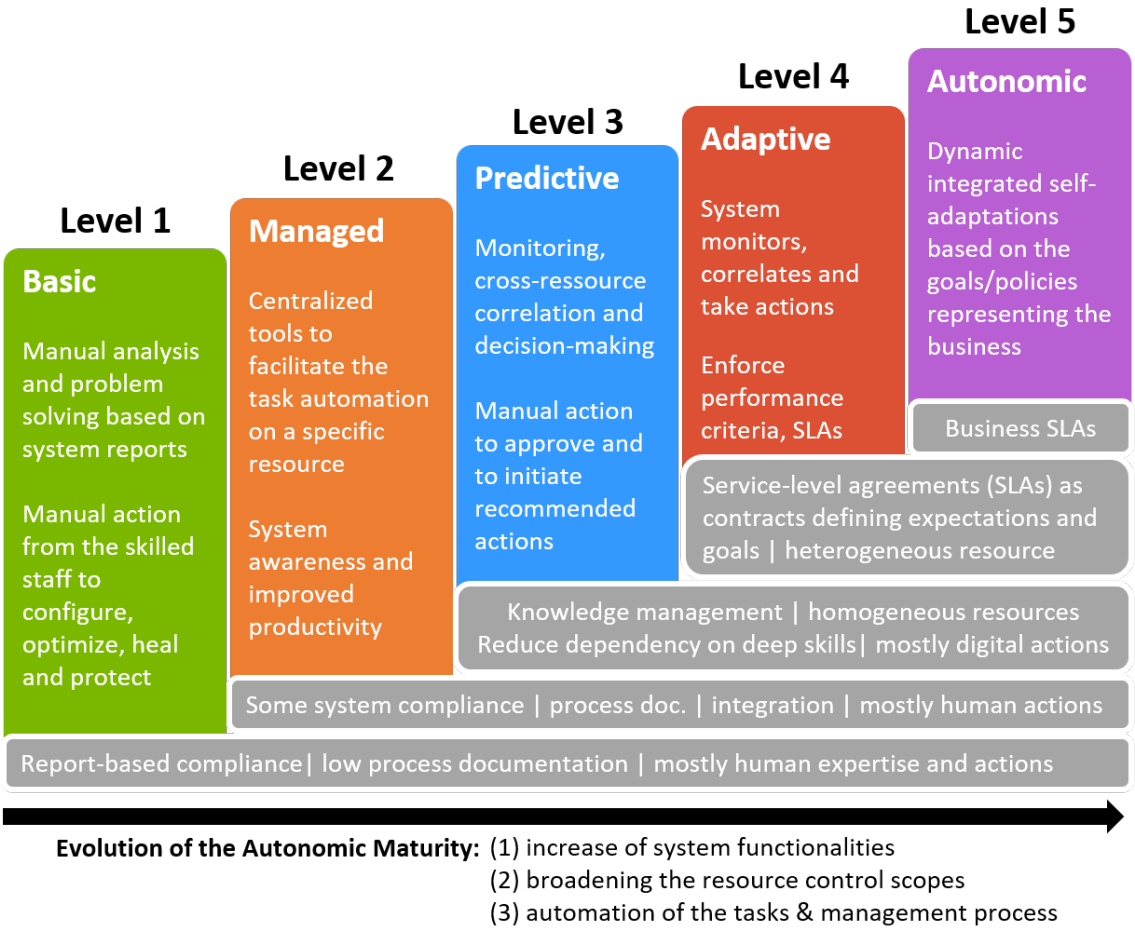


Figure 2.1 Maturity levels [77] for the autonomic computing paradigm

for homogeneous resources.

Level 4 (Adaptive) involves the system to monitor, correlates and takes action. At this level, the human interaction is minimized as only the performance criteria are directly managed by staff. System expectations and goals are defined by contracts (service-level agreements or SLAs). Heterogeneous resource are supported.

Level 5 (Autonomic) performs dynamic self-adaptations based on the goals and policies of the described business. No intervention of staff is required during the operation. Human interaction is mostly between the system and the end-customer as she/he orders new changes that require reconfiguration of the managed resource.

The evolution of autonomic maturity is reflected by: (1) the increase of system functionalities, (2) the broadening the resource control scopes (e.g. from a single subcomponent managing parts of resources to a whole system adaption), and (3) the automation of the tasks and the management process (e.g. from human-based process to its representation with machine-understandable tasks and workflow).

Huebscher et al. [70] provides a complementary classification scheme of five possible classes. They address what they consider as missing in the previous maturity levels, that is the role of autonomicity in the system and the extent of self-optimization achieved.

Support describes that the system possess self-management functions that only operate one particular aspect or component of an architecture. It results in an improvement of the complete system performance.

Core designates systems where the self-management functions is at the core of the system (e.g. end-to-end network management solution for QoS in video streaming). Other parts of the system are not necessarily self-managed. Goals and policies guiding the process are not high-level based.

Autonomous represents an end-to-end management solution to a problem involving intelligent and agent-based technologies. The system is expected to perform self-adaptation upon the context changes and when necessary react to failure. Within this perspective, this type does not measure its own performance to adapt and attain its goals optimally.

Autonomic represents a the Level 5 in autonomic maturity referred before. It concerns a full architecture that performs self-adaptation over its managed resources and itself for optimization guided by higher-level human-based goals (business goals or service-level agreements).

Closed-Loop corresponds to autonomic systems that have the possibility to learn, grow and refine themselves. These closed loop systems mostly refer to the use of diverse forms of reinforced learning (e.g. Markov decision process, partially observable Markov decision process, recurrent neural networks, etc.)

It can be inferred that the self-management as autonomy has a range of actions (i.e. resource-specific to system-wide) but that also requires a toolkit of functions that comes from several fields to be effective (e.g. basic human analysis to continuous learning for classification and decision).

In the field of AV, the SAE J3016 standard [41] distinguishes the preferred term "Automated" rather than the long-time used "Autonomous" term to designate vehicles with a delegation of driving. It designates autonomous vehicles as systems that have "the ability and authority to make decisions independently and self-sufficiently" while operating as stand-alone. This definition would exclude vehicles benefitting of communications technologies to leverage on shared information. Besides, some usage of "autonomous" was misleading in the literature as it was representing the entire system functionality. The SAE J3016 standard prefers the term "automated" to propose recommendations for all types of road systems that include all driving-related systems, architectures, functions, use and interactions.

This thesis employs the term "autonomous" over "automated" as we consider the current systems as fitted to the previous definitions of autonomy. That is, the examined systems can operate as stand-alone as it is the more widely used form for self-sufficient systems and in the fundamental operations of perception, decision and action.

The next section details the considerations for creating self-sufficient systems and understanding its composition.

Understanding the Autonomy in Systems

After having portrayed our understanding of autonomy, we need to consider requirements and constraints to implement such autonomic behaviours. Abeywickrama et al. have proposed a relevant decomposition of autonomy into four attributes, described as follows[5].

Reflexivity implies the system has access to the knowledge of its components that is the current status, capabilities, limits, boundaries and inter-dependencies with other systems and available resources. This implies that the system is self-aware, as it knows itself.

Evolvability (or Evolution) represents the ability of the system to evolve in dynamic situations where new services, new requirements and alternative configurations

can emerge. Thus, it can be self-organizing and self-optimizing to be relevant to its operational needs.

Model-based system engineering (MBSE) enables the system representation as a model, thus maintaining requirements, design, analysis, verification and validation activities throughout the system life-cycle. MBSE provides fundamental guidelines to design and develop complex systems having autonomy capabilities. Further, it provides different perspectives addressing specific concerns like for example system safety.

Uncertainty is usually mitigated by robust solutions at design-time in conventional approaches. However, they reveal to be impractical in highly dynamic environments. Real-time models have been adopted to address uncertainty achieving both reflexivity and evolvability.

These attributes identify a spectrum of cross-domain complexities suggested in the previous works [7, 41, 70, 77, 134]. They help understanding the complexities of designing and specifying autonomy in autonomous vehicles in terms of needs, requirements, processes and the complexity of the overall design. In addition, the application of autonomy in vehicles requires the system to delegate partially or completely the driving tasks, therefore design approaches need to integrate safety, reliability and security dimensions.

Architecting for Autonomy

We have seen that autonomy cannot be implemented from only one or two perspectives as it required to tackle a variety of concerns to appropriately take several variables, actions, requirements, and limitations into account. Therefore we examine different approaches and structure that contribute to capture autonomy in robotic of CPS.

The autonomy architecture presented in [7] suggests that different levels of decision are required while remaining reactive to possible events. These contribute to consider the several variables that drive the decision process in a robot. The first decision level involves mission planning (i.e. refine the mission into tasks), task refinement (i.e. obtain trajectories from expected tasks), and coordination (i.e. merging requests into a plan). The second level includes execution control level, it requests and supervises the plans to operate. The third level is functional, it allows reactive adaptations and follows the selected plan through motion planning, avoidance, world representation, sensor perception and motion execution. The fourth level is Logical, it reflects the interface of the system and mobility in the external world. The Fifth level is Physical, it represents the system hardware that is different sensors and effectors integrated into the robot platform. An architecture involving the different levels is necessary to take appropriate decisions and satisfy the set of distinct variables relevant to the mission.

Durrant-Whyte [48] identifies similar functional elements for autonomous land vehicles: mobility, localisation, navigation, planning and communications. Mobility refers to the design and motion control of mobile vehicles and their kinematic models. Localisation determines the position and attitude of a vehicle in a coordinate system. Navigation provides "the guidance and the control of a vehicle in response to sensor information" from the vehicle surrounding's and vehicle's state. Planning refers to the construction of trajectories up to some time horizon, beyond sensor range and compute possible actions of other vehicles. Communication reflects the ability to request a specific mission or receive reports of the system (e.g. teleoperation requires communication between vehicle and base station). These functions are the basis of current functional AVs architectures as shown in Figure 2.2 on page 41. Duant-Whyte makes emphasis on establishing that effective system engineering is necessary for the deployment of AVs.

Kephart et al. [77] in the Autonomic Computing paradigm refined the scope of the self-management capabilities necessary for systems to demonstrate autonomy, these are as follows: ***Self-Configuration*,** to dynamically adapt to changing environments that includes changes in the system characteristics, the deployment of new components or the removal of existing ones. ***Self-Optimization*,** to tune resources and balance workloads to maximize the use of information technology resources. ***Self-Healing*,** to discover, diagnose and act to prevent disruptions from malfunctions making the system more likely to fail safe. ***Self-Protection*,** to anticipate, detect, identify and protect against threats and hostile behaviours.

These four autonomic properties propose an autonomic response based observations from the sensors, and actions sent to the actuators, and cores that drives each control loop. The authors propose a reference model for the autonomic loops called MAPE-K. Each loop operates the functions of Monitor, Analyse, Plan, and Execute around sources of represented Knowledge. This functional decomposition contributes to identifying the architectural aspects of autonomic systems and separate the concerns of the control and decision process.

Johansson et al. [73] capitalize on the experience acquired during the FUSE project (FUnctional Safety and Evolvable architectures for autonomy) regarding autonomy, safety and system engineering. From a high level point of view, they detail the FUSE reference architecture as "(1) a categorization and description of the key functional components needed for autonomous driving, (2) rationale for these functional components across the architecture, and (3) a three-layer architecture incorporating the described components". The whole makes an echo of the four attributes presented in the Section 2.2. (1) addresses the reflexivity and evolvability attributes, (2) and (3) contribute to build up the model of the architecture and integrate the remaining attributes.

The same approach included in [73] by separating the perception, the decision and control, and the vehicle platform control as three different categories of the key functional components. They suggest that this partition favors high repleacibility of the different functions and components and deployment across different vehicle platforms.

J. Sifakis [134] emphasizes that adaptivity is the appropriate technical answer to manage uncertainty in critical systems. He proposes a paradigm shift stating that conventional hazard analysis in AV have shown limitations to the extent that it became difficult to "foresee at design time all the possible hazards in the system's lifetime due to their poor predictability". He introduces the concept of adaptive control for system resilience and to compensate the lack of human intervention through the use of a generic component named "adaptive controller" based on control theory [16]. It implements the hierarchical control of a property through objective management, planning, and learning functions. The approach proposed in [134] integrates the adaptive controllers in a knowledge-centric architecture. It aims to combine both design-time knowledge (i.e. declarative knowledge about the properties of the designed system) and run-time knowledge (i.e. violation of some property or knowledge based on system run-time monitoring). This approach seeks a compromise between a rigorous design and essential properties that cannot be guaranteed at design time.

Schlatow et al. investigate in [130] the ability to operate in degraded performances and degraded conditions. They suggest that the presence of self-awareness in the several system layers can build truly autonomous systems for AV. The concept of self-awareness refers to the system's capability to know and represent itself by its states, its actions, the consequence of its actions, and its limitations. It has been formalized in the Autonomic Computing paradigm [77], and appears to be close to the reflexivity attribute and the

objective of the learning function previously referenced. It refines the self-awareness concept to a "consistent self-representation of the system" based on the overall monitoring of the layers through metrics. The authors argue that complexity gets out of hands on individual layers if each observed event are locally handled. However, this can be avoided by handling the counteract adaptations through cross-layer solutions. The complexity of the possible observed events to handle is linked to the vastness of the targeted environment and the system.

Challenges of Autonomy

Achieving autonomy in systems consists in pursing and extending these presented aspects:

- *The cognitive complexity*: The mission can become more complex as more detailed, involving more objectives and properties to assert. The heterogeneous goals and policies have to be included in the decision-making to achieve a desired target defined by behaviours, characteristics or constraints as requirements.

- *The operational functions*: The functions need to be relevant in their operating context satisfying their requirements and allocated to components in the architecture. Major complexity comes from the management of the different types of components and architectures, how they are allocated and connected [48, 134, 135].

- *A systematic approach for large distributed evolvable systems*: A rigorous design systematically integrates the necessary functions supported by design tools and development methodologies for an extensible and evolvable architecture.

- *The correctness ensured through adaptation during run-time and over time*: Evolvability of the system can be achieved through adaptations in the system or of the system itself. Self-adaptations contribute to cope with uncertainty but reduce the ability to perform conventional verification. It corresponds to the ability to adjust behaviour through learning and reasoning and to change dynamically the goal management and planning processes [135].

- *The self-awareness in non-predictable dynamically changing environments*: Self-representation of the system (reflexivity) ensures sufficient observability and controllability. Uncertainty needs to be taken into account, especially in the representation of the external world.

The last aspect would be the scientific, technological and societal stakes of the autonomy challenges. Indeed, autonomous systems are mostly designed around human interactions and user experience. Their adoption requires trust through explainability and interpretability. For this purpose, run-time models and knowledge-based systems usually provide the possibility to be transparent to inspection.

Applications of autonomy in cyber-physical systems in certain domains show different context and limitations to consider than others. The next section focuses explicitly on Autonomous Vehicles and details how autonomy have been designed and experienced over the years.

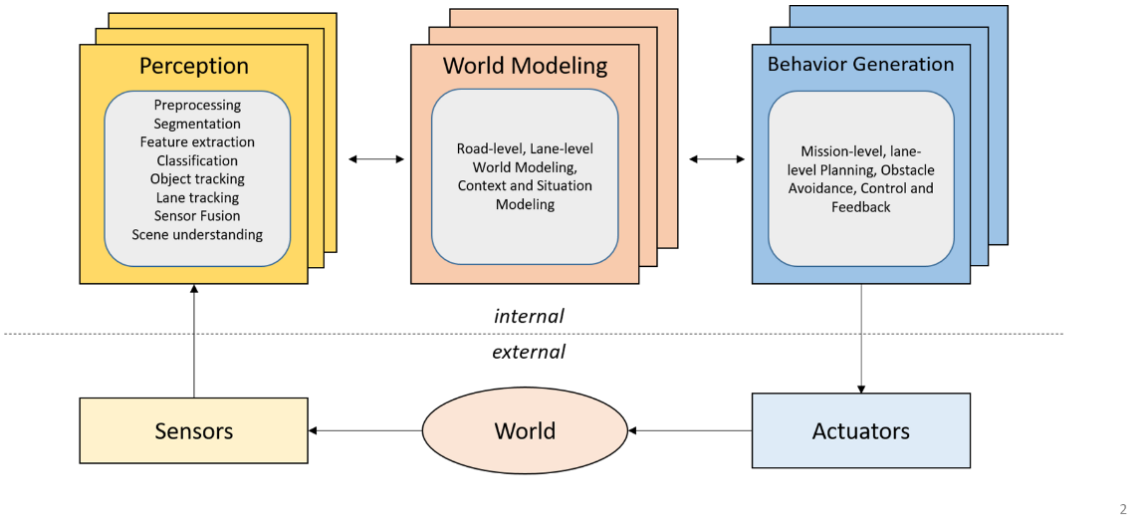


Figure 2.2 Top-level functions for an autonomous navigation system based on the 4D/CS architecture (Courtesy of Renault Research). [10]

Autonomous Vehicles and Autonomy

Autonomy design and driving automation for autonomous vehicles is presented in this section. Two perspectives are considered: the operational description of the AV and their functional architecture.

Operational Description of an AV

The generic functions that allow an AV to operate autonomously are presented as well as the convergence between the Autonomic Computing notions referred previously and vehicle SAE automation levels.

Functions for autonomy Multiple applications for vehicles having autonomy characteristics exist, nevertheless, the essential function are: Perception, World Modelling and Behavior Generation [9, 10].

These three functions represent the manner how an autonomous system can interact with its environment and achieve a purpose. as shown in Figure 2.2. They encompass different of sub-functions, they provide the vehicle with the four autonomy attributes introduced in Section 2.2.

Perception. A digital representation of the vehicle's immediate environment is the result of this function. This is based on the perceived information from the multiple passive (cameras) and active (RADAR & LiDAR) sensors mounted on the vehicles and at times information from the infrastructure.

Perception is a complex function, despite the rapid progress due to dominant machine learning methods like Deep Learning, it remains a challenge. The perception function can be considered as made of several sub-functions: Acquisition & preprocessing, physical attributed from the observed environment are acquired by sensors, these signals are processes to provide data which can be processed by a computer. Segmentation, prepares clusters of observed elements having similar characteristics (e.g. roads, kerbs). Classification, identifies the clusters into different types like pedestrians, cars, road surfaces, etc.

Registration and Object Tracking, the spatial description of the classified objects with respect to the vehicle is inferred, relevant objects like pedestrians or cars are tracked as the move. Multiple sensors are needed to build the digital representation of the environment due to limits on current available sensors.

The sensor fusion techniques make full use of the advantages of each sensor coping with the weaknesses of the other sensors. For example, LIDAR is excellent to provide 3D measurements and is not that affected by scene lighting, however, it suffer from poor resolution compared to camera images. Cameras by contrast provide rich details on the object of the scene, however, they only estimate distance to the object.

Scene understanding corresponds to the contextual representation of the environment where all entities are located, described (e.g. speed), and categorized by their semantic meaning (e.g. labelled as a pedestrian).

As part of the perception function, we include the localisation component, as in order to navigate it is necessary to know the whereabouts of the vehicle. Localisation estimates describe the position of the vehicle with respect to a reference frame. In general GNSS receivers are used like GPS these are augmented though odometric information, inertial sensor, etc.

World Modelling. This is build by enriching the digital representation of the perceived environment projected on an a priori map known as HD-map. The later contains stored information on the road structure at the lane level as well as attributes that enable reasoning to infer the vehicle behaviour.

Context modelling represents a digital model of the world from the perception system and the HD-map. This machine-understandable model of the world reflects the scene of the road elements and the vehicle's current state (e.g. position, speed, acceleration, yaw rate, functional capabilities and degradations, etc.).

Within the World Model it is possible to determine the entities of concern (e.g. pedestrians, vehicles), to predict their motion or intentions. For the vehicle it is possible then to understand its situation and reason what to do next taking into account its mission mandate.

Situation understanding consists in a machine-understandable model of the world that reflects the scene of the road elements.

Behaviour generation. Decision-making and control systems are responsible for generating and executing vehicle behaviour. Different functions are involved in this process.

Mission-level planning considers the path the vehicle should follow at road-level to achieve the strategic tasks as carrying out its high-level objectives.

Lane-level planning, or behaviour planning, intervenes at short range and time to cope with the interaction with other road users and infrastructure, and follow the local driving rules. It determines the path the vehicle should follow within the lanes and the vicinity of the vehicle while pursuing its local objectives (e.g. lane changes, intersection handling, lane keeping, distance keeping).

Obstacle avoidance takes place at the immediate operational range and time to mitigate any potential collision with an obstacle. Avoidance requiring specific manoeuvres also involves the behavioural level of decision making.

Finally, control and feedback are responsible for the motion control that converts the intentions and trajectory defined in the previous level of planning into actions. The control

system provides the necessary inputs to the actuators and generates the expected motion in the external world. Additional measurements contribute to stabilizing the dynamics of the system to the desired state, managing potential disturbances via feedback control.

Through all the observed function safety is a major concern. Perception is not perfect, any error will propagate and result in hazardous situations. The world representation function will suffer of localisation and map errors too in addition to perception issues, further predicting what other entities is very difficult, thus the situation understanding and decisions might also result in hazardous situations. The increased interest on machine learning has resulted on solutions addressing beyond the perception function, that is decision making and navigation. From a safety perspective, this is adding complexity, as machine learning methods are difficult to understand and their results could be erratic when faced by edge cases.

AV minimum behavioural competencies The functions in the three levels necessitate goals, policies and requirements to be finely tuned to appropriately monitor react to the vehicle's environment. From a learning perspective, it needs to differentiate what are a good driver and a bad driver in order to specify or nurture the appropriate behaviours from data. Different works have successively identified these expected behaviour and autonomous capabilities for AV.

Historically, the series of DARPA Grand Challenges (2004, 2005, 2007) encouraged research and development for autonomous vehicles. While the two initiatives require to drive within a desert, the last was more challenging as the vehicle has to drive in an urban environment. This 2007 DARPA Urban Challenge boosted development of unmanned vehicles for urban areas have been the source for numerous technologies, tools, and engineering techniques, both for autonomous vehicles, but also for ADAS. Later, California PATH project [115] published in 2016 a concrete taxonomy of the minimum behavioural competencies required to an AV. These high-level requirements contribute to understanding the decomposition of what and how the vehicle need to perform across a variety of driving environments (e.g. freeway, rural, highway, city streets, valet parking, low-speed shuttles).

Understanding the intricacies, of autonomous operations can be gained through a taxonomy of the behavioural competencies that an AV should have. The PATH programme a UC-Berkeley proposed such taxonomy [115]. This provides an understanding of what and how the vehicle needs, to operate different operational design domains (ODDs). Waymo [153] have proposed an extended list of minimum capabilities based on their progress and field experience of a variety of reasonably foreseeable traffic situations with AV. Table 2.1 lists these behavioural competencies.

Although relevant to understanding the functioning and complexity of vehicle behaviour, the verification and validation (V&V) of each of these competencies through the conventional V-Model methodology seems really inapplicable due to the diversity of events, physical quantities and geometry of the environment [134]. However, they have been the base for generation of driving scenario and source for the refinement [142] of the list of tasks, monitoring and executing actions (OEDR) to specific ODDs.

Defining other behavioural requirements for AV To understand how the driving automation is performed, SAE J3016 [41] investigates the driving tasks performed by the driver during driving and that control the vehicle in the form of different responses and actions. The report defines that AV performs driving automation as a result of sustained

N	Behavioural Competencies for AV
1	Detect and Respond to Speed Limit Changes and Speed Advisories
2	Perform High-Speed Merge (e.g., Freeway)
3	Perform Low-Speed Merge
4	Move Out of the Travel Lane and Park (e.g., to the Shoulder for Minimal Risk)
5	Detect and Respond to Encroaching Oncoming Vehicles
6	Detect Passing and No Passing Zones and Perform Passing Maneuvers
7	Perform Car Following (Including Stop and Go)
8	Detect and Respond to Stopped Vehicles
9	Detect and Respond to Lane Changes
10	Detect and Respond to Static Obstacles in the Path of the Vehicle
11	Detect Traffic Signals and Stop/Yield Signs
12	Respond to Traffic Signals and Stop/Yield Signs
13	Navigate Intersections and Perform Turns
14	Navigate Roundabouts
15	Navigate a Parking Lot and Locate Spaces
16	Detect and Respond to Access Restrictions (One-Way, No Turn, Ramps, etc.)
17	Detect and Respond to Work Zones and People Directing Traffic in Unplanned or Planned Events
18	Make Appropriate Right-of-Way Decisions
19	Follow Local and State Driving Laws
20	Follow Police/First Responder Controlling Traffic (Overriding or Acting as Traffic Control Device)
21	Follow Construction Zone Workers Controlling Traffic Patterns (Slow/Stop Sign Holders)
22	Respond to Citizens Directing Traffic After a Crash
23	Detect and Respond to Temporary Traffic Control Devices
24	Detect and Respond to Emergency Vehicles
25	Yield for Law Enforcement, EMT, Fire, and Other Emergency Vehicles at Intersections, Junctions, and Other Traffic Controlled Situations
26	Yield to Pedestrians and Bicyclists at Intersections and Crosswalks
27	Provide Safe Distance From Vehicles, Pedestrians, Bicyclists on Side of the Road
28	Detect/Respond to Detours and/or Other Temporary Changes in Traffic Patterns
29	Moving to a Minimum Risk Condition When Exiting the Travel Lane is Not Possible
30	Perform Lane Changes
31	Detect and Respond to Lead Vehicle
32	Detect and Respond to a Merging Vehicle
33	Detect and Respond to Pedestrians in Road (Not Walking Through Intersection or Crosswalk)
34	Provide Safe Distance from Bicyclists Traveling on Road (With or Without Bike Lane)
35	Detect and Respond to Animals
36	Detect and Respond to Motorcyclists
37	Detect and Respond to School Buses
38	Navigate Around Unexpected Road Closures (e.g. Lane, Intersection, etc.)
39	Navigate Railroad Crossings
40	Make Appropriate Reversing Maneuvers
41	Detect and Respond to Vehicle Control Loss (e.g. reduced road friction)
42	Detect and Respond to Conditions Involving Vehicle, System, or Component-Level Failures or Faults (e.g. power failure, sensing failure, sensing obstruction, computing failure, fault handling or response)
43	Detect and Respond to Unanticipated Weather or Lighting Conditions Outside of Vehicle's Capability (e.g. rainstorm)
44	Detect and Respond to Unanticipated Lighting Conditions (e.g. power outages)
45	Detect and Respond to Non-Collision Safety Situations (e.g. vehicle doors ajar)
46	Detect and Respond to Faded or Missing Roadway Markings or Signage
47	Detect and Respond to Vehicles Parking in the Roadway

Table 2.1 Behavioural competencies for autonomous vehicles in [115] (1-28) extended in [153]

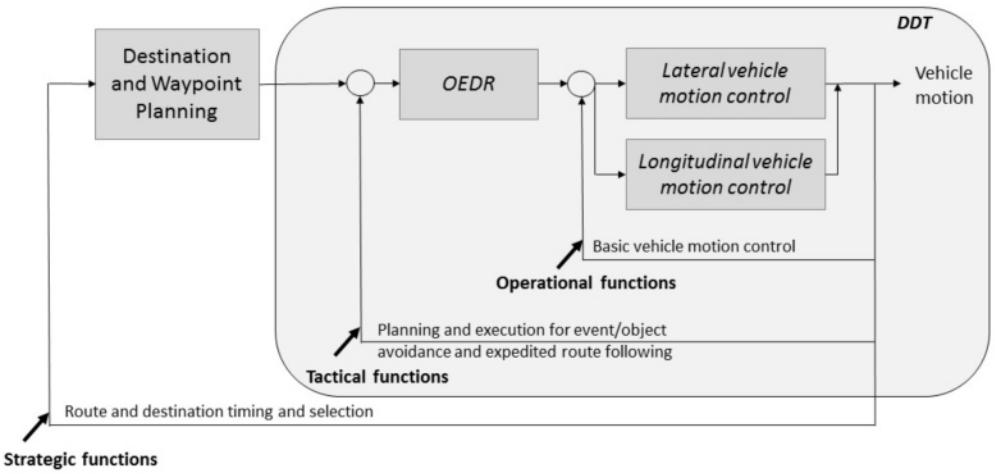


Figure 2.3 Schematic view of the Dynamic Driving Tasks according SAE [41]

Dynamic Driving Tasks (DDT) in a specific environment. Those DDT correspond to the real-time tasks required to operate a vehicle in on-road traffic: (1) Lateral vehicle motion control via steering (y-axis, operational); (2) Longitudinal vehicle motion control via acceleration and deceleration (x-axis, operational); (3) Monitoring the driving environment via object and event detection, recognition, classification, and response preparation (operational and tactical); (4) Object and event response execution (operational and tactical); (5) Maneuver planning (tactical); and (6) Enhancing conspicuity via lighting, signaling and gesturing, etc. (tactical).

The Object and Event Detection and Response (OEDR) corresponds to the tasks (3) and (4). It consists of the monitoring of the driving environment (detection, classification, tracking), preparing to respond to them, and executing an appropriate response. An OEDR is only relevant in the ODD the system has been conceived for.

Figure 2.3 indicates the allocation of the dynamic tasks to the strategic, tactical and operational levels. It illustrates the operation of each task, the frequency rate increases the closer the tasks are to the vehicle motion.

Figure 2.4 allocates the identified DDTs to the top-level functional architecture in order to map the various functions into the DDT tasks.

Other concerns include the handling of a system failure and/or out-of-operational design domain (out-of-ODD) condition. For level 3, 4, and 5 ADS, the SAE J3016 framework distinguishes three concerns as the separate functions of (i) DDT performance, (ii) DDT fallback performance, and (iii) Minimal Risk Condition achievement (MRC). Their interruption, occurrence or completion is performed based on the events the system may detect and follow a set of optional conditions [41]. The next paragraphs summarize and illustrate each of the three functions with the ADS engaged. A summary of the three functions with the ADS engaged is described next:

DDT performance corresponds to the proper operation of the DDT tasks by the system. Two issues can occur: a DDT performance-relevant system failure or an out-of-operational design domain (out-of-ODD) condition. *Minimal Risk Condition (MRC)* is a condition to which a user or an ADS may bring a vehicle in order to reduce the risk of a collision when a given trip cannot or should not be completed.

DDT fallback performance corresponds to the response by the ADS to perform the DDT or achieve the MRC alone after an approaching ODD exit or a DDT performance-

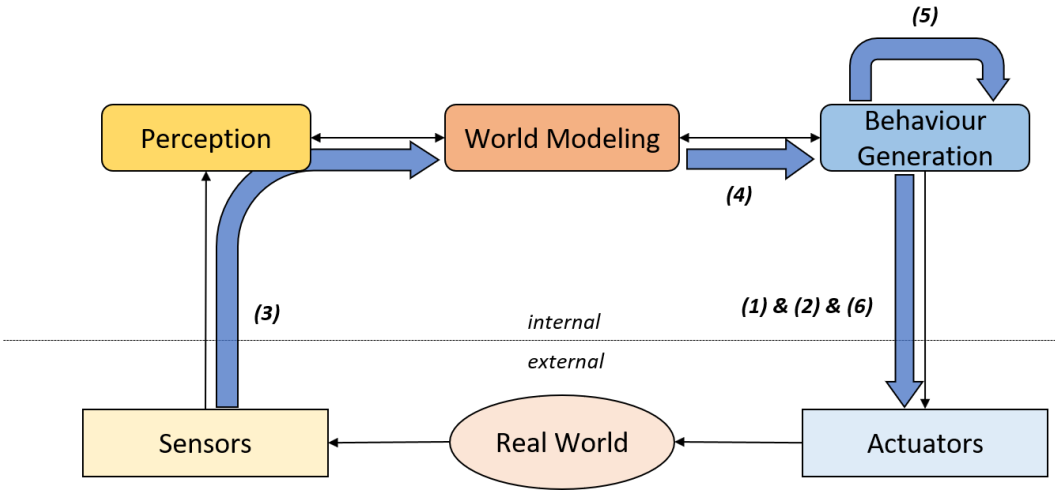


Figure 2.4 Mapping of the DDT to the functional architecture.

relevant system failure are detected.

Whilst this research centres on Level 4 vehicles, it is important to understand the allocation of responsibilities between the vehicle intelligence and drivers. These are mapped with respect to autonomic principles. The top half of Table 2.2 shows the responsibilities between the system and the driver with respect to the DDT tasks and DDT fallback. The bottom half of Table 2.2 provides an identification between the levels of automation and the AC maturity levels [77] and classes [70] presented earlier in Section 2.2. Applied to AV, the maturity levels would consider the staff as the driver, and the self-management functions as two different systems managing the driving tasks and fallback tasks. Next, the SAE automation levels are identified with Autonomic computing, in particularly regarding levels 3 and 4.

The SAE Level 3 includes systems which control the fundamental driving tasks. The system provides self-adaptations to the environmental context and can react to failures with the assistance of the driver (e.g. DDT performance-relevant system failure). Due to the variety of possible combinations of functionalities, range and coverage, these systems are identified with the Adaptive or Autonomic levels and the Core or Autonomous classes but not as performing all goals optimally. For the fallback, maturity levels are Predictive and Adaptive as the system operates and only indicates when it can not. The system can not guarantee the achievement of MRC in all cases within its ODD. The classes are Core or Autonomous because the system still involves humans in its reaction to failures (i.e. Fallback-ready user).

The SAE Level 4 involves no driver for DDT operation. This refers to the Autonomic maturity level. However, the classes are not simple to identify as they can be described as Autonomous, Autonomic or Closed-Loop according to the scope of the system functionalities. A level 4 ADS may either carry out the driving tasks without measuring its own performance to fit its performative goals optimally (Autonomous), or monitor them and optimize them guided by higher-level human-based goals (Autonomic), or optimize its own performance and learn how to improve them by itself (Closed Loop). The fallback for level 4 ADS also needs to operate it without the help of the driver. There are different cases: a system failure or degraded performance which make the system unable to carry out its mission. They both result in questioning the initial mission of the system to see if it is still achievable, if not, navigate to MRC in some cases. The fallback of level 4 ADS

can be described as Autonomic and associated to the Autonomous or Autonomic classes (as not all performance goals may be met optimally with safety first MRM).

The mapping between the SAE levels and the Autonomic Paradigm levels and classes contributes to illustrating the difference in the scope of action of the two functions. The performance of the DDT and DDT fallback in the level 4 and 5 involves a high degree of autonomy where both world and system representations, adaptations and evolution are required.

Autonomous vehicles can be evaluated by their capabilities: we have presented the minimum behavioural competencies from California PATH project [115] and the set of levels, terms and functions of SAE J3016 [41] (i.e. OEDR with ODD, DDT and DDT fallback functions). Two other parts remain relevant to be presented in the operational description of the AV: the system degradation in decision making and the representation of the situational context.

Reschka et al. [20, 109, 121] have proposed a system perspective and an architecture to represent the driving abilities and capture their degradations during the system operation (i.e. during sytem run-time) with skills as altered forms of the initial driving abilities. This representation of degradation (e.g. the vehicle's ability to turn is not as expected) and inducted system restrictions (e.g. turn left becomes impossible) offers enough reflexivity in decision-making to possibly perform safe DDT and DDT fallback. Such characteristics may be included in the OEDR and ODD as adaptation mechanisms and context limitation.

We have seen that designing AV makes possible to come up with a large variety of usage scenarios and specifications within the OEDR and ODD. Different mechanisms are explored to address this variety.

Initially Use Cases (UC) are defined with the purpose of extracting usage scenarios with respect to the Autonomous vehicle expected operation. It consists of a list of high-level interactions the vehicle may perform. Each use case specifies its range of action and the operational situations the vehicle [146]. Since the situations faced by the vehicles can result in a combinatory explosion, the use case formulation provides an abstraction and context. Research literature [151][64][146], vehicle OEMs reports [55, 62, 133, 153] and open projects [103, 143] provide examples and in-depth studies of AV use cases. This can be followed by a representation of decision-making as a set of rules, constraints, policies, goals or mission. These can be abstracted from different capability perspectives, e.g. task, manoeuvre or use case oriented. Others can address specific edge cases and safety concerns.

Conclusion This section presented a taxonomy to highlight the different system representations, adaptation and evolution necessary to grasp the operational context of AV. However, given the complexities of the ODDs when operating in public roads, the hetereogenity of entities sharing the same road network, together with the diverse number of potential automated vehicles means that there is no standard solution [28]. Futher, the complexity of such systems is highlighted and hence the potential for failure. Ensuring the safety is complex and only recently the notion of safety mechanisms is emerging.

Functional Architecture of AV

A functional architecture in the automotive domain is a hardware and implementation independent decomposition of the functionalities the system provides into components and sub-components and the information flows between them. Different views of the same functional architecture may exist to describe each component depending into what

SAE levels of driving automation [40]	Level 0 - No Driving Automation	Level 1 - Driver Assistance	Level 2 - Partial Driving Automation	Level 3 - Conditional Driving Automation	Level 4 - High Driving Automation	Level 5 - Full Driving Automation
Responsibility for performing the Dynamic Driving Task (DDT)	Driver	Driver and System	System	System	System	System
Responsibility for performing the DDT fallback	Driver	Driver	Driver	System if no failure, else Driver (Fallback-ready user)	System	System
Recommended naming	-	Level 1/2 driving automation system-engaged vehicle		Level 3/4/5 Automated Driving System-operated vehicle		
System maturity levels for the DDT	Basic None	Managed or Predictive Core	Managed or Predictive Core or Autonomous	Predictive or Adaptive Core or Autonomous	Autonomic Autonomous or Autonomic or Closed Loop	Autonomic Autonomic or Closed Loop
System maturity levels for the DDT fallback	Basic None	Basic & Managed Support	Basic & Managed Support	Predictive & Adaptive Core or Autonomous	Autonomic Autonomic	Autonomic Autonomic or Closed Loop

Table 2.2 Responsibility in SAE levels of automation while system engaged and identification to Autonomic Computing maturity levels and classes

it is to be addressed. A functional architecture is the key to capture the difficult task of driving autonomously and to present coherently the proposed system [24].

Different AVs architectures have been deployed, the following paragraphs provide a summary view of those regarded as the most representative. Behere et al. [23] surveys architectures from the following fields of study: Intelligent control [12, 104][9, 10], cognitive architectures [88][149] and real-time control architectures [31, 33, 61, 136].

The case study used in this thesis is from the commuter vehicle ADCC from Renault (see Figure 1.2, page 19). It applies the intelligent control approach fomulated by J. Albus [9, 10]. This reference model architecture for intelligent systems design is known as the 4-Dimension Real-time Control System (4D/RCS). It has been used extensively in multiple civilian and defence related vehicles. The ADCC architecture is designed for a vehicle having level 4 capabilities when entering a particular mode.

The robotics and artificial intelligence domains have traditionally driven the development of autonomous vehicles. Providing autonomy, to a driving task is a complex endeavour that "critical look at system architectures and the architecting process since system-level property goes beyond being 'just another requirement' " [76]. For this purpose, a functional architecture displays the decomposition of the essential functions for autonomy as components and their exchanges to enable autonomous driving. This view remains agnostic to specific technology to use and the knowledge to acquire. They also help to judge their potential to scale up towards the broad AV context.

The implementation of a sole functional architecture is insufficient for the automotive domain to provide guarantees of safe driving. Autonomous vehicles are safety-critical systems. Any malfunction or wrongly adopted behaviour may result in potential harm or damage. Integration of these concerns is not a straightforward process. The safety challenges for AV and how system architecture can address safety are examined in the next section.

Operational Challenges: Safety

Safety refers to the ability to deliver services that can justifiably be unharmful and trusted. Some possible harm may be unplanned and unintended but not necessarily unexpected. Safety engineering aims for the AV to avoid the more frequent and more severe expected causes such as failures to protect valuable assets from accidental harm with evidence. Safety engineering proposes methodologies, best practices and tools to achieve an acceptable level of evidence to describe the product as safe. For example, safety analysis are performed to capture and address the internal and external disturbances of the system and their ouputs are used to mitigate the associated risks.

Some operational safety challenges are closely linked to the ones of autonomy. An autonomous vehicle needs to be resilient to challenges that may lead to hazardous situations. The sources of hazards and possible threats are numerous. Some operational safety challenges are closely linked to those of autonomy, affecting different parts of the system because safety is a transversal non-functional quality. An examination of architectures and interpretation of what safe autonomy signifies, resulted in the following safety challenges:

Resilience and robustness of the different operational functions Systems are becoming more complex as operations are expanded and multi-sensors are used. Within this context, uncertainty propagates throughout the system. It plays an important role in

the resilience (stability) or robustness (failure) of these functions, resulting in fluctuating system performance, observability and controllability due to the environment changes.

Understanding and handling the causes and consequences of the complex context The dynamicity of the context imposes real-time constraints on all safety related elements. In addition, the interactions and negotiations between road elements have different levels of comprehension (internal prior knowledge or acquired by learning), predictions, etc. Within this context, an AV would have its performance fluctuate relatively to its context in addition to possible system operational limitations and degradation [81].

Safety characterization as observable features and metrics This addresses a major concern, how to measure credibly the safety of an AV. For example, to know if the result of a fallback solution ensures the vehicle enters a safe state. Conventionally safety is addressed through risks analysis methodologies from dependability that bring structures and frames to iterate on while others are based on metrics and criteria such as time-to-collision and passenger comfort. There are few performance metrics, none as a standard. Data available is mainly on the number of kilometers driven autonomously without disengagement. However, this is only an indicator that has been required by legislation. Safety metrics is a major subject of research [21, 57].

Supervising safety Assessing safety at design with multiple analysis entries and system perspectives only offers a limited scope for safety assurance in AV. Supervising safety during run-time such as the master controller in the ADCC architecture offers ways to correct unsafe behaviours or actions ensured through adaptation. Currently, different components (i.e. functional operations and higher-level components) take these safety requirements and metrics as inputs to ensure safe behaviour and operation [144].

Recommended practices such as safety patterns AVs are a challenging target for software and safety engineering with the possible heterogeneity of components and exchange formats from external sources. Recommended practices in software design suggest the use of interoperable entities and patterns to structure the system (e.g. similar taxonomy for AV context, standardized message exchange). This idea of reuse and pattern from existing solutions as individual validated parts is growing in the field of AV safety [76]. It may ease the process of system safety engineering (analysis, verification, validation). The date of publication, interoperability and rate of adoption remain a limiting factor.

Formal approaches for safety assessment Safety engineering advocates for a rigorous top-to-bottom design that systematically integrates the necessary functions sustained by design tools and development methodologies for a consistent, verifiable and open-to-validation architecture [85]. By contrast, a robotic approach is from the bottom up. System design and software architecture in AV is a challenge, including a safety mechanism to architectures that are still under design, remains a major challenge. In order to facilitate the verification and validation of requirements and systems, machine states or other formal representations (e.g. Petri nets, Graphcet, Automata) are found in the literature. However, they impose to have a wide knowledge of the system at an appropriate level of abstraction which can limit their coherence and reasonable exhaustiveness against real environment.

Addressing all the complexities of autonomy may leads towards *'correctness-by-construction'* architecture [25] involving contract based design, modularization, and composability. Including safety , this is known as *'safe-by-design'* architecture [20] when safety have been purposely integrated as a cross-cutting non-functional quality from the early stage of the development along with the safety analysis, safety goals and requirements.

Despite extensive testing of Level 4 AVs and exploratory work on Level 5 vehicles, there are no long-term services without a safety driver [32]. Lessons learnt from these deployments, and testing procedures are twofold. The recent accidents involving levels 1-3 vehicles have shown technical and usability limitations. Perception is far from perfect and may cause unintended consequences (e.g. Tesla accident on the highway [66]). The monitoring of the system by the driver also shows usability and overconfidence concerns (e.g. driver wrongdoing on system request resulting in a crash). On the other hand, acquisition of experience and achieving shareable and reusable knowledge bases are currently mostly powered by open-road testing programs for both human and ADS. Level 4 vehicle testing has been publicly observed by the actors in the AVs field (e.g. Waymo, Baidu, General Motors, Toyota, Renault-Nissan, Aurora, etc.). Currently, gaining maturity and experience in the field contribute to continuously discover, verify and validate the intended functionalities of the ADS.

To our understanding, these actors are either far from reaching a safe-by-design architecture or short of the safety assurance to move beyond the experimental phase.

2.3 Safety in Autonomous Vehicles

Safety-critical systems (SCS) are systems whose malfunction can result in the harm or loss of human lives or damage to the environment. Our daily's life is surrounded by this type of systems such as airbag systems or ADAS in cars, trains, elevators or even medical equipment. Not only we expect them to provide a correct functional service (e.g. availability, capacity, performance, interoperability with others systems) but also to be acceptably safe and secure. Therefore we can trust them when used. However, acceptably safe or being safe enough cannot be measured that easily for a complex system. Besides, there is no absolute safety because a system cannot be entirely risk-free considering some residual risk may remain. It can be only achieved by reducing the risk of endangering humans or causing damage to the environment to minimum levels.

Safety engineering proposes methods, techniques and tools to reduce and manage those malfunction with accuracy to ensure a minimum level of risks such as failure analysis and fault-tolerance.

This section defines safety and its immediate environment from the literature to understand how the quality attribute is grasped and managed in systems. Then, safety approaches in the automotive industry are detailed for AV but currently face some limitations with the highest levels of autonomy. Finally, we describe novel approaches purposely designed for safety to tackle the specific safety operational challenges of AV.

Safety Definition in Systems

Safety is the absence of unreasonable risk. In other words, it refers to the absence of harm from the system to the user(s) and the environment [17] such as driver, passengers, road users and others road entities for road vehicles [1]. This definition seems simple,

but this quality involves a range of different concerns to address in more complex systems (e.g. cyber-physical). Evaluating safety consists of the observation and traceable evidence of the various aspects of the system as a whole, including hardware, software, humans, environment behaviour. For example, engineering safety in ADS-operated vehicles consists in examining the accidental harm the vehicle can cause to the environment and its source (e.g. through malfunction as internal failures, fault, error, capability limitation), or the external context that can lead the vehicle to possible harm (e.g. obstacles, adversarial weather, unexpected events or behaviours). A system defined as safe proposes an assertable and acceptable degree to which accidental harm is prevented, reduced, and reacted adequately to [52].

Safety is usually referred to as a system property or quality attributes [17][52] within the literature. While its definition is widely accepted, it is necessary to distinguish it from others such as availability (i.e. readiness for correct service), reliability (i.e. continuity of correct service), integrity (i.e. absence of improper system alterations), and maintainability (i.e. ability to undergo modifications, and repairs) [17]. They are composed into the integrating concept of dependability as the ability to deliver service that can justifiably be trusted [17]. A criterion for a dependable system is its ability to avoid service failures that are more frequent and more severe than is acceptable. Additional attributes of confidentiality (i.e. absence of unauthorized disclosure of information) and security (composite of confidentiality, integrity and availability) are not related to safety but sometimes are misinterpreted or may overlap. Firesmith [52] distinguishes safety from security (i.e. the degree to which malicious harm is prevented, detected, and reacted to) and survivability (i.e. the degree to which both accidental and malicious harm to essential services is prevented, detected, and reacted to).

Thus, engineering safety in a system requires only to specify, develop and deploy mechanisms to observe, predict and mitigate accidental harm in the whole life-cycle of the system. However, the scope of safety implications is vast and may affect components all across the entire system. It usually implies the creation of a variety of new process, sub-systems, components and algorithms to handle the prevention, reduction and reaction. It results in the generation of cross-cutting safety requirements all across the system. Firesmith [53] indicates that safety requirements may be confused with reliability ones (i.e. advocating the continuity of service) and come from a variety of shareholders highlighting the cross-cutting scope that safety can have. For this purpose, the author decomposes the safety as four distinct safety-related requirements for a system that may have significant safety implications as follows: (1) Safety requirements, (2) Safety-significant requirements, (3) Safety constraints, and (4) Safety system requirements.

The first type of safety requirements refers to any requirements that specify mandatory amounts of a subfactor of the safety quality factor (i.e. specifications to protect assets from accidental harm, detect safety incidents, and respond to safety incidents). Contrary to functional requirements that specify what the system shall so, they define what the system shall not do or prevent from happening.

The second type is the safety-significant requirements. It relates to any non-safety primary mission requirement that can all negatively impact the safety of a system as it can cause hazards and safety incidents (e.g. an accident or near-miss). They are the results of the safety analysis of the primary mission specifications of the system (i.e. functional requirements, data requirements, interface requirements, and non-safety quality requirements such as performance, robustness or reliability). These safety analyses regroup the analysis of the system (i.e. asset, hazard, and safety risk analysis), and their combination and exploitation (i.e. categorizes accident/hazard severities, accident/hazard likelihoods,

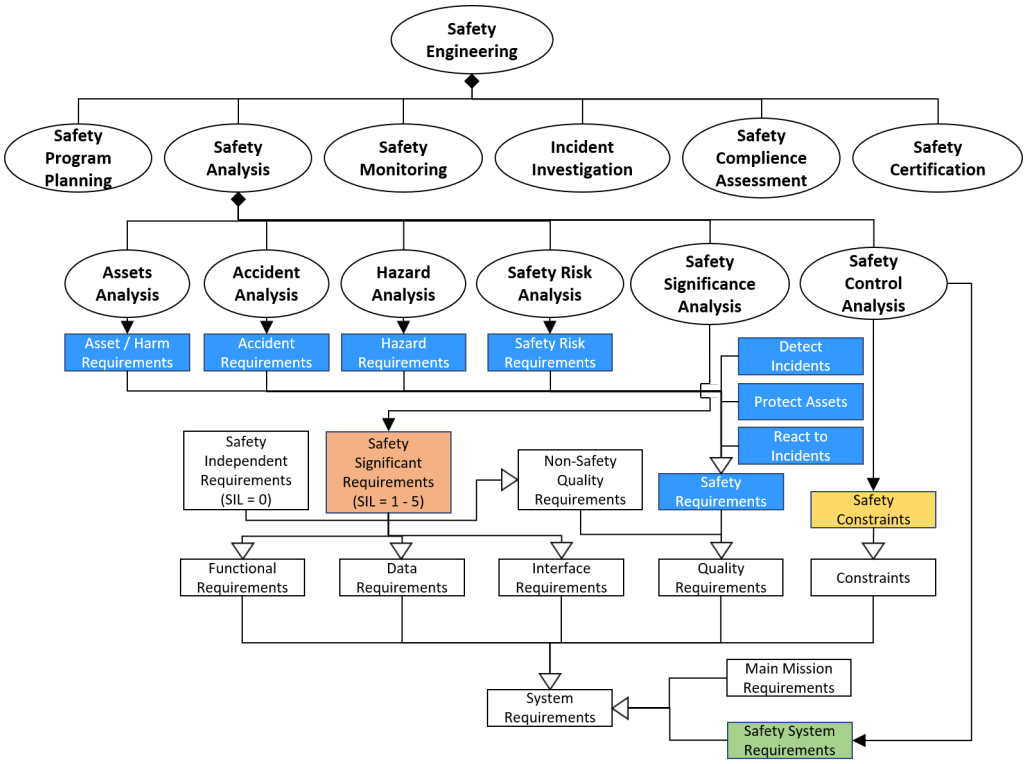


Figure 2.5 Taxonomy for Safety Engineering including the four types of safety-related requirements. From [53]

and associated safety risks) to output new safety-significant requirements. It is during these analyses that Safety Integrity Levels (SIL) are calculated. They are a target level of risk-reduction that a future safety function will need to provide for a specific system or performance.

The third type refers to safety constraints as business rules or engineering decisions like certifications that are treated during requirements engineering (e.g. compliance to regulation, standard, law). It corresponds to mandated safety policy or safeguards that affect the architecture development, design, implementation, integration, or testing (e.g. presence of OEDR, ODD, fallback, and a safety driver and procedures for testing for AV).

The fourth type corresponds to the safety system requirements. They are requirements that specify the safety systems and sub-systems built around the primary system already covered by the three previous types. The safety systems only exist to ensure the safety of the main system (e.g. emergency coolant system of a nuclear power plant). Their specification also involves some the three previous types of requirement since it is considered as a new system and may operate with different concerns.

Through these different types, we were able to see the extent to which safety studies could be carried out and the integration of their results as additional specifications. Figure 2.5 illustrates the different safety fields, the respective colored types of safety-related requirements and how the pieces of evidence are obtained to aim towards a safe system. At the end of the first process iteration, new analysis determine if the mitigation strategies meet the safety goals so that all the identified risks are brought to a minimum acceptable level according the selected safety standard. If it is not the case, the multi-iteration cycle is repeated until the hazards, goals and analyses converge (i.e. identification of hazards and safety risks, definition of safety goals and more detailed safety requirements, and

mitigation approaches for hazards). The choice of standard and the multi-iteration cycle takes are part of the safety plan that gives the big picture for the project safety engineering.

Next section presents the current vision of safety in AV, the additional perspectives that are adopted or in the way, and the new concerns and implications that need to be tackled to ensure the safety holistically in AV.

Safety Approaches for AVs

Safety engineering in autonomous vehicles is mainly guided by the ISO26262 standard [1]. It provides a collection of methodologies, techniques and tools such as safety analyses, validation of evidences and evaluations that cope with the analysis, design and assurance of a safe AV. However, the standard alone fails to define and provide delimitation for what 'safe driving' is and put it into requirements.

The next section presents the conventional approaches towards safety as performed in the ADAS (level 1-3), outlines its current gaps and introduces complementary standards or guidelines. Then, we detail a variety of architectural concepts, methodology and metrics that fills those gaps by representing, understanding and validating the safety by both the human (e.g. architectural designers, developers, testers) or by the system itself (e.g. monitoring, adaptation).

Conventional Approach: Functional safety from ADAS

The AV domain is relatively new in the vehicle industry. Therefore, there are no international standards explicitly developed for AV safety yet (and also security). Currently, the ISO 26262 standard [1] is being used for road vehicles and AV safety. It describes the functional safety for these systems and provides guidance, recommendation and argumentation for a safety-driven product development in the automotive area. Among others, it gives methodologies, techniques and tools applicable for road vehicles. This section briefly presents the standard and its related entities and identifies what the safe-by-design AV means. Finally, it reports the issues of the current standard and how new additional long-overdue standards and recommendations propose to address them.

Definition of ISO 26262 Historically, the standard is the adaptation of the generic norm IEC 61508 for systems comprised of electrical and/or electronic elements standard to road vehicles. ISO 26262 alone defines functional safety for automotive equipment applicable throughout the life-cycle of all automotive Electronic and Electrical (E/E) safety-related systems [128]. Functional safety refers to the "absence of unreasonable risk due to hazards caused by malfunctioning behaviour of Electrical/Electronic systems"[1]. In other words, it contributes to formalize and ensure the trustworthiness of the way of performing safety analysis and how the safety requirements are integrated to make the architecture safer. The standard specifies how to perform risk analysis and the consideration of their results as the first step of the system design process. It acts as acknowledging the problems that can meet automotive safety-critical systems.

ISO 26262 is structured into 10 parts that describe the different safety activities, and Part 3-7 target the safety life cycle.

Part 1 defines the vocabulary of terms used in the ISO 26262 series of standards. Part 2 describes a project management framework for functional safety to assist in the development of safety-related E/E systems. Part 3 specifies the requirements for the concept phase for automotive applications by defining the item (e.g. system or array of systems,

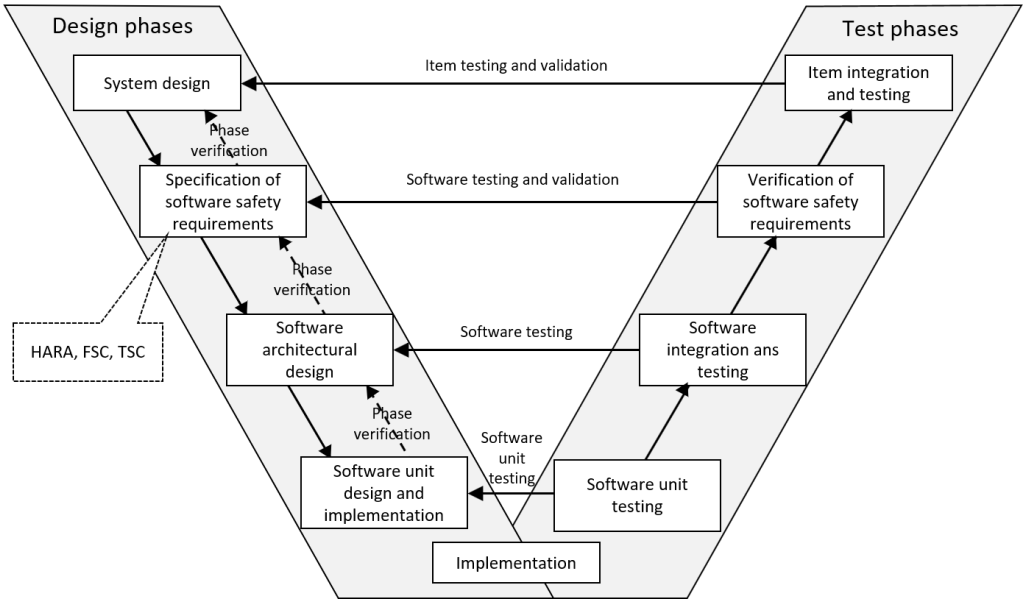


Figure 2.6 The V-Model for system and embedded software life-cycle in ISO 26262. Adapted from [1]

or function), performing the hazard and risk analysis for the item and providing a Functional Safety Concept (FSC). This FSC is the analytical result of safety goals for all hazards. They are derived and classified with an Automotive Safety Integrity Level (ASIL) rating that is a risk classification scheme defined in Part 9. Part 4 focuses on the next development phase and provides the requirements for product development at the system level for automotive applications. Part 5 and 6 intervene in requirements at the hardware level and software level for automotive applications. Part 7 specifies the requirements for production, operation, service and decommissioning. Part 8 specifies the requirements for supporting processes (e.g. configuration management; change management; verification, hardware and software tool qualification, proven in use argument). Part 9 specifies the requirements for Automotive Safety Integrity Level (ASIL)-oriented and safety-oriented analyses.

Appendix B (see page 163) presents the definitions of terms from ISO 26262 Part 1 relevant for this thesis. The adoption of this taxonomy aims to facilitate the discussions about easing the understanding of safety.

V-Model for system life-cycle ISO 26262 advocates for two pillars in the rigorous design of a safe system: Verification and Validation (V&V). Figure 2.6 illustrates the V-cycle and how V&V intervene between each phase, and in between during the design (i.e. each step from the earlier design, down to the specification of system and safety requirements, to the implementation), and testing (i.e. between the implementation, up to unit testing, software and integration testing and final product testing). What is important to note here is the coexistence of two design cycles - one linked to System Engineering and the other to Safety Engineering - which maintain links between them. In general, coordination of the activities of the two business lines is necessary. This coordination is reflected in particular in need to share data (documents, analyses, models) between the actors of the System and Safety Engineering. More information is available in the standard [1].

The specification of software and hardware safety requirements is the result of safety analyses. They perform the hazard and risk analyses for each item and provide related Functional Safety Concepts. The primary concern is the management of hazards. They can be addressed through three different ways and related techniques as follows: identification of hazard is performed through analysis; elimination of hazard is achieved by design; finally, control of hazard is implemented by management.

ISO26262 considers functional safety at the beginning of the safety-related system design with the analysis of risks and identification of hazards for relevant safety-related systems. This point corresponds to the second step of the design phase in Figure 2.6. A variety of for low-level risk analysis are used, like Hazard Analysis and Risk Assessment (HARA), Failure Mode and Effect Analysis (FMEA), probabilistic FMEA (FMECA), Fault Tree Analysis (FTA), Event Tree Analysis (ETA), Hazard and operability study (HAZOP). These tools result in the creation of safety requirements that may provide a safer system design (i.e. Functional Safety Concepts (FSC) and Technical Safety Concepts (TSC)).

The safety and reliability engineering analysis usually results in design changes that concentrate on dealing with component failure such as redundancy and barriers (rail-operational design to prevent failure propagation), high component integrity and overdesign (trustworthiness and robustness of the inputs), fail-safe design (safe mode as it does not endanger lives or property when it fails, e.g. the human driver is asked to retake control of the DDT) or fail-silent (continue operating properly in the event of the failure into a graceful degraded mode, e.g. mission remains pursued but with a loss of performance) or other operational procedures (e.g. safety plan, checklists, training). The adoption of the whole cycle contributes to consider safety from the ground up and incorporate safety at every system level and every development stage, from design to testing and validation. A similar approach needs to be pursued for AV regarding the design of autonomy and the safety-related. It specifies enough the requirements of the different components so that a change made to specific sub-systems do not affect the verification of the entire vehicle (i.e. 'side-effect-free' integration of new individual components).

Limitations and bottlenecks However, the application of ISO26262 meet some limitations that are predominantly due to the autonomy required in AV. This results in a difficulty in applying the methods and scale up even with some adaptations. The fact that ISO 26262 has gaps in the later steps of the design phase and flaws in its application to AV is acknowledged across the domain [57, 76, 105, 128, 144, 147]. It is at the heart of the debate as to whether it is applicable to the development of autonomous vehicles and its functions involving cognition or artificial intelligence. However, ISO 26262 is the only document that positions itself as a standard for every type of road vehicles, AV included. Moreover, adherence with the safety standard in any AV is essential not only for the comprehensive coverage it provides throughout the product safety life-cycle (from concept to decommissioning) but also because demonstrating compliance with the prevailing best practices is an important factor for the exclusion of liability in case of a system malfunction [76]. Using the assessment methods and already qualified tools in an existing non-AV tool-chain remains some crucial factors in determining how trustworthy a system could be (i.e. providing partial analysis and evidence) but might also add new faults. We detail these bottlenecks as follows:

Conventional solutions alone may be worse. On one side, this conventional solution may longer work but may be even worse applying not answering to key questions. Törngren et al. [144] reports some difficulties from applying ISO 26262 to provide an acceptable safety assurance and not sufficient. They remarks that not only safety assurance problem remains, but also that common cause failures are difficult to deal with and duplicating the main functionality (i.e. providing redundancy) may need negotiation between the system if they disagree. On the other side, functional approaches for autonomy alone makes the safety hard to handle afterwards. Most of the research functional approaches claims to be saef-ready will encouter difficulties to integrate safety design and assurance while it should have been done at the start of the system conception making it hard to perform. In brief, this duality makes the problem of safe autonomy for AV bigger and enhances the complexity of the management to be done.

Knowledge gaps and Incomplete identification of hazards. Previous claims are also sustained by the NHTSA report of Hommes [147] that mentions a lack in guidance on hazard identification and elimination (FTA, FMEA) in the concept phase. They concern the systemic and interaction related problems for complex software intensive E/E systems (emergence). These lack of detailed design specifications at the early design phase may lead to an unacceptable level of residual risks. Moreover, some vocabulary introduced may help but the related requirements are missing. For instance, the operational safety & behavioural (roadworthiness) is defined as the "property or ability of any kind of automobile to be in a suitable operating condition or meeting acceptable standards for safe driving and transport of people, baggage or cargo in roads or streets" but not specified.

Observability and controllability of autonomy. The Automotive Safety Integrity Level (ASIL) becomes difficult near complex to determine in AV. It results that most of the system become described as ASIL D in level 4-5 vehicles. These levels are no longer relevant for the highest levels of automation, when there is no driver for fallback. In addition, conventional methods to meet ASIL D (e.g. redundancy) are questionable [144]. New concepts for observability and controllability may be necessary as the system replaces the human being for monitoring and performing fallback.

Towards architecting AV with autonomy and safety. Performing both autonomy and safety engineering becomes difficult with AV since both of the perspectives intertwine but go to different directions. While the first way adopts a bottom-up approach to define continuously new requirements, data, tests, and other necessary features, the safety prones out a top-down analysis that need to be systematic and based on detailed design specifications (i.e. complete list of systems, functions). Missing some specifications, not taking the dynamic behaviour of systems into consideration (i.e. emergence due to feature interaction), or failing to provide safe evidence of safety requirements integration could result in some autonomy-related functions not to provide enough safety.

Performance of the autonomy functions. We have seen that driving automation is a difficult operation for levels 4 and 5 (i.e. perceive and understand the vicinity and take appropriate decisions as fallback). Not performed well, they can cause hazard and may result in critical accident but this performance not good enough to be safe is not necessary due to a malfunction. It can be the source of a wrong definition and specifications of the roadworthiness. It is also possible that performance limitations may be the

cause. For example, sensors may cause improper/erroneous internal representations such as false negatives. In addition, sensors in degraded mode may also contribute to lower the robustness of the perceived data.

Lack of big picture on safety and observability. The automotive industry has followed the excellent architectural principle of 'separation of concerns' the systems having limited knowledge of the existence, purpose and functioning of the other sub-systems. However, it fails when no system-wide reasoning is available in the presence of severe uncertainty, disturbances or internal failures whereas some tighter coupling based on machine learning for autonomy perform faster. Machine learning has brought increased progress to a variety of functionalities involved in the autonomy. Computer vision has improved the ability to detect surrounding objects in various conditions and better classification. Decision-making also increased its ability to produce actions that are effective and robust under uncertainty with the partially observable Markov decision process (POMDP). However, they have the disadvantages to currently be brittle to environment changes and data degradation. They also are like a black box with opaque behaviours reducing the potential for observability and verifiable granular specification (e.g. prediction of failures, how to perform validation against requirement when inductive training give no requirement nor design insights). Using ML in conventionally engineered safety-critical systems is still an open question according to [129]. Moreover, the uncertainty that originated from these probabilistic approaches may cause improper/erroneous internal representations such as false negatives and propagates to the rest of the system. Thus, it needs to be addressed with evidence to identify potential unknown emergence. Computer vision detection requirements issues may jeopardise the expressiveness and relevance of the perceived information (object, properties, behaviours, expectation). Consequently, with approximate sensors robustness, it becomes hard to achieve a holistic vision for safety in the AV within the system as and or even granular run-time views for each component.

Limitation of human supervision. Lesson learned from the start of deployment and testing of AV shows that the people are sometimes relying on automation to a degree at which manual skill decreases. Extreme situations where driver failed to perform their expected tasks or identify the wrong-doing of the system have led to critical collisions (see 1.3). Although there are less fatalities with these types of vehicles, societal and ethical issues remain in providing and testing *beta* safety-critical system while acknowledging that driver attention is mandatory.

Raised questions In addition to previous consideration, recent accidents emphasize the need for safety in the AV life-cycle (e.g. limitation of human customer supervision and distracted safety driver without monitoring). Consequently, these systems, which are considered critical to safety, must *systematically* investigate the perspectives for robustness, reliability, safety, ethics and social acceptance. The remaining question is "How safe is safe enough?" for AVs. We have seen that safety engineering corresponds to having residual risks to a minimum level supported by substantial evidence. However, verification and validation are tricky. Even if the vehicle may identify the conditions and perform for a safe state [122][157], how many miles in closed, virtual, and public roads provides acceptable evidence [74]? Which other criteria are acceptable (e.g. MRM performance, number of disengagement, incident per miles, failure injection while driving)

In [58], Lex Fridman and Elon Musk reach a consensus on how to win the race of AV that many companies would not face the risk to miss. It consists of acquiring a large amounts of real-world data (high confidence, analysis and replay value), accumulate miles driven (functionality to validation), and finally improve the relative safety of AV (iterative refinement of requirements, analysis) and reduce knowledge gaps (wrong model assumptions, edge cases, unknown unknowns [127]). Elon Musk also projects that safe enough means performing better than humans; two times to be acceptable and up to four times less than crashes, injury, death to be really impactful. To illustrate that keeping humans in the loop may be dangerous, he takes the example of the automated elevators that are no longer operated by humans for safety and reliability reasons.

Progressive coverage by extensions with recommendations and future standards Over the years, several organizations have proposed recommendations, voluntary guidance and best practices for the development and deployment of autonomous vehicles.

Historically, three leading organizations including NHSTA (National Highway Traffic Safety Administration) [107], BASt (Germany Federal Highway Research Institute) [60] and SAE International (Society of Automotive Engineers) [39] have individually published slightly different standards defining the levels of automation of autonomous vehicles. The number and the differentiation of levels for each proposal are somewhat different, mainly due to the diversity of point of views regarding automation capabilities and technical usage. However, the three schemes globally converge on the key automation definitions. Finally, the propositions result to be subsumed in the SAE J3016 and latest updates [40, 41]. As we have been able to present in previous sections, the document covers the autonomy of the driving tasks with guidance on the concepts of DDT, DDT fallback, OEDR, ODD. Although it does not address every part of the non-functional analyses, this is the first document to give enough guidance for the driving automation and the environmental conditions to be used as reference.

Based on these new levels, NHTSA [108] describes its vision for ADS and its lifecycle by including the considerations of ISO 26262 and regulations. This document aims to aid industry as it moves forward with testing and deploying ADS and each state to draft and establish plans for ADS.

Waymo publishes its first safety report in 2017 [152, 153] that conveys the list of good practices the company adhere to. It also defines different aspects of safety, such as the usual functional safety, crash-analysis safety. The relevant one in our scope is the behavioural safety that finally "*focuses on how a system should behave normally in its environment to avoid hazards and reduce the risk of mishaps*". Even it may intertwine with the functional operations of driving, it is identified as resulting from both the collaboration of autonomy and safety engineering.

The ISO/PAS 21448: Road Vehicles or SOTIF (Safety of the Intended Functionality) [2] aims to address whether a driving function under normal operation behaves in a safe way. It only address the part of the concept of behavioural safety that is not in the scope of the ISO 26262 (i.e. malfunctions can occur even with defect-free equipment). It focuses on the hazards due to functional insufficiencies (e.g. sensors which do not function in all environmental conditions) or reasonably foreseeable misuses (e.g. inappropriate driver monitoring). SOTIF is all about the scenarios as it provides tools and guidance in enumerating scenarios and triggering events to generate a giant scenario database. The iterative process of data collection and risk reduction is performed until the residual risks are acceptable. However, this new standard fails to address the rest of the problem: so-

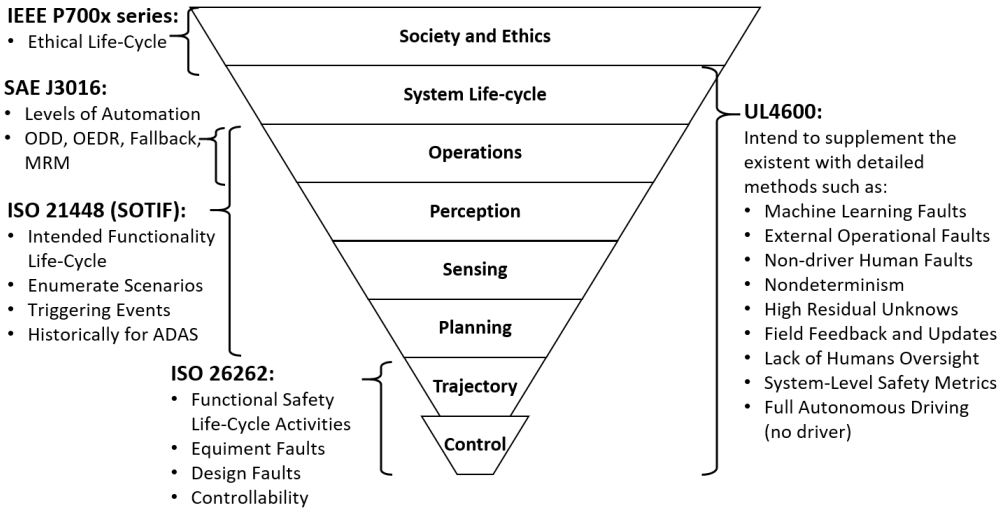


Figure 2.7 Coverage of the AV concerns by the different existing and future standards. Adapted from [82]

ciety and ethics (i.e. How safe is enough?), system life-cycle (e.g. updates), vehicle operations (i.e. driver do more than just drive), and perception for object classification remain (i.e. validation of inductive training).

Finally, the recent goal-based and agnostic UL4600 standard intends to be the first comprehensive document to address the safe deployment of autonomous vehicles and mobile robotic products in Q4 2019 [124]. It [80] explains how validation of the safety of a autonomous vehicles can be performed without stating any design approach or specific technology to used via methodology and metrics for V&V. The standard focuses on addressing the complexity of the combined ODD, OEDR, Manoeuvres and Faults [83] and supplement some safety activities. The society and ethical considerations are left to under production ethics standards drafted around autonomous and intelligent systems in the IEEE P700x series. Figure 2.7 illustrates a comprehensive vision of the presented standards and summarizes their coverage around key aspects.

Conclusions As suggested by UL4600, adopting a goal-based and agnostic approach may be a suitable and methodical way to objectively measure how one AV comply with its expected intended functionalities and performances as well as to include perspectives for robustness, reliability, safety, ethics and social acceptance. These documents advocate for more transparency, a minimum performance and autonomy software safety. Although these recommendations and new standards may guide for better autonomy and safety integration in AV, the variety of documents addressing each intertwined concerns shows that safety is achievable through multiples ways. An everlasting question remains to solve the safe autonomy puzzle: What coverage pieces of additional methodologies or mechanisms remains necessary to obtain a complete safe design?

Next section answers this question by presenting six different pieces of puzzle to provide and ensure safety in AV.

Purposely Designed Approaches to AV Safety

One of the significant challenges in deploying a fully autonomous vehicle is to design, develop and validate the system with an acceptable level of safety. Pursuing better safe

design copes with the issues and gaps of the complexity of safe autonomy by proposing pieces to improve the current state-of-the-art in the respective areas. These solutions take different forms. They are methodologies, modular analysis, structural patterns, shifts in how the problem is managed, or behavioural adaptations.

To our knowledge of the current literature, numerous approaches exist to progress towards a safe-by-design architecture as well as systems perspectives to be more robust, easier to design, and easier to verify and validate. We have identified the following categories:

1. Novel trends and evolution of the methods in safety management
2. Robustness of systems and functions driving the autonomy
3. Designing Safe Embedded Software
4. Model-based Safety Supervisors and shifting some safety assessment to run-time
5. Safety architectural patterns for fail-safe and fail-operational systems
6. Formal safety approaches to design and manage dependable systems
7. Safety metrics to guide AV safety assessment and certification

Novel trends and evolution of the methods in safety management The ISO/PAS 21448 standard fills the gaps of ISO26262 on the design and development activities for functional safety. It targets insufficient item definitions that result in inadequate safety requirements and safety goals that could be violated even when the system is fault-free. However, the promoted safety analysis methods still lack guidance on hazard identification and elimination in the concept phase [147].

Other approaches have been proposed in the literature for hazard analysis to address the lack of safety requirements and constraints of the system before detailed design. Different works have reported STAMP/STPA to identify different hazard than conventional ISO 26262 HARA and FMEA methods [139, 147].

Systems-Theoretic Accident Model and Processes (STAMP) [97] have the particularity to describe the individual and multiple components failures but also covers the design and nominal performance flaws. This inappropriate system performance includes management process, requirements, human behaviour, indirect or non-linear interactions and technical parts.

Systems-Theoretic Process Analysis (STPA) is the method to analyze systems based on STAMP [98, 99]. The difference of the type of accident model and how risks are considered with FMEA results in finding a different coverage of the possible hazards and thus leads in to further safety goals that were previously not handled. The method consists of three steps.

The first step corresponds to the preliminaries of the analysis. It consists of the identification of the accidents and hazards based on the outputs of the HARA approach and the identified control structure of the system. The control structure diagram identifies the major components and controllers and is labelled with the control/feedback arrow.

The second step identifies the critical safety control actions on the high level of abstracted view of the system. They are obtained by crossing the control action from the control/feedback arrows with four general hazardous types (i.e. not providing causes hazard, providing incorrect, providing at wrong timing/order, and stopped too soon/applied

too long). This way we know whether or not they lead to hazardous events. Then, each unsafe control action is translated into a corresponding safety constraint by using the guide words (e.g. "shall" or "must").

Finally, the third step identifies the causal factors of the hazardous events to implement the safety countermeasures changes to fit the new requirements. It consists of determining the controller process models, and analyze and modify the different controller, control path or feedback path, and process accordingly.

Different works have applied the STPA method to vehicles and have reached positive outcomes [4, 19, 42, 90, 119, 128, 144]. Moreover, STPA is detailed as additional hazard analysis method in the safety report of some AV actors [54, 55, 62, 152, 153].

The literature advocates for the combined application for STPA and functional safety ISO 26262 in order to discover more hazard and gather more goals and requirements [4, 19, 128] and integration to standards [148]. Some work also advocates for the even more extensive integration of all standards and recommendations for Safety analysis as they combine both ISO 26262, STPA, SOTIF, augmented faults trees and Goal Structuring Notation (GSN). GSN is a graphical notation for modeling a safety argument, which is the core part of every safety case that gives a clear, comprehensive and defensible argument that a system is acceptably safe to operate in a particular context.

The outcomes of the STPA method are positive. The possible combination brings more perspectives for other standards. The iteration is possible until satisfying granularity in software code and software requirement is reached. But it actually requires some trade-off regarding the difference in vocabulary addressed in [3], and the manual interventions from a safety engineer on the modeling tools. An industry-ready version of the STPA would involve tool-chains need to be systematic and automated as possible to ease the development and verification.

We can also expect the UL4600 standard to propose combined methodology with STPA to address other parts of safety life-cycle in AV from the conception to run-time. This recent standard intends to address changes required from conventional safety practices to accommodate autonomy.

Perception building robustness and stress testing to adversarial inputs Perception systems are particularly affected by unusual situations. However, safety needs to know if the detected object is a person while the machine learning-based system can only answer that it matches as a person in its training data with a relative confidence. The functions responsible for the autonomy need to be either more efficient and more robust with additional training.

To measure, discover and patch, three directions for testing have been performed with autonomous vehicles. Closed course testing is conducted in a managed environment, making it quite safe, but have the disadvantage to be expensive and not scalable.

The second is public road testing and excels in identifying "easy" cases. This method is costly as it requires to deploy a fleet to acquire data and experience, and potentially dangerous (see accidents during testing in Section 1.3, page 25). Brute force road testing should not be for debugging (ethical issues) but for data collection purposes only according to [81]. However, the validation of the driving system with this method would require 100M miles per critical mishap or surprise. Moreover, it would take the unrealistic distance of 1 billion miles of testing if tests need to cover from 3 to 10 times the mishap rate.

The third allows reducing these number by performing driving simulation and focus on the specific scenario and situational training. This method is highly scalable (cloud processing power), less expensive and the safest, but it only creates scenario the developer of the simulator have thought of.

The most challenging part of testing is to cope with these limitations of scenario and unknowns. The study of edge cases has shown promising results in acknowledging those surprises and unusual, rare, unique or unknowable things not seeing during testing. They correspond to gaps in training data that can lead to perception failure that the developer, tester, safety practitioner did not catch. Moreover, they are due to a variety of factor: the limitation of the perception system within specific environment (e.g. camouflage, sun glare, occlusion), unusual road obstacles, unexpected or strange human behaviour (e.g. person with an animal costume), non-understood interaction (e.g. human perform form of signalization in a construction area, human being bare legs). Edge case testing is mostly context-dependent [81], and their listing contributes to create a "Zoo". On this basis, robustness testing consists in a variety of methods such as altering the scene (remove or alter object), data augmentation (add new object to the scene), data degradation (adversarial inputs), data fuzzing (add noise) and injecting "unrealistic" faults.

To conclude, reaching an acceptable level of testing for validation should refer to the ability of the system to react to objects and events from edge case "zero".

Designing Safe Embedded Software and System Architecture Leveson [96] advocates for safer system engineering that systemically integrates system hazard analysis, user task analysis, traceability, and informal specifications combined with executable and analyzable models. The complexity of most embedded software limits our ability to assure safety after the fact. The proposed approach requires considering it from the start of system development and designing the software. Thus, it reduces the potential for hazardous behaviour that achieves higher confidence in safety.

The recommend method to be adopted by INCOSE (International Council On Systems Engineering) and the French AFIS (Association Française d'Ingénierie Système) is model-based engineering. This type of engineering is no longer based on documents but on models manipulated in the being formal or semi-formal. The change mentioned is therefore accompanied by the introduction of the implementation of formal methods.

From an architectural perspective, Behere et al [26] report that the intended functionality can be addressed through two complementary approaches. The first one, named *'correctness-by-construction'*, refers to principles and mechanisms of composing systems out of subsystems in a way that there are no unexpected side effects or emergent behaviour. The constructionist design methodology involves the integration of a large number of functionalities that must be carefully coordinated to achieve coherent system behaviour [25]. The second approach consists of eliminating the feature interaction both at the design phase and run-time. The feature interaction occurs "when the operation of a subsystem/feature interferes with the operation of another subsystem/feature leading to unexpected and undesirable system level behaviour" [26]. It is a direct result of the cognitive complexity [89] from continuously and mandatory compliance to a variety of high-level goals and policies. The authors [26] suggest it can be achieved using a formal representation of both the architecture, a feature interaction, and use model checking and verification methods to search for the feature interaction in the architecture. Some run-time detection and resolution have been performed in the networking and telecom domains, but far from the assurance of robustness and safety an AV may need.

Furthermore, Bagschik et al. [20] translate this vision to safety proposing and redefining the term '*safety-by-design*'. Only multidisciplinary and cross-domain efforts from academic disciplines may provides such design. It results in the need for a holistic engineering process to carry out and manage the emergent properties of safety of a complex system both during design and run-time.

Therefore, a shift is occurring from manually designed to self-organizing architectures that can learn and grow to accommodate to autonomy (e.g. learn to learn). According to [141], such solution replaces top-down design approach with methods that allow the system to manage its growth. Thus, it requires the solution to have a high degree of architectural plasticity. The authors identify three efforts to pursue autonomous architectural adaptation and system growth: system components shall have the possibility to be dynamic and not fairly static; the architecture shall include as many as components as possible and not be limited by what a designer or team can handle; and finally, components and their interconnections shall be self-managed in the architecture to pursue higher flexibility and scalability.

To conclude, designing safe embedded software consists in providing scope for holistic safety in the constructionist approaches and prepare the transition towards self-organizing architectures.

Model-based Safety Supervisors and shifting some safety assessment to run-time A supervisor plays the role of providing a holistic safety assessment at a specific scale of the system: a nominal channel is monitored by another channel with an higher integrity. In safety engineering, supervisors are conventionally safety systems designed to monitor, analyze, predict, and mitigate failures or faults happening in the nominal system. They belong to the safety countermeasures established for the system safety of the nominal channel. Safety engineering conventionally prefers that predictable critical properties are established at design time. However, dealing with the uncertainty that comes from the unpredictable environment forces the system to be adaptive in order to keep it acceptably safe.

A vehicle-centric supervisor in AV aims to maintain the coherence of the underlying systems to pursue safe navigation [71]. It can involve a variety of fail-safe, fault-tolerance and fail-silent mechanisms and monitoring processes that are triggered in case of performance degradation or malfunction of any of the systems.

In application, the system run-time monitoring shall identify and predict future violations of safety-critical properties detection of violations and mitigate them. Two subclasses for safe adaptations can be distinguished. The first, adaptive fault detection, prevents the system to be brought into an unsafe state by performing threat assessment that warrant a set of predefined rules. The second, adaptive failure mitigation, involves self-adaptation in the sense that the system can detect, evaluate and react to a potential failure (e.g. caused by uncertainties faults) and bring the the system into a safe state. This mitigation implies the supervising system to also be self-adaptive so that it adapts to the uncertainties, uses knowledge (a-priori and acquired by learning based on experiences) to make own decisions, and operates correctly by controlling and changing the system behaviours.

For example, Törngren et al. [144] propose a fault-tolerant functional architectural (functional safety) in which a safety supervisor that complements the nominal functionality of driving. The safety supervisor represents a redundant and simpler set of functionalities that should take over if the nominal channel fails. The supervisor consists

of autonomic structured functions (Monitor, Analyze, Plan, and Execute) and performs run-time monitoring over internal and external safety constraints to assess. Internal constraints refer to the possible violations in the status of the nominal channel and the vehicle platform while external ones target any causes of violation in association to the driving scenarios. Such approach imposes to have an observable nominal channel at different key interfaces (e.g. sensor status, sensed objects and properties, near-term trajectory and planned set-points send to control) and enough knowledge to appropriately perform the various run-time verification and plausibility checks upon the context. However, the way of formulating and deriving safety constraints, as conditions or risk measures that cover all relevant hazardous events, is recognized as difficult by the authors [144] as possibly hand-crafted. This limits the ability of the architecture to scale within a new and evolving context.

Such concerns aim to be handled with run-time safety assurance. In essence, it consists of shifting not only the safety countermeasures but also the whole safety engineering considerations and activities from development time to run-time. It covers the different steps of HARA, V&V, Assurance and Certification. A general scheme of the process and the related functions is proposed in [37, 145] to dynamically manage functional properties as well as non-functional properties and assurances following the model@run.time design [15]. It requires quite a variety of sources of knowledge about the new tasks and their corresponding high-level cultures (multi-concern and multi-level models to specify goals, policies). It also needs the mechanisms to operate that knowledge and templates for the possible generated countermeasures.

Some research works are pursing in this direction, and not only address the needs for knowledge representation [11, 131], but also propose constructivist architectures that evolve and change how the safety assurance is performed [154][155].

To conclude, safety supervisors and the operating knowledge will shift to run-time in order to purse a more composed and personalized safety assurances (e.g. functional safety and other types). This is also extendable for other system qualities which was not possible to handle at design time and in a human-scale development, but now possible with the large models that documents the different safety engineering considerations, activities, tasks and mechanisms.

Safety architectural patterns for fail-safe and fail-operational systems Supervisors are a form of safety controllers that act at usually the high-level scale of the system. However, other safety patterns exist to provide fault tolerance and mitigation at a different level of granularity for the system. They usually consist of multi-channel approaches that ensure the safety of nominal usage. A typical application is the avoidance of any single point of failure in the software. Patterns are accepted and generalizable solutions to a specific set of needs, such as a safety argument. They oppose the anti-patterns that reflect how some ad-hoc solutions or clever shortcuts fail to make the system safe. The usual pitfalls for safety to be avoided are mixed-SIL software without isolation, no redundancy for high critical functions and confusion between fault detection and availability. Thus, appropriate patterns would provide solutions to cover those cases in the forms of cross-checked redundancy for fault detection, standby redundancy for availability, and clear separation between high and low SIL functions.

For example, the doer/checker generic pattern provides a simplex architecture between two systems to provide a fail silent channel. A low integrity doer operates the main functionality that has a low safety integrity level (SIL). A high integrity checker monitors

its inputs and outputs. It has a high SIL and acts as a safety envelope checker involving simple functions. Different versions of this pattern exist upon the integrity of the doer and checker, their software/hardware allocation and the trade-offs (pros/cons). They can provide different behaviours for the system to fail: fail-silent (discontinued operation) or fail-operational (full or degraded operation of a function even if a failure occurs). In [144], the supervisor consists of several doer and checker couples that safely operate different steps of the driving operation (M, A, P). The checkers diagnose the safe operational state space of the doer and any unsafe state space that triggers the performance of a safety manoeuvre or complete take over of the doer by the checker.

To conclude, the architectural patterns that focus on safety, autonomy, or combined safe autonomy are instruments for reducing the complexity of system design and system integration referred to composability and correctness-by-construction [76]. They introduce principles for generic reconfiguration and control over the different elements of the system.

Formal safety approaches to design and manage dependable systems Formal approaches aim to work on the correctness and completeness of the requirements, which involves the creation of one or more models. A model is an abstract description of the system and/or process. This simplified representation facilitates the manipulation among shareholders and its verification as well as reduces the system complexity.

For the design of dependable systems, formal approaches are well suited as they can make sure that for each possible event, a system reaction is specified — this way each foreseen event are considered and take place in the design of the model. Thus, they are used to define and prove system safety properties (e.g. model checking, control system analysis, use of validated synthesis tools, kinematic analysis, and correct by construction approaches).

However, formal approaches meet limitations in a partially known environment and in their scaling up to the entirety of an autonomous vehicle, although they remain particularity valuable for the safety arguments [85]. For example, edge cases do not allow to have a complete list of events to mitigate (OEDR). Moreover, safety remains an emergent property that results of the intertwine between the system (design, implementation, knowledge) and the environment in ways that often reach beyond the formal verification and modelling.

Some research works aim to tackle those challenges of formal approaches such as [49] that bring probability in Petri-nets with a fallback application or [100] that surveys the formal specification and verification of autonomous systems.

Safety metrics to guide AV safety assessment and certification A metric is a measurement or a Key Performance Indicator (KPI) for an observable feature. A set of generally applicable metrics or signals that presage an accident can contribute to identifying and measuring the risks and system performance for different outcomes. The safety domain refers to two types of indicators that can exist [93]. Lagging indicators analyze past performance where actual safety outcomes involved harm (e.g. crashes, injuries, deaths, costs). Leading indicators identify the potential for an accident before it occurs and influence future performance (i.e., proxy measures of driving behaviours correlated to safety outcomes). The leading indicators are proactive measurements as context feedback and provide a means for predicting and anticipating the risk or mitigating the future perfor-

mance results. Their identification and control can significantly affect the effectiveness of the safety control structure in the system to prevent misbehaviours or accidents.

Obtaining an effective set of leading indicators is a clear objective for safe systems. Their identification conventionally results from the safety analysis and other control processes. Leveson [93] indicates that good quality indicators need to achieve several goals as being:

- Complete, a process for determining what should be checked, how, and when will be a critical part of the identification of the leading indicators. No process is perfect in its coverage, but it needs to address all critical assumptions.
- Consistent, the leading indicators may underlay some inconsistencies that need to be identified and handled.
- Effective, the indicators should appropriately address the underlying assumptions, uncertainties, and vulnerabilities and accurately evaluate risk.
- Traceable, each leading indicator and the action attached to it should be identified as a response to one or more assumptions.
- Minimal, there should be no extraneous assumptions, checks, or actions that are not necessary to prevent accidents.
- Continually improving, the leading indicators should be frequently updated over time in response to feedback about their effectiveness.
- Unbiased, the leading indicator process should minimize standard biases in risk assessment and management. Reducing the exposure to bias may involve updating the structured method for identifying, detecting, and managing leading indicators.

Much effort has been spent on trying to identify these indicators in AV to assess safe driving and autonomy. Metrics as miles driven and the frequency of human intervention are used for open-road testing and are displayed as evidence for the public. However, they are reported as insufficient to demonstrate the safety of the AV [57]. Moreover, additional metrics in use for AV accident analysis have also proved to be not sufficient even after car crashes [36] and need improvements [51]. The AV research effort is shifting from the old race for "quantity of miles" to the "quality of coverage" required for safety awareness, testing, verification and certification.

The report [57] aims to propose a global understanding and measuring of the safety in AV. The authors are surprised about the unclear and no standard definition of safety in AVs. They take the lagging and leading indicators as AV safety characteristics to define, measure and communicate on the safety puzzle. They identify three types of relevant leading indicators: (1) *infractions* as failures to follow traffic rules, (2) *disengagements* as the occasion when human has to take-over (e.g. during testing), and (3) *roadmanship* as the ability to drive safely and appropriately within the road space. The roadmanship concept is an attractive leading measure of safety conceptually but complicated definitionally [57]. The general idea behind this concept is to regroup all indicators that cover the ability to drive without creating hazards and responding to the ones created by other road users. It is not defined yet by standards, but it is partially integrated into existing AV ad-hoc approaches using terms like appropriate, behavioural or safe are used.

Moreover, the report identifies additional goals for good quality AV safety indicators to be objective, physics-based, available using current technology, and reflective of the

official and unofficial rules of the road. Surveyed measurements include the potential for near-misses or near-crash, the events of rapid acceleration or deceleration, time-to-collision, the probability of an unavoidable crash, the definition of a safety envelope, and effectiveness of evasive manoeuvres. Like all types of measurements, practitioners have to make a trade-off between the advantages they offer (e.g. meet some of the expected goals) and drawbacks (e.g. subjective, incomplete, irrelevant in certain conditions, covers only partial indicator effectiveness goals, or techniques and tools not yet available).

Recent work and industry best practices have started to report publicly the rules and best practices they are using to grasp the safety [103] and the concept of roadmanship.

For example, Shalev-Shwartz et al. [133] proposes a safety envelope, called Responsibility Sensitive Safety, established around five formal logic and rules. Some consist in managing numeric metrics that intervene during driving with limits and conditions under which the system must operate, such as the longitudinal distance (don't hit the car in front of you) and the lateral distance (don't cut recklessly). Others are more high-level rules and detail how the autonomous vehicles should behave and interact: the right-of-way is given not taken (i.e. negotiation), be careful in these zones with limited visibility and occluded areas (e.g. hidden pedestrian behind a wall or a car), and safely and legally avoid collision without causing another one (i.e. don't be the initiator of unsafe decision leading to accident).

Other research works [122][157] have also covered with metrics what constitute a safe state and a safe spot to immobilize the vehicle for its fallback.

For the coverage of technology-agnostic metrics, we also have high expectation for the future UL4600 standard [124] to formalize safety measurements and awareness across level 4-5 autonomous vehicles. It may provide farsightedness on existing recommendations (e.g. RSS requires to be more about edge cases and road conditions [86]). The future standard might also provide a library of detailed metrics, recommendations of application in specific contexts, and processes and tools for the identification of new indicators and maintenance of existing ones.

To conclude, identifying a practical set of leading indicators for an AV needs to meet some key characteristics to be integrated into the system. They contribute to providing risk identification, assessment, prediction and anticipation to prevent future misbehaviours or accidents, once each drawback is acknowledged and managed. A library of metrics, tools and processes for metric identification and maintenance would help the AV domain to purse safe driving.

Synthesis

Winning the race to safe level 4-5 AV is not only achieved by acquiring large amounts of real-world data and accumulate miles driven but mostly through the improvement and assessment of the "quality of coverage" of safety in AV (e.g. reduction of its knowledge gaps, run-time assessment, combined metrics).

Safety analyses are complex with different methodologies and approaches in the specification of the system requirements that have to be combined to reduce any gaps or residual risks. The entire process of analyzing and managing the safety of the system needs to be systematically performed as it ensures the coherence (correctness-by-design and safety-by-design) and the traceability during all the life-cycle of the vehicle (design, implementation, assembly, operation, maintenance). In addition, the association of safety with autonomy and run-time needs to face the incremental gain of maturity of the future AV system towards adaptive or constructivist architecture.

With this in mind, we propose to designate the global discipline as "self-safety". We consider this autonomic meta-property as covering the intelligent representation, composition, and orchestration of the safety qualities in components based on self-configuration, self-optimization, self-healing and self-protection properties. Self-safety refers to the ability of the system to provide both safety-awareness and a dynamic safety management in order to guarantee the overall safety of the system based on different and heterogeneous autonomic properties and mechanisms.

The different categories and related works have lead to the identification of four mandatory quality system (NFP) for safe autonomy to progress towards safe-by-design architectures as well as ones that are more robust, easier to design, and easier to verify and validate.

Observability Measurements and indicators requires observability points in the system architecture to produce both machine and human-interpretable data that detect defect and presage an accident (e.g. to know if the system is not just getting lucky to successfully avoiding an obstacle). They contribute to the construction of insights on the design, performance, and intention of the autonomous system. Availability of the measures depends on the specific AI uses (e.g. black-box IA, grey-box AI, functional), the measure type (design time or run-time), and its management (operational, tactical, strategical). Such access to a variety of data thought built-on or built-in monitoring contribute to provide different adaptive services such as fault-tolerance, evaluation of the health and performance and roadmanship of the system, and also assessment of the current contextual situation.

Traceability The observability and traceability contribute to demonstrating that the system is doing the right thing for the right reason in run-time. To design and operate such monitoring and adaptation, the system requires traceability to be consistent, to have rational reconfiguration decisions that can be explained with justified reasoning, and to ease verification and validation upwind. Safety assurance traces the different evidences such as safety analyses, system requirements, system design and implementation, knowledge representation, behavioural rules and metrics. In the scope of adaptive systems, the system need to have and provide access to them, and even make them (e.g. learning).

Reconfigurability and Evolvability Approaches have arisen in the field of AV systems towards run-time monitoring and assessment to manage identified risks, catch assumption violations, uncertainties (e.g. unknown unknowns) and safety assurance. Only a systematic approach to the vehicle's context and its system capabilities of driving, reconfiguration and evolution can achieve to cope with the safe autonomy complexity. Involving abstraction on the context and the components implies the use of a knowledge model to capture the relation between the available components, their interfaces and specifications. Based on such understandable knowledge of the system, the compositions of those described components can be the study of design and run-time to provide the most relevant adaptation mechanisms. Perceiving the system as multiple control loops contributes to extending the ability of the system to manage the uncertainty using dynamic and flexible run-time adaptation mechanisms. In application, the leading indicators are expected to change over time as well as the architecture of the system to cope with the new context (e.g. updates by ODD, OEDR, mission, goals, beliefs, learning).

Flexibility for managed Composability and Maintainability The design and development of autonomous systems are heading towards highly dynamic and flexible self-adaptive reference architectures. They seek to achieve a greater extensibility, modularity, agnosticy, interoperability, composability and maintainability of behavioural and structural system adaptations [138]. Extensibility refers to the ability that new components can be easily plugged to be added to the system. Modularity emphasizes loose coupling and high cohesion to take system to the next level of abstraction. Agnosticy refers to the unawareness or noncommittal willingness of global system to know the specific nature of the components until their integration (e.g. add knowledge as respective logic) or interaction (e.g. add scheme to exchange). Interoperability allows the components to be discovered and to easily communicate dynamically (e.g. caused by the heterogeneity, the reconfiguration or the evolution of components). Composability suggests that the components with generic structures and interfaces can be combined to fit new actual goal or create a new metric. Maintainability covers the replacement, update and disposal of the component and knowledge. All of these attributes contribute to building the different management services of a self-adaptive architecture (i.e. discovery, composition, orchestration, deployment and binding capabilities) and makes their customization available at design and run-time.

2.4 Conclusions

This chapter has shown how safe autonomy evolve through systematic analysis of AVs, specifically through the creation of architectural representations (functions, supervision for holistic safety, patterns, formal models), representation of knowledge (metrics, rules) and process (safe system design). Figure 2.8 illustrates the four domains where some long term perspectives enable to progress towards safe autonomy, i.e. Concepts, Architecture, Systems Engineering and Technical Implementation. On this behalf, they advocate for approaches to provide greater observability, traceability, reconfigurability and flexibility for which the AV domain have only done very brief and occasional excursions yet [25]. Progress from the contemporary low-level concepts (inner cycle) to the arising high-level concepts (outer cycle) involves these four non-functional properties. Their support can be integrated by extension, refactoring, redesign or composition of existing concepts, architectures, methodologies and implementation.

Architectures. Both safety supervisors and nominal channels operations is shifting to run-time in order to pursue a more composed and personalized safety management and assurance. The future of the safety domain lies in flexible and reconfigurable structures for run-time system level reasoning and their openness for architectural evolution (e.g. learn and grow to accommodate to autonomy). Achieving such autonomic enhanced design not only enhances the observability and traceability of safety in the driving system architecture but also promotes reconfigurability and flexibility to progress towards a scalable adaptable architecture.

Concepts. Potential solutions address the problem of validating machine learning by separately and independently define what “safe”, “appropriate” and “behavioural” operation means. This separate set of safety requirements could be imposed as a set of independently monitored behavioural requirements on the autonomous vehicle’s autonomy [75]. Such monitoring and approach to holistic supervision can be used during validation,

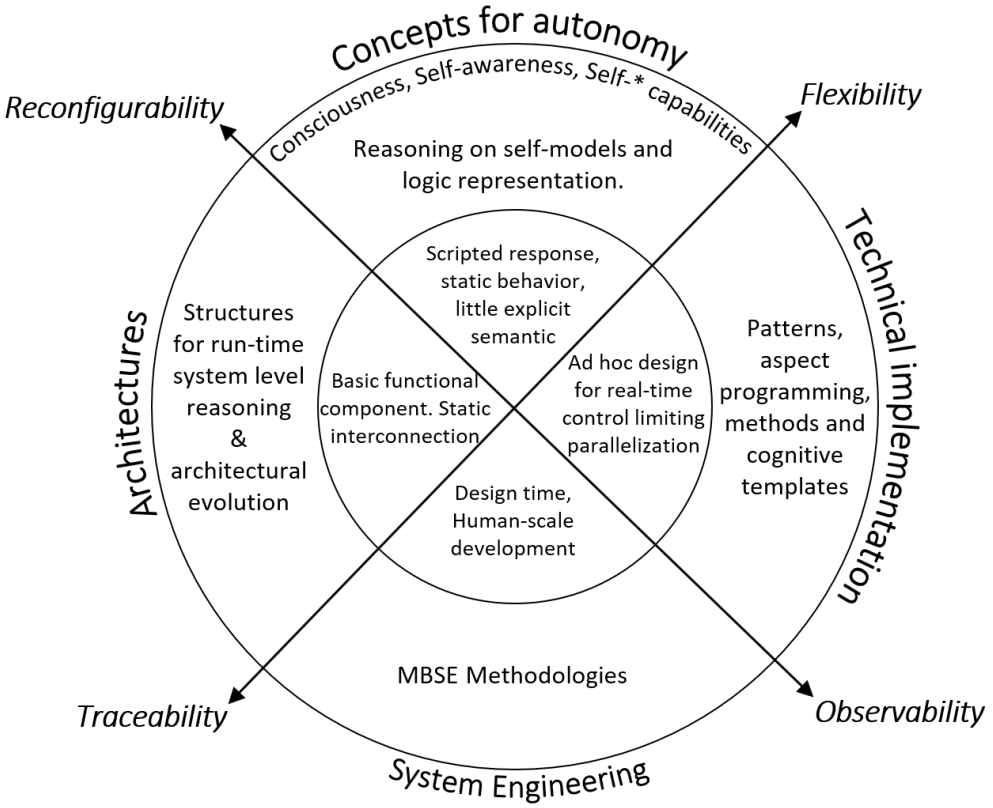


Figure 2.8 Evolution towards observable, traceable, flexible and reconfigurable systems and self-safety. Adapted from [25]

on-road testing, and perhaps deployment to ensure that the vehicle does not exhibit unsafe behaviours, even if gaps or faults still exist in systems based on machine learning [79]. Such future autonomic or adaptive architecture should integrate knowledge bases, goals and policies orientation as well as self-managing and orchestration functionalities. The result is that both behavioural and structural adaptation strategies should reason on models (e.g. system self-model, environment), actions, outcomes and their desirability (e.g. links between context and behavioural safety). Moreover, the concept of safe autonomy bases its trust in the construction of behavioural and structural adaptations that progress towards the creation of consciousness, self-awareness, and self-organization.

System Engineering. Improvement of the current practices, methodologies and tools for modelling safety-critical complex systems will favor the integration and enforcement of qualities that were not possible to handle at design time and in a human-scale development. It makes reference to the design of safe embedded software that consists in providing scope for holistic safety in the current constructionist approaches and prepare the transition towards self-organizing architectures. System engineering provides a scope for composing both the new types of architectures and templates in implementation to contribute to easing the design, development and traceability.

Implementation. The shift to adaptive or constructivist architecture will undoubtedly introduces paradigm shift in the way the system is implemented. A pattern for such autonomous system will require to support reflectivity, knowledge acquisition, learning, massive parallelization, real-time control (i.e. from simple monitor to a hierarchy of

controllers) [25]. This will require new programming languages, capabilities, tools and patterns to fit the new needs.

What is next. The following chapters explore the four introduced categories. They provide our contributions that compose existing approaches from different domains to progress towards these desired high-levels levels.

Chapter 3 tackles system engineering practices and surveys potential approaches that contribute in managing and providing behavioural safety to autonomous cyber-physical systems. We also explore a MBSE methodology to design and model autonomous vehicles.

Chapter 4 proposes a microservice-oriented autonomic architecture for safety-awareness and safety dynamic management in coherence with our NFP of observability, traceability, reconfigurability and flexibility.

Chapter 5 provides the application of the two previous contributions with models and implementation of the AV safety management system in a pedestrian case study.

CHAPTER 3

Methodology and Architecture for Safety Management

"All models are wrong; some models are useful."

- George E. P. Box, *Statistics for Experimenters, 2005*

Contents

3.1 Introduction

Ensuring the trustworthiness of autonomous systems can not only be based on evidence from some rigorous design methodology and critical system engineering techniques and standards [135]. The enforcement of AV safety when the system is designed, implemented, and operated have led us to the identification of four mandatory non-functional properties for safe autonomy (NFP). Their considerations make the system progress towards safe-by-design architectures as well as ones that are more robust, easier to design, and easier to verify and validate.

Their considerations also traduce into providing evidence that the abstract system model fosters the system's nominal behaviours, guarantees its critical system goals and

manages any deviations from them. Their goals, coordination and composition may only result of a global model-based analysis with a symbolic and conservative representation both human and machine-readable. Therefore, the whole design process requires a more or less exhaustive analysis to identify all kind of harmful events and their possible effects.

Then, the implementation of DIR (Detection, Isolation, Recovery) mechanisms contributes to detect those symptoms and mitigate them as it keeps or brings the system back to a set of trustworthy states. Moving them from correctness at design time to run-time autonomic management may reduce existing gap between critical and best-effort systems engineering on autonomous systems. They require not only cutting-edge theory but also to find adequate trade-offs between quality of control and performance. Proposing a variety of DIR-type adaptive processes within the supervised system involve complex decision methods by keeping critical goals and plan best-effort goals according to resource availability. Immediate downsides are the ability to react promptly for timely recovery or the creation possible conflicts between the different control loops.

Shifting the mitigation and assurance to run-time to cope with uncertainty and context complexity constitutes a critical change in System Engineering. This turning point involves new types of theory and architecture to handle these system-level functionalities and new kinds of architecture coping with the different NFP (e.g. large distributed evolvable autonomous systems with non-predicable dynamically changing environments).

This proposed computational model for autonomous systems suggested in [135] can provide a basis for studying model-based autonomous system design. However, such abstract model-based architecture may appear difficult to grasp in its entirety at the first sight since the design and run-time operations have become more complex. Nonetheless, this additional complexity is necessary to appropriately integrate the results from the environment analysis and symbolic representation that contribute to tracking the possible trade-offs and evidence. Design flows involving these solutions are not yet mature and rigorous in the automotive industry, but some initiatives start to provide the different pieces to build up such a solution.

This chapter focuses on the study of model-based autonomous system design and investigates the design flows and initiatives grasping with the previous computational model.

Section 3.2 details how System Engineering allows coping with the complexity of designing and integrating the different concerns and quality properties using views, in particular, safety. We also survey self-adaptive software solutions in the literature that contribute to performing safe operations from a quality assurance perspective.

Section 3.3 formulates and deriving safety constraints as conditions or risk measures that cover all relevant hazardous events in regard to the vehicle behaviour. The scope of the work turn around behavioural safety for a level 4-5 Autonomous Vehicle so that functional safety is not directly addressed but remains acknowledged for the later technical design of the supervisor. An abstract model of the vehicle is proposed using the Arcadia methodology involving the expected operational functionalities from the vehicle, the system, the environment and the actors (i.e. stakeholders, actors, regulators, recommendations).

Section 3.4 reformulates the solution independent from any technical solution as logical and technical requirements to be met in the framework proposed later in this thesis.

Section 3.5 summarizes our findings.

3.2 AV Safety and Systems Engineering

This section investigates how safety in AV is managed in SE. We pursue the finding and analysis by exploring research works and solutions in the different domains limited to QoS regarding one or more properties.

System Design Complexity in Autonomy

The combination of multiple functions having different and complementary capabilities enables the emergence of Autonomous Vehicles. Their deployment is limited by the level of complexity they represent together with the challenges encountered in real environments with strong safety concerns. Thus a major concern prior to massive deployment is on how to ensure the safety of autonomous vehicles despite likely internal (e.g. malfunctions) and external (e.g. aggressive behaviours) disturbance they might undergo.

System Engineering has partially contributed to respond to the complexity of autonomy, context, system architecture and knowledge. Autonomy precises two challenges in both System Engineering methods and Knowledge representations.

In the first place, System Engineering offers to manage complexity of the AV functional and non-functional decomposition. The System Theory is a way to cope with complexity alongside analytic reduction and statistics. Systematic approach to manage complexity with appearance of new defined needs: the necessity to provide holistic and systematic engineering (through methodology, tools, processes and definition). It offers traceability and flexibility of the architecture (systems, components, features) in the life cycle. In addition, it may provide top-down, bottom-up and middle-out possibilities to enhance and integrate learning processes, updates, new regulations facilitating the acquisition and adaptability of the knowledge mandatory for autonomous systems.

"Safe-by-Design" Architectures towards run-time Safety Representation and Assurance

Despite the fact that the automotive industry practices a bottom-up approach to the engineering of autonomous driving systems focusing on functions and technologies first, architectures have been including those concerns and architecting towards these properties. They capture both AV functional decomposition, AV non-functional decomposition (i.e. safety) and AV contextual decomposition. The compliance to them in standards and research towards more self-awareness and safety in AV have successively been introduced at different scale of adaptation to the system.

Hybrid architecture In the first place, the structure of the framework of Albus on 4D/RCS [10] introduces a hybrid control architecture being both deliberative (i.e. actions based on reasoning and planning) and reactive (i.e. fast actions based on direct and simple condition on feedback). On the one hand, the framework structure ensures reasoning and planning processes based on goals and priorities of the decision entities within a control hierarchy. On the other hand, reactive loops that provide a faster and controlled response are introduced at each level of the control hierarchy, and can locally modify planned actions to adapt to new events. This approach is close in its concept and struc-

ture composition to the autonomic computing paradigm [77] introduced by IBM, with a hierarchy of autonomic orchestrating managers in IT infrastructure.

Degraded states and modes In their work to define robust system architectures, Tas et al. [43] emphasize the relation between architectural design and the use of degraded operation modes, introduced by [101]. The authors refer to the design of a functional and layered system architecture as one of the main focus of autonomous driving to deal with safety-critical challenges in systems engineering. Moreover, they suggest that the use of an effective monitoring system is necessary to give proper feedback to the vehicle about its state and to allow to take well-adapted decisions. Modes and graceful degradations continue to be investigated in the literature [50].

Self-awareness on system's abilities and limitations In the matter of monitoring, Reschka et al. [123] report the needs to provide a permanent online monitoring of vehicle capabilities, as well an adequate modelling tool to support appropriate and safe decisions. Accordingly, the framework perceives the current performance of the system during operation, by knowing the range of actions of the system and its limitations. With respect to ISO26262 standard [1], the authors propose the use of ability and skill graphs to design the functional architecture of AV. Those skills reflect the performance feedback of the system and then are used for system self-perception. Moreover, Colwell et al. [38] also introduce degradation for subsystem functionalities and ODD restriction to grasp the limit of the safe system operation.

Constructivist and layered evolvable architecture

Regarding the long-term evolution of AV in systems engineering, Behere et al. [25] overview the key functional components and orientations needed for autonomous driving, to establish a layered evolvable functional architecture. They report the necessity of constructivist architectures managed by Artificial Intelligence to tackle the limitations of current engineering practices for scaling to more complex systems. These so-called constructivist architectures introduce a fundamental shift from manually designed to self-organizing architectures that evolve at run-time. The current challenge resides in the possibility of run-time reasoning and run-time verification of the desired properties as safety constraints. The design and implementation of such architectures will involve paradigms and technologies from non-automotive domains. For example, the use of reflective intelligent control systems based on a high-level supervisor on the functional layer that will allow the monitoring and change of its behaviour for better adaptations.

Those approaches have been introduced progressively in the automotive domain in order to propose solutions for well-adapted decisions; to apprehend the range and limitation of actions of the system; to evolve the structure to provide better adaptation behaviours; and finally, to integrate non-functional dimensions. Researchers have started to investigate the application of such models in run-time environments combining safe-by-design and run-time mitigation across the system and its functionalities. This software change beyond typical software evolution approaches which has led to the vision of self-adaptive software.

Ensuring Safe Operations with Self-adaptive Software

Several run-time models that have been experienced and surveyed to cope with uncertainty have been proposed [67, 110]. They are presented as an appropriate solution to

cope with the concerns of safety in AV emphasized previously. This type of architecture involves constructivist or evolvable or dynamic behaviours or systems and are referred as self-adaptive architecture. They act as an orchestration solution that can support acceptable trade-off to pursue the assurance of some non-functional quality such as safety in evolvable and reconfigurable system [25].

Composable and flexible systems

Introduction to microservice

Regarding software architecture, modularization and abstraction have been the first stages to break down the complexity of systems in programs and architectures. However, according to Dragoni et al. [47], those stages still result into the creation of monoliths, where monolithic applications are generally built as a single unit with functionalities spread in modules and components that are not independently executable. The final product and code result in difficulties for scalability, maintainability and evolvability. In order to cope with these considerations it suggests the use of micro-services as an architectural style in software development to create a single application as a suite of small composable services. In order to confront the previous issues, the components are forced to only implement one functionality that results in the expression of one capability. Inspired by service oriented computing, Martin Fowler et al. [56] formalize the terminology of this current movement in software development as alternative to monolithic style applications. The approach suggests to isolate all business capabilities of the systems into simple and small specialized services similar to components but decoupled enough to be independently deployable by an automated deployment machinery. A micro-service results to be an independent functional process that express one capability, also defined as cohesive [47], interacts with messages and owns its domain logic locally.

Microservice in self-adaptive or safety-critical systems

[125] is an application of micro-service architecture for aerial unmanned vehicle. The intended approach in its ability to discover new services at run-time and provide an automated deployment machinery requires micro-services to be auto-descriptive by semantically describing their business capability (behaviour), goal (system), logic, structure (component) and interface (communication). The advantages of using microservices in designing is their capacity to fail small. Not only they allow smart adaptation based on observation of such faults, they also make the fault-tolerance mechanisms easier to bound.

3.3 Modeling for Safety Assessment with MBSE in AV

Application scope

Current work in the domain have proposed relevant approaches for the refinement, integration and enforcement of AV safety. Considering safety as a dynamic control problem proposed by STPA/STAMP in [98, 99] has shown promising applications and results in [19, 42, 90, 119], and a complementary approach in [4, 128, 148] towards the AV standards and the traditional failure analysis for hazards coverage [55, 139]. However, most conclusions of these works insist on the difficulty to provide a scalable automation for

Hazard	Through
identification	analysis
elimination	design
control	management

Table 3.1 Mitigation of hazards in safety engineering

production-ready analysis processes to the AV context and a flexible enforcement of these safety constraints in parallel. Consequently, they only result in a partially-holistic view of the ADS restricting its ability to argue (e.g. on decisions, observation restrictions). The flexibility and manageability of performing safety assurance at run-time in AV require expertise and related work from others domains than automotive to address the presented concerns.

Control structure for safety analysis towards reduce residual risks

The combination with the conventional ISO26262 analysis (FMEA, HARA, Safety case) and STPA allows to perform safety analysis in order to provide a large scope of detection of hazard and mitigation of the residual risk. The goal remains the same as it is to create systems that integrates easily the safety requirements of any types that may have impact on design or can only be mitigated during system operation.

The primary concern of System Safety is the management of hazards as described in Table 3.1. It presents the 3 ways of managing hazards leading to the reduction of the risk to a manageable and acceptable level.

Composition of the Safety Analysis and Assessment in AV architecture

In our approach, we propose to apply to integrate the awareness of the hazard analysis in the system model via some machine-understandable knowledge. We also consider the strategies of the hazard management in the form of control loops. We propose to call these loops the "safety assessment processes" as they constitute workflows involving different functions like monitoring, classification, evaluation, planning or mitigation. To determine the rightful allocation between the different functions, dataflows and each workflow, we propose to perform a combined safety analysis with STPA/STAMP and ISO that were both presented in Section 2.3 (see page 60).

The combination of both STPA/STAMP and ISO hazard analysis have been performed and shown an extended coverage of the hazard [4]. However, such application requires to consider a specific ODD and reveals to be an iterative process with continuous refinement. Experiments [42] have been applying it to AV and formalized this context-dependant analysis method using a taxonomy (operational keyword to generate a sentence creating the different scenarios).

In the first place, we propose to start the hazard analysis at the high-level representation of our architecture with the behavior competencies and regulations. Consequently, we start by identifying some general use cases involving the vehicle competencies (e.g. perceive and react near a pedestrian crossing). Then, we determine the hazards, the control structure of the system and identify the unsafe control actions to construct the hazardous scenarios. Finally, safety constraints can be identified and allocated to the different systems or components or functions involved. This integration may result in the creation of a new version of the system architecture (i.e. design constraints). The constraints that

suggest some run-time management through control loops are called “safety assessment processes” (SAP) in our research. They will be the focus of our work as we want them to be deployed when the context requires it.

This process is then iteratively performed by applying it these new version but also when the system architecture is continuously refined and more defined in-depth (bottom-up).

Background on MBSE

MBSE Methodology and Techniques

The previous point were dealing with the process of designing and maintaining architecture. However, what language or formalization can be used to describe software architecture?

UML (Unified Modeling Language) is one of object-oriented solutions used in software modeling and design. UML is a language (of object modeling) and therefore proposes a notation and a semantics associated with this language (i.e. models), but no process methodology (i.e. an approach proposing a sequence of steps and activities that lead to the resolution of a problem). Therefore, UML is not a method. UML unifies object methods with a design process that was strongly influenced by its intended use in object programming. UML has a whole approach (i.e. covering the entire development cycle: analysis, design and implementation) that is object-oriented (and not functional): the system is broken down into collaborating objects (rather than tasks broken down into functions that are easier to perform). However, its application is not particularly adapted to modeling complex systems, and suited to system engineering in the first place.

The Architecture View Model designed by Philippe Kruchten, also called 4+1 view model represents the functional and non-functional requirements of software-intensive systems. They are represented in the different views that offer and describe the system from the viewpoints of different stakeholders (e.g. end-users, developers, system engineer, and standards). The four views of the model consist of the logical, development, process and physical views. Each one achieve a set of specific concerns regarding the system in addition to a use case or scenario view that describes the sequences of interactions between objects and between processes involving the system and guiding the other views. The logical view is high-level and focuses on abstraction and encapsulation, it models the main elements and mechanisms of the system. It identifies the elements of the domain, as well as the relationships and interactions between these elements "notions of classes and relationships". The component view expresses the physical perspective of the code organization in terms of modules, components and especially the concepts of the language or implementation environment. From this perspective, the architect is mainly concerned with the aspects of code management, compilation order, reuse, integration and other pure development constraints. To represent this perspective, UML provides adapted concepts such as modules, components, dependencies, interface. The process view is very important in multitasking environments; it expresses the perspective on concurrent and parallel activities. The deployment view expresses the distribution of the system through a network of computers and processing logic nodes. This view is particularly useful for describing the distribution of a distributed system.

SysML is the new modeling language defined by the OMG. It can be seen as an extension of UML for modelling a broad spectrum of complex systems. It is based on UML and replaces class and object modeling with block modeling for a vocabulary more

adapted to System Engineering. A block includes any software, hardware, data, process, and even people management concept

Much more than a simple modeling tool, Capella is a model-based engineering solution that has been successfully deployed in a wide variety of industrial contexts. Based on graphical modeling, it provides system, software and hardware architects with rich methodological advice based on Arcadia, a complete model-based engineering method. The DSML Arcadia/Capella is based on the UML/SysML and NAF standards, and shares many concepts with these languages. It is the result of an iterative definition process driven by software systems and architects working in a wide range of business areas (transport, avionics, space, radar, etc.). Many industrial companies, such as Airbus, Areva, Thales, Continental and Renault are currently interested in using Capella and running pilot modelling projects with this tool. It exists because Arcadia allows you to:

- Ensure collaboration at the engineering level by sharing the same reference architecture.
- Manage the complexity of systems and architectures.
- Define the best optimal architectures through compromise analysis.
- Manage different levels of engineering and traceability through automated transition and refinement of information.

The couple Capella/Arcadia associates both the tool and the language: referring to MBSE's three well-known pillars, one could say that ARCADIA provides both a modeling language and a modeling approach, and that Capella knows the language and method perfectly.

With a similar scope around adaptation, EUREMA is an integrated MDE approach as it rigorously and consistently uses models for engineering feedback loops [150]. The models are used (1) to represent the adaptable software to achieve self-adaptation as proposed by the idea of models at run-time (Models@run.time [27]), as well as to design (2) individual adaptation activities for feedback loops, (3) feedback loops as a whole, and (4) coordination between these loops. Finally, (5) these models are used throughout the life cycle of the self-adaptive system, i.e. to specify, execute and evolve feedback loops with their adaptation activities. This approach introduces a model-driven architecture approach that guide the design and development of complex and evolving systems while claiming to guarantee the portability, the interoperability and the reusability of the final system.

Modeling with Arcadia/Capella

Introduction to Arcadia/Capella

ARCADIA (ARChitecture Analysis and Design Integrated Approach) is a model-based engineering method for the architectural design of systems, hardware and software. It was developed by Thales between 2005 and 2010 through an iterative process involving operational architects from all Thales businesses (transport, avionics, space, radar, etc.). It applies a structured approach over successive engineering phases that establishes a clear separation between requirements (operational needs analysis and system needs analysis) and solutions (logical and physical architectures), in accordance with the IEEE 1220 standard. ARCADIA recommends three mandatory interdependent activities at the

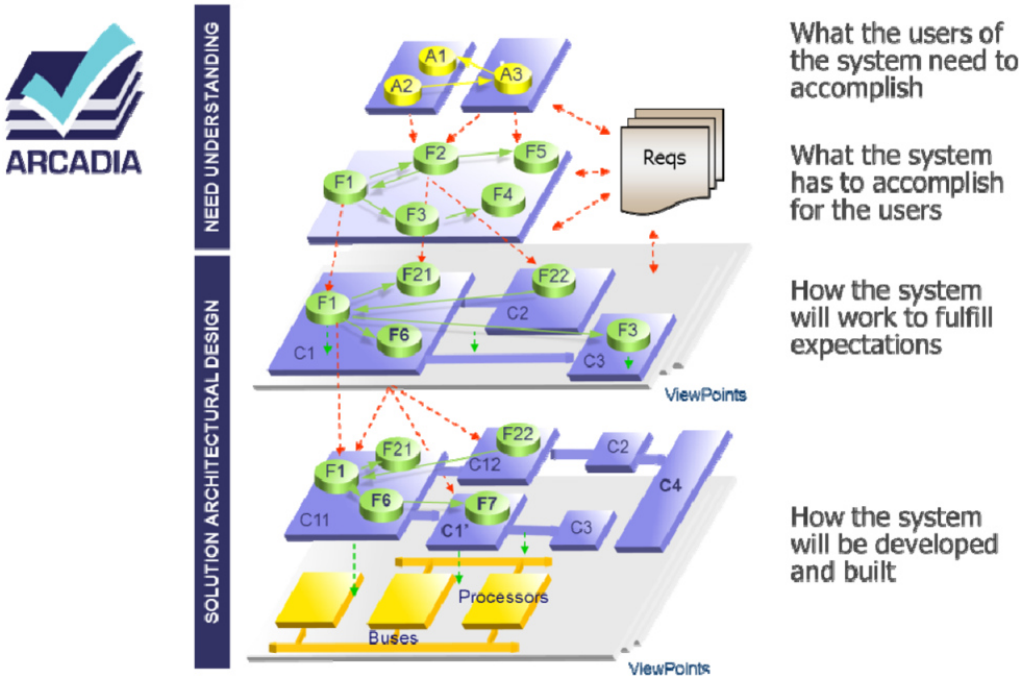


Figure 3.1 The main engineering levels of Arcadia and transition steps [126]

same level of importance: Needs analysis and modeling; Construction of architectures and validation; and Requirements engineering.

Arcadia DSML (Domain-Specific Modeling Language) is based on UML/SysML standards and NAF and shares many concepts with these languages. But a modeling language was preferred in order to facilitate ownership by all parties stakeholders. Arcadia is mainly based on functional analysis, then the allocation of functions to the components. The richness of Arcadia DSML is comparable to SysML with about ten types of diagrams: data flow diagrams, scenario diagrams, state and mode diagrams, component distribution diagrams, component distribution diagrams, component distribution diagrams functional distribution, etc. There are also diagrams available from ARCADIA at different levels that are detailed in Figure 3.1.

The Capella tool provides the different set of tools to carry out the system modelling and automated transition steps from one level to the next providing traceability in the refinement of the different element of the system model architecture.

It is noticeable that the Arcadia method is presented in a top-down manner by nature. However, the representation can also be bottom-up if we start from an existing system for instance. In the next chapters, our example will assume the extension of the ADCC architecture. For this purpose, the Arcadia methodology relates more about levels providing key representations objectives rather than phases or steps. In addition, all the architectural levels are not stated as mandatory. Since the methodology aims to guide the analysis and modelling processes, some levels may be skipped depending the system complexity and model expectations (e.g. operational analysis, logical architecture, EPSB are optional).

Operational Analysis

The Operational Analysis represents the highest level of the Arcadia method. It focuses on answering the question “what the users of the future system need to accomplish?”. At this level, the future system is not yet represented as a modeling element but the identification

of the needs and objectives of the system start in this step. This level constitutes the first representation of the system environment by modelling the jobs of future users as activities, roles to fulfill while precising the operational conditions of the system. A series of activities and of interactions constitutes a process and contributes toward an operational capability. The creation of models and scenarios representing the different elements allows to check the adequacy of the system to these operational needs.

This level of Operational Analysis consists in:

1. Capture and formalize the operational needs from the different stakeholders.
2. Define what the vehicle (System) need to accomplish and how the other actors contribute, influence or impact.
3. Also identify the different entities in each separate domain (entities, actors, roles, activities and concepts).

System Analysis

The System Analysis places the system as the central element of the dataflow representations in interaction with the external actors. It focuses on answering the question "what the system must accomplish for the users?". The System Analysis involves the identification of system capabilities and functions that satisfy the operational needs defined in the previous level. At this point, the system is still considered as a black-box as no internal structure should be decided yet. Only function and communication exchanges may be allocated during this external functional analysis. Consequently, this analysis consists in refining the system functions with the decomposition of the top-level functions and adding the constraints of non-functional properties. For these aspects, these diagrams on Capella provide rich mechanisms for managing complexity: simplified links calculated between high-level functions, categorization of exchanges, etc.

Functional chain can also be displayed as highlighted paths. They represents a specific path involving a sequence of the system functions and exchanges and are relevant for assigning constraints (latency, criticality, etc.), as well as organizing tests on a specific system features.

This level of System Analysis consists in:

- Define what the system have to accomplish for the users.
- Identify the boundary of the black-box system, consolidate the requirements (sourced from the actors).
- Model functional data-flows and dynamic behaviours.

Logical Architecture

The Logical Architecture aims to identify the different logical components that form the component structure inside the system. It focuses on answering the question "how the system will work to fulfill expectations?". This level provides tools to represent the relations between components and their content, independently of any considerations of technology or implementation.

The previous model described in the System Analysis can be imported via an automated transition to Logical Architecture model. Thus, it keeps all the previous definition

and description of the top-level function and exchanges. On this basis, the Logical Architecture carries out the internal functional analysis where all sub-functions subsumed by the previously identified top-level functions need to be identified and allocated to a specific logical component while keeping tracks of the integration of the non-functional constraints .

This level of Logical Architecture consists in:

- Provide a white-box vision of the system identifying how the system is fulfilling the expectations.
- Propose a first step for trade off-analysis.

Physical Architecture

This level of Physical Architecture possesses the same objective than the Logical Analysis but defines the final architecture of the system providing information on how the system will be built. This level reflects the technical choices and defines the different components (software and hardware).

This level of Physical Architecture consists in:

- How the system will be developed and built.
- Software vs. hardware allocation, specification of interfaces, deployment configurations, trade-off analysis.

EPBS and integration contracts

An additional level EPBS (End Product Breakdown Structure) is available to define the "*conditions that each component must fulfill to satisfy the architecture design constraints and limitations, established in the previous phases*" [126]. This last level answers to the question "what is expected from the provider of each component" and is usually used with sub-contracting.

Integrating dynamic safety analysis and management in Arcadia/Capella

Overview of the Composed Methodology

We propose to design a methodology based on the Arcadia to appropriately represents the safety-related adaptive processes by synchronizing the step of the Arcadia methodology with the STPA analysis for the set of defined scenarii. For each step of Arcadia, we propose to associate an STPA analysis for any representation that contains the equivalent of a control loop.

Process of Integration for Safety Analysis

Operational Analysis In the first place, it is necessary to base our analysis on what the behavioural requirements the AV system and vehicle needs to accomplish and the source of these requirements. In fact, multiple actors are the sources of requirements that can highly impact how the vehicle need to behave, how it is design and implemented. Standards, recommendations, guidelines, state-of-the-art, car manufacturer, regulators,

renting or end customers impose diverses requirements to the AV system that can be both functional or dysfunctional coping with totally distinct disciplines (e.g. functions, safety, ethics, social acceptance, security, etc).

The operational analysis of the Arcadia methodology contributes to capturing those operational needs from the different stakeholders. It defines what the vehicle need to accomplish and how the other actors contribute, influence or impact. Indeed, we model the whole process of actors and requirements that identify the different entities in each separate domain with a specific vocabulary of entities, actors, roles, activities and concepts. The functional part definition of the system takes as inputs the different behavioural competencies, OEDR, ODD, DDT and requirements from different shareholders available in the literature [142].

The actors of the non-functional part also takes place as they require specific operational capabilities to manage properly conditions or risk measures that cover all relevant hazardous events in regard to the vehicle behaviour. The following figures contribute to illustrate how high-level capabilities, entities, actors, interactions, activities and process. It is important to notice that the creation of diagrams contributes to refine step-by-step the model of the system. The next figures are views (or windows) to observe the semantic model that have been incrementally built using the Capella tool and following the Arcadia methodology.

Figure 3.2 defines the operational entities and the capabilities involving both the functional and the safety concerns. This first diagram "Operational Capabilities Blank" that helps defining the domain or business around the product is created empty by default (blank). At this level, we start to represent the "real" needs of the various shareholders without the architecture, design or technical implementation in mind. We describe the model with Operational Capabilities (OC), Operational Entities and Operational Actors (OA) and the relation between them. For autonomous vehicle, shareholders often decompose the needs between functional and non-functional concerns. We also propose to distinguish functional and behavioral safety strategies from the functional part of driving to initiate the technical design of some sort of the core of the supervisor (both in the model and in the future physical architecture). For this purpose, we have identify some functional oriented capabilities at the top-half of the Figure 3.2 with the driving automation and some non-functional strategies at the bottom-half (safety and social dimension). In addition, we have performed a additional step in the definition by refining each OC into some subsumed OCs. To illustrate, the DDTs are represented as OC as they can not be translated into activities and later as functions since they remain high-level definition of general service for operational objectives. In our case, their description and traceability is performed with some involved activities and operational processes (e.g. SENSE, PLAN, ACT) that other diagrams detail (e.g. Figure 3.4).

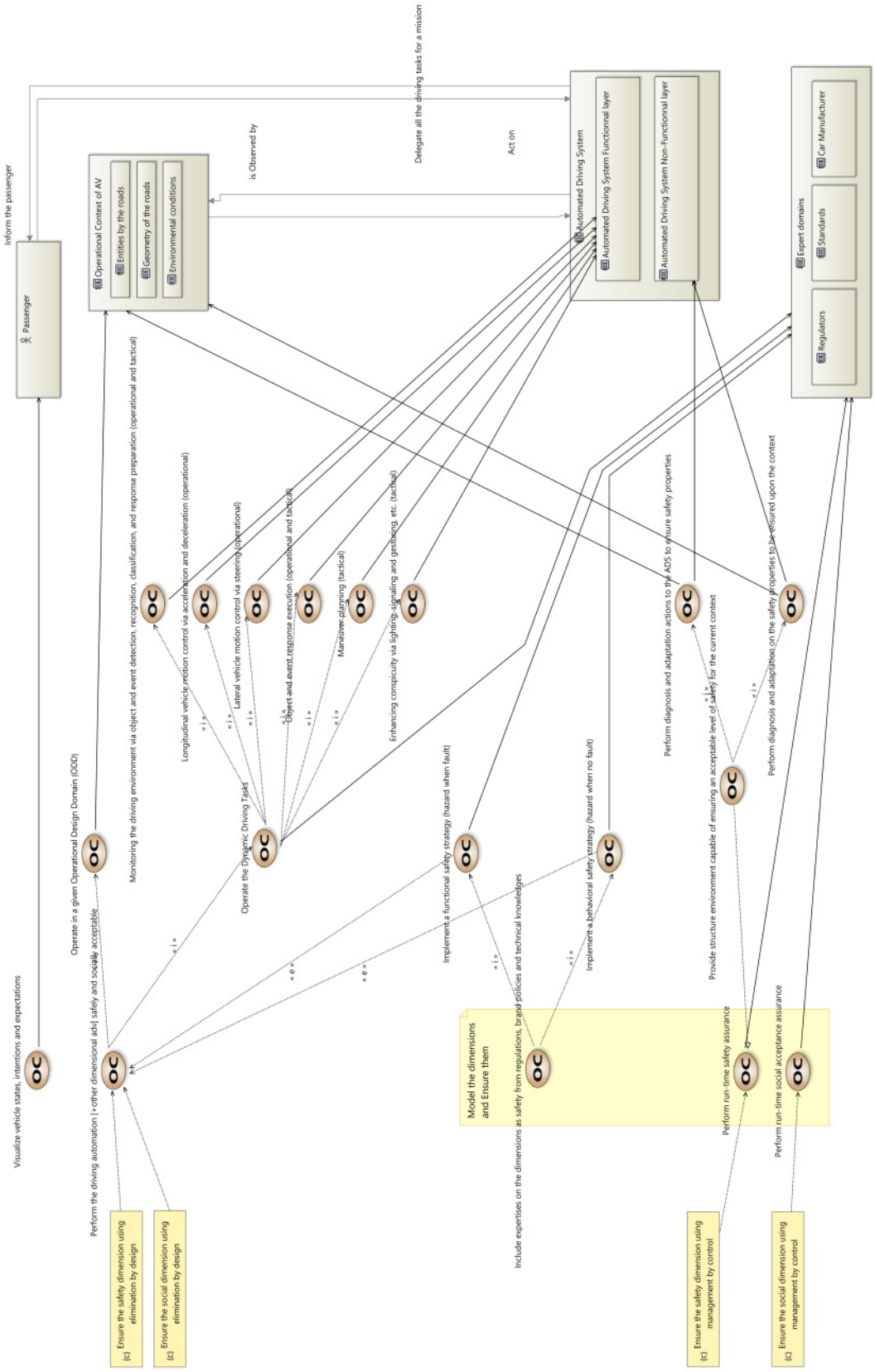


Figure 3.2 [OCB] Operational Capabilities

Figure 3.3 defines the architecture between the operational activities and the interactions between the different actors to grasp the operational context of the system. The

activities (in yellow) correspond to the different process steps carried out in order to reach a precise objective for the entities. They might need to use the future system in order to do so, or the system might use it (e.g. Provide ODD definition). For example, we allocated to the operational context of the autonomous vehicle as the surrounding environment it may perceive (external observations) as well as its internal observations (system states). These observation are then consumed by the driving automation activity of the functional layer. This diagram also represents activities related to the ODD as it is a major source of the environment representation and formalization of needs from the different stakeholders with expressions (e.g. rules or contracts to require behaviour X in situation Y, or to prevent X in Y). The construction (structure) and definition (content) of the ODD play a major role in our definition as well as the understanding of the current situations in which the system is involved. Finally, additional activities are also represented in interaction with the previous activities to address the non-functional concerns and strategies to use.

To define with more detail the activities and the respective interactions for each capability (OC), Capella proposes to create an OAIB view (Operational Analysis Interaction Blank). Figure 3.4 and Figure 3.5 are OAIB diagrams. They respectively represent the capabilities of "*Operate in a given Operational Design Domain (ODD)*" and "*Operate the Dynamic Driving Tasks*".

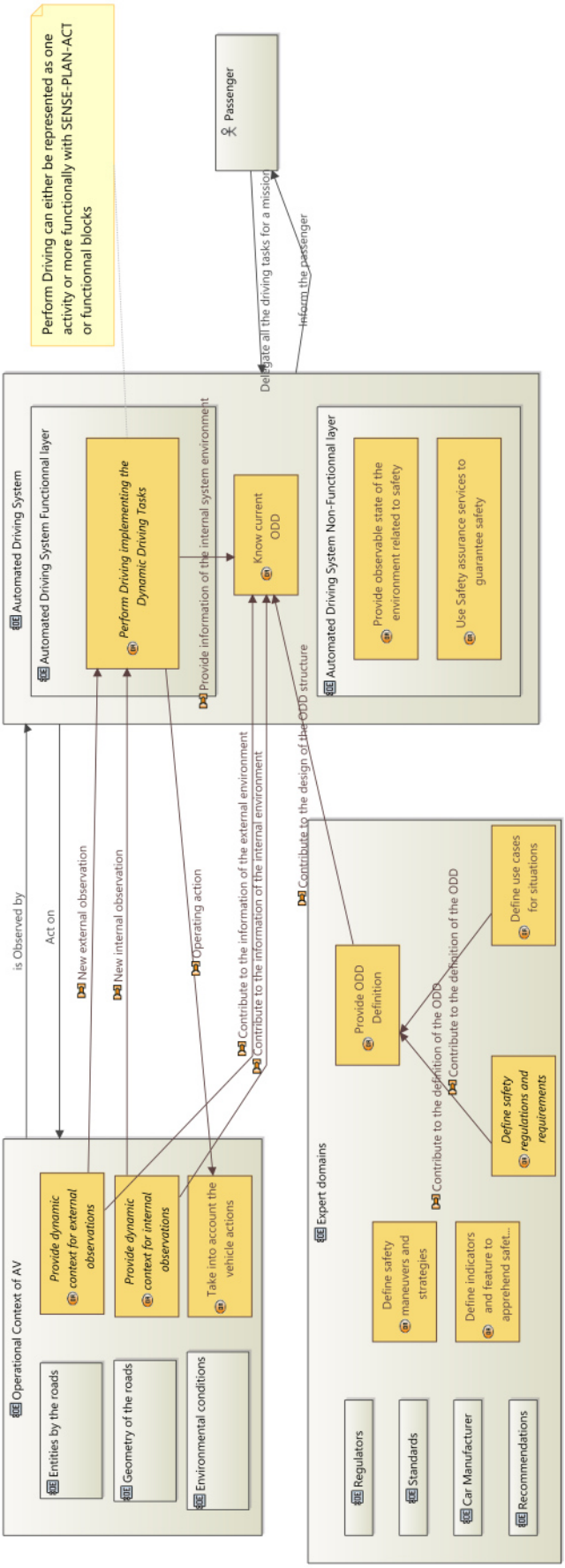


Figure 3.3 [OAB] Operational Context

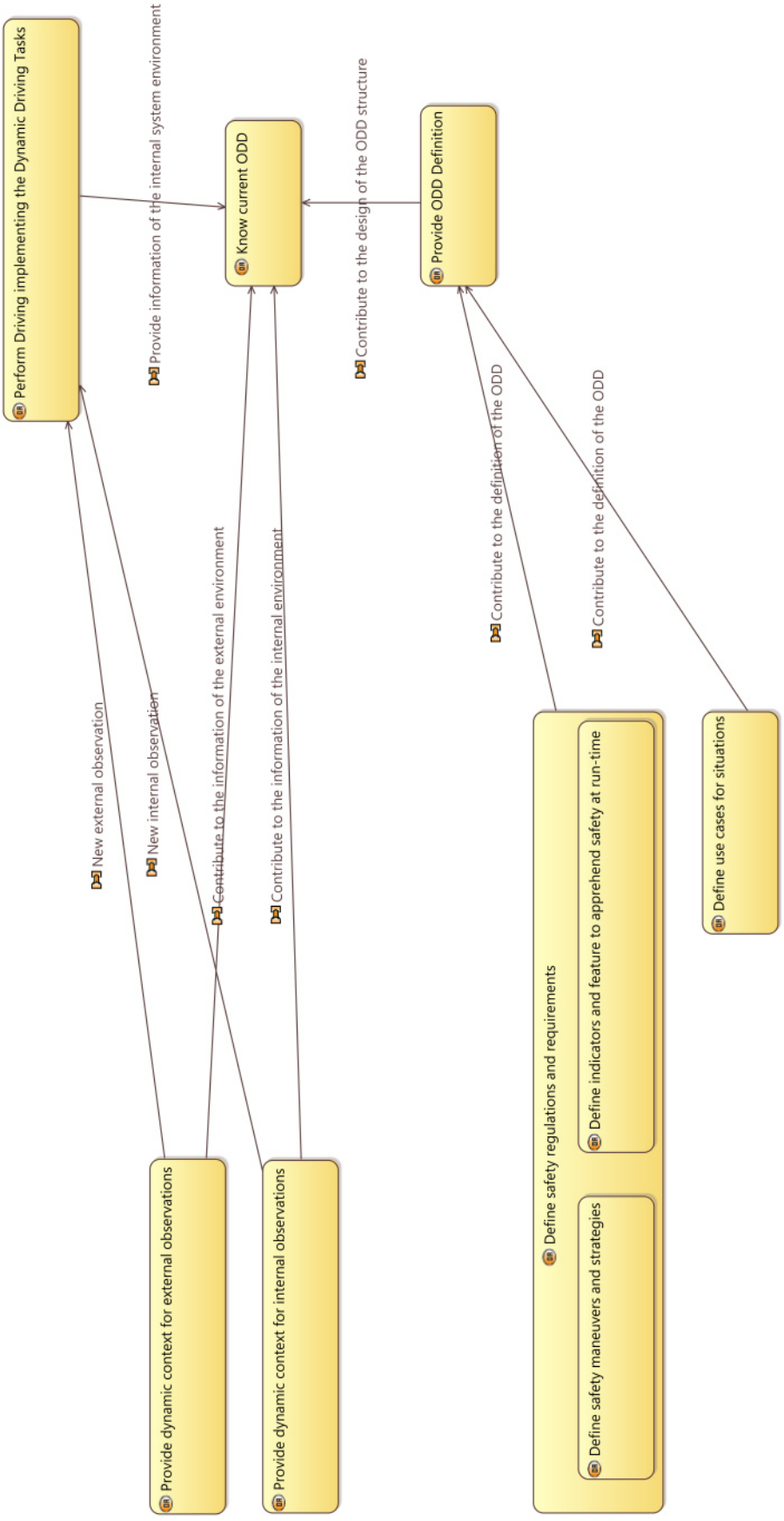


Figure 3.4 [OAIB] Capability “Operate in a given Operational Design Domain (ODD)”

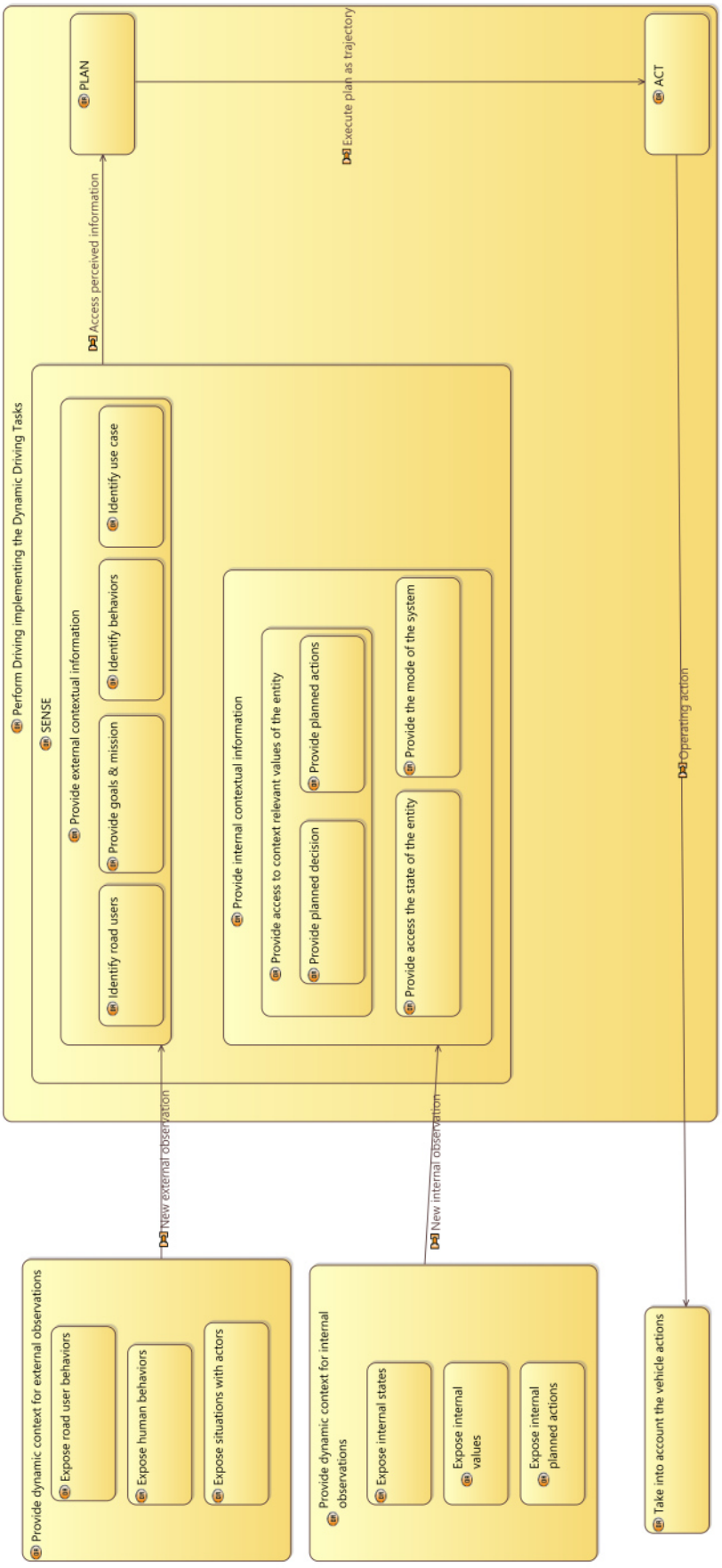


Figure 3.5 [OAIB] Capability “Operate the Dynamic Driving Tasks”

System Analysis The role of this next level is to identify the different functions or system services necessary to its users and specify them with the possible constraint from the non-functional properties. First, we have started with a definition of the system as a black-box and creating functions that realize the operational activities from the previous level. Figure 3.6 is a synthesis view of the system functions we have identified as parent functions: to perform the driving automation, to obtain access to the ADCC resources (Layer 1), to supervise them with specific loops of adaptation (Layer 2), to perform run-time adaptions to fit the ODD and safety concerns (Layer 3), to store this information for possible update or learning purposes with Knowledge Bases (KB), and finally to display some system internal information. At this step, we have already made some choices regarding the design and how we wanted to structure and organize the control loops. Figure 3.7 provides further details about the design choices for the architecture with the introduction of MAPE-K loops with a external knowledge base and service orientation.

The next levels that are Logical architecture and Physical Architecture will be presented later in the Chapter 5 with the implementation of the framework. They are in charge of incorporating a white-box analysis and allocation management that integrate the design choice and mechanisms we present in the next chapter.

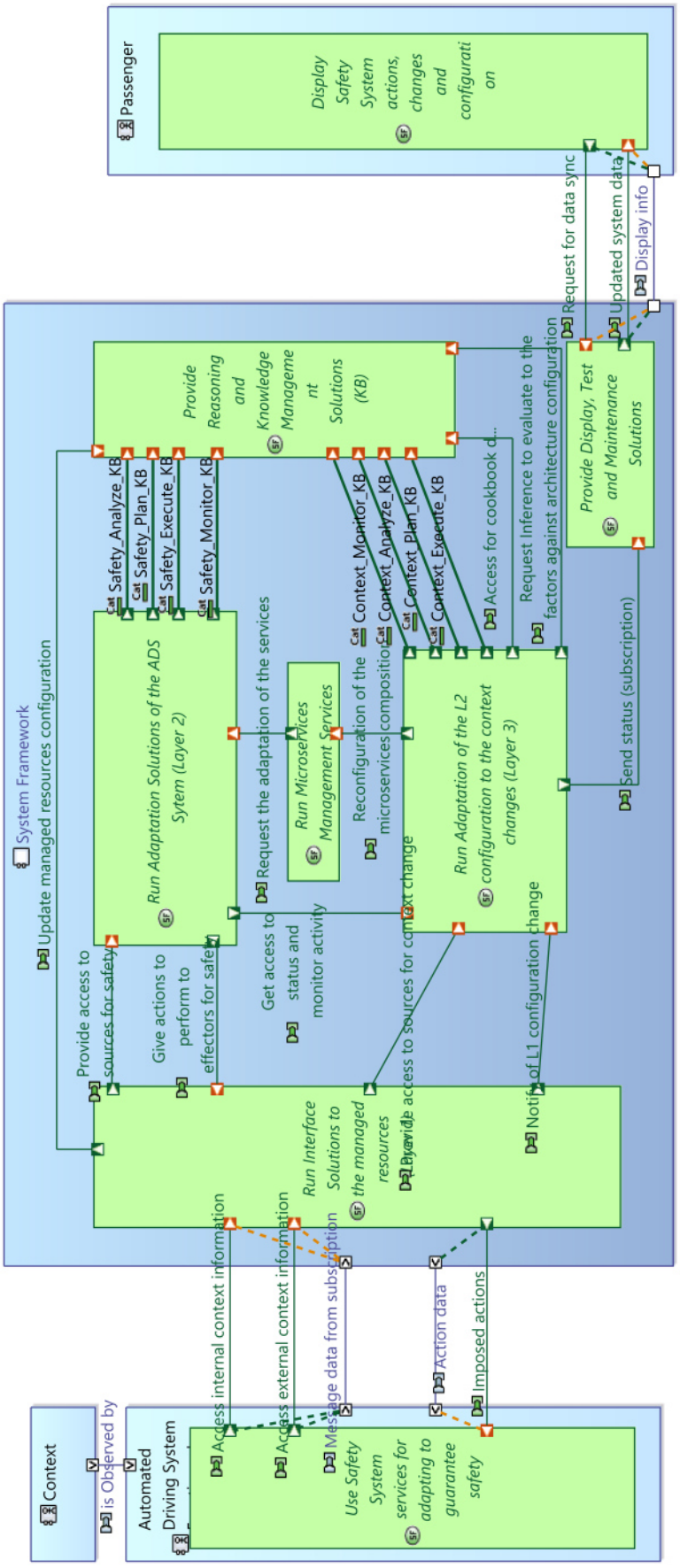


Figure 3.6 [SAB] Top Level System Overview

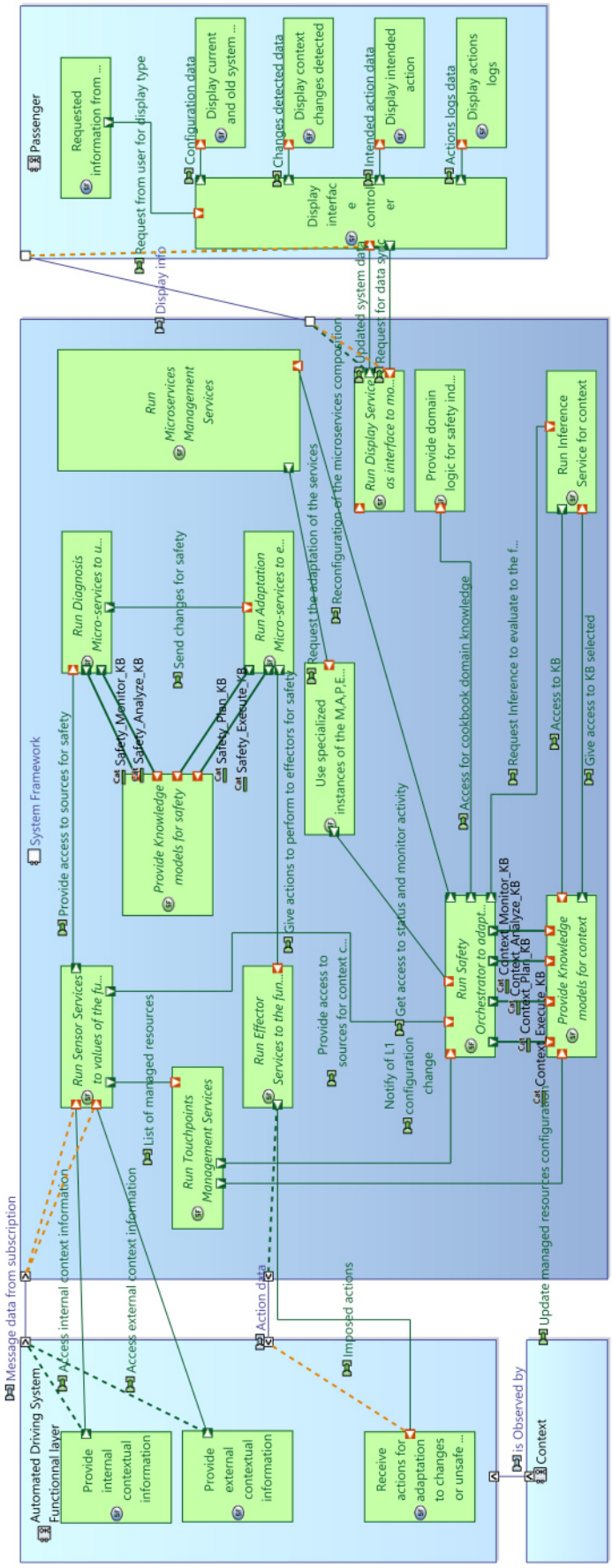


Figure 3.7 [SAB] High Level System Overview integrating the MAPE-K loop as types of function

3.4 Problem reformulation

We have proposed and experienced a way to model an AV architecture with safety requirements in a separate viewpoint than the functionalities of the AV. These requirements are affecting both the design and the operation of the system. While one is integrated in the current design, the assurance of the other safety requirements as constraints to monitor and control remains.

Rationales for potential solutions

Such approach makes new needs appearing.

How the transition between the functional view to the logical and physical architecture can be achieved? We need to determine what types of architectural solutions can support the safety management loops and mitigation within the system. In addition, we also need to address the management of the constraints resulting from the AV context analysis. Determining the different operational contexts needs to be open for iterative and thorough safety analysis. Finally, we need to guarantee the constraints as defined distributed autonomic processes.

How to assess the safety that remain behavioural? Hence, we want to ensure the safety and similar system attributes than in the design by managing the complexity of the systems for observability, traceability and flexibility. It may not be possible to solve by only tweaking profile or configuration: it is required to provide guarantees at any system or sub-system levels of adaptation.

Potential technical solutions in logical and physical view

Use of Abstraction for ODD and Composition of the Scene In the effort to reduce the complexity, we propose as a first step to design hierarchies in the ODD with the use of the abstraction to subsume some behaviours.

As well, we are looking to treat specific situations as sets of simple situations which can be superimposed. Hence, small set of simple, primary situations is proposed and described from which all more complicated situations can be obtained.

We propose to elevate the treatment of the other ‘objects’ on the road – notably other vehicles – to equal level with the system vehicle intention. Using the “lane” concept for the system vehicle’s intended path, the actual geometry encountered is relegated to a second level, but otherwise all the elements of the hierarchy are kept.

Self-managed Autonomic Layers The framework performs autonomic loops at two scale of the system to provide reflexivity and evolvability.

One layer provides behavioural adaptation loops driven by the assurance of safety properties of the capabilities deployed in the services. This way it perceives and acts directly on the managed resources (sensors and actuators) guaranteeing the satisfaction of the properties at run-time.

Another layer orchestrates the structural adaptation of these deployed services and properties to validate. By orchestrating these first-level loops at run-time, the system can provide the best composition of the system’s capabilities and assert relevant properties within the current context.

The system as a whole is constantly orchestrated by these autonomic loops at run-time to carry out the adequate composition of services to ensure behavioural safety upon the relevant context.

Context-dependent Semantic Model of Capabilities The introduced self-managed autonomic layers are driven by respective three knowledge bases as ontological models. One represents the relation between the safety requirements and capabilities with the system components. Another describes the context and represents the different use cases, situations, edges cases or combinations the vehicle can meet. Finally, the last knowledge base represents semi-formally the context-dependence relations between the system components and their applicable context. Regarding current practices, these knowledge bases consists in specifying the safety requirements derived from behavioural safety and decomposing the capabilities into context-specific services (allocation to the Operational Domain Design).

Micro-service Oriented Architectural Style The micro-services oriented reference architecture sustains a flexible environment of component-services and knowledge bases to manage its dynamic nature. Those component-services are implementing only functionalities strongly related to their concern, are independently deployable and are accessible via messages. In a such way, the use of micro-services eases the composability that serves the orchestration of the system at run-time. Transforming the task-oriented system architecture of autonomous vehicle into a capability-driven grained one offers more scalability and helps to achieve a more optimized engineering. The mentioned scalability involves the following factors: the size and complexity of the context-dependence relations of service and capabilities, the number of service-components and the manageability of these knowledge bases at conception.

3.5 Conclusions

In this chapter, we have explored the way to study and build L4-L5 AV using MBSE with the key functionalities and safety considerations as design and run-time mitigation. We focus on the key functionnalities for providing autonomy. Technological and technical solutions should not be pushed over them to provide manageable system complexity with sufficient level of safety coverage (quality and quantity) and NFP. It also enables the possibilities to perform trade off between the various alternatives, combinations, variants and implementations of the various architecture. Nonetheless, we are far from ensuring that the conditions are in place to develop rigorous design flows.

To conclude, the proposed computational model for autonomous systems can provide a basis for studying model-based autonomous system design. Nonetheless, we are far from ensuring that the conditions are in place to develop rigorous design flows.

CHAPTER 4

Framework for Safety in Autonomous Vehicles

"The control of an undertaking consists of seeing that everything is being carried out in accordance with the plan which has been adopted, the orders which have been given, and the principles which have been laid down. Its object is to point out mistakes in order that they may be rectified and prevented from recurring."

- Henri Fayol, *General and industrial management, 1918*

Contents

4.1 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 centered 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. in the STPA/STAMP method [98, 99] shows promising applications and results in [19, 42, 90, 119]. Complementary research investigate the trade-off with the traditional failure analysis for hazards coverage [55, 139] while others address the compatibility of the approach with the current AV standards in [4, 128, 148]. 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 assurance at run-time using adaptive behaviours has been shown to require a complex combination of AV system non-functional properties including observability, traceability, reconfigurability and scalability.

The main contribution presented in this chapter is the reference architecture that incorporates the notion of self-safety into an existing AV and copes with the previous identified non-functional properties. 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 chapter presents the fundations of the framework architecture, namely, requirements, functional design, functions involved in self-adaptation, abstractions, knowledge representation, and templates that connect and operate the safety control-loops as composable and agnostic processes.

Section 4.2 reminds 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 4.3 presents the reference architecture specificities and details the overall design of the control approach.

Section 4.4 details the implementation guidelines of the reference architecture applied to AV. We firstly update the system requirements to integrate the AV's non functional properties (NFP) in the overall architecture with a safety management system. We also detail how the related works integrate those required NFP in potential solutions and how they contribute to satisfy them through system capabilities. Our architectural framework is structured based on the Autonomic Computing paradigm, the adoption of the flexible microservice architectural style and the use of knowledge representations in the form of semantic and semi-formal models. The successive steps of combinations of design schemes detail the design of the main parts of the framework and their interactions to obtain a flexible, orchestrable and composable agnostic microservices architecture. Then, we detail how the knowledge bases are structured, their role and their allocation to drive the agnostic system.

Finally, in Section 4.5, a few conclusions are presented.

4.2 Requirement Analysis

Our approach aims to propose 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 AVsafe system (i.e. observability, traceability, reconfigurability and scalability), we propose to put emphasis on the following design principals:

1. 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.

2. 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.

3. 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 integration 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.

4. 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 requires 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.

5. 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 have 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.

4.3 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.

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 selected this architecture as it will be able to be enhanced 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 4.1 illustrates the reference architecture and details the perception, navigation and vehicle control mechanisms that are connected to the vehicle world via the sensors and acturators 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 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).

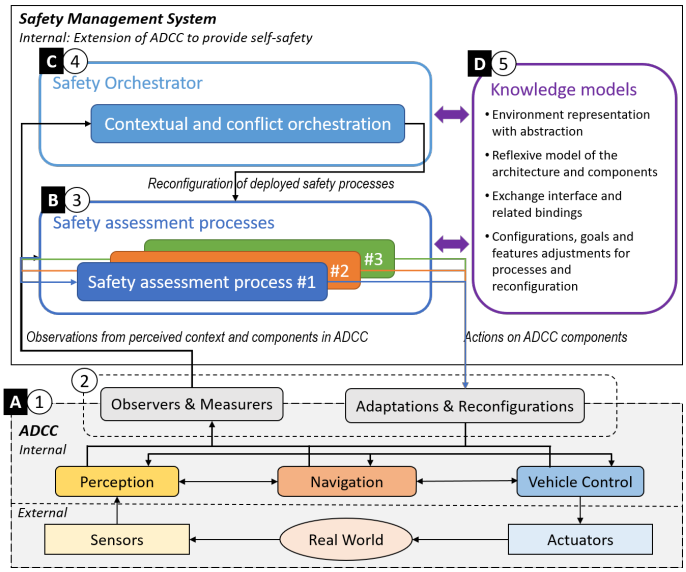


Figure 4.1 Reference architecture involving several levels of adaptations

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

Component Descriptions

The architecture of the Safety Management System (SMS) involves three main components extending a base architecture. These components are illustrated in 4.1 and are described as follows:

A. 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 maneuvers), 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, maneuvers or imposed trajectory for the ADS to adopt (i.e. affecting the configuration of ADS functions at different levels).

B. 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.

C. Safety Orchestrator and Context-Dependance 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.

D. 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 agnostics. 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.

Conclusion

In this section, we have introduced a reference architecture to answer the identified needs for a reconfigurable, flexible, traceable, and observable Safety Management System (SMS). We have proposed an extended architecture plugged to ADCC that operates at two levels of adaptations to facilitate the management of the safety assurance processes and to capture their specifications to guarantee context-dependence and possible conflicts.

Based on the previously introduced requirements, and with regard to the reference architectural components, our effort in the next section focuses on the development of reference implementation architecture. We analyze the requirements introduced by the design choices of the reference architecture for a reconfigurable, flexible, traceable, and observable management system for safety. We also identify the structures and behaviours that satisfy our previous requirements from the analysis section.

4.4 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. Then, Section 4.4 presents 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, the last section 4.4 details how the knowledge of the self-managed system should be structured and illustrates its applications to the AV.

Implementation Requirement Analysis

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.

Autonomic Computing

The Autonomic Computing paradigm proposed in [77] 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, Analyzing, 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). They 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 [77] 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 favors 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 comply 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).

Microservices

The microservice architectural style [56] 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.

Scalability is one of the major driving forces behind any kind of architectural solution. According to [102], it can be attained from three axis: (1) horizontal duplication (i.e. achieved by cloning components such as running copies), (2) functional decomposition (i.e. achieved by decomposing component's functions), and (3) data partitioning (i.e. achieved by splitting the data or knowledge bases and access). The microservice architectural style mostly insists on the second point by favoring the service decomposition by splitting the system into multiple and different services. It becomes easier to understand the functionality of a service and catch its range of action.

An other drive force for microservices is flexibility. Indeed, they offer the possibility to be individually replaced so that they keep performing the same function but with a new technique or in a different language. Dependencies and versioning becomes also easier to manage each microservice individually.

The adoption of a template design in the interfaces of communication of the microservices promotes an higher level of pluggability for the whole software architecture. Indeed, using common or standard interfaces also promotes observability in the system by facilitating the observation and monitoring of the channels of communication.

Microservices have also a relevant ability to improve fault isolation. Systems based on microservices may commonly remain unaffected by the failure of a single one service. In this type of software architecture, the implementation of fault tolerance mechanisms and adversaries mechanisms to test the resilience (e.g. fault injection as Netflix's chaos monkey) are made easier.

We believe that microservices architecture allows to extend the current traceability of the knowledge in the existing ADCC modular architecture. In this architecture, the acknowledged functions in the autonomous vehicle are tightly coupled for performance and uncertainties purposes like most AV architecture. They act like a black box with opaque behaviours reducing the potential for observability and traceability which we want

to promote. In fact, the source of knowledge like goals and policies are not directly understandable or accessible as a service. We want to explore how the separation of concern and granular specification well-fitted for specific concerns can be operated while remaining traceable with the use of microservices. In this purpose, the microservice separation approach between operation and agnostic orientation towards knowledge may provide a traceable way to observe, trace, verify, validate and finally trust the system.

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 contribute to cover the structural range of our expected attributes as it specifies the structure of intercommunication between components and allocation as microservices.

Knowledge Representation and Semantic Model

Model-Driven Engineering (MDE) have contributed to facilitating the development of architecture for complex systems including self-adaptive systems by addressing theirs representation problems (i.e. flexibility, scalability, and traceability) [44]. 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 as suggested in Systems and Software Engineering practices [140]. 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 an 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 [45, 78]. In our perspectives, it rigorously and consistently uses models for engineering feedback loops as it capture the adaptation mechanisms and exchanged information.

The knowledge representation with semantic models contributes to cover the structural range of our expected attributes regarding the capture of knowledge and how it can be documented and structured.

Conclusion

This section have assessed that the required attributes can be addressed by the Autonomic Computing, Microservices and Knowledge Representation with Semantic Models approaches. Within this objective, and with regard to the architectural components im-

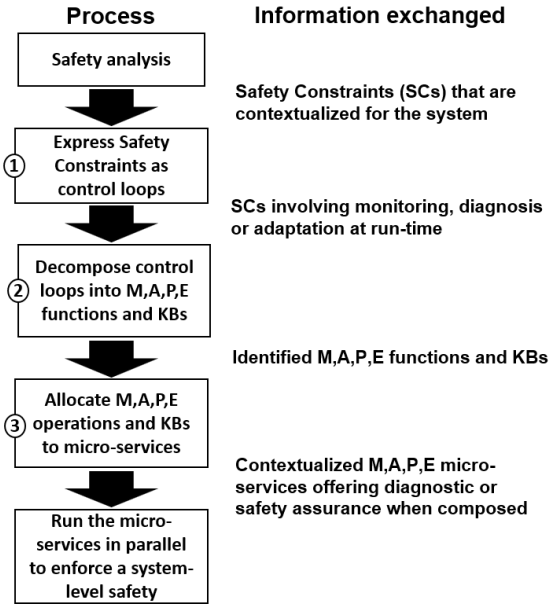


Figure 4.2 Process of integration of the safety constraints in the system

plementation, our effort focuses on the development of procedural correct rules for the dynamic reconfiguration of multilevel microsevice-based software architectures.

Next section performs the successive combination of them to obtain a resulting architecture that have our expected system attributes and their individual advantages.

Structural and Behavioural Implementation

This section introduces the structural and behavioural implementation of the framework by four successive steps. Figure 4.2 illustrates the step-by-step methodology to obtain a composable and orchestrable microservice-oriented reference architecture offering a flexible and adaptable run-time safety assurance capability from a set of safety constraints.

Section 4.4 introduces how the safety constraints that are originally expressed in the model and can be assessed at run-time are represented as a collection of several safety assurance processes to be ensured at run-time. Then, each process is decomposed into four distinct functions according to the composition scheme in Section 4.4. The shared knowledge and exchanged information are also identified at this step. Section 4.4 describes the allocation step of each MAPE functions to a distinct typed microservice. Section 4.4 characterizes the orchestration scheme that manages of the safety constraints into structured and manageable safety assurance processes and their respective functions.

Finally, section 4.4 summarizes the resulting architecture, the identified layers and how the attributes are covered in each system capability.

Management of the Safety Constraints in Parallel: the Safety Assurance at Run-time

In the previous chapter 3, we have raised our interest on the automated driving system of the AV as a whole. By considering safety as a control problem to enforce safe behaviours, we have proposed safety constraints determined for our level-4 automated vehicle and ODD. These safety constraints can be separated in two categories: one enforces

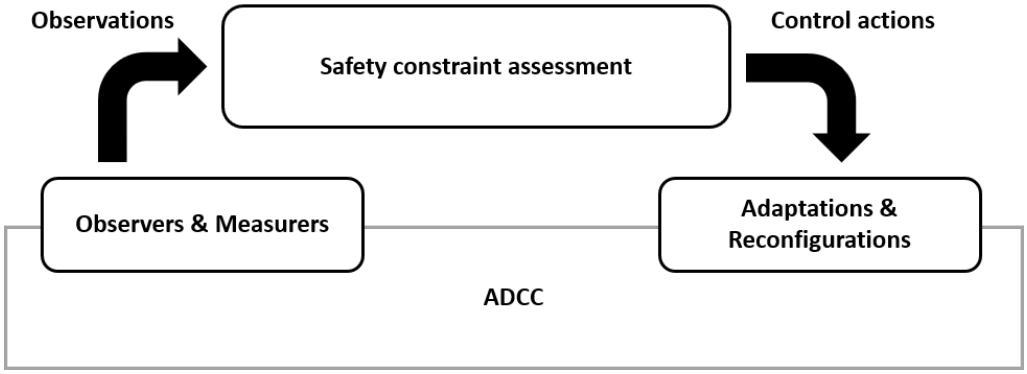


Figure 4.3 Illustration of the assessment of a single constraint related to safety

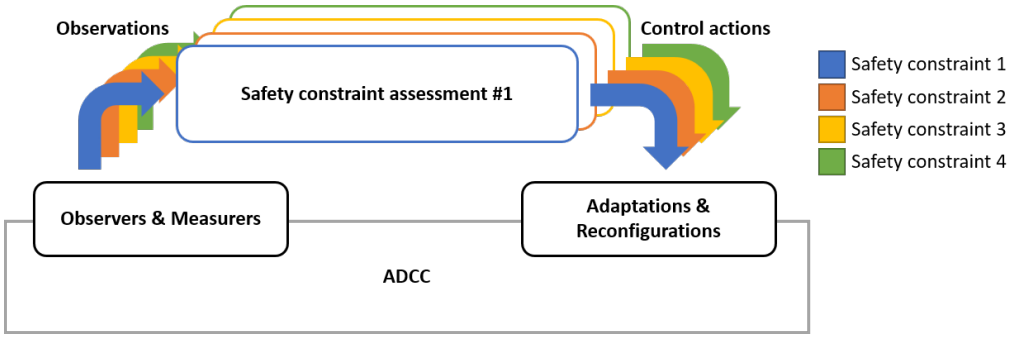


Figure 4.4 Running assessment of multiple constraints related to safety

safety through hazard elimination by refining the design while the other though the hazard control using management to reduce adverse events.

We now aim ensure the constraints from this last category at run-time by performing operation monitoring, diagnosis or adaptation on the ADS. Figure 4.3 illustrates a safety assurance process that can be performed during the operation of the functional system. In our scope of work, the sensors and the actuators can actually be real or virtual. As an example, they can respectively act as sources of observations from external and internal sensors (e.g. perceived objects, proprioceptive observations of a component) and as consumers of control actions (e.g. a new trajectory to follow, a service to deploy, a new configuration to adopt).

In our approach, we perform these safety assurance processes at run-time and in parallel on the functional architecture of the AV as shown in Figure 4.4.

MAPE-K as Architectural Solution for Automation: Decomposition of Control Loop With Mape-k Autonomic Loop

Then, to favor observability we identified our control loop to an autonomic manager from the vision of the Autonomic Computing paradigm. At this regard, we decompose it into Monitor, Analyze, Plan and Execute functions sharing respective Knowledge. Figure 4.5 proposes a representation where the four parts work together to provide the autonomic loop realizing the safety assurance process. Communication is performed between the functions in the form of exchanges of symptoms, change requests, change plans and actions using callback or shared memory. It is important to notice that not all the safety assurance processes are necessarily performing a closed control loop on the ADS. In-

deed, it is possible that some only aims to achieve part of the full autonomic loop such as measurements (M+K), diagnostic (M+A+K), planning (M+A+P+K) or adaptations for the full loop (M+A+P+E+K).

Monitor The Monitor function accesses the information of the managed resources via the sensor interface. The retrieved information is filtered and only a relevant part is transmit or stored in the Knowledge. The aggregation, correlation and filtering determine if a resulting symptom need to be analyzed. The symptom is then transmitted to the analyze function. It is important to note that it can exist more than one Monitor function for a specific autonomic loop if the correlations are multiple or the filtering differs.

The raised symptom can relate to a particular combination of events resulting from the aggregation and correlation between the content of different message or event from multiple resources. As an example, we can consider that detecting the presence of a pedestrian as a possible symptom. Also a pedestrian crossing the road can also result in a more complex symptom. The granularity of the process within the monitor functions determines if the symptoms are either simple or complex.

Analyze The Analyze function compares the observed data from the expected values such as anomaly detection. More complex analysis can necessitate the use of behaviour-aware techniques such as partially observable Markov decision process (POMDP), probabilistic approaches or learning bases approaches. As a consequence of some policy not being met or going to be, a diagnostic is established in the form of a change request.

This change request contains the modification to overdo that the analyze function have considered as necessary or beneficial.

Plan The Plan function triggered by the diagnostic selects or determines the strategies in order to achieve the intended goals (creation, correction, prevention). Various plans are studied and one final plan is transmitted to the execute function.

The change plan consists of a prescribed collection of changes to apply to the managed resources (e.g. a trajectory)

Execute The Execute function executes the corrections actions via the effector interface and caches the process for future analysis and planning.

From this step, the Knowledge does not only represent the policy that influence how the decisions have to be made. It also includes the information retrieved from the observations, the environment's configuration and inside communication (i.e. symptoms, change requests, change plans and actions). Moreover, the Monitor function can contribute to the creation of knowledge regarding the recent observations as well as the Execute function with the actions that have been applied.

Implement MAPE-K as Microservices

To favor the composability and flexibility of the resulting architecture, we assign each of the MAPE functions and the Knowledge management to distinct and functionally described microservices. At this regard, the allocation results in the creation of small, independent, deployable and simple typed services offering each distinct functions (i.e. Monitoring, Analyzing, Planning and Execution) and communication interfaces.

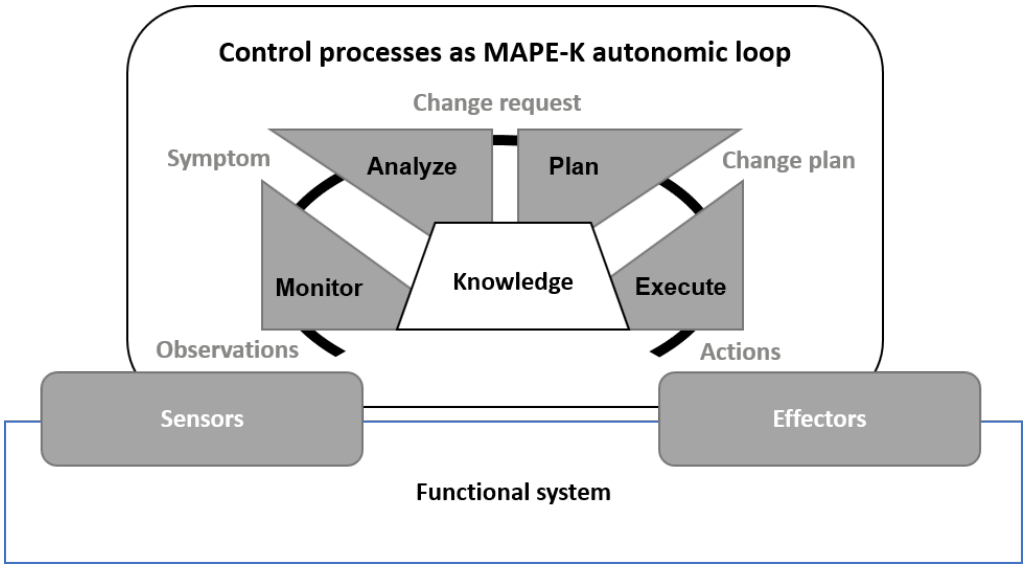


Figure 4.5 Decomposition of the control process into M, A, P and E functions that share Knowledge

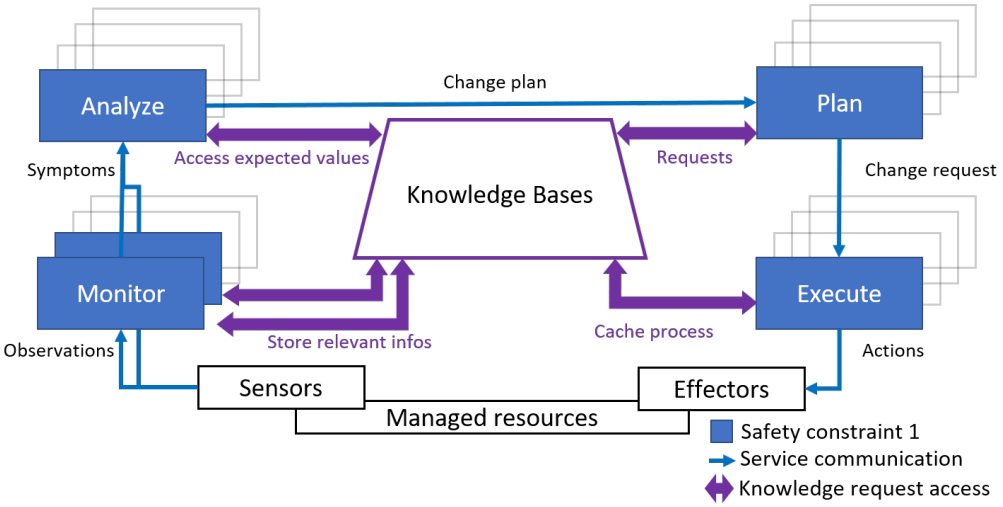


Figure 4.6 One Autonomic loop is composed by M,A,P,E entities to observe, diagnosis or ensure the safety

From a functional perspective, these microservices continue to implement the autonomic loop as described previously. Figure 4.6 illustrates the decomposition of one safety constraint as MAPE-K into the M, A, P, E, K microservices and their respective communications channels. The K microservice or Knowledge Bases Service (KBS) provides access to the knowledge bases through an interface.

Consequently, the structure of the system have been impacted in its design, implementation and on how the knowledge is used and managed. Exchanges are no longer callbacks but become messages or client-server relations. For instance, a Monitor microservice is designed to access some observations following a specific nomenclature of the touchpoint, publishing the symptom and storing relevant information or directly the symptom in the KBS.

The microservices are constantly running, operating their function or waiting to be triggered.

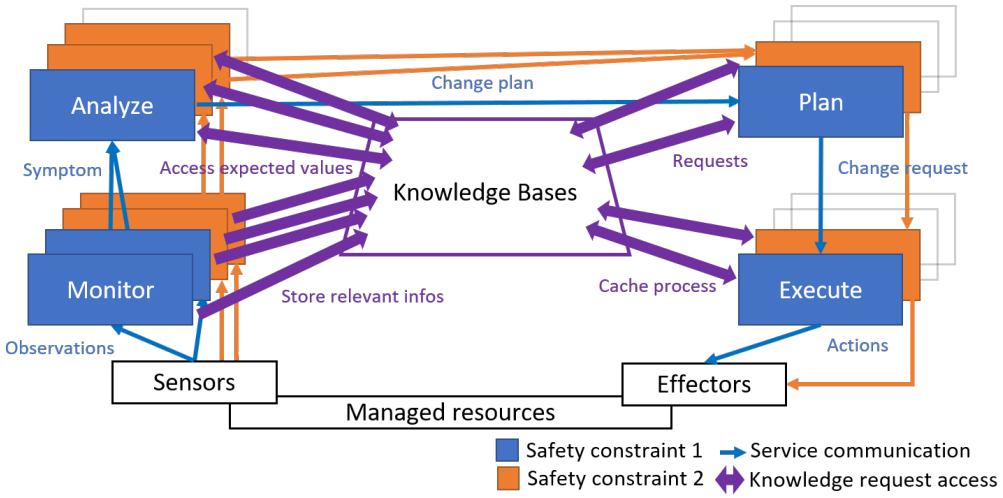


Figure 4.7 Several Autonomic loops are running in parallel using M,A,P,E microservices illustrating the Hyper-dimension

However, adopting this microservice structure to have its benefits implies to support the goals of service orientation. Additional features as service repository, discover service (running or not), service contract standardization will need to be included in the logical and physical architecture of the framework.

The advantages of using composition of the respective instances of microservices to perform multiple autonomic loops are twofold.

Hyper-dimension: Running multiple composable autonomic loops in parallel First, microservices instances can be reused to perform multiple autonomic loops impacting both the design and the run-time. It would be to allow service descriptions and data models to become highly discoverable. On one hand, the design of new safety constraints with MAPE-K can reused existing microservices as far as the needed function is the same. On the other hand, only one microservice can be deployed for the same function in the whole system at run-time. The introduction of this hyper-dimension impacts the design as the system is able to know what are the services currently deployed or available for deployment.

Figure 4.7 illustrates the case of two safety constraints running in parallel using the MAPE decomposition scheme. They are operating separately and involve several microservices to perform the Monitor and Analyze functions. Their functions on the system can be easily defined due to their small size and their mapping to a particular MAPE role.

Let us now suppose that some roles of these Monitor and Analyze functions are identical and can be shared between the two safety processes. Figure 4.8 highlights this reuse of some deployed M,A,P,E microservices in the different loops running in parallel. In terms of design, as long as the operations and observations remain the same, the communication service allows reuse.

Multi-dimension: Running multiple autonomic loops involving different dimensions of constraints in parallel Second, others dimensions (e.g. social acceptance, ethics, legal, performance) can also be integrated along the autonomic loops of safety assurance processes in our approach. However, they still require to be represented using the MAPE-K microservice scheme we previously introduce. Figure 4.9 illustrates the introduction

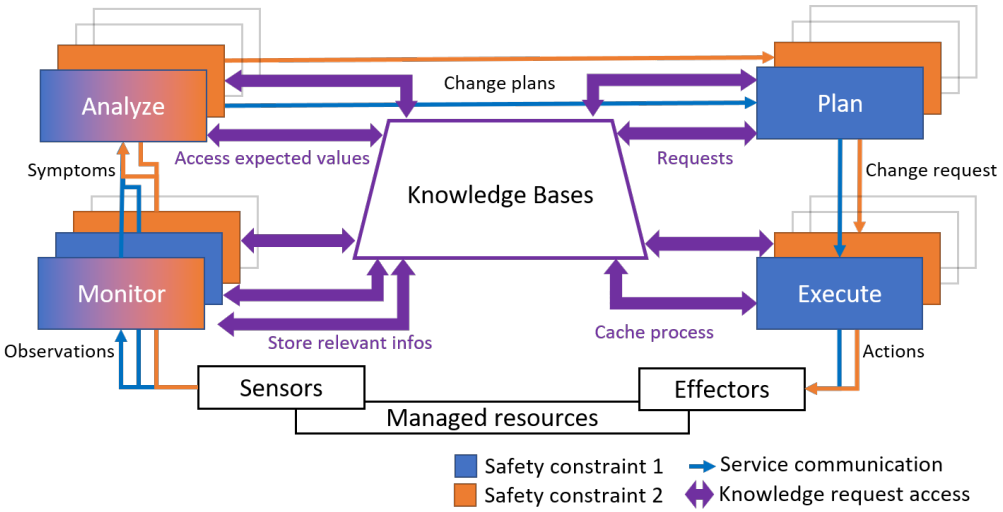


Figure 4.8 Several Autonomic loops are running in parallel and reusing deployed services as M,A,P,E entities illustrating the Hyper-dimension and reusability property of the framework

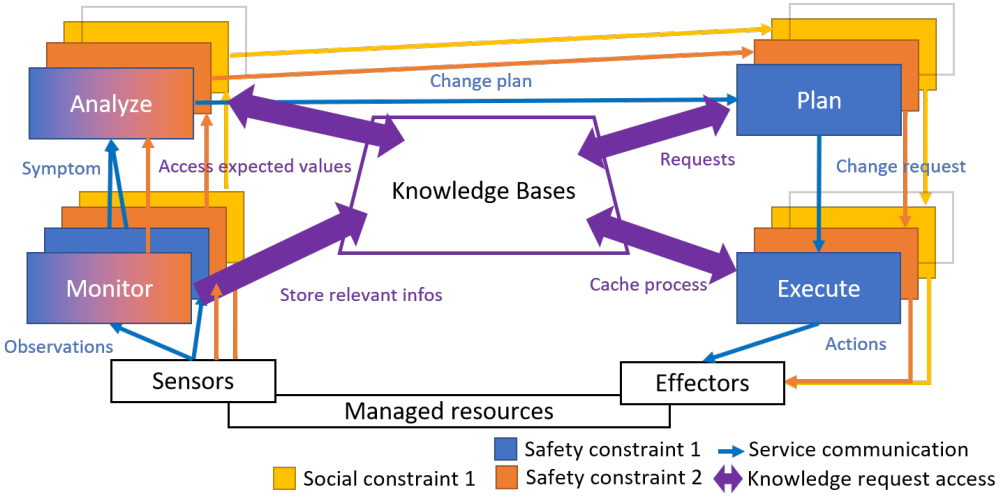


Figure 4.9 Several Autonomic loops are running in parallel and reusing deployed services as M,A,P,E entities: Illustration of the Multi-dimension and extendability property of the framework

of one social constraint to be ensure at run-time as autonomic loop. This example just considers the previous example with the addition of a social constraint. For instance, the social constraints targets the features for socially acceptable distance to vulnerable users (i.e. maintain d distance between a pedestrian).

Moreover, attention can be drawn on the hyper-dimension of the framework where similar services from other dimensions can reused or be reused as long as they are involving the same function and same concerns (e.g. gathering road and lane position from the pedestrians crossing the road).

Orchestration of MAPE-K Microservices: Managing the Autonomic Loops

The goal of this subsection is to understand how the association of the previous schemes are key enablers for monitoring safety-critical software systems. We will also clarify their role and fundamental interests for the framework in order to finally define a reference architecture for our concerns.

We are using the hierarchical coordination of an Orchestrating Autonomic Manager (OAM) from the Autonomic Computing to perform a framework-wide autonomic behaviour in completely different disciplines than the resources it manages.

This OAM considers the AMs of the autonomic loops presented in Figure 4.9 (i.e. performing our safety and social assessment processes) as managed resources. It also performs the function of monitoring, diagnosis, planning or adaptations and is composed of M,A,P,E,K microservices. Figure 4.10 illustrates this hierarchization between the safety and social assessment processes as *Managed resource #2* and the OAM as a higher managing instance.

An example of application is the intended use of this hierarchy in this thesis as an autonomic loop to reconfigure the safety assessment processes deployed for the current context. We have seen previously that the AMs are constantly performing the autonomic loop realizing the safety assessment process for a given context (i.e. where they are relevant, where they can be deployed and where they are sufficiently efficient). However, a change of context may result in the assessment of safety constraints no longer relevant or even unacceptable for the new context leading to possible violations for the current context. Aware of the context-dependance of these safety assessment processes, the OAM monitors those AMs and the current context to determine and deploy the most appropriate configuration for the safety assurance of the ADS (and other dimensions).

This OAM can also include hyper-dimension and multi-dimension considerations according how the assessment can be managed upon (e.g. context of system, performance of AM, consistency of AM, stability of the safety assurance, ...).

Resulting Architecture

The architecture results in the following logical parts:

1. Touchpoints that interfaces the system to the functional architecture of the ADS.
2. Autonomic Managers that performs collectively safety assurance allocated as a grid of M, A, P, E microservices. It represents the first level of adaptation.
3. Orchestrating Autonomic Manager that performs the reconfiguration of the safety assurance to adapt it upon the context. It represents the second level of adaptation.
4. A Knowledge Service offering aside access to the different shared knowledge bases where symptoms, diagnostics, change plan are stored. It manages the access to the distinct knowledge sources for each of the two levels of the adaptation.

Conclusion

In this section we have presented the structural and behavioural implementation of the reference architecture using the four successive steps based on safety constraints. The proposed microservice-oriented architecture offers a flexible and adaptable management of safety during run-time by using composition and orchestration mechanisms.

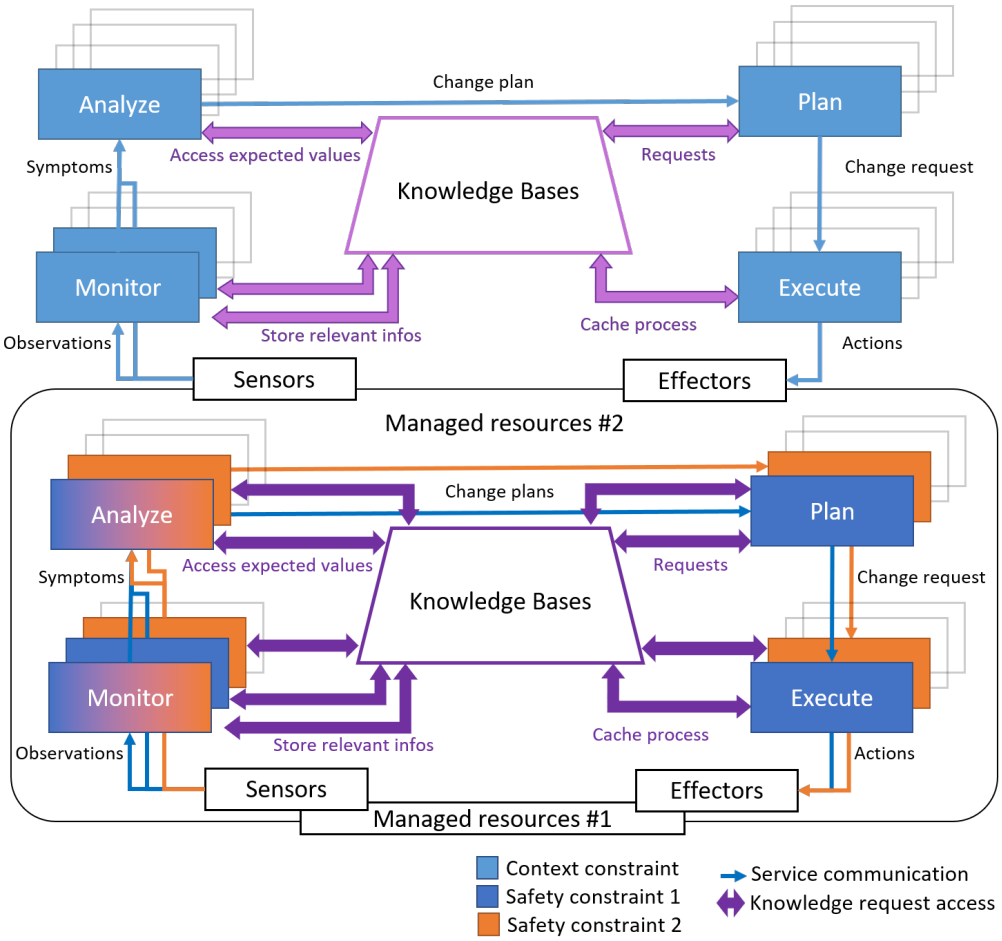


Figure 4.10 Hierarchy of several Autonomic loops for different disciplines: Illustration of the Orchestration and reusability property of the framework

However, the different sources knowledge to operate those mechanisms still need to be identified and defined with clear boundaries and semantics. Next sections presents how these different sources are represented as ontologies (e.g. system's operating context, the vehicle capabilities and actions and interfaces).

Design of the Knowledge Bases and Semantic Models

The MAPE-K loops require the presence of different sources in the knowledge base to be able to operate the autonomic adaptations of the two levels. Indeed, each of the M,A,P,E functions has a particular behaviour and requires the presence of different relationships and specified interactions between entities within different knowledge bases. Hence, we propose to model the respective knowledge of the levels using ontologies as semantic models.

The two levels address distinct purposes regarding the aspect of the property of safety. The first level of behavioural adaptation requires information on performing the safety assurance processes. The second level of the structural adaptation ensures the composition of the deployed instances based on context-dependence requirements. These different purposes shall result in creating two separate knowledge bases to support the adaptations of the levels. Although their purposes are distinct, some knowledge may actually be

shared between the two levels in the abstraction of the context or the system representation.

To cover the knowledge requirements of the MAPE functions, we use a set of five types of models adjusted for each level of adaptation proposed in the literature [15]. We also detail the mapping for each of the levels of the reference architecture and illustrate their application on AV safety. Finally, we present the sources used to represent the knowledge models for each type at design time and how they plan to be used and be changed at run-time.

Main Types and Information in the Run-time Models

To clearly identify the scope for each source of knowledge we adopt the abstraction of models proposed by Aßmann et al. [15] in the model@run.time architecture. They use a set of five types of models to cover the needs of knowledge representation and the definition of boundaries. Therefore, the following paragraphs present the particular purposes of those models, and their coverage, and the identified operating knowledge in our approach.

Context models propose an observable state of the environment containing relevant information on the entities surrounding the managed system. Semantic understanding as context interpretation is expected to be performed by the Monitor microservices and stored as symptom in the context models. Therefore, this model is expected to have two different parts: an abstract representation of the context serves as a static terminology basis; and a part that evolve at run-time and solely represent the symptoms detected from environment. In our scope of work, the context models are close to the definition of the Operational Design Domain (ODD) in AV. It represents the extent of various conditions and scenarios where the vehicle is designed to operate intently and safely. The ODD can be constructed as multiple layers describing the dynamic entities (e.g. car, buses, pedestrian, scooters), the localization (e.g. geo-fenced), the scenery (e.g. roads geometry, infrastructure) or the allowed activity (e.g. highway driving, self-parking), and help building scenario. As a result, the terminology available for abstract representation shall include those different concepts. The symptom shall also compose them to create identified context abstractions.

Configuration models depict the interfaces of the managed resources in how they can be accessed and controlled. They contribute to characterize the knowledge required by the touchpoints to operate the autonomic loops (i.e. manageability interface). These models can be seen as providing an architectural view of the interaction with the managed resources. In a generic service-oriented approach, they would reflect which communication channels and managed services are currently available and deployed on which configurations and platforms. In our framework, they are the sources for initialization of the *Monitor* and *Execute* functions to interface with the managed systems appropriately. They affect the deployed AMs microservices managing the ADS and the microservices forming the OAM.

Both context and configuration models help the capture of a human and machine-understandable representation of the external and internal observations.

Capability models characterize how the available features can influence the monitored resources by which actions and involving which effectors to interface with (e.g. adjust parameter for requiring slower speed, deploying new microservices). This model type is expected to be rather static as it expresses all the possible capabilities of the system but can be updated (e.g. new design, new structure). In our scope, the features are the

mechanisms that influence the AD system to enforce safety in AMs and the mechanisms that enforce its acceptable contextualization.

Plan models represent the different sets of actions as plans to perform the autonomic adaptation. The Plan functions contribute to the model storing the requested change, and the Execute translate them into the actions to send to the effectors. Contrary to the capability models that define the system and interactions structure (T-Box), plan models use the structure of the capability model to describe each action as individuals (A-Box). Therefore, they collect a history of all of the requested actions sent to the managed system.

Goal & Adaptation models define how each of the MAPE function may be performed (i.e. specify a mechanism) and tuned (i.e. specify a configuration). For instance, they characterize how the *Plan* should evaluate the alternative future configurations and how the *Monitor* should trigger itself to create symptoms in our framework. Furthermore, these models are the basis for a choice of design.

Indeed, we decide that the Goal & Adaptation models only describe a terminology of the system (T-Box) and shall not change at run-time. The main idea behind this particular choice of design is to highlight the traceability between the configuration of the different microservices, and the context (i.e. internal and external) with the ability to rightfully orchestrate them. The first level performs the safety assessment with the stateless AM microservices while the second level continually verifies that the constraints of context-dependence (i.e. restrictions based on the configuration and deployment of specific AMs in particular contexts). Thus, only the configurations of the available microservices are expected to change and to be composed at run-time. It results that the goals and mechanisms models specify for all microservices their different static configurations and context deployment restriction constrained.

Besides, these models are expected to change over time in design iterations to manipulate new mechanisms or adjustments for specific microservice during the life cycle of the system.

Table 4.1 synthesizes all the roles and applications for each type of models. It is noticeable that concerns and knowledge to represent differ according the levels of the reference architecture. Finally, Figure 4.11 illustrates the different conceptual models and their relationships in the resulting ontology of the framework.

Applications, justifications and examples are detailled in Appendix B (see page 163). They detail the requirements, modelling choices and implementation of the knowledge bases applied to the context of the case study in Chapter 5 (see page 117).

Discussions on the Advantages of the Semantic Orientation

We see this approach over knowledge as a run-time operable way to have requestable and updatable reasoning ability. Thus, it opens the possibility of arguing in the system. It also allows comparison between reasoners and to perform trade-offs (i.e. the language of implementation, how the ontology is used, and how the relations we are exploiting). The addition and integration of new uses cases, situations or equivalent rules can have their impact studied directly on the model and do not necessitate additional implementation.

We also gladly inherit of the ontology interoperability that serves the representation of the system from an MBSE tool as Capella to *Monitor*, *Analyze*, *Plan* and *Execute* ontology classes.

Besides, we think that the ontology representing the context model proposes a reasonable way of composing the *Abstract context* as a set of scenes, situations and other

Table 4.1 Type and description of the models for each levels of the reference architecture

Model types	Definition	Grid of microservices forming Autonomic Managers (AMs)	Orchestrating Autonomic Manager (OAM)
Context model	• Relevant information on environment state of the managed system	• Current status of the environment of the vehicle from the ADS observations	• Current context of the vehicle from the ADS observations (through symptom as identified use cases or situations) • Current status of the AMs microservices monitoring (through symptom as event)
Configuration model	• Access and control interface to the managed system • Current state of the managed system	• Explicit the access and control interfaces of the AD system • Current configuration of the AD system, sensors and effectors status (operated by Touchpoints management)	• Explicit the access and control interfaces for each of the AMs • Current configuration of the safety assessment and MAPE microservices currently deployed and running on which configurations
Capability model	• Features available to influence the managed system • Available effectors to interface with • How the environment is affected by actions	• System functions on specific mechanisms available to influence the AD system (except context-related application)	• System functions on the mechanisms available to influence the AMs
Plan model	• Set of actions to perform to realize the adaptation	• Actions to ensure the safety related to specific maneuvers	• Actions to reconfigure the deployed AMs microservices changing their goal
Goal & Adaptation	• How the alternative future configurations should be evaluated as fulfillment of intended goals	• Assessment of the behaviour and the intention of the ADS in the scope of one safety constraint	• Distance and semantic coverage of the current use case from the previous configuration

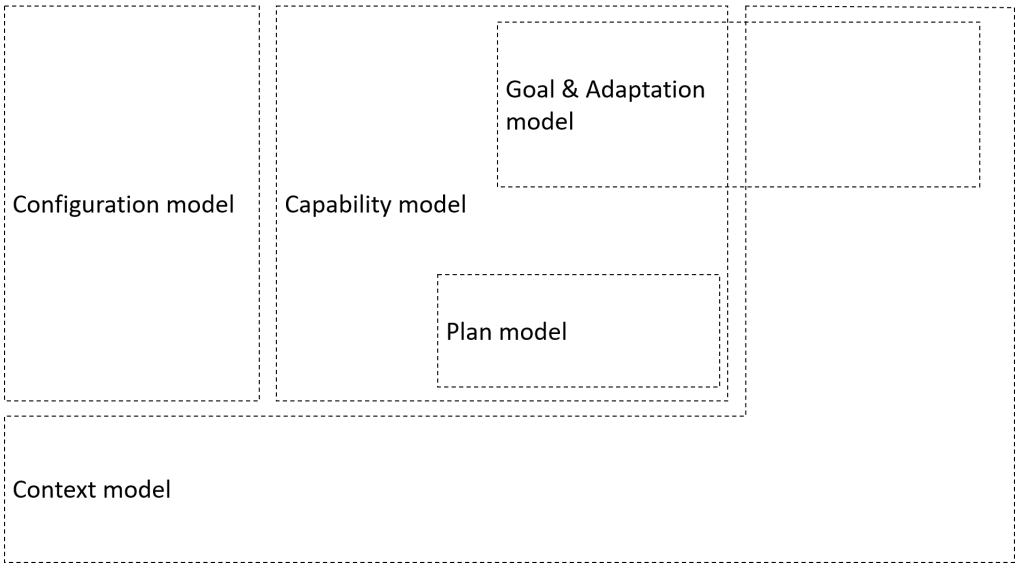


Figure 4.11 Overview of the knowledge represented through the different models as modules

inherited classes. It acts as a catalogue for those different abstractions that cover a different range of representation and granularity.

The representation of axioms from the ontology as classes also satisfies their reuse as such directly in the code for better interoperability. Such import of original ontology concepts as classes and seamlessly use are made possible by the some package. This specific part takes place in future works regarding creating a toolchain and the streamline of the knowledge from the model, to the ontology and the actual executable code.

Finally, lesson learned by the Automated Vehicle community from recent crashes, future standards, regulations and expected behaviours intensify the specialization towards particular Operational Design Domain and verification. Our approach can integrate those requirements into its a-priori knowledge thanks to the composable context abstractions and explicit context-dependence restrictions of the microservices. On this basis, the consistency of the updated model with the actual system is still performed continuously as being investigated by the second level of adaptation.

4.5 Conclusions

In this chapter, we delineated our microservice-oriented 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 illustrated the use of patterns for components and knowledge bases to meet the NFP.

The next chapter applies both the introduced patterns and models for safety assessment in a case study range of scenario with respect to the safety analysis from the previous chapter.

CHAPTER 5

Applications and results

"The good thing about computers is that they do what you tell them to do. The bad news is that they do what you tell them to do."

- Ted Nelson,

Contents

5.1 Introduction

Pedestrian crossing is one of the most important interaction that autonomous vehicle have with pedestrian [114]. Pedestrian are vulnerable actors in some conditions and collision with a vehicle can result in severe human damage. In some specific settings, a collision is also likely or inevitable and before impact occurs the vehicle is expected to act to reduce the total harm and injuries. Research have shown that it exist 38 factors that can potentially impact the way pedestrians behave with autonomous vehicles [118]. However, those factors reveals variations upon the geographical locations where culture and social norms may differ. Thus, appropriate reactions as communications and negotiation are expected from each individual vehicle. It also implies that the autonomous vehicle needs understanding pedestrian's intention.

In this chapter, we aim to apply our methodology and framework to the detection and mitigation of possible unsafe situations with pedestrians. The resulting system should be able to, first, identify the relevant elements in the scene, second, reason about the interconnections between these elements, and third, infer the upcoming actions of the road users. In addition, the flexibility and reconfigurability of the system are expected

to appropriately respond to the various traffic scenarios (e.g. different street structures, traffic signals, crosswalk configurations).

Section 5.2 presents our general implementation of the framework as respective elements for the different layers. We also details the implementation of the containers and knowledge representations to grasp the flexible, adaptable and observable perspectives.

Section 5.3 presents the generation and analysis of the scenarios as well as the application of the different containers that constitute our framework. To conduct the experimentation of the framework, we have allocated the safety constraints that we considered as relevant in the scenario as mitigation mechanisms within the presented containers. Our applications contains some symbolic approaches to understand of pedestrian's intention and mitigate their behaviours to ensure safety at run-time. Our first results shows that the integration of the framework to the ADCC architecture on the robotic operating system (ROS) architecture is feasible. This delivers significantly better results due to reducing the unsafe behaviours of the vehicle around pedestrians.

5.2 Framework Implementation

System Scope and System Architecture

The following general design guides the implementation of our framework. The main features that forms the composition, adaptation, and knowledge mechanisms in the implementation are also illustrated.

Implementation Overview

Embedded and restrictive limitations

System Specifications

This section introduces the different framework, techniques and methods we are using to implement system behaviours to compose our physical design architecture and final implementation. In particular, we will introduce the robotic operating system (ROS) middleware that will be used to implement and evaluate our proposal.

ROS: ROS is a middleware that enables concrete component based software engineering techniques, as well as smarter communication patterns as publish-subscribe [116]. Not only ROS is already used in ADCC, but it also impose a service-oriented implementation for the technical architecture. Although ROS based systems are not ready for customer open road operations, it presents a variety of advantages regarding the development of any prototype or research project [156]:

1. Packages to interact which are already implemented within the ROS environment. For instance, one package may provide a functionality that may not need to be implemented in the first place, but may be replaced later.

2. ROS is already a distributed service-oriented architecture. It favors modular packages over a monolithic development and run-time environment. This "thin" ideology discourage accidental creation of monoliths.

3. It provides efficient tools as introspection features (e.g. message payload), graph of nodes and data visualization.

4. Nice and broad community has been created around ROS which has been able to grow and exchange with the organization of conferences (ROSCON), tutorials and the provision of packages.

5. ROS is language independent, interaction is done at the interaction layer via messaging.

6. Replay and emulation ability and facilitators (i.e. launch files, configuration files) are a key perspective for the thesis in our ability to create, simulate or react on simulated and recorded situations. It ease the experimental procedure and favor the possible comparisons of various algorithms per node.

ROS presents a variety of additional advantages that goes into the orientation of our solution. The service-oriented middleware already provides the basic features associated to microservices architecture. Its use allows to access the following existing and well-experienced functionalities:

- Fundamental elements in a ROS-based system are nodes, messages, topics, and services. In our case, we will use nodes as microservices following the ROS best practices: "ROS is designed to be modular at a fine-grained scale".

- Service registry supports a peer-to-peer topology with the master as lookup mechanism to allow processes to find each other at run-time.

- Nodes are communicating by sending messages to topics using a publish/subscribe mechanisms (event streams). Service calls are also possible for synchronous transactions.

- Introspection for logging, monitoring, and visualization of the network dataflows are possible with the provided tools and API in the form of the graphs (node can communicate between each other). Creating introspection for a specific interface can be easily performed, customized and extended later.

Microservice oriented architecture: As raised by Dirk Thomas, software engineer at the non-profit Open Source Robotics Foundation, in the ROS2 open-source GitHub repository[1], the creation of microservices in ROS (in its version 1 or 2) does not required any package or extension since the communication core "makes no specific assumptions about the context they are being used in". From this statement, it is easily feasible to design the ROS nodes as separate microservices. Indeed, the management of different ROS nodes becomes handy when some of their life-cycles are longer than others or involves specific reconfigurations.

Regarding the deployment, commonly used in the world of web services and microservices, Docker[2] is a solution for deployment to standardized development environments. It provides containers that can host a set of services. They refer to be convenient for pursuing some goals of software development such as continuous integration and continuous development. ROS deployment possibilities on Docker have been synthesized in

[1] https://github.com/ros2/design/issues/85
[2] https://www.docker.com/

[156]. It precise to follow the convention of keeping to one process per container like the one suggested for microservices architecture. Indeed, the launch of a set of nodes from a same terminal (using roslaunch or subprocess calls). A unified deployment with Docker is considered as further work.

Design patterns favoring extendability, replaceability and flexibility: The use of several design patterns (i.e. abstraction, strategy, decorator and composite patterns) contributes to the readability and maintainability of the physical structure framework implementation. Design patterns are not generally code specific but provide a way to solve a common problem with a limited trade-offs than ad-hoc solution [59].

Languages: Python, C++ and Java code: The ROS middleware environment allows the creation of different nodes which can be implemented in different programming languages. The keystone for the exchange of data between the different services is the communication layer. It requires a definition of the *ROS message* structure that is shared and compiled for all possible languages. For illustration, we can use some node in python for monitoring others coded in C++ and access a knowledge base running with Java.

State Machine with the SMACH package: The use of template state machines from the SMACH package[3] and their customization avoids recreating a full library of components with a introspection functionality. The package offers the possibility to create customized state machines with state names, event triggers (e.g. message reception or personalized) and operations to run in each state.

Ontology management with the ARMOR package: The ARMOR package[34][4] provides a ROS service implementation that can load multiple ontologies. It can also operate Protegé-like functionalities, but at run-time (load, add instance, infer, check consistency).

We have presented both the limitations and system specification to propose an acceptable environment to apply the framework and to implement it. The next section overviews the choice of implementation of the framework with the presented technical choices (ROS, state machine, ontology service).

Implementation of the Logical View of the Refined Model-based Design of the Framework Architecture

This section presents by functions the different instances and abstractions that compose and constitute the framework implementation.

Service Architecture and Container Allocation

Our framework follows a microservice-oriented architecture. Thus, the functions are allocated to distinct components as microservice so that they perform their main operating function. Figure 5.1 illustrates the whole design and the clustering of the microservice by their layers (L1, L2, or L3), and role (Monitor, Analyze, Plan, or Execute). It is noticeable that the container regrouping all the safety assessment processes (L2) is presented in its

[3] http://wiki.ros.org/smach

[4] https://github.com/EmaroLab/armor

generic version. In addition, Figure 5.1 also precises the main channels of components exchanges between the different instances (publish-subscribe or client-server messages).

Stereotypes for the Framework Microservices

Stereotypes are the main component abstraction of microservice-components that constitute our framework. They contribute to help to handles cross-cutting concerns within the microservice chassis framework. Their specialization at run-time provides the adequate capabilities required by the system. Those stereotypes respect a composite architectural pattern that creates the template functions and interfaces to allow these mechanisms to be deployed, composed and specialized. In this section, we explain the roles, capabilities and specializations that we implement in our framework.

Some mechanisms are similar and shared between each of the instances of Monitoring, Analyze, Plan and Execute. Thus, we propose to abstract and aggregate those common specifications to sustain a similar structure for each of the microservices. Consequently, all microservice in the system have a common bare-bone structure we called a stereotypes. Each of the microservice differs in the expression and implementation of their capabilities according to their types. As a consequence, we push a common structure, illustrated by Figure 5.2, and strategy expression in each of the types of microservices (i.e. Monitor, Analyze, Plan and Execute). As an example, Monitor typed microservices are subscribing to information from heterogeneous sources, accessing their goals and values in the KBs to setup their evaluation, update the knowledge bases with the perceived information and finally raise a symptom if necessary.

The run-time specialization mechanisms of the microservices follows their respective logic stored from the knowledge bases and accessed during its initialization. It aims to provide dynamic and manageable capabilities to the system. The knowledge bases stores the attributes of the semantic descriptions: exposed service interfaces; declarative functionality to be deployed, discovered and reused (deployment recipes); description of requirements, functions and mechanisms (capability properties); preconditions, post-conditions, and invariant rules specific to the behaviour and composition of components.

Further, four types of interfaces are included to grasp the inputs and outputs to communicate with the managed resources, the monitoring to observe the state of each of the microservice, the adaptation controlled by a managing entity and the methods to access and interact with the knowledge bases through a specialized service. These interface are designed and implemented to be open for extension and decorations which can affect all or some specific microservice according the need.

Template interface of I/O The I/O interface allows to dynamically perform the semantic bindings for the exposed service interfaces with the use of a semantic interpreter service. The I/O port can either be as static I/O or be mediated based on the needs and requirements that have been described semantically. Both of the I/O types implement its semantic description that contains information regarding some properties (integrity with interval, limitation, errors, ...).

Template interface for KB The Knowledge Interface corresponds to the interface with the Knowledge Management Service that manages the access to the ontology. It contributes to have some homogeneity and synchronization in the ontology access between the multiple microservice (e.g. shared access methods, types of exchanges and logs).

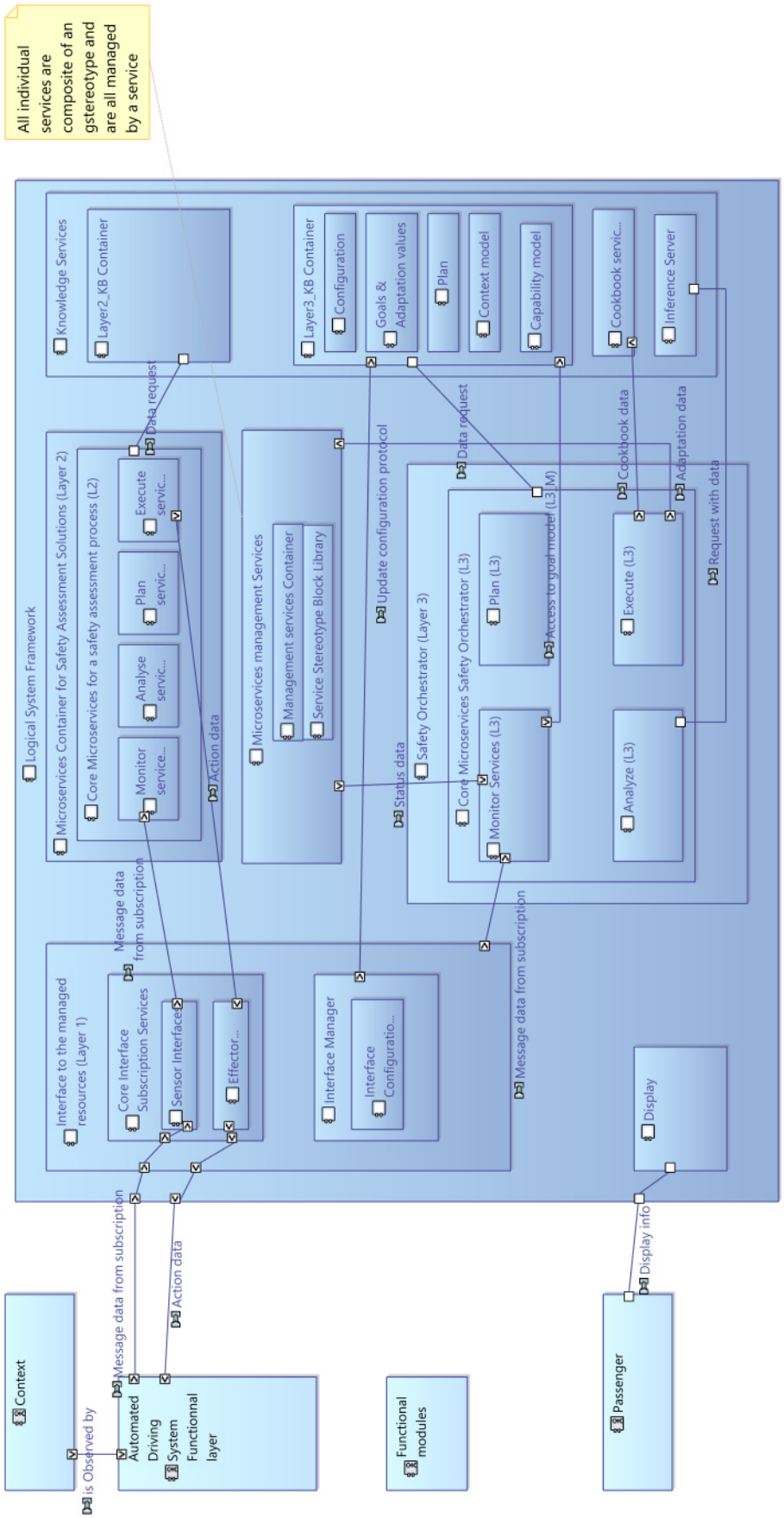


Figure 5.1 Components as service container and component exchanges of the framework (Logical Architecture Blank view)

Template interface for monitoring The abstraction of the stereotypes allows to easily integrate a monitoring interface to composite microservices. This monitoring interface will serve monitoring capabilities with default behaviours (i.e. heartbeat, data consistency, service performance, state) but it can be customized for specific services or advanced computation (e.g. QoS properties).

Template interface for adaptation The framework also imposes the presence of adaptive ports for implementing external adaptation strategies. It provides an interface to the stereotypes to receive adaptation requests from an external component and also to send adaptation request to managed entities or resources. In our implementation, we have linked the execution of our code, the start and the reconfiguration to a state-machine. This way the adaptation process can only reconfigure the service in the state Reconfiguration.

Stereotype top-level view We have presented the composite structure, the capability inheritance for the system architecture capabilities (communication, monitoring, adaptation) of the framework at run-time and the possibilities to customize them during conception. Figure 5.2 shows the stereotype template as part of the logical view modeled using the Arcadia/Capella modelling software. This system representation specifies the structure, interfaces and knowledge accesses of a generic microservice. Five role-specific models are also displayed to represent the type of local knowledge that the microservice can use and access from a knowledge service. This service is detailed in the next paragraphs.

Knowledge Management Service

The stereotypes possess an interface to connect to the Knowledge Management Service (KMS) to perform change or request data from the ontology. The KMS plays a major role in the microservice architecture as the role of a centralized server and access to the knowledge. The other microservice that are its clients are identified. Any changes and requests are stored to be synchronized and logged. The different layers of the reference architecture and types of microservices may interact differently with the KMS to store observations and request information. Those exchanges are illustrated by Figure 5.3.

In our implementation, they are represented as ontologies accessed through as knowledge service. The implementation of the KMS is mainly based on the definition of interface with the microservice that manipulation the ARMOR API at run-time.

Safety Assessment Process (SAP) Decomposition in Monitor, Analyze, Plan and Execute

The safety assessment processes follow the MAPE decomposition in microservice as shown in Figure 5.4. The functional chain in blue illustrates the dataflow of a safety assessment process which involves the observations from perceived context and components in ADCC, the successive M, A, P, E allocated microservices and finally the action on ADCC components.

This illustration is voluntarily generic as the application of a safety assessment process may perform different functions. However, the M,A,P,E decomposition offers a generic template on how to use information stored locally or in the knowledge models.

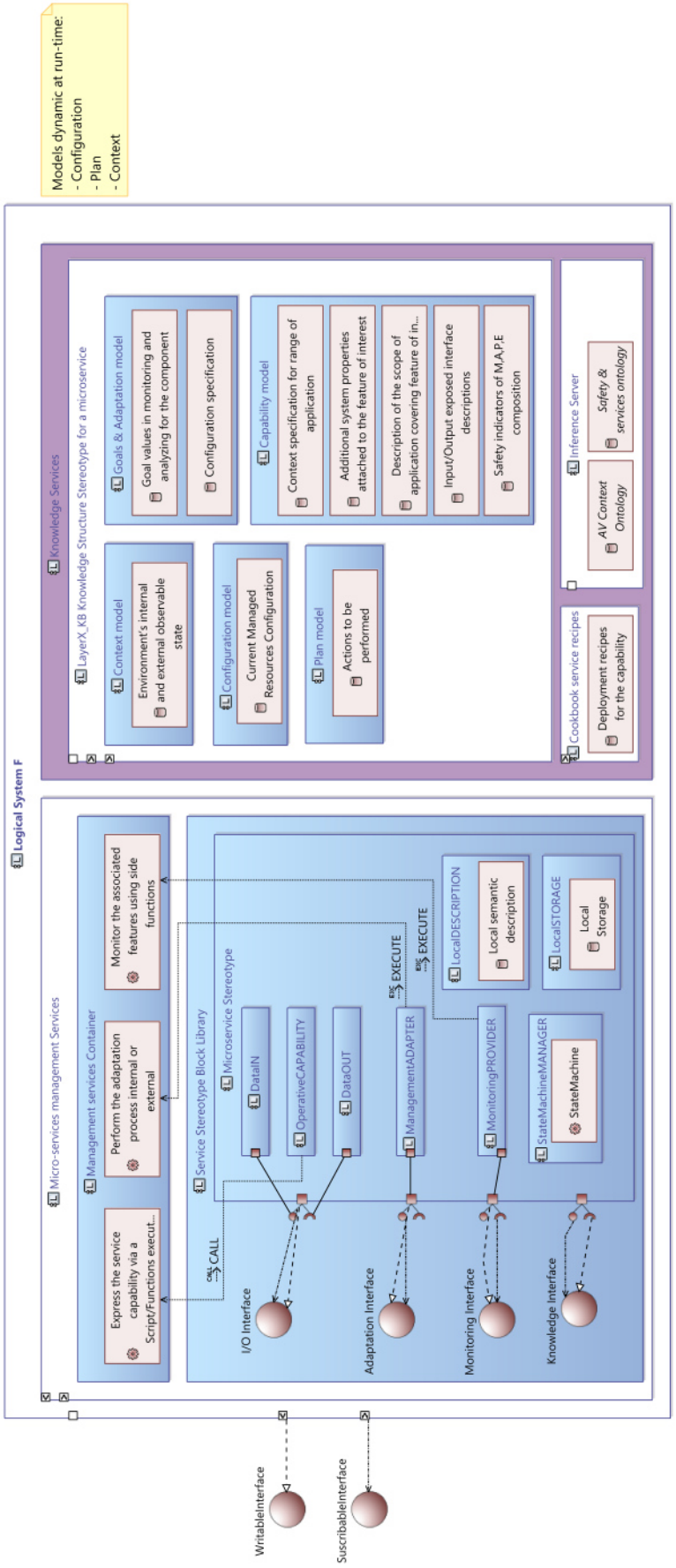


Figure 5.2 Stereotype interfaces and capabilities (Contextual Internal Interface)

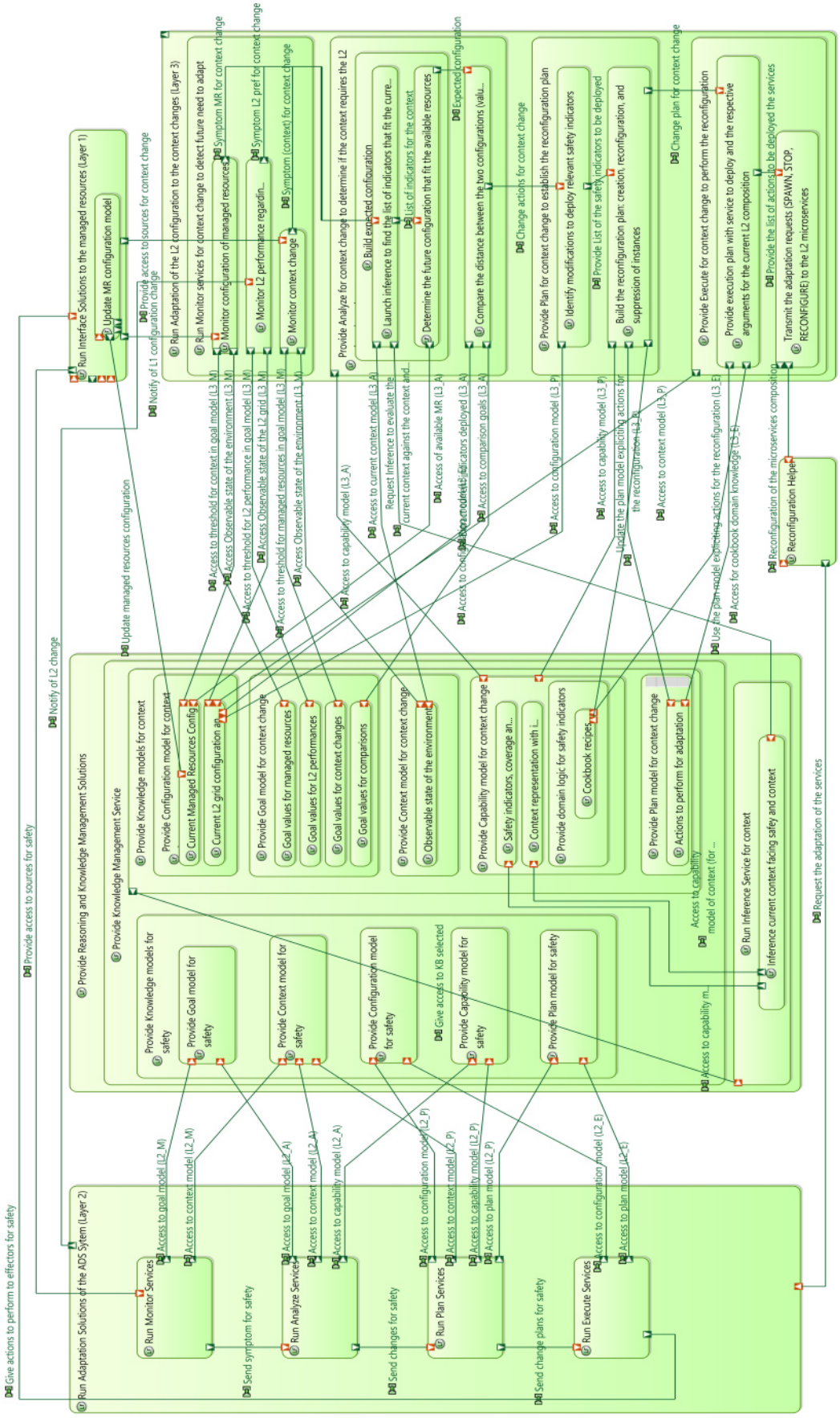


Figure 5.3 Knowledge management (Logical Functional Dataflow Blank Diagram)

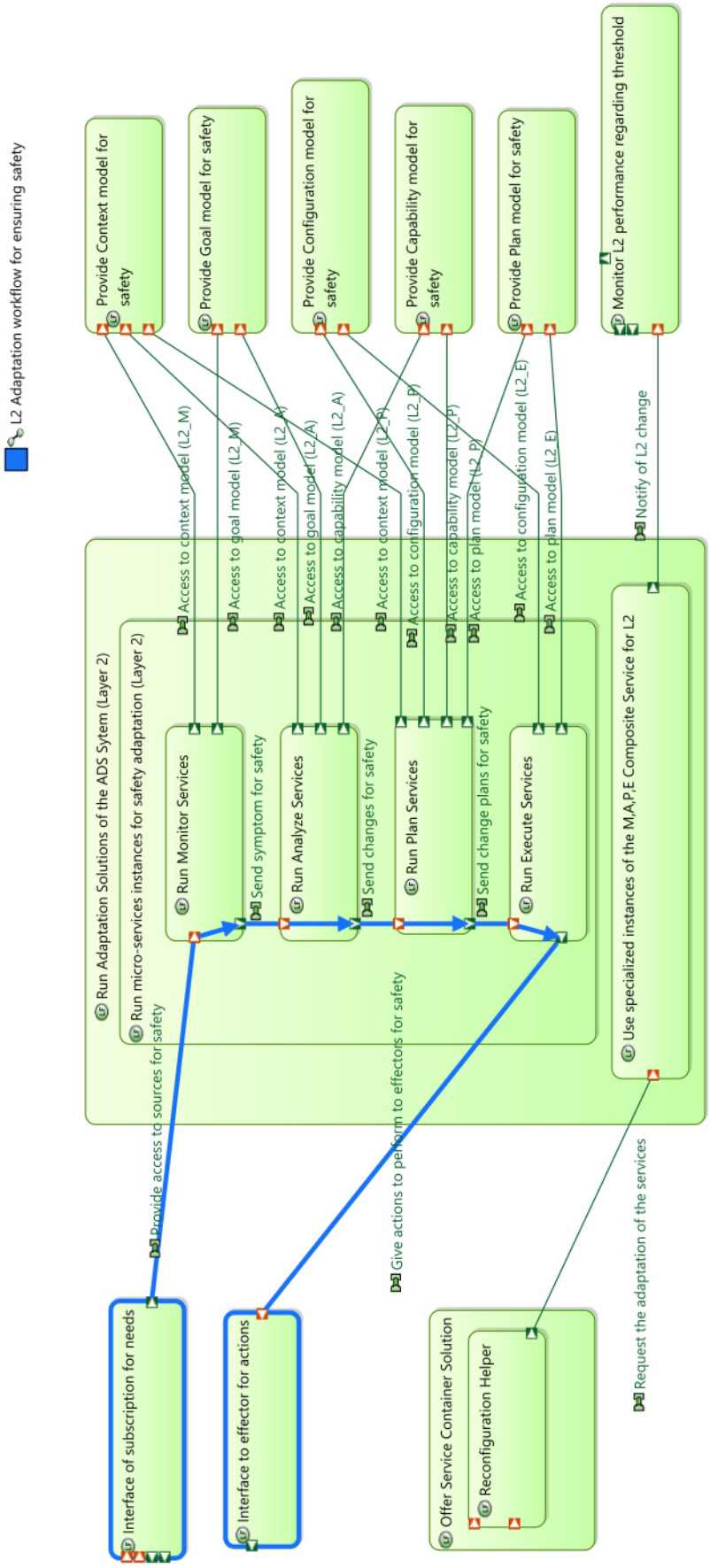


Figure 5.4 Logical Functional Dataflow Blank Diagram for the Safety assessment process (Layer 2). *Referred as B in the reference architecture.*

Safety Orchestrator decomposition in Monitor, Analyze, Plan and Execute

In a similar manner, the safety orchestrator perform a closed loop involving the M, A, P, E functions. The origin of the functional chain is threefold: the orchestration is performed either by the observations from the perceived context and components in ADCC (new context detected by a new symptom), either by a change in the managed ressources (ADCC components changes), and either by the unsufficient performance of a safety assessment process.

Top views of the Arcadia/Capella Model of the Framework

To conclude, the framework design and implementation can be summarized as Figure 5.6 where six main functions can be identified in this figure. The two levels that regroup the autonomic adaptations are designated as L2 and L3 whereas L1 corresponds to the interface with the existing system and vehicle platform. The knowledge management service provides interfaces for requests and reasoning results. Finally, the microservice management service proposes a framework as a library to run the allocated stereotypes and allows to have some customized introspection.

5.3 Case Study: a Pedestrian Crossing Application

This section is aimed at illustrating how to use the framework for a specific feature of interest: the pedestrian presence and how it ensures the safety of the situation by acting appropriately. We have defined a scenario based on data acquired from haulage. The framework is currently its second iteration and does not fully implement all the features. However, the framework can illustrate its behavioural adaptation and structural adaption for the pedestrian presence within the scope of this case study.

Presentation of the Case Study and Illustration

This case study focuses on assessing the risk management and generation of appropriate corrective adaptations by the framework in situations involving vulnerable entities as pedestrians. In this scope, the framework is implemented in connection with the existing structure of ADCC to acquire observation data from perception and provide recommendations or corrective trajectory to the navigation system. The tests are mainly carried out with simulated data on ROS and replayed data that are close to real-data.

Scenario definition: An example scenario description and safety assessment The case study is composed of three initial situations where the car start on the straight road and then engage into the crossing area as illustrated in Figure 5.7. The assests and nomenclature used to define and create the scenario with respect to the toolkit of the Voyage project[5]. Pedestrians are positioned in the crossing area along the straight road and far enough from the Ego vehicle detection range. At a certain position of Ego in the scenario, the pedestrian decides to move according: (A) Ego approaches a crosswalk. Pedestrian's calculated trajectory will be in the crosswalk when Ego is predicted to arrive.; (B) Ego approaches a crosswalk. Pedestrian is idle while being close to the pedestrian crossing. Pedestrian shows interest to cross the road.; (C) Ego approaches a pedestrian crossing the

[5]Using assets from Voyage testing toolkit available at https://oas.voyage.auto/testing-toolkit/

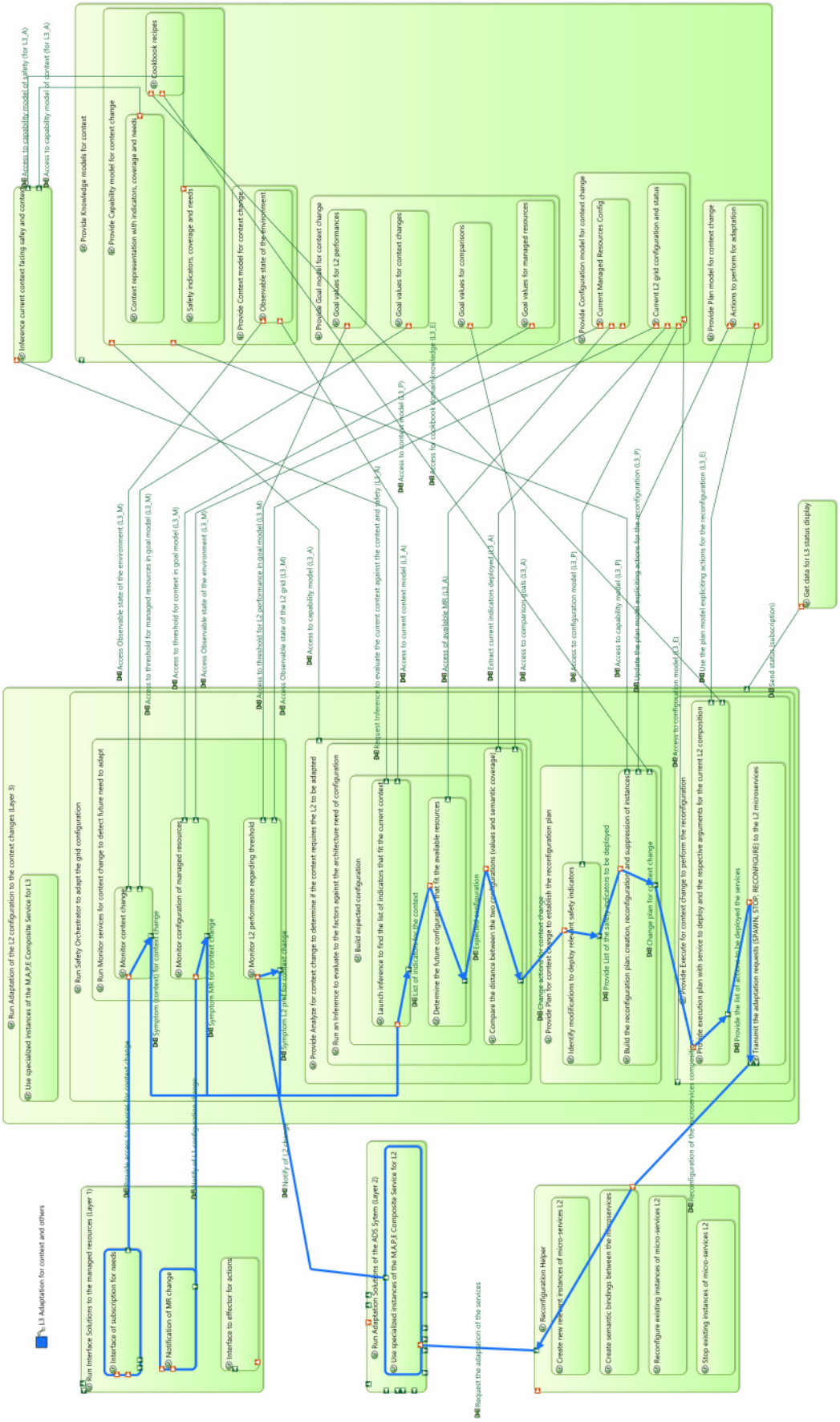


Figure 5.5 Logical Functional Dataflow Blank Diagram for the Safety Orchestrator (Layer 3). *Referred as C in the reference architecture.*

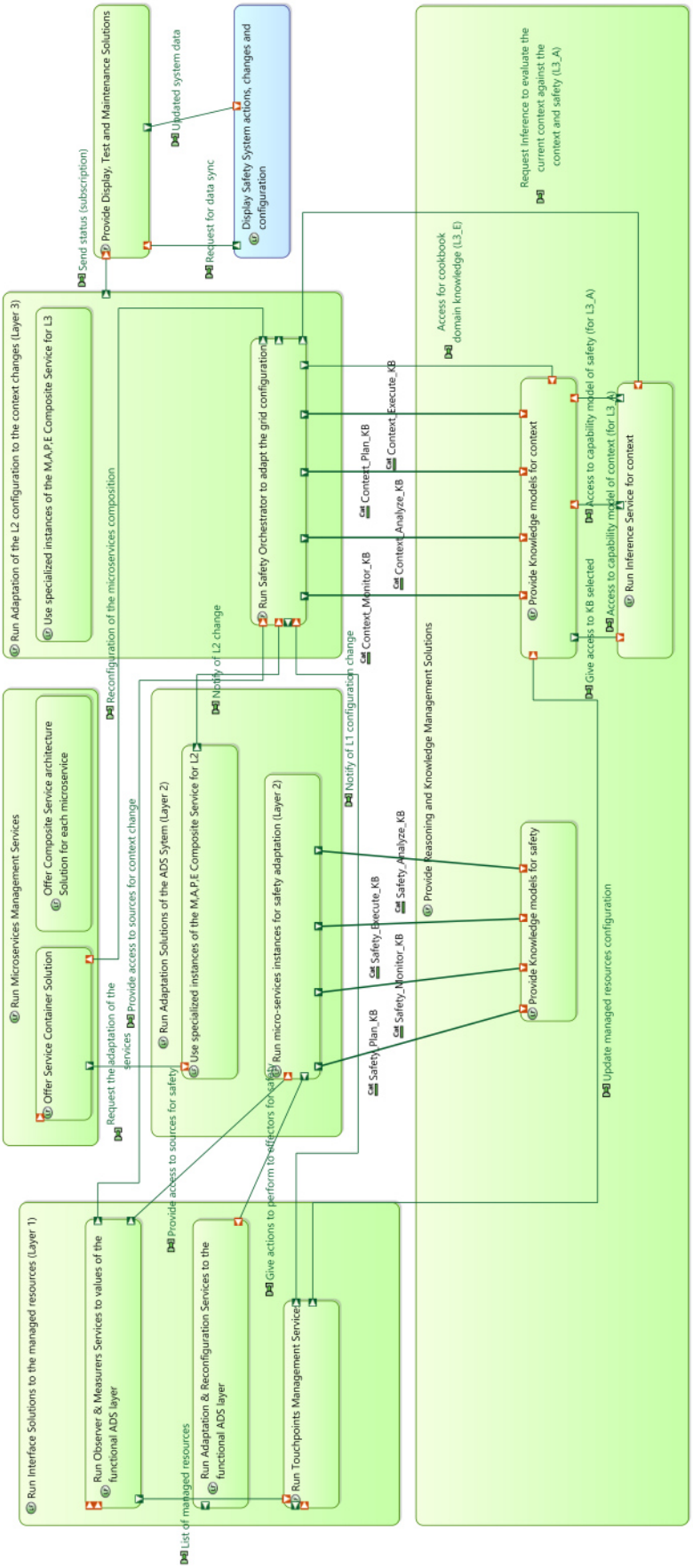
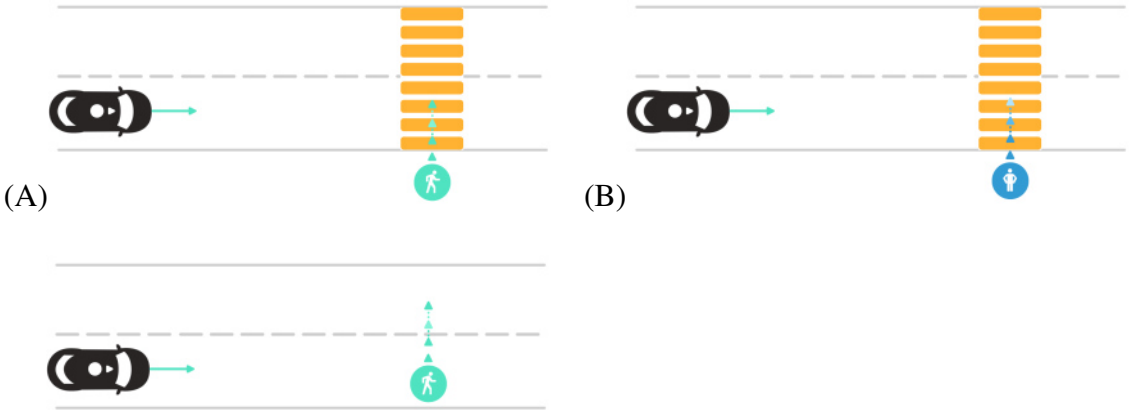


Figure 5.6 Top Level Logical Functional Dataflow view including the three layers and support systems.



(A) (B)

(C)

Figure 5.7 Illustrations of the case study involving different scenarios for pedestrian crossing.

road on no crosswalk. Pedestrian's calculated trajectory will not be in the crosswalk when Ego is predicted to arrive.

The feature of interest of pedestrian presence currently focuses on the activity of the pedestrians and determines through different processes if the pedestrian and the ego vehicle are safe (i.e. ensuring safe distance, ensuring appropriate Ego response).

In this thesis, the scenario is not expected to solely trigger an actual structural adaptation of the microservices since no major context change appears. However, such scenario would be possible if the initial condition were extended to an actual curve or involved different speed limitation that in final would require different configurations of the microservices being part of the safety assurance processes.

Application of the Framework

Methodology for Knowledge Bases Representation for the Case Study

In our ontology-driven approach, we can distinguish two specific usages for the ontology on how the framework should perform structural adaptation.

The first practice represents the framework structure and services composition based on the semantic description of the service-component (deployment recipes, capability properties, capability deployment...). We follow the joint OGC and W3C standard SOSA/SSN ontology [68] that describes sensors, observations, sampling, and actuation and applied it to our AV scope.

The second use aims to specify the behaviour of components upon the context. For this purpose, we need to detail the dependency between the actual service, the feature of interest, its property (functional or non-functional property), and the current context. These relations are defining a safety requirement according Waymo [153]. We have adopted partially the context decomposition proposed by [158] and the additional scene decorations in [18].

The ontology we are currently using is illustrated in Figure 5.8. The methodology we adopted for the creation of this structure and the safety requirements follows these 6 steps:

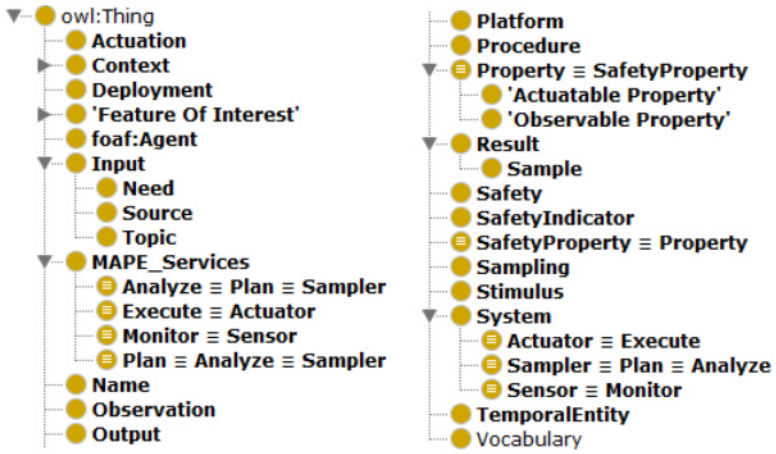


Figure 5.8 Ontology topology with SOSA/SSN integrated

1. Capture needs from safety indicators/requirements and features relative to risk assessment methods or motion planning.

2. Identify those needs in the sources of information in the functional layer (modules, events) and create associated touchpoints to access these sources of characterized information.

3. Represent the capability of these methods with respective local logic and databases. Their role needs to be represented across the discipline (e.g. safety).

4. Associate the needs of these capabilities to the respective semantic described service interfaces of those micro-services.

5. Ensure the consistency of the local graphs that are part of the knowledge representing the discipline and allocate the capabilities between instances of Monitor, Analyze, Plan, Execute and Touchpoints.

6. Specialize the composition and orchestration according the context (or system modes).

The set of scenarios presented in 5.7 enables the construction of our first models as ontologies that represent the operating context of the AV and the managed resources (i.e. configuration and accessibility of ADCC features). At this step, testing of the detection verifies their correct classification based on generated and simulated situation. Their rightful detection leads to the populating of the ontology with new instances for the specific situation during the operation of the system.

Then, each elicited scenario is analysed and bounded by safety analysis to produce a set of assessable run-time safety constraints. The decomposition into MAPE and symptoms, change request, etc., defines the autonomic processes that the microservices can perform. It specifies the required contents in the knowledge bases (i.e. relations between context and observations, symptoms for safety assessment). Thus, a second knowledge representation step provides ontologies representing the system regarding its structure, capabilities, data and component exchanges.

The context ontology is illustrated in Figure 5.9. It stores the symptoms from monitoring and enables the composition of abstracted context (e.g. a situation) with different road objects (e.g. Pedestrian, EgoCar) and relevant attributes. In addition, each of the road

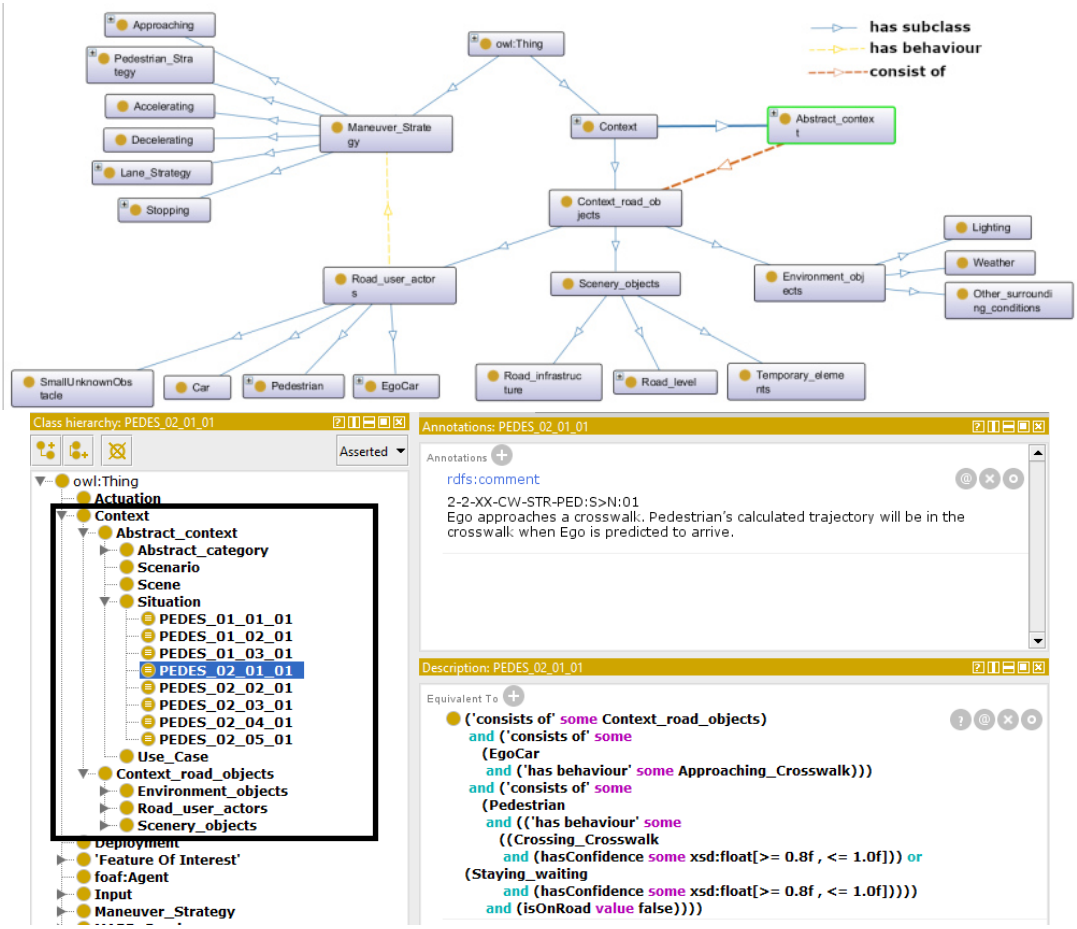


Figure 5.9 Ontology representation in the framework for context (top) and situations tagging equivalence (bottom)

users can have several maneuvers allocated (e.g. Decelerating, Stopping) with attached confidence to represent the uncertainty of perceived behaviours. The zone delimited by the black rectangle in Figure 5.9 corresponds to the different situations we have model and that we are trying to identify through the use of the reasoner on the stored symptoms. Inference can then be performed by evaluating the available individuals by the reasoner at run-time, as one situation identification rule. One situation detection rule is illustrated in the 'Equivalent To' field displayed on the bottom-right of the black rectangle of Figure 5.9.

A more detailled ontological representation of the other MAPE function is available in Appendix B.

Implementation of the Technical Solutions: First Feedback From Design Phase

The framework encompasses numerous different fields and interplay at different scale. The complexity of managing such structure can easily be misguiding and generate enormous amount of code. For this purpose, we have used the MBSE methodology and tool Arcadia/Capella to manage and use as much as existing libraries or API. The methodology Arcadia used iteratively have empowered the successive definition and refining of the architecture model in Capella up to code implementation in ROS. In addition, the

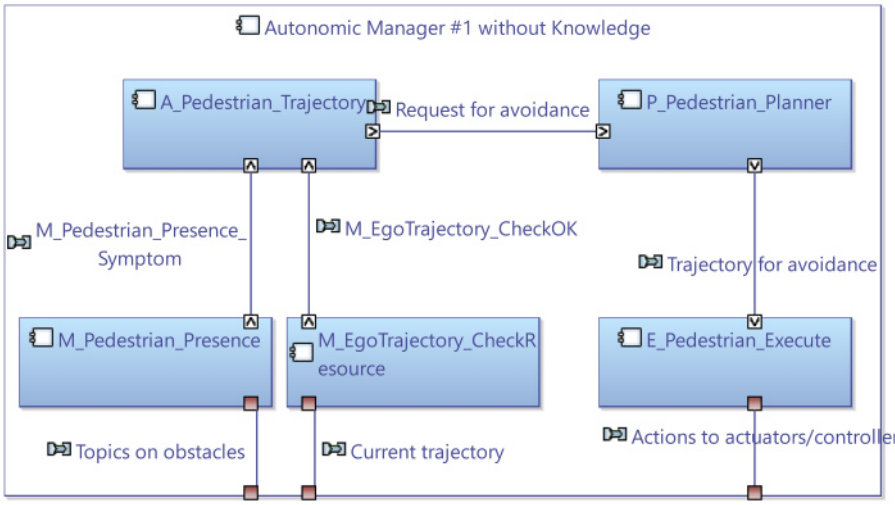


Figure 5.10 Composition of microservices for the behavioural adaptation

use of template state machines using SMACH[6], and AMOR[7] for a ROS service implementation to work with multiple ontologies at run-time, and several design patterns (i.e. abstraction, strategy, decorator and composite patterns) contributes to the readability and maintainability of the framework physical structure and code.

A such framework encompasses numerous different fields and at different scale. The complexity of managing a such project can easily misguided or enormous amount of code. For this purpose, we have used the MBSE tool Capella to manage and use as much as existing libraries or API.

The methodology Arcadia used iteratively have empowered the successive definition and refining of the architecture model in Capella up to code implementation in ROS.

On ROS, we are using SMACH for state machine (and templates), a ROS service implementation of AMOR framework to work with multiple ontologies and design patterns in our code to deal with abstraction, strategies, decorations and composite pattern.

Experimentation

Our first tests have proven that the integration of the framework to an existing ROS architecture is possible as adaptive actions were performed. To verify and validate the adaptations and the SMS behaviours, we have used and extend the simulator of ADCC navigation. This simulated environment helps in creating scenario involving the EgoCar vehicle, others road actors and road infrastructure and in assessing behaviours.

This simulation focuses on the scenario (A) illustrated in Figure 5.7 and aims to run the SMS for the set of microservices presented in Figure 5.10. These microservice have been represented in the ontology for this experiment and are implemented in respective ROS node following the stereotype. During the simulation, we expect the SMS to correctly detect the situation PEDES_02_01 so that the microservices get deployed and start to perform adequately their function resulting in adaptives actions.

[6]http://wiki.ros.org/smach
[7]https://github.com/EmaroLab/armor

Results Analysis

The analysis and scenario we present is in this section aim to show that the framework can correctly identify the situation, deploy the safety assessment process illustrated by Figure 5.10. The monitor interfaces publish all the data (state, consistency, heartbeat) that can be logged or inspect by a future micro-service. Monitored external KPI shows the difference in the perceive behaviour of the EgoCar and ADCC alone.

Analysis of a simulated scenario

To illustrate our findings, we propose to consider the scenario *PEDES_02_01* where "*Ego approaches a crosswalk. Pedestrian's calculated trajectory will be in the crosswalk when Ego is predicted to arrive*". We have formalized it as 2-2-XX-CW-STR-PED:S>N:01 in our scenario database. The detection of the situation correspond to "Detect pedestrian not on road close to or intending to cross crosswalk". The expected behaviour are the following: (1) Ego slows and comes to a stop in front of the crosswalk and remains stopped. (2) Ego proceeds through the crosswalk. Figure 5.11 illustrates this different steps the vehicle and pedestrian had during the record in the simulated environment: Vehicle is approaching the pedestrian crossing (keyframe 3), Detecting the pedestrian on the side of the road with the intent to cross (keyframe 5), Reacting to the pedestrian behaviour by slowing (keyframe 6), Pedestrian has crossed (keyframe 7), Vehicle crosses the pedestrian crossing and regains speed (keyframe 8-10),

In these screenshots of the simulated environment, the Vehicle is represented by a blue rectangle, the pedestrian by a square blue dot with a black circle and the pedestrian crossing by the white rectangle.

To illustrate the behaviour change with the framework, we propose to follow some physical and safety metrics obtained by the introspection of ADCC and our framework. Figure 5.12 presents the baseline with the measurements of key metrics for the non-altered vehicle behaviour with pedestrian. The two setups involved the simulation of the ADCC vehicle where velocity (speed in x axis), acceleration (in x axis) and detection of the pedestrian by the sensors are displayed. The scenario starts at 45 seconds with an initial acceleration to engage the vehicle on the road and to come closer to the pedestrian crossing. Maximum speed is attained at 54s. The pedestrian is then detected by the vehicle sensors at 57s which triggers the slowing behaviour (acceleration is negative and speed is deceasing). The vehicle is maintaining a slower velocity the closer the vehicle is to the pedestrian. At 63s, the pedestrian has been overtaken and the vehicle starts to attain its maximum speed.

To create a simulated scenario involving our framework, we have altered the ADCC behaviour to be more aggressive and susceptible to violate some safety constraints we have identified in our analysis in order to trigger our system. To this end, we have introduced a x1.3 factor to the original values and reduce the potential of acceleration. The results are illustrated in Figure 5.13. The framework intervenes with the detection of the PEDES_02_01 situation the moment we start to have pedestrian instances populating the ontology. This situation is then ignored after the pedestrian crossing has been passed. It is noticeable to see that the experience with only one pedestrian have created multiple instances as we keep some information history of the road users identified. The drop in the number of individual between 63s to 69s is the result of system we have implemented to overcome excessive numbers of instances that slows the inference. More limitations are detailled in the next section.

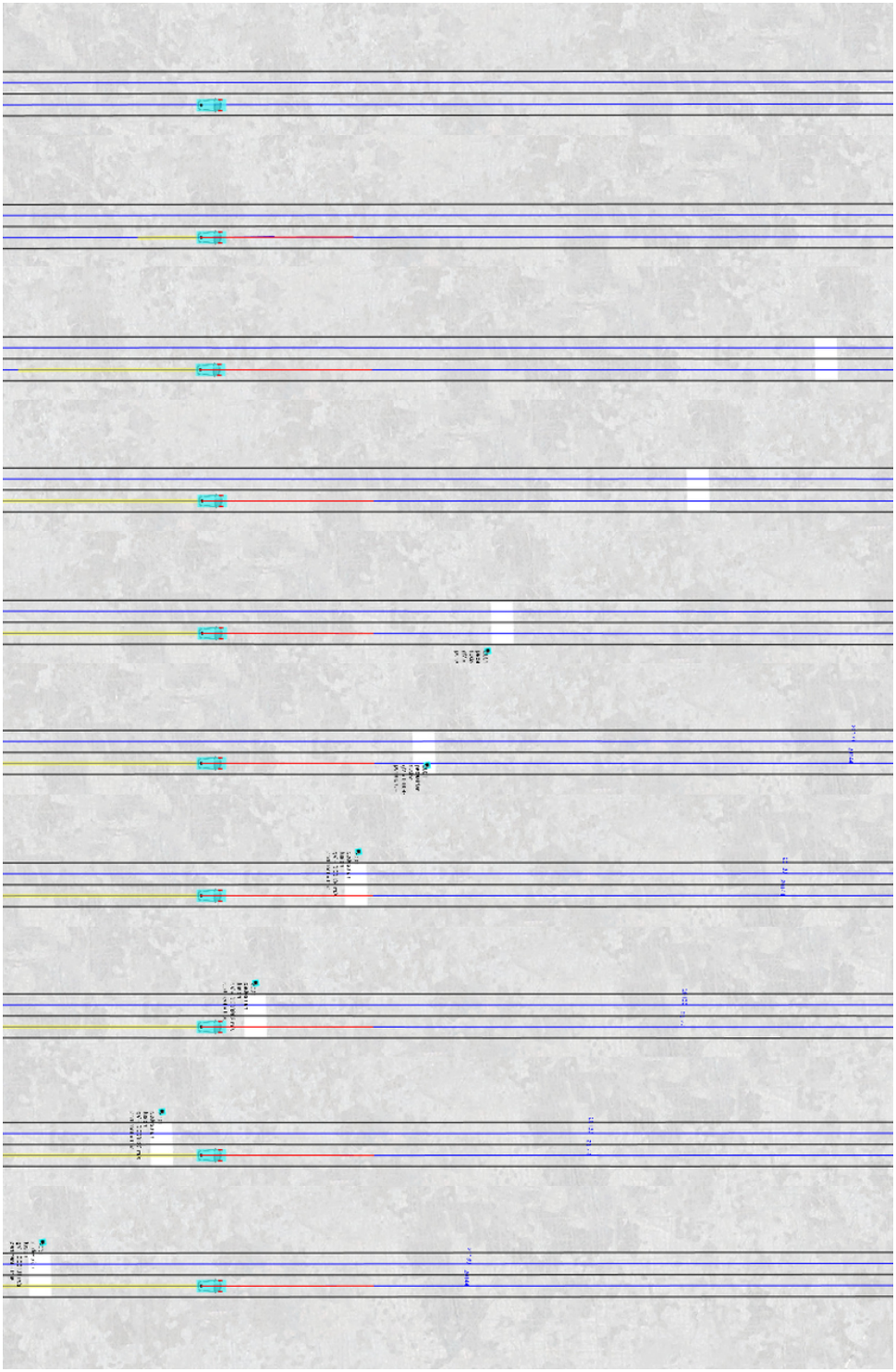


Figure 5.11 Key frames of the Scenario PEDES_02_01 in the simulated environment.

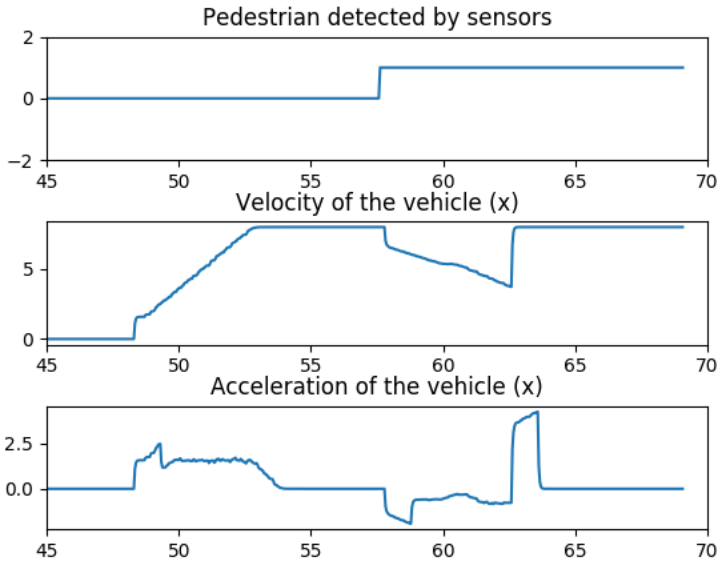


Figure 5.12 Observed metrics for the Scenario PEDES_02_01 for ADCC with agressive behaviour.

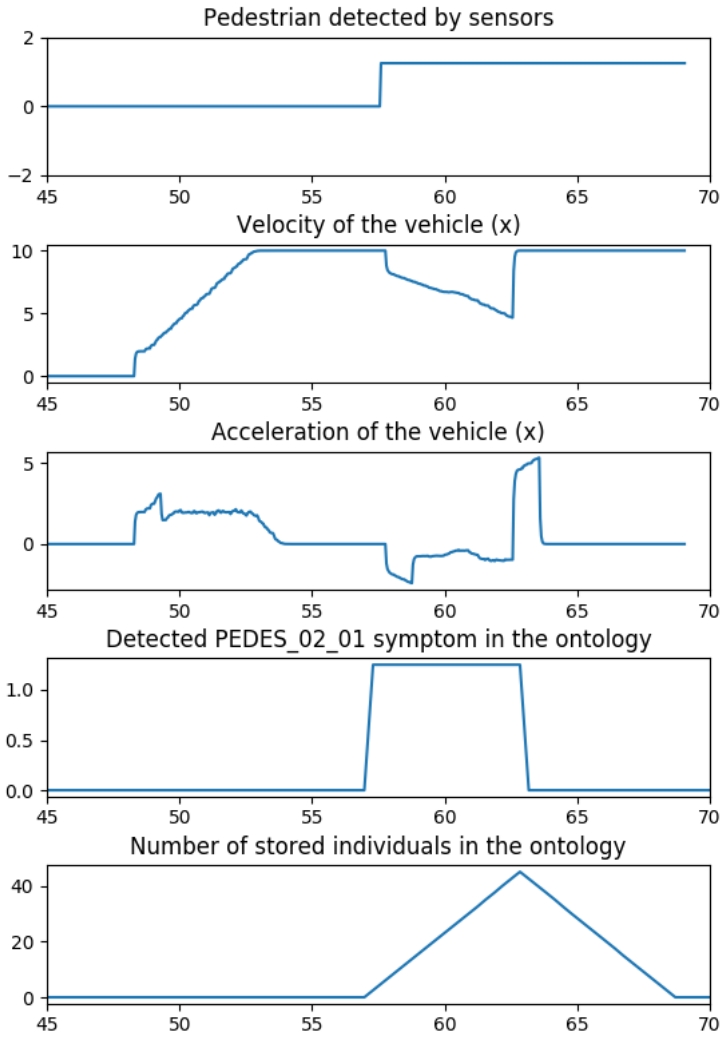


Figure 5.13 Observed metrics for the Scenario PEDES_02_01

Inherited and Overcome Limitations

During the implementation, we have met several difficulties and overcome some limitations using different solutions.

In the first place, the distance to pedestrian were never violated as ADCC was performing as too protective for the presented scenario. Consequently, we had to make ADCC more aggressive to trigger the appropriate adaptations and display the correct behaviour when corrected.

We had also difficulties storing observations to create symptoms in the ontology as the number of element were increasing significantly. For this purpose, we have created of a custom time window to limit the range of the elements concerned by the inference and regularly save the ontology for debugging purposes. To illustrate, in the case of more than one sensor, the observations and result can be stored with different timestamp (due to sensors rate acquisition or monitoring aggregation or correlation processing time). Thus, synchronization between the data were needed for the scene to contain most of the entities. We also limited the total number of element stored in the ontology for a road user concept to keep the inference time relevant for run-time operation.

Finally, the adoption of microservice required a rigorous design and implementation of the different nodes and library. Coordinating the whole system through its numerous node have been difficult the first time.

5.4 Conclusions

To assess a "safe by design" adaptive architecture, we have presented the efforts to systematically integrate and address the dynamic complexity from environment, context-dependent system's capabilities, uncertainty and safety. The safety for an AV is addressed as a system orchestration and composition problem. By using a systematic analysis and MBSE approach, we help specifying different aspects of software architecture development into the knowledge base that can be later be used by the system to perform self-adaptations. These system attributes and required knowledge are represented to offer a flexible, composable and observable autonomy. It allows for the system to assess and perform safely its driving behaviours.

The approach is applied to only address non-functional system properties related to the safety from a service perspective. Thus, the framework extends the existing ADCC managed system to provide higher observability and maintainability. The semantic description helps to expose the complexity of safety mitigation and management at run-time.

Current iteration of the framework and case study have successfully shown the applicability of the framework to AV safety. However, the scale of AV scenarios analysis and knowledge may require much human effort to build models and ontologies. Systems engineering tools could include the proposed framework to include from inception safety assurance at run-time while guaranteeing traceability and consistency.

Future work on this research needs to address additional perspectives to first fit future industry standards to be appropriately designed and implemented to include safety assessment procedures throughout the design and development life-cycles. For example, the manner of observing the context changes, providing interfaces for fault injection or testing the ability of the different services to learn to avoid and mitigate collisions. As-

sessment of the objectives, additional concerns and future work are discussed in the next conclusion chapter.

CHAPTER 6

Conclusions and Perspectives

"Somewhere, something incredible is waiting to be known."

- Carl Sagan, *Perhaps*

Contents

This chapter presents the major findings of this PhD thesis, which addresses the feasibility of self-adaptive microservice-oriented safety management framework into an existing autonomous vehicle system and its impacts on safety in a manageable and scalable manner. This chapter draws consistent conclusions based around the original objectives detailed in Chapter 1 and the NFPs identified in the literature (observability, traceability, reconfigurability and flexibility). Following the reviews of the reiterated research objectives, this chapter outlines the key contributions and achievements of this research. Then, discussions on some issues met and potential future work are presented. In closing, insights and additional perspectives that enrich this work are suggested.

6.1 Research Objectives Achievement

The main aim of this thesis has been to specify a framework that enables behavioural safety in a manageable and scalable manner. The refined aims and objectives for this framework were to identify all aspects and mechanisms necessary to assess internal and external AV safety as constraints with the non-functional properties of observability,

traceability, reconfigurability and flexibility in mind. The proposed framework results from the fulfillment of the following research objectives detailled in the next section:

- **Objective 1:** To examine in detail the safety implications related to AD functions and the methods used in different domains to ensure the safe operation of critical systems.
- **Objective 2:** To model all of the AV behaviours and structures of functions with a systematic approach that connects how the AV system reactions to the different internal or external events whilst it operates.
- **Objective 3:** To formulate a control mechanism that configures the functions determining the behaviour of the AV to maintain the safety constraints, whatever changes in the vehicle evolving context occurs.
- **Objective 4:** To include a hierarchical coordination function (i.e orchestration) to ensure a reduction in complexity and management of the different available mechanisms.
- **Objective 5:** To demonstrate the application of the proposed safety approach via a use case study.

6.2 Conclusions

This section demonstrates how the thesis objectives reiterated in Section 6.1 have been investigated and achieved. The benefits provided by this research are also discussed in relation with the practical application.

.

Understanding the key aspect of the puzzle that is Autonomy and Safety in Autonomous Vehicles

Key solutions to grasp and solve the safe autonomy puzzle have been presented in Chapter 1 and 2 with composed and personalized safety management and assurance solutions. Adopting a goal-based, context-based as well as an agnostic methodical approach is a suitable way to objectively measure and assess how one AV comply with its intended functionalities, expected performances and meeting the perspectives of robustness, reliability, safety, ethics and social acceptance. The identified NFPs also advocates for more transparency, a minimum performance and long term perspectives to progress towards safe autonomy.

These approaches need to combine (1) the new architectural structures (create reconfigurable and flexible supervision for holistic safety), (2) the representation of the safety knowledge (concepts of what it is to be safe via independently monitored behavioural requirements like metrics or rules), (3) the system engineering methodology and processes (integration and enforcement of qualities for safe system design) and (4) their rightful implementation (programming languages, capabilities, tools and patterns to fit the new needs). The appropriate composition of these aspects and the orchestration of the components shall guide the design of AV towards the creation of consciousness, self-awareness, and self-organization for the AV safety.

Identifying AV behaviours in relation with safety constraints from composed hazard analysis approaches

Chapter 3 have explored the way to study and build L4-L5 AV using MBSE with the key functionalities and safety considerations as design and run-time mitigation. The literature have shown that different hazard analysis can be combined and applied to identify safety constraints that eliminate the hazards by design and control them through management. We have proposed to integrate the awareness and the result of the management strategy to the system through control loops that we called safety assessment processes.

Identifying suitable run-time supervision and management Architecture

The literature review about supervising safety in Chapters 2 and 3 presents approaches that examine the outcomes that may lead to violation of safety-critical properties of the system. They supervise system operation at run-time, adapt to uncertainties that may cause faults, and mitigate failures presents applicable ways to keep the system acceptably safe by using behavioural adaptations. Different classes of behavioural adaptations exist as they perform fault detection (i.e. detect the violations that can bring a system into an unsafe state) or failure mitigation (i.e. evaluate the risk, control and to change behaviour by necessary actions to recover to a safe state). These adaptations are the means to integrate, ensure and guarantee the safety constraints that result from the hazard analysis in the form of as safety assessment processes. Chapter 3 has explored the way to study and build self-adaptive systems to design L4-L5 AV using MBSE with the key functionalities and safety considerations of design and run-time mitigation.

Self-adaptive framework providing run-time management of safety assessment processes in a manageable and scalable manner

Chapter 4 has presented our 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 that compose, orchestrate and define the safety assessment processes. 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. 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. To reduce the complexity of the context representation, our work proposes an ontology to foster the integration of heterogeneous data stemming from road environment. This ontology also relates the context-dependance of the safety assessement processes.

Feasibility of monitoring, diagnosis and adaptation at run-time to real-time

Chapter 5 has detailed the application of both the introduced patterns and models for safety assessment in a case study involving pedestrians. This study considers a range of scenario and L4 driving system for which we apply the safety analysis methodology presented in Chapter 3. We also present the knowledge-based implementation in order to integrate this kind of scenarios. Our first results have demonstrated that the integration of the framework to an existing ROS architecture is feasible.

6.3 Discussions and Perspectives

Research conducted during this PhD thesis helped addressing some challenges which hinder the development of systems for autonomous vehicles. The issues met and outcomes of our research work open important and interesting research perspectives. This section discusses about the different issues met during the thesis and introduces to possible perspectives.

Hazard Analysis issues

Hazard Analysis and Constraints Generation

Although the STPA contribute to generate numerous potentially hazardous scenarios and propagate the safety constraints by causality, STPA remains an iterative process and refinement that can be time consuming. In addition, analysis can still grow very large and become confusing without the use of an appropriate tool and versioning.

Constraints Stability

The hazard management required by the constraints shall not impact the system overall stability. For instance, multiple system adaptations in a short time-window may result in system instability. It may be the consequence of the overprovisioning or wrongdoing of the safety assurance processes, but also an inappropriate detection of the context.

Coherence and Non-blocking Constraints

Our design also imposes to formalize the safety constraints that we transform into safety assurance processes. In fact, it is necessary to verify that constraints occurring in the same context are not in conflict. The possible conflicts may be detected directly at the design by referring to the definition of the different safety assurance processes (actions and managed resources). In addition, additional check can be performed during the choice of SAP to deploy so that they or other dimensions do not conflict. This idea has yet to be studied.

Design Issues

Framework Tool chain for Correctness and Completeness

Our approach orients our framework to propose the system model similar to a digital twin of the architecture. Indeed, the model created in Capella/Arcadia and the implemented

systems are closed and shared most of their elements (components description, context-dependence). Creating a tool chain between the two would fusing the frontier between both representations. Such project can contribute to a higher flexibility in the design process (extension) and facilitating the traceability of the different non-functional properties and their operational context (via context-dependence relations).

Framework Rigorousness and Consolidation

However, we are far from ensuring that the conditions are in place to develop rigorous design flows. In fact, our design and implementation are containing some design flaws as our implementation is a mixed-SIL software without isolation (anti pattern).

Microservice also results in the addition of system complexity if the architecture and code is not well managed and rigorous. However, the microservices' flexibility still are worth the complexity and communication load we are adding by using them. It ease the frequent functionality change (orchestration) and facilitate the interaction with the operational context (composition). In our case, the context dependence is also identified that serves the traceability in design and during system operation.

Framework and MBSE: Applications of Modeling Processes

The model of the framework presented in the different chapter have been done progressively and by iteration with Arcadia/Capella. As we were focusing the architecture design first, we may have partially integrate concerns of other levels during the function refinement (i.e. working with "grey" boxes in System Analysis, or performing some allocation in the Logical Architecture). In general, we have tried to avoid common pitfalls and anti-pattern in the way to work and to model. However, we are far from ensuring that the conditions are in place to develop rigorous design flows.

Another interesting perceptive with the framework would have been the shift of most of the ADCC components and functions to microservices following the stereotype.

Semi-formal Interface Between Services

At the start of the thesis, we have introduced binding contracts between services where semantic mediator instance analyzes the service needs and bind it to the appropriate sources. This concept have been represented in the touchpoint level (L1) of the Capella framework model. However, this feature is not implemented yet and replaced by the ROS communication layer as it was not mandatory for a simple proof of concept.

Introducing Learning on Safety in the Framework

Learning is a key for autonomous systems to adapt to totally new contexts. The creation of orchestration of the meta-learning and meta-understanding of the system can be investigated to pursue safety assurance.

Implementation issues

Ontologies as a Choice for Symbolic Knowledge Representation

Ontologies have their advantages to model the domain of application with inference on the concepts, relations and instances. However, a close attention to the system architecture

and design should be paid to ensure appropriate data storage, inference and management such that the right consistency (strong or eventual) is achieved.

Their ability to be used to learn new knowledge remains limited. Some research such as Deep Learning for Ontology Reasoning (RDFox) may be relevant.

Additional challenging issues have also been experienced during the analysis, design and implementation that lead to the creation of new appropriate solutions (e.g. asynchronous storage of the perceived objects, expressiveness of the OWL2 for the inference, performance limitation of ontologies for real-time operations). They are detailed in Section 5.3.

Perspectives

In closing, our research outcomes open important and interesting research perspectives.

A final perspective would be to replace some of our symbolic implemented functions with probabilistic reasoning. While the composition and the orchestration of the framework remains, this type of multiple "black box" approach might challenge end-to-end solution in terms of performance. However, it would definitely provide sufficient observability and traceability if deployed with respect to the framework. Further, the ability to learn would be facilitated (e.g. detecting X) while its conditions of training (e.g. learning to detect X in context Y, preconditions, post-conditions) are already defined for specific contexts of operation. The solution may reside in providing a framework were they can appropriately expand and operated while being supervised at run-time to assess their good performances and meeting their objectives (e.g. contract-based appropriate reactions).

Bibliography

[1] Iso 26262 - road vehicles – functional safety, 2018. URL http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=43464.

[2] ISO/WD PAS 21448 - road vehicles - safety of the intended functionality., January 2019. URL https://www.iso.org/standard/70939.html.

[3] Asim Abdulkhaleq and Stefan Wagner. Integrated Safety Analysis Using Systems-Theoretic Process Analysis and Software Model Checking. In Floor Koornneef and Coen van Gulijk, editors, *Computer Safety, Reliability, and Security*, pages 121–134, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24255-2.

[4] Asim Abdulkhaleq, Stefan Wagner, Daniel Lammering, Hagen Boehmert, and Pierre Blueher. Using STPA in compliance with ISO 26262 for developing a safe architecture for fully automated vehicles. *CoRR*, abs/1703.03657, 2017. URL http://arxiv.org/abs/1703.03657.

[5] Dhaminda B. Abeywickrama and Eila Ovaska. Reflexive and evolutional digital service ecosystems with models at runtime. *CEUR Workshop Proceedings*, 2019: 184–192, 1 2017. ISSN 1613-0073.

[6] Rachel Abrams and Annalyn Kurtz. Joshua brown, who died in self-driving accident, tested limits of his tesla. *The New York Times*, July 2016. URL https://www.nytimes.com/2016/07/02/business/joshua-brown-technology-enthusiast-tested-the-limits-of-his-tesla.html.

[7] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand. An architecture for autonomy. *The International Journal of Robotics Research*, 17(4):315–337, apr 1998. doi: 10.1177/027836499801700402. URL http://dx.doi.org/10.1177/027836499801700402.

[8] Nourhène Alaya, Sadok Ben Yahia, and Myriam Lamolle. What makes ontology reasoning so arduous?: Unveiling the key ontological features. In *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics*, WIMS '15, pages 4:1–4:12, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3293-4. doi: 10.1145/2797115.2797117. URL http://doi.acm.org/10.1145/2797115.2797117.

[9] J. S. Albus. Outline for a theory of intelligence. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(3):473–509, May 1991. ISSN 0018-9472. doi: 10.1109/21.97471.

[10] James S Albus, Hui-Min Huang, Elena R Messina, Karl Murphy, Maris Juberts, Alberto Lacaze, Stephen B Balakirsky, Michael O Shneier, Tsai Hong Hong, Harry A Scott, et al. 4d/rcs version 2.0: A reference model architecture for unmanned vehicle systems. *NIST Interagency/Internal Report (NISTIR)-6910*, 2002.

[11] Tiago Amorim, Denise Ratasich, Georg Macher, Alejandra Ruiz, Daniel Schneider, Mario Driussi, Radu Grosu, and Andrea Hoeller. 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*, pages 137–168. IGI Global, 2018. doi: 10.4018/978-1-5225-2845-6.ch006.

[12] P. J. Antsaklis, K. M. Passino, and S. J. Wang. Towards intelligent autonomous control systems: Architecture and fundamental issues. *Journal of Intelligent and Robotic Systems*, 1(4):315–342, Dec 1989. ISSN 1573-0409. doi: 10.1007/BF00126465. URL https://doi.org/10.1007/BF00126465.

[13] Alexandre Armand. Situation Understanding and Risk Assessment Framework for Preventive Driver Assistance. *IV'14*, (2016SACLY008), #May# 2016. URL https://pastel.archives-ouvertes.fr/tel-01421917.

[14] Alexandre Armand, David Filliat, and Javier Ibañez-Guzmán. Ontology-based context awareness for driving assistance systems. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 227–233, June 2014. doi: 10.1109/IVS.2014.6856509.

[15] Uwe Aßmann, Sebastian Götz, Jean-Marc Jézéquel, Brice Morin, and Mario Trapp. *A Reference Architecture and Roadmap for Models@run.time Systems*, pages 1–18. Springer International Publishing, Cham, 2014. ISBN 978-3-319-08915-7. doi: 10.1007/978-3-319-08915-7_1. URL https://doi.org/10.1007/978-3-319-08915-7_1.

[16] K.J. Åström and B. Wittenmark. *Adaptive Control: Second Edition*. Dover Books on Electrical Engineering. Dover Publications, 2013. ISBN 9780486319148. URL https://books.google.fr/books?id=4CLCAgAAQBAJ.

[17] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl Landwehr. Basic concepts and taxonomy of dependable and secure computing. *Dependable and Secure Computing, IEEE Transactions on*, 1(1):11–33, 2004. doi: 10.1109/TDSC.2004.2. URL http://drum.lib.umd.edu/bitstream/handle/1903/6459/TR_2004-47.pdf?sequence=1.

[18] Gerrit Bagschik, Till Menzel, and Markus Maurer. Ontology based scene creation for the development of automated vehicles. *CoRR*, abs/1704.01006, 2017. URL http://arxiv.org/abs/1704.01006.

[19] Gerrit Bagschik, Torben Stolte, and Markus Maurer. Safety analysis based on systems theory applied to an unmanned protective vehicle. *Procedia Engineering*, 179:61 – 71, 2017. ISSN 1877-7058. doi: http://dx.doi.org/10.1016/j.proeng.2017.03.096. URL http://www.sciencedirect.com/science/article/pii/S1877705817312122. 4th European {STAMP} Workshop 2016, {ESW} 2016, 13-15 September 2016, Zurich, Switzerland.

[20] Gerrit Bagschik, Marcus Nolte, Susanne Ernst, and Markus Maurer. A system's perspective towards an architecture framework for safe automated vehicles. *CoRR*, abs/1804.07020, 2018. URL http://arxiv.org/abs/1804.07020.

[21] Anand Balakrishnan, Aniruddh G. Puranic, Xin Qin, Adel Dokhanchi, Jyotirmoy V. Deshmukh, Heni Ben Amor, and Georgios Fainekos. Specifying and evaluating quality metrics for vision-based perception systems. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, mar 2019. doi: 10.23919/date.2019.8715114.

[22] Christian Basarke, Christian Berger, and Bernhard Rumpe. Software & systems engineering process and tools for the development of autonomous driving intelligence. *CoRR*, abs/1409.7121, 2014. URL http://arxiv.org/abs/1409.7121.

[23] Sagar Behere. *Architecting autonomous automotive systems: With an emphasis on cooperative driving*. PhD thesis, KTH Royal Institute of Technology, 2013.

[24] Sagar Behere and Martin Törngren. A functional reference architecture for autonomous driving. *Information and Software Technology*, 73:136 – 150, 2016. ISSN 0950-5849. doi: http://dx.doi.org/10.1016/j.infsof.2015.12.008. URL https://sagar.se/files/wasa2015.pdf.

[25] Sagar Behere and Martin Törngren. *Systems Engineering and Architecting for Intelligent Autonomous Systems*, chapter 13, pages 313–351. Springer International Publishing, Cham, 2017. ISBN 978-3-319-31895-0. doi: 10.1007/978-3-319-31895-0_13. URL https://doi.org/10.1007/978-3-319-31895-0_13.

[26] Sagar Behere, Fredrik Asplund, Andreas Söderberg, and Martin Törngren. Architecture challenges for intelligent autonomous machines. In Emanuele Menegatti, Nathan Michael, Karsten Berns, and Hiroaki Yamaguchi, editors, *Intelligent Autonomous Systems 13*, pages 1669–1681, Cham, 2016. Springer International Publishing. ISBN 978-3-319-08338-4.

[27] Nelly Bencomo, Robert France, Betty H. C. Cheng, and Uwe AÃmann. *Models@run.time: Foundations, Applications, and Roadmaps*. Lecture Notes in Computer Science 8378 Programming and Software Engineering. Springer International Publishing, 1 edition, 2014. ISBN 978-3-319-08914-0,978-3-319-08915-7. URL http://gen.lib.rus.ec/book/index.php?md5=FE3FB31034F4BF988FFCF0E220BA76F6.

[28] Christian Berger and Bernhard Rumpe. Engineering autonomous driving software. *Experience from the DARPA Urban Challenge*, pages 243–271, 2012.

[29] David Bissell. Automation interrupted: How autonomous vehicle accidents transform the material politics of automation. *Political Geography*, 65:57 – 66, 2018. ISSN 0962-6298. doi: https://doi.org/10.1016/j.polgeo.2018.05.003. URL http://www.sciencedirect.com/science/article/pii/S0962629817303943.

[30] National Transport Safety Board. Preliminary report highway hwy19fh008, March 2019. URL https://www.ntsb.gov/investigations/AccidentReports/Reports/HWY19FH008-preliminary.pdf.

[31] R. Peter Bonasso, R. James Firby, Erann Gat, David Kortenkamp, David P. Miller, and Mark G. Slack. Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental & Theoretical Artificial Intelligence*, 9(2-3):237–256, 1997. doi: 10.1080/095281397147103. URL https://doi.org/10.1080/095281397147103.

[32] Neal E. Boudette. Despite high hopes, self-driving cars are 'way in the future'. *The New York Times*, Jul 2019. URL https://www.nytimes.com/2019/07/17/business/self-driving-autonomous-cars.html.

[33] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation*, 2(1):14–23, March 1986. ISSN 0882-4967. doi: 10.1109/JRA.1986.1087032.

[34] Luca Buoncompagni, Alessio Capitanelli, and Fulvio Mastrogiovanni. A ROS multi-ontology references services: OWL reasoners and application prototyping issues. *CoRR*, abs/1706.10151, 2017. URL http://arxiv.org/abs/1706.10151.

[35] CA-DMV. Reports of Traffic Accidents Involving an Autonomous Vehicle - OL316, 2019. URL https://www.dmv.ca.gov/portal/wcm/connect/3946fbb8-e04e-4d52-8f80-b33948df34b2/Google+Auto+LLC+02.14.16.pdf?MOD=AJPERES.

[36] CA-DMV. Reports of Traffic Accidents Involving an Autonomous Vehicle - OL316, 2019. URL https://www.dmv.ca.gov/portal/dmv/detail/vr/autonomous/autonomousveh_ol316+.

[37] Betty H. C. Cheng, Kerstin I. Eder, Martin Gogolla, Lars Grunske, Marin Litoiu, Hausi A. Müller, Patrizio Pelliccione, Anna Perini, Nauman A. Qureshi, Bernhard Rumpe, Daniel Schneider, Frank Trollmann, and Norha M. Villegas. *Using Models at Runtime to Address Assurance for Self-Adaptive Systems*, pages 101–136. Springer International Publishing, Cham, 2014. ISBN 978-3-319-08915-7. doi: 10.1007/978-3-319-08915-7_4. URL https://doi.org/10.1007/978-3-319-08915-7_4.

[38] I. Colwell, B. Phan, S. Saleem, R. Salay, and K. Czarnecki. An automated vehicle safety concept based on runtime restriction of the operational design domain. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1910–1917, June 2018. doi: 10.1109/IVS.2018.8500530.

[39] SAE On-Road Automated Vehicle Standards Committee et al. Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems. Technical report, 2014.

[40] SAE On-Road Automated Vehicle Standards Committee et al. Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems. Technical report, SAE International, September 2016. Revision of J3016_201401.

[41] SAE On-Road Automated Vehicle Standards Committee et al. Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems. Technical report, SAE International, June 2018. Revision of J3016_201609.

[42] Shawn A. Cook, Hsing-Hua Fan, Krzysztof Pennar, and Padma Sundaram. Building behavioral competency into stpa process models for automated driving systems. March 2018.

[43] Ö. Ş. Taş, F. Kuhnt, J. M. Zöllner, and C. Stiller. Functional system architectures towards fully automated driving. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 304–309, June 2016. doi: 10.1109/IVS.2016.7535402.

[44] Romain Cuer. *Démarche de conception sûre de la Supervision de la fonction de Conduite Autonome*. PhD thesis, 2018. URL `http://www.theses.fr/2018LYSEI091`. ThÃšse de doctorat dirigÃ©e par Niel, Eric Automatique Lyon 2018.

[45] Codé Diop, Guillaume Dugué, Christophe Chassot, Ernesto Exposito, and Jorge Gomez. 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, nov 2012. doi: 10.1108/17427371211283001.

[46] Dmitri A. Dolgov and Christopher Paul Urmson. Us9367065b2. U.S. Patent, Jun 2016.

[47] Nicola Dragoni, Saverio Giallorenzo, Alberto Lluch-Lafuente, Manuel Mazzara, Fabrizio Montesi, Ruslan Mustafin, and Larisa Safina. Microservices: yesterday, today, and tomorrow. *CoRR*, abs/1606.04036, 2016. URL `http://arxiv.org/abs/1606.04036`.

[48] Hugh Durrant-Whyte. A critical review of the state-of-the-art in autonomous land vehicle systems and technology. In *Sandia report*, volume SAND200I-3685, Albuquerque, 2001. Sandia National Laboratories.

[49] Yrvann Emzivat. *Safety System Architecture for the Design of Dependable and Adaptable Autonomous Vehicles*. PhD thesis, 2018.

[50] Yrvann Emzivat, Javier Ibanez-Guzman, Herve Illy, Philippe Martinet, and Olivier H. Roux. A formal approach for the design of a dependable perception system for autonomous vehicles. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, nov 2018. doi: 10.1109/itsc.2018.8569903.

[51] Francesca Favarò, Sky Eurich, and Nazanin Nader. Autonomous vehicles' disengagements: Trends, triggers, and regulatory limitations. *Accident Analysis & Prevention*, 110:136 – 148, 2018. ISSN 0001-4575. doi: https://doi.org/10.1016/j.aap.2017.11.001. URL `http://www.sciencedirect.com/science/article/pii/S0001457517303822`.

[52] Donald Firesmith. Common concepts underlying safety, security, and survivability engineering. Technical Report CMU/SEI-2003-TN-033, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2003. URL `http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=6553`.

[53] Donald Firesmith. A taxonomy of safety-related requirements. In *Proceedings of the International Workshop on Requirements for High Assurance Systems*. International Workshop on High Assurance Systems (RHAS'05), 2005.

[54] FiveAI. Certification of highly automated vehicles for use on uk roads: Creating an industry-wide framework for safety. Technical report, FiveAI, 2019.

[55] Ford. A matter of trust fords approach to developing self-driving vehicles. techreport, Ford, 2018. URL https://media.ford.com/content/dam/fordmedia/pdf/Ford_AV_LLC_FINAL_HR_2.pdf.

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

[57] Laura Fraade-Blanar, Marjory S. Blumenthal, James M. Anderson, and Nidhi Kalra. Measuring automated vehicle safety: Forging a framework. Technical report, RAND Corporation, Santa Monica, CA, 2018. URL https://www.rand.org/pubs/research_reports/RR2662.html.

[58] Lex Fridman. Elon musk: Tesla autopilot | artificial intelligence (ai) podcast, Apr 2019. URL https://www.youtube.com/watch?v=dEv99vxKjVI.

[59] Erich Gamma. *Design patterns: elements of reusable object-oriented software*. Pearson Education India, 1995.

[60] Tom M. Gasser, Clemens Arzt, Mihiar Ayoubi, Arne Bartels, Jana Eier, Frank Flemisch, Dirk Häcker, Tobias Hesse, Werner Huber, Christine Lotz, Markus Maurer, Simone Ruth-Schumacher, Jürgen Schwarz, Wolfgang Vogt, and Projektgruppe Rechtsfolgen zunehmender Fahrzeugautomatisierung. Rechtsfolgen zunehmender fahrzeugautomatisierung (bast-bericht f 83). techreport vol. 83., BASt, Wirtschaftsverl. NW Verl. fÃŒr neue Wissenschaft, Bremerhaven, 2012. URL http://bast.opus.hbz-nrw.de/volltexte/2012/587/pdf/F83.pdf. Gemeinsamer Schlussbericht der Projektgruppe. Berichte der Bundesanstalt fÃŒr Strassenwesen - Fahrzeugtechnik (F).

[61] Erann Gat. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, AAAI'92, pages 809–815. AAAI Press, 1992. ISBN 0-262-51063-4. URL http://dl.acm.org/citation.cfm?id=1867135.1867260.

[62] General Motors. 2018 self-driving safety report. Technical report, General Motors Inc., 2018. URL https://www.gm.com/content/dam/company/docs/us/en/gmcom/gmsafetyreport.pdf.

[63] Xinli Geng, Huawei Liang, Biao Yu, Pan Zhao, Liuwei He, and Rulin Huang. A scenario-adaptive driving behavior prediction approach to urban autonomous driving. *Applied Sciences*, 7:426, 04 2017. URL http://www.mdpi.com/2076-3417/7/4/426.

[64] S. Geyer, M. Baltzer, B. Franz, S. Hakuli, M. Kauer, M. Kienle, S. Meier, T. Weissgerber, K. Bengler, R. Bruder, F. Flemisch, and H. Winner. 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, May 2014. ISSN 1751-956X. doi: 10.1049/iet-its.2012.0188.

[65] Samuel Gibbs. Crash involving self-driving google car injures three employees. *The Guardian*, July 2015. URL https://www.theguardian.com/technology/2015/jul/17/crash-self-driving-google-car-injures-three.

[66] Samuel Gibbs. Tesla model s cleared by auto safety regulator after fatal autopilot crash. *The Guardian*, January 2017. URL https://www.theguardian.com/technology/2017/jan/20/tesla-model-s-cleared-auto-safety-regulator-after-fatal-autopilot-cr

[67] Holger Giese, Nelly Bencomo, Liliana Pasquale, Andres J. Ramirez, Paola Inverardi, Sebastian Wätzoldt, and Siobhán Clarke. *Living with Uncertainty in the Age of Runtime Models*, pages 47–100. Springer International Publishing, Cham, 2014. ISBN 978-3-319-08915-7. doi: 10.1007/978-3-319-08915-7_3. URL https://doi.org/10.1007/978-3-319-08915-7_3.

[68] Armin Haller, Krzysztof Janowicz, Simon JD Cox, Maxime Lefrançois, Kerry Taylor, Danh Le Phuoc, Joshua Lieberman, Raúl García-Castro, Rob Atkinson, and Claus Stadler. 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, 2018. doi: 10.3233/SW-180320.

[69] C. Hubmann, M. Becker, D. Althoff, D. Lenz, and C. Stiller. Decision making for autonomous driving considering interaction and uncertain prediction of surrounding vehicles. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1671–1678, June 2017. doi: 10.1109/IVS.2017.7995949.

[70] Markus C Huebscher and Julie A McCann. A survey of autonomic computing – degrees, models, and applications. *ACM Computing Surveys (CSUR)*, 40(3):7, 2008.

[71] Javier Ibañez-Guzmán, Christian Laugier, John-David Yoder, and Sebastian Thrun. *Autonomous Driving: Context and State-of-the-Art*, pages 1271–1310. Springer London, London, 2012. ISBN 978-0-85729-085-4. doi: 10.1007/978-0-85729-085-4_50. URL https://doi.org/10.1007/978-0-85729-085-4_50.

[72] J.Kaplan, R. Glon, S. Edelstein, and L. Chang. Deadly uber crash was 'entirely avoidable' had the driver not been watching hulu. *Digital Trends*, 2018. URL https://www.digitaltrends.com/cars/self-driving-uber-crash-arizona/.

[73] Rolf Johansson, Jonas Nilsson, Carl Bergenhem, Sagar Behere, Jörgen Tryggvesson, Stig Ursing, Andreas Söderberg, Martin Törngren, and Fredrik Warg. *Functional Safety and Evolvable Architectures for Autonomy*, pages 547–560. Springer International Publishing, Cham, 2017. ISBN 978-3-319-31895-0. doi: 10.1007/978-3-319-31895-0_25. URL https://doi.org/10.1007/978-3-319-31895-0_25.

[74] Nidhi Kalra and Susan M. Paddock. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? Technical report, RAND Corporation, 2016. URL http://www.sciencedirect.com/science/article/pii/S0965856416302129.

[75] Aaron Kane, Omar Chowdhury, Anupam Datta, and Philip Koopman. A case study on runtime monitoring of an autonomous research vehicle (arv) system. In Ezio Bartocci and Rupak Majumdar, editors, *Runtime Verification*, pages 102–117, Cham, 2015. Springer International Publishing. ISBN 978-3-319-23820-3.

[76] Viktor Kaznov, Johan Svahn, Per Roos, Fredrik Asplund, Sagar Behere, and Martin Törngren. *Architecture and Safety for Autonomous Heavy Vehicles: ARCHER*, pages 571–581. Springer International Publishing, Cham, 2017. ISBN 978-3-319-31895-0. doi: 10.1007/978-3-319-31895-0_27. URL https://doi.org/10.1007/978-3-319-31895-0_27.

[77] J Kephart, D Chess, Craig Boutilier, Rajarshi Das, and William E Walsh. An architectural blueprint for autonomic computing. *IBM White paper*, June 2006. doi: 10.1.1.150.1011. URL https://pdfs.semanticscholar.org/0e99/837d9b1e70bb35d516e32ecfc345cd30e795.pdf.

[78] R. Koh-Dzul, M. Vargas-Santiago, C. Diop, E. Exposito, and F. Moo-Mena. 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*, pages 416–421, Dec 2013. doi: 10.1109/UIC-ATC.2013.35.

[79] P. Koopman and M. Wagner. Autonomous vehicle safety: An interdisciplinary challenge. *IEEE Intelligent Transportation Systems Magazine*, 9(1):90–96, Spring 2017. ISSN 1939-1390. doi: 10.1109/MITS.2016.2583491.

[80] Phil Koopman. An Overview of Draft UL 4600: Standard for Safety for the Evaluation of Autonomous Products. *Medium*, Jun 2019. URL https://medium.com/@pr_97195/an-overview-of-draft-ul-4600-standard-for-safety-for-the-evaluat

[81] Philip Koopman. Autonomous vehicles: Safety validation and edge case testing, June 2018. URL https://users.ece.cmu.edu/~koopman/lectures/2018_chinaav.pdf.

[82] Philip Koopman. The big picture for self-driving car safety, April 2019.

[83] Philip Koopman and Frank Fratrik. How many operational design domains, objects, and events? *Safe AI 2019: AAAI Workshop on Artificial Intelligence Safety*, January 2019. URL https://users.ece.cmu.edu/~koopman/pubs/Koopman19_SAFE_AI_ODD_OEDR.pdf.

[84] Philip Koopman and Michael Wagner. Toward a framework for highly automated vehicle safety validation. In *SAE Technical Paper*. SAE International, 04 2018. doi: 10.4271/2018-01-1071. URL https://doi.org/10.4271/2018-01-1071.

[85] Philip Koopman, Aaron Kane, and Jen Black. Credible autonomy safety argumentation. 2019.

[86] Philip Koopman, Beth Osyk, and Jack Weast. Autonomous vehicles meet the physical world: RSS, variability, uncertainty, and proving safety. In *Lecture Notes in Computer Science*, pages 245–253. Springer International Publishing, 2019. doi: 10.1007/978-3-030-26601-1_17. URL https://www.slideshare.net/PhilipKoopman1/autonomous-vehicles-meet-the-physical-world-rss-practical-experience

[87] John Krafcik. Waymo one: The next step on our self-driving journey. *Medium*, Dec 2018. URL https://medium.com/waymo/waymo-one-the-next-step-on-our-self-driving-journey-6d0c075b0e9b.

[88] John E. Laird, Allen Newell, and Paul S. Rosenbloom. Soar: An architecture for general intelligence. *Artificial Intelligence*, 33(1):1 – 64, 1987. ISSN 0004-3702. doi: https://doi.org/10.1016/0004-3702(87)90050-6. URL http://www.sciencedirect.com/science/article/pii/0004370287900506.

[89] Edward A Lee. Cyber physical systems: Design challenges. In *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, pages 363–369. IEEE, 2008.

[90] Stéphanie Lefèvre. *Risk estimation at road intersections for connected vehicle safety applications*. PhD thesis, INRIA Grenoble, 10 2012.

[91] Stéphanie Lefèvre, Christian Laugier, and Javier Ibañez-Guzmán. Risk assessment at road intersections: Comparing intention and expectation. In *2012 IEEE Intelligent Vehicles Symposium*, pages 165–171, June 2012. doi: 10.1109/IVS.2012.6232198.

[92] Stéphanie Lefèvre, Dizan Vasquez, and Christian Laugier. A survey on motion prediction and risk assessment for intelligent vehicles. *ROBOMECH Journal*, 1 (1):1, 2014. ISSN 2197-4225. doi: 10.1186/s40648-014-0001-z. URL http://dx.doi.org/10.1186/s40648-014-0001-z.

[93] Nancy Leveson. A systems approach to risk management through leading safety indicators. *Reliability Engineering & System Safety*, 136:17–34, apr 2015. doi: 10.1016/j.ress.2014.10.008.

[94] Nancy Leveson. Engineering a safer and more secure world, March 2018.

[95] Nancy G Leveson. *Safeware: System Safety and Computers*. Addison-Wesley, New York, NY, USA, 1995. ISBN 0-201-11972-2. URL http://ocw.alfaisal.edu/NR/rdonlyres/Aeronautics-and-Astronautics/16-358JSpring-2005/DD631F49-C42B-4804-8536-2EA6D67DC3FC/0/book_1_2.pdf.

[96] Nancy G. Leveson. An approach to designing safe embedded software. In Alberto Sangiovanni-Vincentelli and Joseph Sifakis, editors, *Embedded Software*, pages 15–29, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-45828-9.

[97] Nancy G. Leveson. *Engineering a Safer World - Systems Thinking Applied to Safety*. The MIT Press, 2012. URL https://mitpress.mit.edu/sites/

default/files/titles/free_download/9780262016629_Engineering_a_Safer_World.pdf.

[98] Nancy G. Leveson and John P. Thomas. *STPA Handbook*. MIT Partnership for a Systems Approach to Safety (PSAS, March 2018. URL http://psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf.

[99] Nancy G. Leveson, John P. Thomas, and MIT. *STPA Primer*. MIT Partnership for a Systems Approach to Safety (PSAS), 2015. URL sunnyday.mit.edu/STPA-Primer-v0.pdf.

[100] Matt Luckcuck, Marie Farrell, Louise Dennis, Clare Dixon, and Michael Fisher. Formal Specification and Verification of Autonomous Robotic Systems: A Survey. *arXiv preprint arXiv:1807.00048*, 2018.

[101] J. Lygeros, D. N. Godbole, and M. E. Broucke. Design of an extended architecture for degraded modes of operation of ivhs. In *American Control Conference, Proceedings of the 1995*, volume 5, pages 3592–3596 vol.5, Jun 1995. doi: 10.1109/ACC.1995.533806.

[102] Michael T. Fisher Martin L. Abbott. *The Art of Scalability*. Addison Wesley, 2015. ISBN 0134032802. URL https://www.ebook.de/de/product/23633910/martin_l_abbott_michael_t_fisher_the_art_of_scalability.html.

[103] M.Sc. Matthew Wood, Dr. Philipp Robbel, Dr. Michael Maass, Dr. Radboud Duintjer Tebbens, M.Sc. Marc Meijs, M.Sc. Mohamed Harb, B.Sc. Jonathon Reach, Karl Robinson, M.Sc. David Wittmann, M.Sc. Toshika Srivastava, Dr.-Ing. Mohamed Essayed Bouzouraa, MBA Siyuan Liu, BS, MA Yali Wang, Dr.-Ing. Christian Knobel, Dipl.-Inf. David Boymanns, Dr.-Ing. Matthias Löhning, Dr. Bernhard Dehlink, M.Sc. Dirk Kaule, Dipl.-Ing. Richard Krüger, Dr. Jelena Frtunikj, Dr. Florian Raisch, Dipl.-Math. Miriam Gruber, M.Sc. Jessica Steck, Dipl.-Psych. Julia Mejia-Hernandez, Dipl.-Ing. Sandro Syguda, Dipl.-Ing. Pierre Blüher, Dr.-Ing. Kamil Klonecki, Dr. Pierre Schnarz, Dr. Thomas Wiltschko, Dipl.-Inf. Stefan Pukallus, Dr.-Ing. Kai Sedlaczek, M.Sc. Neil Garbacik, BSAE David Smerza, Dr. Dalong Li, Dr. Adam Timmons, Marco Bellotti, BS Michael O'Brien, Michael Schöllhorn, Dipl.-Ing. Udo Dannebaum, M.Sc. Jack Weast, BS, BS Alan Tatourian, Dr.-Ing. Bernd Dornieden, Dr.-Ing. Philipp Schnetter, Dr.-Ing. Dipl.-Wirt.Ing. Philipp Themann, Dr.-Ing. Thomas Weidner, and Dr. rer. nat. Peter Schlicht. Safety first for automated driving (safad). Technical report, Aptiv Services US, LLC; AUDI AG; Bayrische Motoren Werke AG; Beijing Baidu Netcom Science Technology Co., Ltd; Continental Teves AG & Co oHG; Daimler AG; FCA US LLC; HERE Global B.V.; Infineon Technologies AG; Intel; Volkswagen AG, 2019. URL https://www.daimler.com/documents/innovation/other/safety-first-for-automated-driving.pdf.

[104] A. Meystel. Intelligent control: A sketch of the theory. *Journal of Intelligent and Robotic Systems*, 2(2):97–107, Jun 1989. ISSN 1573-0409. doi: 10.1007/BF00238683. URL https://doi.org/10.1007/BF00238683.

[105] Helen Monkhouse, Ibrahim Habli, John McDermid, Siddartha Khastgir, and Gunwant Dhadyalla. Why functional safety experts worry about automotive systems having increasing autonomy. In *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pages 1–6. IEEE, 2017.

[106] Hausi A Müller, Liam O'Brien, Mark Klein, and Bill Wood. Autonomic computing. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 2006.

[107] NHTSA. Preliminary statement of policy concerning automated vehicles. techreport, NHSTA, 2013. URL https://www.nhtsa.gov/staticfiles/rulemaking/pdf/Automated_Vehicles_Policy.pdf.

[108] NHTSA. Automated driving systems 2.0: A vision for safety. Technical report, U.S. Department of Transportation, 2017. URL https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/documents/13069a-ads2.0_090617_v9a_tag.pdf.

[109] Marcus Nolte, Gerrit Bagschik, Inga Jatzkowski, Torben Stolte, Andreas Reschka, and Markus Maurer. Towards a skill- and ability-based development process for self-aware automated road vehicles. *CoRR*, abs/1708.02532, 2017. URL http://arxiv.org/abs/1708.02532.

[110] Milos Ojdanic. Systematic literature review of safety-related challenges for autonomous systems in safety-critical applications. Master's thesis, MÃlardalen University, School of Innovation Design and Engineering, VÃsterÃs, Sweden, 2019.

[111] Sean O'Kane. Tesla defends Autopilot after fatal Model X crash. *The Verge*, March 2018. URL https://www.theverge.com/2018/3/28/17172178/tesla-model-x-crash-autopilot-fire-investigation.

[112] Sean O'Kane. Tesla hit with another lawsuit over a fatal autopilot crash, August 2019. URL https://www.theverge.com/2019/8/1/20750715/tesla-autopilot-crash-lawsuit-wrongful-death.

[113] Kelly Pierce. Udot makes safety upgrades after autonomous shuttle accident injures man in salt lake city. *KSL News Radio*, Jul 2019. URL https://kslnewsradio.com/1908556/udot-makes-safety-upgrades-after-autonomous-shuttle-accident-injures

[114] Horia Porav and Paul Newman. Imminent collision mitigation with reinforcement learning and vision. *CoRR*, abs/1901.00898, 2019. URL http://arxiv.org/abs/1901.00898.

[115] PATH Program. Peer review of behavioral competencies for avs. techreport, University of California, Oakland, CA, USA, February 2016. URL https://www.nspe.org/sites/default/files/resources/pdfs/Peer-Review-Report-IntgratedV2.pdf.

[116] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, volume 3, page 5, May 2009. URL `http://www.willowgarage.com/sites/default/files/icraoss09-ROS.pdf`.

[117] A. Ramaswamy, B. Monsuez, and A. Tapus. An extensible model-based framework for robotics software development. In *2017 First IEEE International Conference on Robotic Computing (IRC)*, pages 73–76, April 2017. doi: 10.1109/IRC.2017.21.

[118] Amir Rasouli and John K. Tsotsos. Autonomous vehicles that interact with pedestrians: A survey of theory and practice. *CoRR*, abs/1805.11773, 2018. URL `http://arxiv.org/abs/1805.11773`.

[119] Thomas Raste, Hagen BÃ¶hmert Ali, and Ali Houry. Fallback strategy for automated driving using stpa. In *3rd European STAMP Workshop*, 2015.

[120] Art Raymond. Utah driverless shuttle mishap doesn't slow 76-year-old state employee. *Deseret News Utah*, Jul 2019. URL `https://www.deseretnews.com/article/900080444/utah-driverless-shuttle-accident-state-employee.html`.

[121] A. Reschka, G. Bagschik, S. Ulbrich, M. Nolte, and M. Maurer. Ability and skill graphs for system modeling, online monitoring, and decision support for vehicle guidance systems. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 933–939, June 2015. doi: 10.1109/IVS.2015.7225804.

[122] Andreas Reschka and Markus Maurer. Conditions for a safe state of automated road vehicles. *it - Information Technology*, 57(4), jan 2015. doi: https://doi.org/10.1515/itit-2015-0004.

[123] Andreas Reschka, Jürgen Rüdiger Böhmer, Tobias Nothdurft, Peter Hecker, Bernd Lichte, and Markus Maurer. A surveillance and safety system based on performance criteria and functional degradation for an autonomous vehicle. In *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pages 237–242. IEEE, 2012.

[124] Edge Case Research and Underwriters Laboratories. Ul 4600: The first comprehensive safety standard for autonomous products, June 2019. URL `https://edge-case-research.com/ul4600/`.

[125] D. Rodrigues, R. de Melo Pires, E. A. Marconato, C. Areias, J. C. Cunha, K. R. L. J. Castelo Branco, and M. Vieira. Service-oriented architectures for a flexible and safe use of unmanned aerial vehicles. *IEEE Intelligent Transportation Systems Magazine*, 9(1):97–109, Spring 2017. ISSN 1939-1390. doi: 10.1109/MITS.2016.2611038.

[126] Pascal (Consultant) Roques. *Systems Architecture Modeling with the Arcadia Method*. ISTE Press Ltd - Elsevier Inc, 2017. ISBN 9781785481680. URL `https://www.ebook.de/de/product/29025440/pascal_consultant_roques_systems_architecture_modeling_with_the_arcadia_method.html`.

[127] Donald Rumsfeld. *Known and Unknown: A Memoir*. Penguin Group USA, 2011. ISBN 978-1-59523-067-6. URL https://www.ebook.de/de/product/12544690/donald_rumsfeld_known_and_unknown_a_memoir.html.

[128] Giedre Sabaliauskaite, Lin Shen Liew, and Jin Cui. Integrating autonomous vehicle safety and security analysis using stpa method and the six-step model. *International Journal on Advances in Security*, 11:160–169, July 2018. URL https://www.researchgate.net/profile/Giedre_Sabaliauskaite/publication/326504334_Integrating_Autonomous_Vehicle_Safety_and_Security_Analysis_Using_STPA_Method_and_the_Six-Step_Model/links/5b595f430f7e9bc79a656f09/Integrating-Autonomous-Vehicle-Safety-and-Security-Analysis-Using-ST
pdf.

[129] Rick Salay, Rodrigo Queiroz, and Krzysztof Czarnecki. An analysis of iso 26262: Using machine learning safely in automotive software. *ArXiv*, abs/1709.02435, 2017.

[130] J. Schlatow, M. Moostl, R. Ernst, M. Nolte, I. Jatzkowski, M. Maurer, C. Herber, and A. Herkersdorf. Self-awareness in autonomous automotive systems. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, pages 1050–1055, March 2017. doi: 10.23919/DATE.2017.7927145.

[131] Daniel Schneider and Mario Trapp. B-space: dynamic management and assurance of open systems of systems. *Journal of Internet Services and Applications*, 9(1): 15, Aug 2018. ISSN 1869-0238. doi: 10.1186/s13174-018-0084-5. URL https://doi.org/10.1186/s13174-018-0084-5.

[132] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *CoRR*, abs/1610.03295, 2016. URL http://arxiv.org/abs/1610.03295.

[133] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. On a formal model of safe and scalable self-driving cars. *CoRR*, abs/1708.06374, 2017. URL http://arxiv.org/abs/1708.06374.

[134] Joseph Sifakis. System design in the era of iot - meeting the autonomy challenge. In Simon Bliudze and Saddek Bensalem, editors, *Proceedings of the 1st International Workshop on Methods and Tools for Rigorous System Design, Thessaloniki, Greece, 15th April 2018*, volume 272 of *Electronic Proceedings in Theoretical Computer Science*, pages 1–22. Open Publishing Association, 2018. doi: 10.4204/EPTCS.272.1.

[135] Joseph Sifakis. Autonomous systems – an architectural characterization. In *Models, Languages, and Tools for Concurrent and Distributed Programming*, pages 388–410. Springer International Publishing, 2019. doi: 10.1007/978-3-030-21485-2_21.

[136] R. G. Simmons. Structured control for autonomous robots. *IEEE Transactions on Robotics and Automation*, 10(1):34–43, Feb 1994. ISSN 1042-296X. doi: 10.1109/70.285583.

[137] Santokh Singh. Critical reasons for crashes investigated in the national motor vehicle crash causation survey. (traffic safety facts crash stats. report no. dot hs 812 115). techreport, Washington, DC: National Highway Traffic Safety Administration., February 2015. URL https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812115.

[138] Ion Stoica, Dawn Song, Raluca Ada Popa, David A. Patterson, Michael W. Mahoney, Randy H. Katz, Anthony D. Joseph, Michael Jordan, Joseph M. Hellerstein, Joseph Gonzalez, Ken Goldberg, Ali Ghodsi, David E. Culler, and Pieter Abbeel. A berkeley View of Systems Challenges for ai. Technical Report UCB/EECS-2017-159, EECS Department, University of California, Berkeley, Oct 2017. URL http://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-159.html.

[139] Sardar Muhammad Sulaman, Armin Beer, Michael Felderer, and Martin Höst. Comparison of the fmea and stpa safety analysis methods–a case study. *Software Quality Journal*, Dec 2017. ISSN 1573-1367. doi: 10.1007/s11219-017-9396-0. URL https://doi.org/10.1007/s11219-017-9396-0.

[140] Phil Tetlow, Jeff Z. Pan, Daniel Oberle, Evan Wallace, Michael Uschold, and Elisa Kendall. Ontology driven architectures and potential uses of the semantic web in systems and software engineering, 2006. URL https://www.w3.org/2001/sw/BestPractices/SE/ODA/.

[141] Kristinn R. Thórisson. A new constructivist AI: From manual methods to self-constructive systems. In *Atlantis Thinking Machines*, pages 145–171. Atlantis Press, 2012. doi: 10.2991/978-94-91216-62-6_9.

[142] Eric Thorn, Shawn Kimmel, and Michelle Chaka. A framework for automated driving system testable cases and scenarios. techreport, U.S. Department of Transportation, National Highway Traffic Safety Administration, NHTSA, 1200 New Jersey Avenue SE., Washington, DC 20590, September 2018. URL https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/documents/13882-automateddrivingsystems_092618_v1a_tag.pdf.

[143] Debra Topham, Gary Ford, David Ward, Christian Zott, and Piero Mortara. Vehicle probe use cases and test scenarios. resreport, SAFESPOT, 2006. URL http://www.safespot-eu.org/documents/D1.2.1_Vehicle_probe_use_case-and_test_scenarios.pdf.

[144] Martin Törngren, Xinhai Zhang, Naveen Mohan, Matthias Becker, Xin Tao, DeJiu Chen, and Jonas Westman. Architecting safety supervisors for high levels of automated driving. In *the 21st IEEE Internal Conference on Intelligent Transportation Systems*, 2018.

[145] Mario Trapp and Daniel Schneider. *Safety Assurance of Open Adaptive Systems – A Survey*, pages 279–318. Springer International Publishing, Cham, 2014. ISBN 978-3-319-08915-7. doi: 10.1007/978-3-319-08915-7_11. URL http://dx.doi.org/10.1007/978-3-319-08915-7_11.

[146] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, and M. Maurer. Defining and substantiating the terms scene, situation, and scenario for automated driving. In

2015 IEEE 18th International Conference on Intelligent Transportation Systems, pages 982–988, Sept 2015. doi: 10.1109/ITSC.2015.164.

[147] Qi Van Eikema Hommes et al. Assessment of safety standards for automotive electronic control systems. Technical report, United States. National Highway Traffic Safety Administration, 2016.

[148] Mark A. Vernacchia. Gm presentation for introducing stamp/stpa tools into standards. March 2018. URL http://psas.scripts.mit.edu/home/wp-content/uploads/2018/04/SAE-STPA-Recom-Pract-Task-Force-Overview-Mark-Vernacchia-GM-27mar18-pdf. MIT STAMP Workshop.

[149] David Vernon. *Artificial cognitive systems: A primer*. MIT Press Ltd, 2014. ISBN 9780262028387. URL https://www.ebook.de/de/product/23292892/david_professor_of_informatics_university_of_skovde_vernon_artificial_cognitive_systems.html.

[150] Thomas Vogel and Holger Giese. *Model-Driven Engineering of Self-Adaptive Software with EUREMA*. PhD thesis, 2018. URL http://arxiv.org/abs/1805.07353.

[151] Walther Wachenfeld, Hermann Winner, J. Chris Gerdes, Barbara Lenz, Markus Maurer, Sven Beiker, Eva Fraedrich, and Thomas Winkle. *Use Cases for Autonomous Driving*, pages 9–37. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016. ISBN 978-3-662-48847-8. doi: 10.1007/978-3-662-48847-8_2. URL http://dx.doi.org/10.1007/978-3-662-48847-8_2.

[152] Waymo. Waymo safety report - on the road to fully self-driving. techreport, Waymo, Google, 2017. URL https://storage.googleapis.com/sdc-prod/v1/safety-report/waymo-safety-report-2017.pdf.

[153] Waymo. Waymo safety report - on the road to fully self-driving. techreport, Waymo, Google, 2018. URL https://storage.googleapis.com/sdc-prod/v1/safety-report/waymo-safety-report-2017.pdf.

[154] Ran Wei, Jan Reich, Tim Kelly, and Simos Gerasimou. On the transition from design time to runtime model-based assurance cases. 10 2018.

[155] Ran Wei, Tim P. Kelly, Xiaotian Dai, Shuai Zhao, and Richard Hawkins. Model based system assurance using the structured assurance case metamodel. *Journal of Systems and Software*, 154:211–233, aug 2019. doi: 10.1016/j.jss.2019.05.013.

[156] Ruffin White and Henrik Christensen. *Robot Operating System (ROS)*, chapter 9, pages 285–307. 707. Springer International Publishing, 2017. ISBN 978-3-319-54927-9. doi: 10.1007/978-3-319-54927-9.

[157] David Wittmann, Cheng Wang, and Markus Lienkamp. Definition and identification of system boundaries of highly automated driving. 2015.

[158] Lihua Zhao, Ryutaro Ichise, Seiichi Mita, and Yutaka Sasaki. Core ontologies for safe autonomous driving. *International Semantic Web Conference*, 2015.

[159] Lihua Zhao, Ryutaro Ichise, Tatsuya Yoshikawa, Takeshi Naito, Toshiaki Kakinami, and Yutaka Sasaki. Ontology-based decision making on uncontrolled intersections and narrow roads. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 83–88. IEEE, June 2015. doi: 10.1109/IVS.2015.7225667.

[160] Chris Ziegler. A google self-driving car caused a crash for the first time. *The Verge*, Feb 2016. URL `https://www.theverge.com/2016/2/29/11134344/google-self-driving-car-crash-report`.

APPENDIX A

Safety vocabulary from ISO26262

The next definitions of relevant terms from Part 1 contribute to the best understanding of safety. They are divided into three categories and will be applicable in the next discussions and referenced works in the thesis.

A.1 Safety

harm, physical injury or damage to the health of persons.

fault, abnormal condition that can cause an element or an item to fail.

failure, termination of the ability of an element, to perform a function as required.

error, discrepancy between a computed, observed or measured value or condition, and the true, specified, or theoretically correct value or condition.

hazard, potential source of harm caused by malfunctioning behaviour of the item.

malfunctioning, behavior failure or unintended behaviour of an item with respect to its design intent.

unreasonable risk, risk judged to be unacceptable in a certain context according to valid societal moral concepts.

residual risk, risk remaining after the deployment of safety measures.

safety, absence of unreasonable risk.

safety architecture, set of elements and their interaction to fulfil the safety requirements.

safe state, operating mode of an item without an unreasonable level of risk, e.g. intended operating mode, degraded operating mode or switched-off modes.

safety mechanism, technical solution implemented by E/E functions or elements, or by other technologies, to detect faults or control failures in order to achieve or maintain a safe state.

degradation, strategy for providing safety by design after the occurrence of failures.

safety goal, top-level safety requirement as a result of the hazard analysis and risk assessment. NOTE: One safety goal can be related to several hazards, and several safety goals can be related to a single hazard.

safety case, argument that the safety requirements for an item are complete and satisfied by evidence compiled from work products of the safety activities during development, e.g. the safety case is a collection of documents, goals, arguments and evidences which provide reasoning why the system is justifiable safe enough to not produce a hazard NOTE: Safety case can be extended to cover safety issues beyond the scope of this standard.

functional safety concept (FSC), specification of the functional safety requirements, with associated information, their allocation to architectural elements, and their interaction necessary to achieve the safety goals.

functional safety requirement (FSR), specification of implementation-independent safety behaviour, or implementation-independent safety measure, including its safety-related attributes.

technical safety requirement (TSR), requirement derived for implementation of associated functional safety requirements.

A.2 System

item system or array of systems to implement a function at the vehicle level, to which ISO 26262 is applied.

architecture representation of the structure of the item or functions or systems or elements that allows identification of building blocks, their boundaries and interfaces, and includes the allocation of functions to hardware and software elements.

component non-system level element that is logically and technically separable and is comprised of more than one hardware part or software unit.

life-cycle entirety of phases from concept through decommissioning of the item.

functional concept specification of the intended functions and their interactions necessary to achieve the desired behaviour.

availability capability of a product to be in a state to execute the function required under given conditions, at a certain time or in a given period, supposing the required external resources are available.

controllability ability to avoid a specified harm or damage through the timely reactions of the persons involved, possibly with support from external measures.

intended functionality behaviour specified for an item, system, or element excluding safety mechanisms.

baseline version of a set of one or more work products, items, or hardware or software, elements, that is under configuration management and used as a basis for further development through the change management process.

robust design design that has the ability to function correctly in the presence of invalid inputs or stressful environmental conditions. NOTE: Robustness can be understood as follows: for software, robustness is the ability to respond to abnormal inputs and conditions, for hardware, robustness is the ability to be immune to environmental stress and stable over the service life within design limits.

A.3 Process

model-based development development that uses models to describe the functional behaviour of the elements which are to be developed. NOTE: Depending on the level of abstraction used for such a model, the ‚model can be used for simulation or code generation or both.

assessment, examination of a characteristic of an item or element.

verification, determination of completeness and correct specification or implementation of requirements from a previous phase or sub-phase.

safety validation, assurance, based on examination and tests, that the safety goals are sufficient and have been achieved.

APPENDIX B

Environment model & SOSA/SSN

This chapter is an appendix to detail how the different knowledge bases of the MAPE-K loops are represented.

B.1 Existing representations of architecture models, contextual ontologies and system description

Existing work have already proposed some representations of some identified purposes in Table 4.1 for the following topics:

- **Road environment**: representation of the road entities and the scenery (i.e. road geometry) in [13, 14, 18, 63, 64, 146, 159].
- **System capability and functions**: behaviour representation and the ability to infer the most corresponding future behaviour that it should have.
- **Relationships and interactions between entities**: problem of interaction between road entities, and construction of scenery in [13, 14, 18, 63, 64, 146, 159].
- **Uncertainty**: use or capture result of probabilistic approaches taking into account uncertainty on data or decision.
- **Chain reactions**: models that do not take into account chain reactions alone in [14].
- **System description**: model of the system architecture (e.g. capabilities, functions, features of interest, and the relations between components and system concepts) and configuration (e.g. topography of the components, states) that is machine readable and understandable in [68].

All of those solutions store the knowledge of entities and their relation to be used by the system after design. Besides, increased interests arise out of representations that involve enable reasoning and have been using in the cited references to provide some decision-making, situation assessment or behaviour planning. As we have presented earlier in Section 4.4, run-time reasoning can infer additional knowledge from the terminological (i.e. concepts) and assertional boxes (i.e. individuals). The reasoner can perform association between similar concepts and individuals (e.g. relations of inclusion or equivalence), identification of missing hierarchical terminological concepts, and consistency check over the knowledge bases (e.g. assessment of restrictions like disjoints or cardinality relations between concepts).

As a matter of fact, we will mainly use ontologies to model the relevant semantic pieces of information (i.e. symptoms, change request, change plan and performed actions) and to perform monitoring and diagnostic operated by the reasoner. We are using an OWL-DL ontology using the Terminological box (T-box) to represent the structure of the

system representation and the Assertional Box (A-box) to store run-time observations or deployed microservices as individuals. On this basis, we adopt the guideline to make the T-Box remain static during the system operation while the content of the A-Box evolves to represent the different changes of the system.

The next sections present how those different pieces of knowledge are captured and represented using five different models.

B.2 Context models

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

A first ontology captures the key concepts and structure of the observable context. The representation of the environment of the vehicle is performed using classes representing an urban environment marked inspired by the layers presented in [18]. In Figure B.1, we illustrate the considered entities such as dynamic objects (e.g. pedestrian, car, bus), static objects (e.g. stationary obstacles, traffic lights), the road geometry (e.g. lane, intersection, crosswalk), their interactions and also possible maneuvers (e.g. crossing activity from the pedestrian). Each road objects that is a road user is considered as dynamic and has the possibility to have different maneuvers. It helps to describe how the road entity is operating or acting within the scene. Current perceived actions (i.e. maneuvers) 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 maneuver 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 contributes semantically enhance the scene. We see the solution for their representations as twofold.

First, we satisfy the fact that maneuvers 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 [59]. 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 *Maneuver 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 maneuvers, 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*.

Second, the identification of the perceived maneuvers are subject to uncertainties due to observability restriction raised in [144] 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 [144]. The confidence is largely used to express a weighting on how much we are sure of a specific sampling. In order to allow more

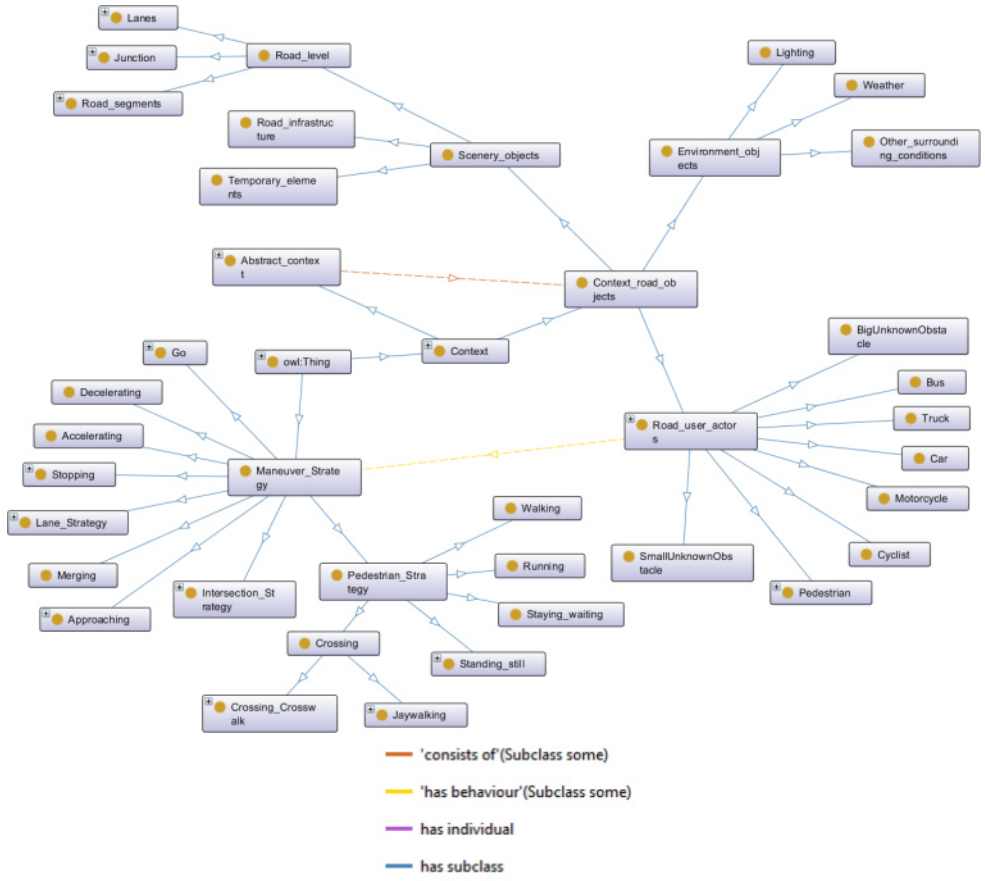


Figure B.1 Ontology representation of the environment of the vehicle with road entities

than one cardinality for road-user/maneuvers relation, we choose to create an *Strategy* instance for each maneuvers and attach the confidence of the observed result as a data property. The link between road user and possible maneuvers are then represented by the *has behaviour* semantic relation. The existence of *Strategy* individual associated to a specific road users 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 shows 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 a short time. Certain ontology features can also impact the reasoner performances and might causing unexpected reasoning results according the analysis presented in [8]. However, ontologies tend to remains the most efficient when reasoning with individuals and relations where composition are involved.

An application on the observation of a pedestrian is illustrated in Figure B.2. The pedestrian *Ex01_Pedestrian* is represented as a individual (marked by purple diamond) from the perceived context.

On one hand, the object can be described by data properties (i.e. id, age, pose, speed, heading, IsOnRoad, IsOnLane ...) and by its current maneuvers. Applicable strategies

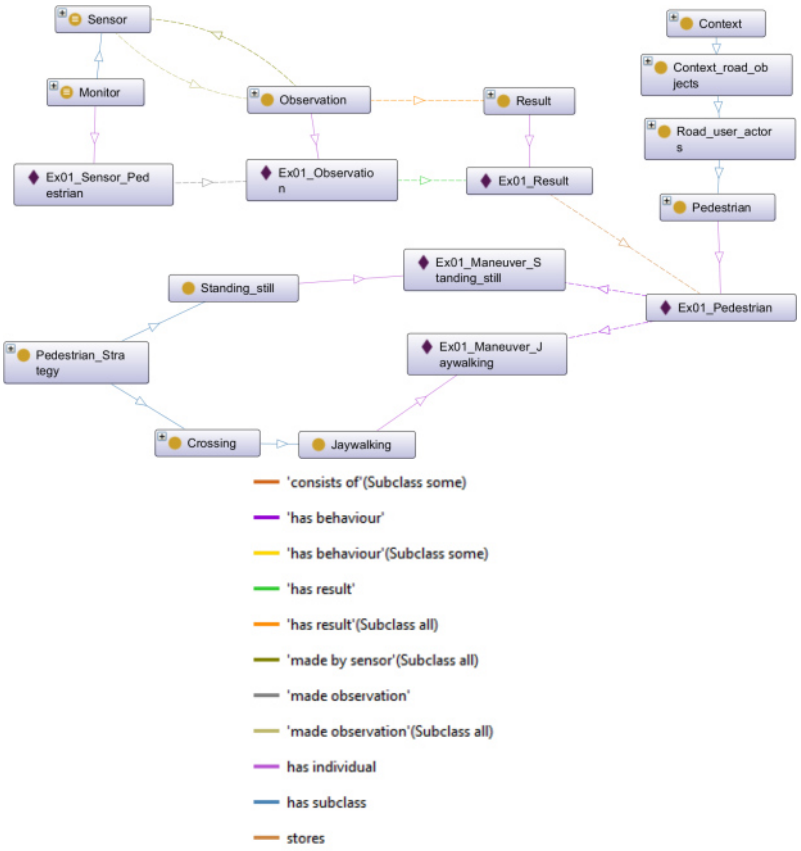


Figure B.2 Representation of the environment of the vehicle using observations of road entities

for pedestrian are a subclass of *Pedestrian Strategy*. In our example, the *Ex01_Pedestrian* have two possible maneuvers: *Standing still* and *Jaywalking*. The existence of two maneuvers (i.e. possibly more than one) helps integrating the uncertainty within the observation of the context. The confidence is then stored in the dataproperty *hasConfidence* for each strategies.

On the other hand, the chain of relations between the *Result*, *Observation* and *Monitor* serves the traceability of the observation. Indeed, we are able to know which microservice has produced or contributed to the information creation or semantic augmentation by timestamping.

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 aims to offer a higher-level representation of the context to favor scene understanding, scene tagging and identification. For example, they serve as a support for inference to identify known compositions of contextual road objects. They examine the enriched symptoms from perceived information from sensors (e.g. perception, localization) and the a-priori knowledge (e.g. HD digital map data, identification rules, context-dependence). We can envisage them as patterns that match the different parts we intend to study like the concepts introduced in [146]. Figure B.3 illustrates the different forms of context abstractions we propose to capture based on the following specific

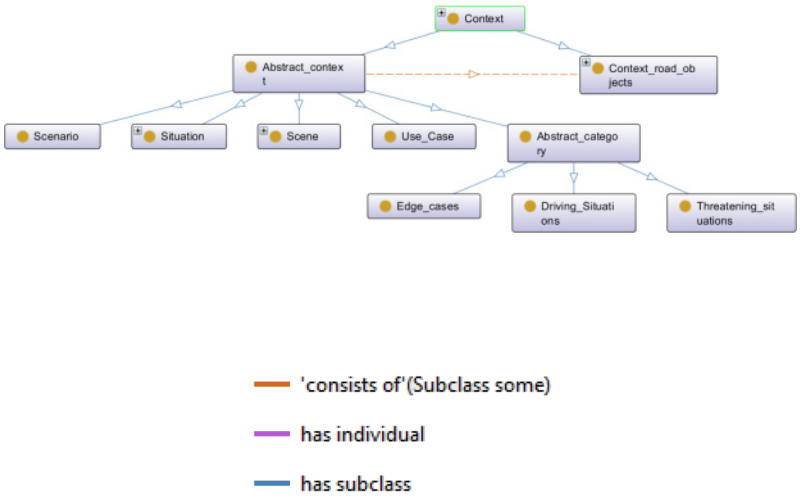


Figure B.3 Ontology representation of the use cases for situations encounter using composition of road entities

representation goals and prerequisites.

A S*cene* 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 S*ituation* 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 S*cenario* 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.

An U*se 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 ISO26262 [1].

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 B.4 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 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 B.2 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.

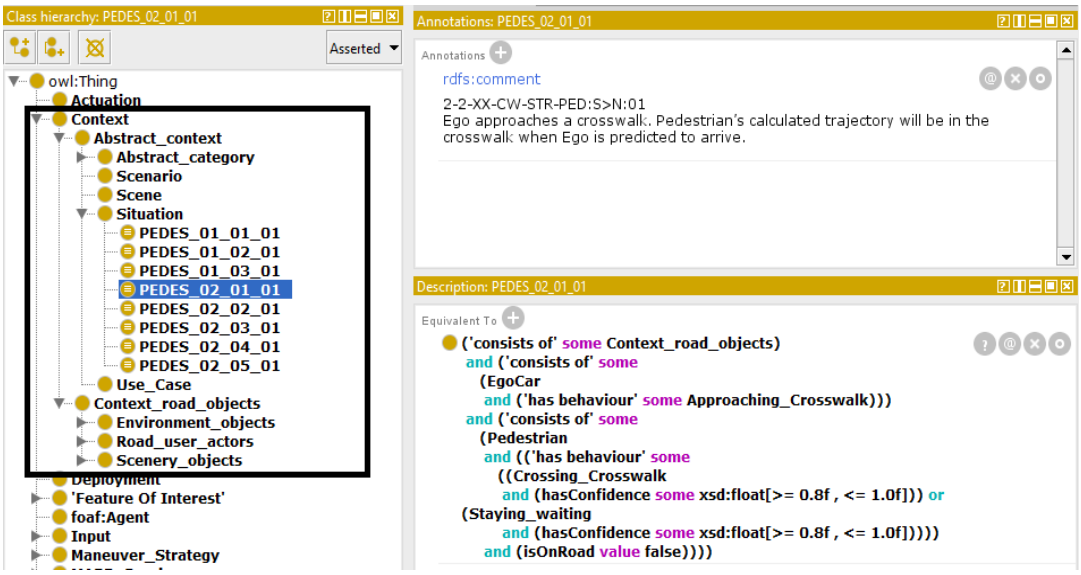


Figure B.4 Ontology representation of the use cases for situations encounter using composition of road entities. *Partial repost of Figure 5.9*

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 constitute symptoms in the OAM adaptation loop.

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.

We have introduced the use of an ontology as representation of different situations where some can be consider 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. Further explanations on the use and the composition will be presented in Chapter 5 within a more thoroughgoing pedestrian crossing case study.

Ontology for the symptoms characterizing the current status of the AMs

The second level of adaptation assures that all safety constraints relevant to the current context are appropriately guaranteed and none out-of-context are wrongly ensured. The assurance of context-dependence between context abstractions and the status of the AMs requires a rightful model and symptoms to be defined at design. At run-time, the reasoner infers on the specific model and contributes to deploying the different appropriate AMs upon the vehicle context.

For this purpose, the model shall be able to answer to the following question: What are the microservices that are required to the current context? Answering it requires two requests to different sources of information: “What are the current context abstractions identified?” refers to the symptoms identified from context; “What are the relevant

microservices for these identified abstractions?" points out the existence of specified context-dependence relation between the microservices operating range and the current identified context abstractions.

Thus, we create a reflexive model of the system that captures the configuration, structure, properties, deployment and operating range of the AMs. It actually means that we need to semantically describe the ODD for each safety assurance process and their possible restrictions. Additional sources of information can also be integrated to this structure and contribute to increasing the relevance of the reasoning (e.g. integrating some properties promoted by functional safety). For instance, we are targeting supplementary representations of insufficient performance, abnormal heartbeat, error state and data inconsistency for each microservices involved in the AMs. More information regarding those properties, the introspection functions, individual monitoring and the logical structure of the microservices are have been presented in Section 5.2 on page 121.

In brief, we need to design a run-time representation that abstract our system's configuration and is operable by the OAM. For this purpose, the use of the existing SOSA/SSN ontology from [68] contributes to the descriptive model of our system as a collection of *Sensor*, *Sampler*, *Actuator* systems and their respective *Observation*, *Actuation* and *Sampling* records. However, the generic vocabulary introduced by the SOSA/SSN may bring confusion over some actual terms (e.g. sensor and actuator are used differently in Autonomous Vehicles and Autonomic Computing). To solve any ambiguity, we have created equivalent classes for the reference *System* classes (i.e. *Sensor*, *Sampl*er, *Actuator*) to manipulate our *Monitor*, *Analyze*, *Plan*, and *Execute* types of microservices rightfully. We illustrate the created equivalences between the SOSA/SSN classes and our MAPE types of systems in Figure B.5 on the following page. We represent them as follows: A *Monitor* service is identified as a *Sensor* system and produces *Observation* that contributes to building up symptoms. *Analyze* and *Planning* services are *Sampler* systems and produce *Sampling* that constitutes the change request and change plans respectively. Finally, an *Execute* service identified as *Actuator* and produces *Actuation* to changes the states of the outside world.

The extended ontology now represents a collection of microservices where each deployed system are identified to a particular type, and the associated individual contains information regarding its current configuration and restrictions.

Furthermore, the use of the optional ssn-system namespace[1] normalizes the representation of system capabilities, operating ranges and conditions which open rooms for traceability and extension. The representation of the system with these concepts contributes to specify the primary purpose of the system, how capabilities and properties are affecting it and define a representation of the system's normal operating environment. The information that is collected through these concepts is close to what the ODD and OEDR needs to capture in the field of AV. *OperatingRange* describes the *Conditions* in which the system is expected to operate. If the *Conditions* are not met, the system is consider as operating out of operating range and the *SystemCapability* specifications may no longer hold.

Based on these concepts of the ssn-system ontology, we identify an equivalence between the *Condition* and the *Abstract Context* axioms as they both specify ranges for qualities and compositions. The equivalence between the two concepts also facilitates the manipulation and readability of the ontology. As a result, we can semantically represent

[1] http://www.w3.org/ns/ssn/systems/

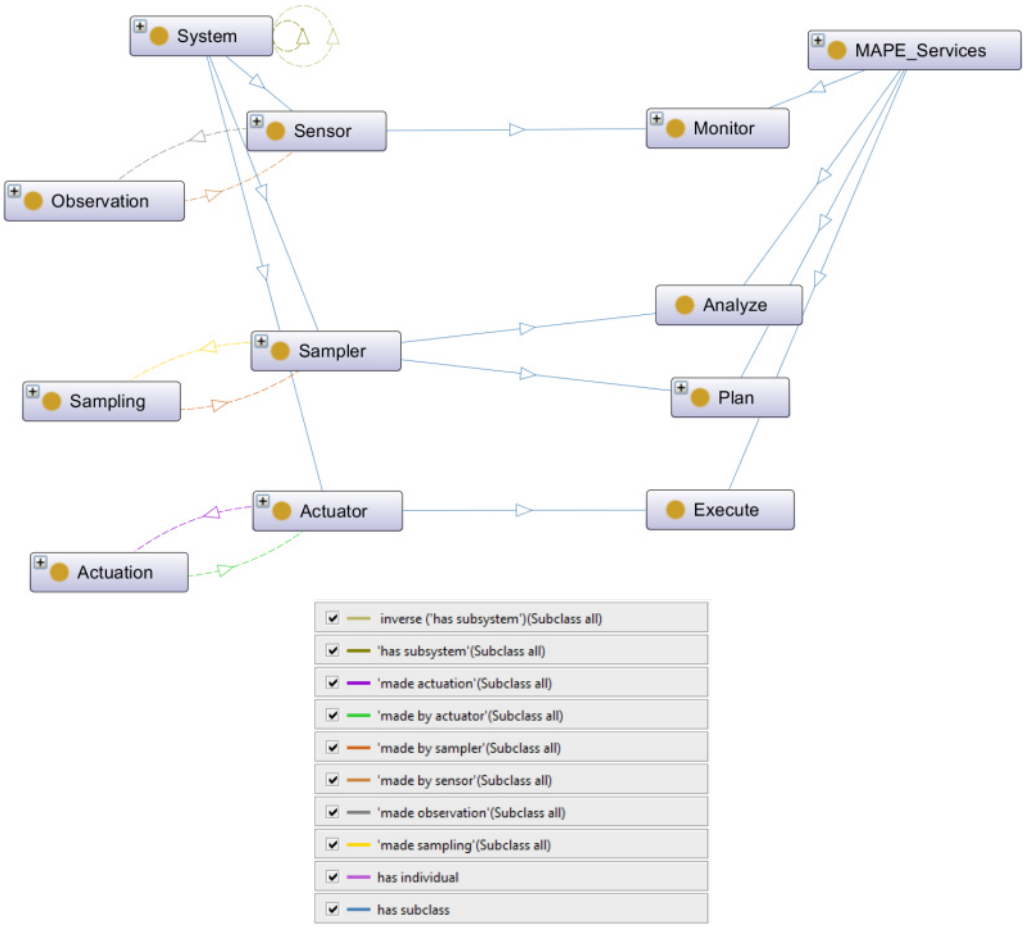


Figure B.5 Ontology representation of the microservices of the framework as system using SOSA/SSN

the relation between the system components and the different context abstractions (i.e. symptoms) in which they aim to be operated only.

Figure B.6 on the next page illustrates the representation of the context-dependence relation between the Monitor *Pedestrian_Detection* microservice called *Ex03_Pedestrian_Detection_Presence* and some specific context abstractions. In this example, we show the operating range as only one situation: *PEDES_02_01_01*. The operating range *Pedestrian_Detection_ORange* semantically represents the relationship with both the situation and the microservice.

Those relations between the AM microservices and the context abstractions (i.e. use cases, situations, scenes, etc) are a-priori knowledge stored in the T-Box and aims to reflect relevance and restrictions of the microservice deployment.

Besides, system properties (e.g. frequency, detection limit, resolution, etc) can be represented as well using the *Pedestrian_Detection_SCapability* property to describe the system further.

B.3 Configuration models

The configurations models aim to provide an architectural view of the current state of the managed resources for both design and run-time. It defines what the external managed resources are, how they can be accessed and how our system can control them. In our

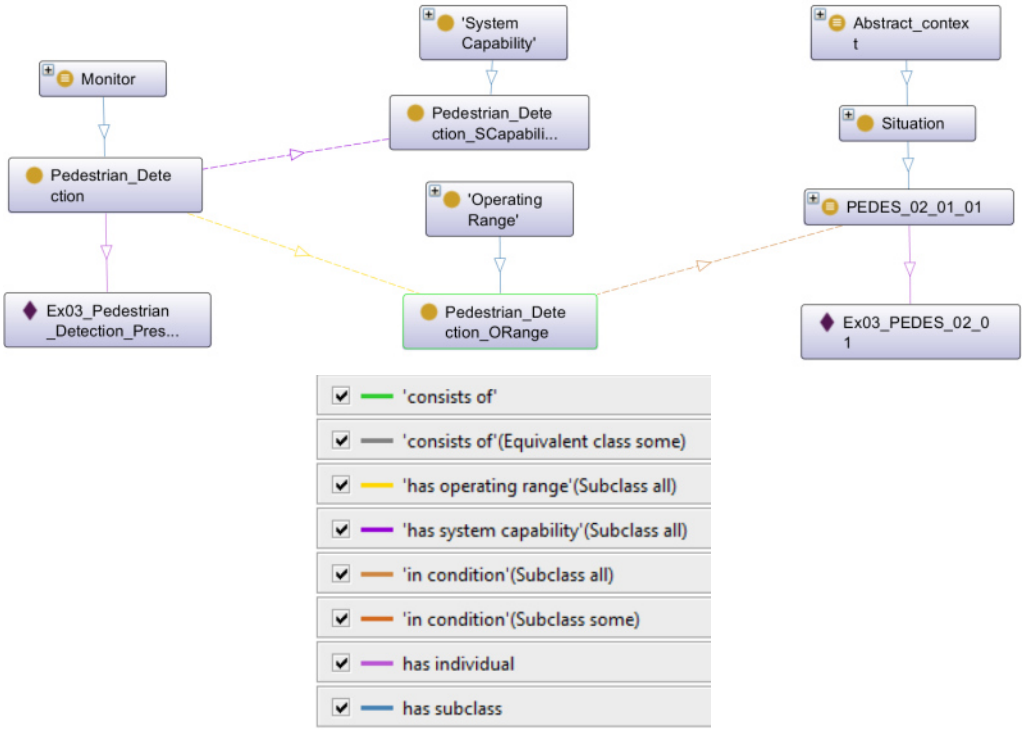


Figure B.6 Ontology representation of a *Monitor* microservice performing *Pedestrian* detection isolating speed and range

approach, this configuration model contributes to describe a part of the mechanisms provided by the manageability interfaces of the AC paradigm. Hence, we can model, store and access log files, events, commands, application programming interfaces (APIs) and configuration files related to the managed resources. Events and commands are directly related to the observations and control actions. By using a microservice architecture, the managed resources share and get that information from a publish/subscribe communication mechanism.

Furthermore, adopting a microservice architecture provides service repository mechanisms that capture the state of all deployed services. The service repository acts as a catalogue of services to see what services are registered and available in the system. At run-time, it holds a list of all currently deployed or deployable microservices.

Our contribution does not aim to redefine the service repository but integrates it into our knowledge management as a source of information for context understanding. Consequently, the model holds two parts: a static part characterizing the access and control over the managed resources (i.e. interface specifications); a changeable part reflecting the current configuration of the system (i.e. managed resources status).

In the first place, the model of the system logical architecture represents and characterize these two parts as interactions between as components, interfaces and components exchanges. To pursue our idea of having a reflexive system, a second step uses the resulting model information and convert it into an ontology representing the possible interactions between the system and its managed resources.

We have it as a part of the configuration model as ontologies. More information on the system interfaces and use of the models is provided in Section 5.2 with the presentation of the resulting logical architecture of a template microservice.

In our implementation, the adoption of the same service-oriented middleware than

in the ADS makes us use the same publish/subscribe communication mechanisms. The application and the coding of the interfaces result to be simplified. Although the implementation of the complete touchpoints and manageability interfaces (i.e. events, formats, metrics, state) is made more accessible, the ontology still needs to carry information on the interfaces and current status of the managed resources. The following paragraphs detail the configuration models for each of the two levels of adaptation available in our approach.

Current configuration of the AD system, sensors and effectors status (operated by Touchpoints management in layer 1)

The first configuration model aims to represent a run-time catalogue of the available sensors and effectors of the ADS. It also characterizes what the ADS can offer in terms of observations, and control actions, and their exchange interfaces and formats.

In this first level of adaptation, the identified external managed resources are the ADS perception and navigation. They are accessible through specific topic names and type of messages. The AMs control the ADS behaviour through the navigation system (e.g. topic and interface to alter the vehicle motion with new trajectory or speed).

The representation of the configuration model is mainly based on the axioms of *Input* and *Output* and their relation with *Procedure* and *System* from the SOSA/SSN ontology. Such use allows us to model the relation between the systems, and to precise the type of input, and to define how the measuring, sampling or executing need to be performed. However, those axioms alone are not sufficient to express our type of exchanges between services. Hence, we introduce new axioms in the ontology that precise which topic or which need the system consumes or provides. The axioms *Topic* or *Need* precise what information is available for publish/subscribe in the system. A *Topic* corresponds to the direct source of information the system needs to subscribe. A *Need* represents the type of information the system need to provide or to operate with. The information of a need gives relative access to the information and requires to be treated by an interpreter. Thus, a semantic mediation shall provide the rightful set of topics based on the expressed needs.

Figure B.7 on the facing page shows the representation as an ontology of the configuration model for the managed resources of the ADS. The axioms of *System*, *Procedure*, *Output*, *Input* on the left of the figure are from the SOSA/SSN ontology. *Managed_resources*, *Managed_Procedures*, *Source_to_Publish*, *Source_on_Subscribe*, *Need* and *Topic* are the axioms we have added to fit the use of service-oriented approach. Figure B.7 on the next page justifies of their origin from the original axioms. Then, the remaining classes based on the extension contribute to define how the managed resources.*ADS_Perception_Pub*, *ADS_Navigation_Pub* and *ADS_Navigation_Sub* are interacting. Hence, the ADS provides two sources of observations and one source that consume actuation. In the presented example, the *Topic* axiom regroups all observations from the fusion-based perception sensor on the *Ex03_obstacle* topic and from navigation predictions on *Ex03_planned_trajectory*. It also collects the actuation on *Ex03_controlled_trajectory* in charge of the control of the navigation trajectory.

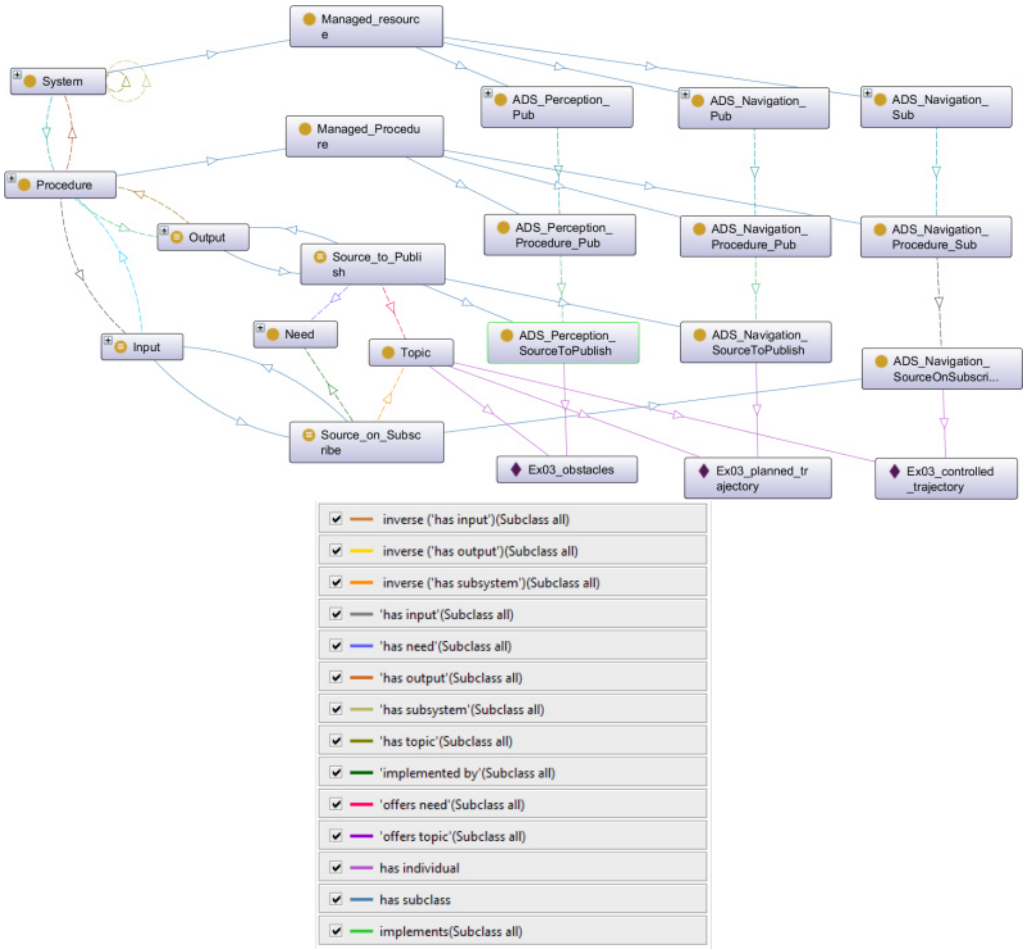


Figure B.7 Ontology representation of the configuration model for the ADS

Current configuration of the safety assessment via the currently deployed and running MAPE microservices

The second configuration model similarly represents the AMs that are contributing to the safety assessment. Contrary to the previous model, the managed resources are part of the reference architecture we are designing so that some models of the first level already covers part of the definition for the touchpoint and manageability interfaces. However, interfaces and communications exchange still needs to be specified and detailed. Figure B.8 on the following page illustrates the configuration model for the managed resources (i.e. MAPE microservices) of the AMs.

B.4 Capability models

We are using distinct capability models to support each of the two levels of adaptation loops that both aim to semantically explicit how the decisions can be made. They detail the knowledge to support the decision using the relations between the managing system capabilities (i.e. available plans to study) and the behaviours (i.e. possible maneuvers or reconfigurations). They also contribute to specify how the maneuvers influence the managed resources through their touchpoint manageability interface's mechanisms and which effectors have to be interfaced with. We diligently use the term "manageability

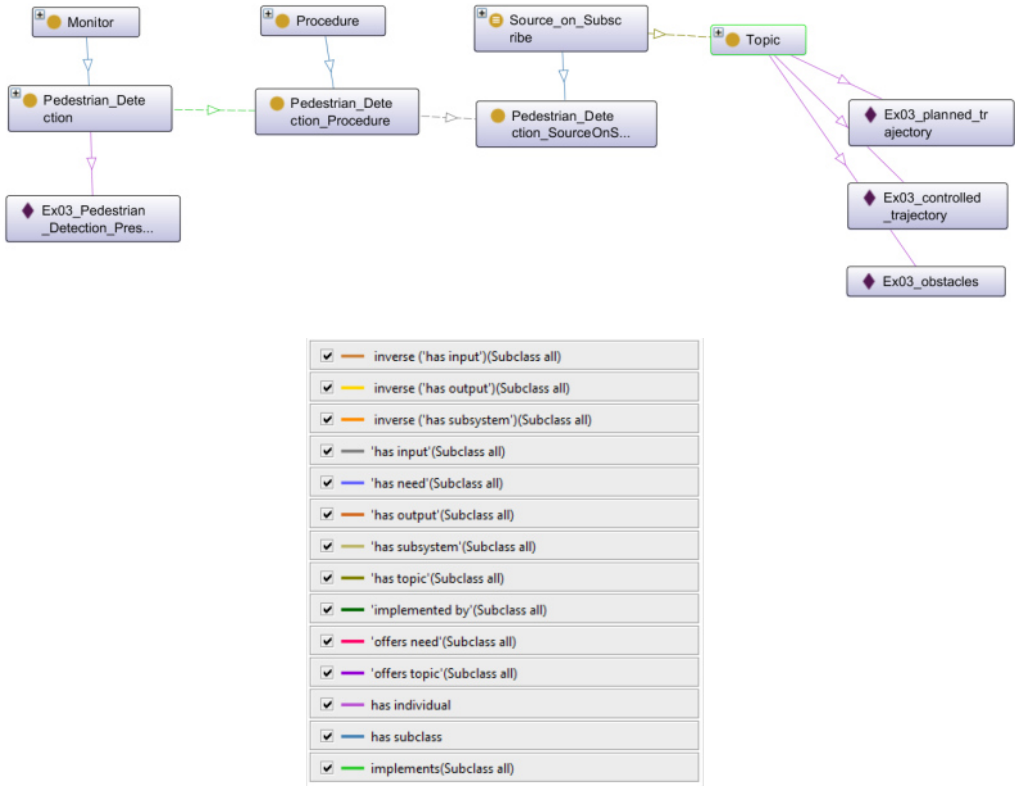


Figure B.8 Ontology representation of the configuration model for the AMs

mechanisms" with the capability and plan model to designate the effector behaviour as the way of affecting the state, attributes or variables of the managed resources.

With this model, we do not express the selection of the behaviours using directly hard-coded functions. We prefer to exploit the semantic relations established between the Plan microservices in the AMs or OAM, the available mechanisms that can affect the behaviour of the managed resources (i.e. high-level or low-level control as plan of composed of maneuvers or constrained value) and the actions to undergo on the managed resources via the effectors to perform such plan. The Plan microservice achieves the plan selection by requesting and evaluating plans for a set of relevant planning services in parallel. Finally, it selects the most relevant plan according to its current goals and triggers the next step of Execution.

The plan models of the framework are based on the definitions and relations provided by the T-Box of an ontology. Hence, the plans can be built up and composed of instances involving the capabilities, behaviour, actions, managed resources, and relations between them. The two levels of adaptation of the reference architecture require two distinct capability models.

Capability model of the AMs to influence the ADS using maneuver-based mechanisms

The first capability model aims to provide knowledge on the planning maneuvers mechanisms and traces how the ADS behaviour adaptation are performed. It entails the relations between the AMs *Plan* microservices that request plan generation using sets of vehicle

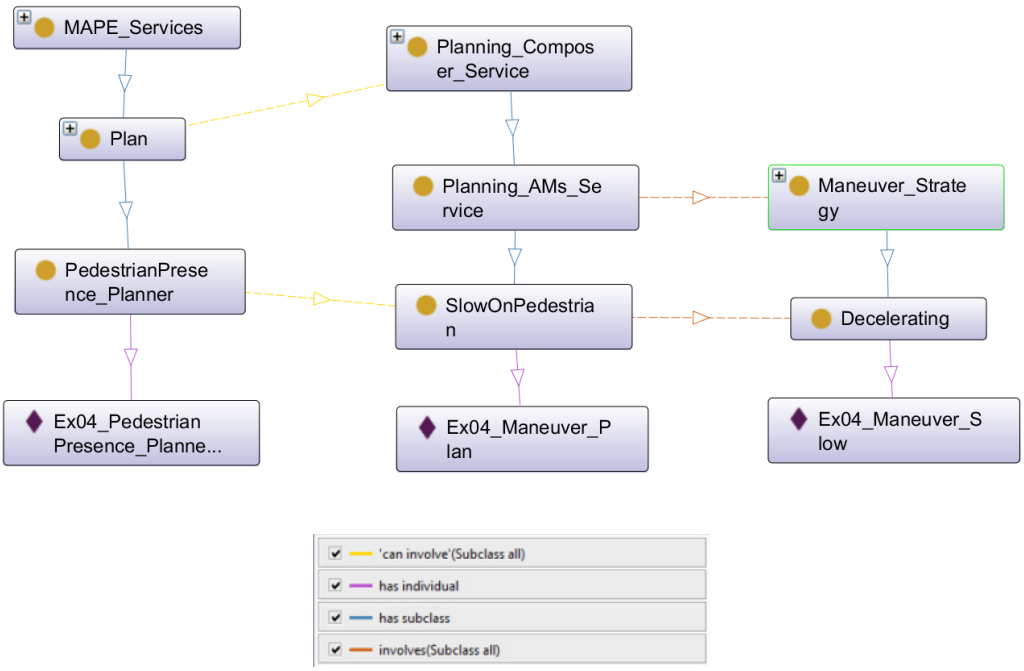


Figure B.9 Ontology representation of the capability model involving the planning and possible vehicle maneuvers the AMs can perform on the ADS

maneuvers for a given change request. According to the catalog of available and acceptable plan generation services, several service of plan generation plans can be requested (i.e. plans consisting of simple maneuver or of a composition of maneuvers which the EgoCar can perform) and then selected.

As an example of application, Figure B.9 illustrates an individual *Planner* microservice named *PedestrianPresence_Planner* that can request a plan from *SlowOnPedestrian* that involves a *Maneuver_Strategy* called *Decelerating*. This slowing maneuver can be operated by the EgoCar and specify how it actually affects the vehicle behaviour or configuration. It can be achieved by affecting new values such as speed or heading, or directly impact the selected trajectory of EgoCar, or imposing additional restriction for the choice of ADS-computed future maneuvers (e.g. via top-level mechanisms as minimum horizontal spacing or low-level as speed). For instance, *Ex04_Maneuver_Slow* contains information regarding a new speed and heading to adopt.

Capability model of the OAM to influence the AMs using service reconfiguration mechanisms

The next capability model possesses a similar set of relations but involves the reconfiguration of microservices as a mechanism of the second level of adaptation. This mechanism involves the management of the current deployment status and configuration of each microservice. Figure B.10 presents the T-Box that details the concepts and relations between the Plan microservices of the OAM, the microservices that compose plans to evaluate and the actions to interact with the microservice's state. Four types of actions are possible: *Launch*, *Idle*, *Reconfigure*, and *Stop*. Indeed, a resulting change plan uses the structure proposed by the capability model to compose a change plan for each relevant and available microservices.

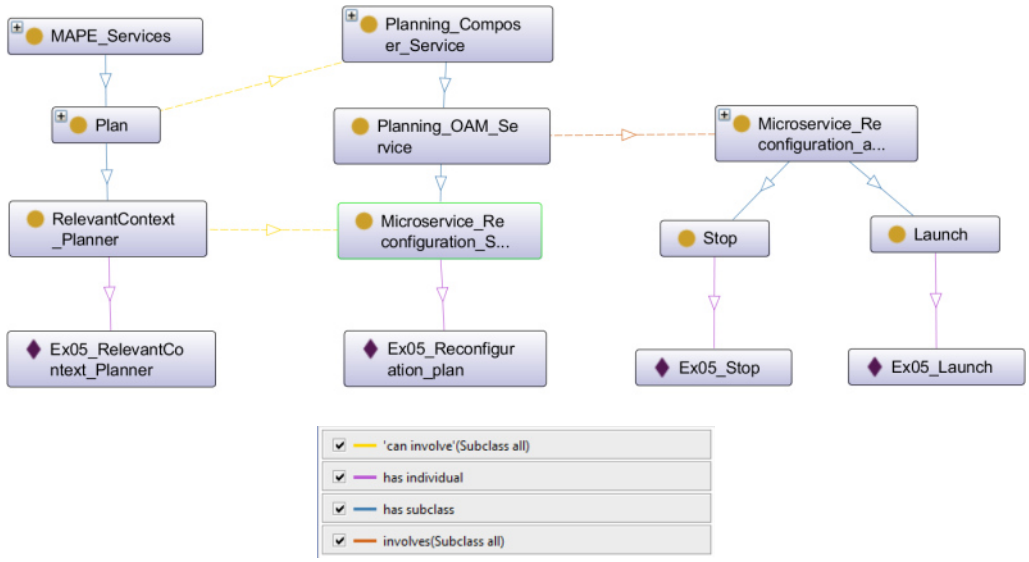


Figure B.10 Ontology representation of the capability model involving the planning and possible service reconfiguration the OAM can perform on the AMs

B.5 Plan models

The plan models acts as the collection of the change plans that are evaluated by the *Plan* microservices. They are based on the structure of the capability model in the framework and stored as individuals. For example, individuals in Figure B.9 and B.10 respectively represents plans for the two levels.

The shared structure between the plan model and capability model ensures that no "out-of-range" or inappropriate actions may be performed according the given design. Such context restrictions impose the evaluation of only a specific set of change plan or can narrow the set of possible actions in the specific context abstractions.

Each evaluated plans have also an unique identifier with a timestamp. Hence, the plan model contributes to store an history of the previous evaluations and selected plans to detail the called functions performed output actions to the effectors.

Goal & Adaptation models

The goal & adaptation models seek to semantically represent where and how the evaluations of the different configurations, symptoms or plans are performed and tuned.

Firstly, the semantic relations between the context and capability models already specify the context-dependence restrictions (i.e. plans and actions to be evaluated and performed upon the context). They associate the means of adaptation of the system (i.e. how it is expected to change over time and its mechanisms based on specific requirements inherited from the MBSE design) with the different contexts that may require system adaptations (e.g. meeting a specific or combined situation). It is achieved using the *OperatingRange* to specify where the system operates as *Conditions* as presented in the section of the context model.

Secondly, goal & adaptation model also contributes to define the different settings and the tuning of the *Monitor*, *Analyze*, *Plan,* and *Execute* functions. We use *OperatingProperty* instances as identifiable characteristics that represent how the system operates. More than one *OperatingProperty* can extend a single *OperatingRange* to specify the config-

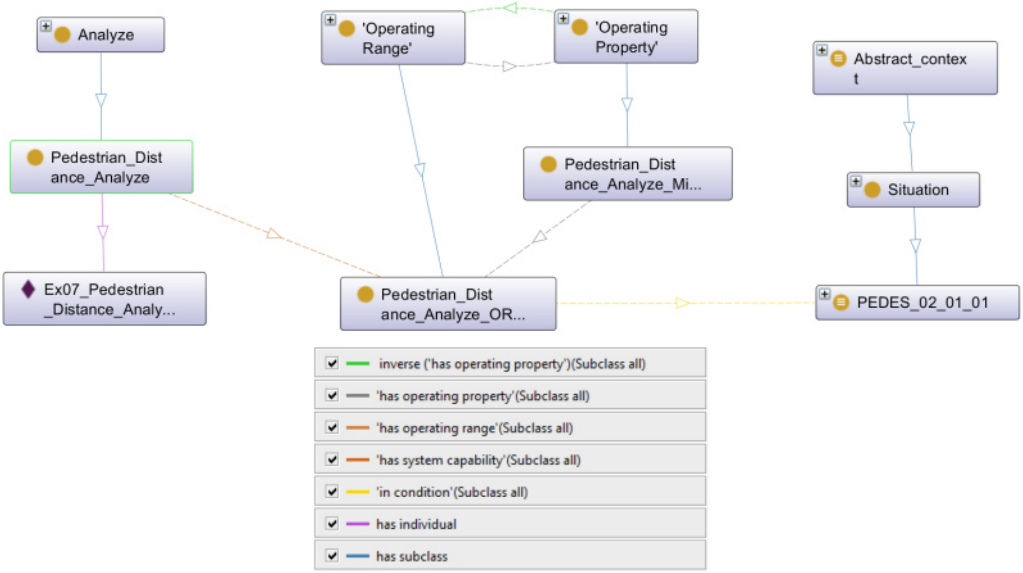


Figure B.11 Ontology representation of the goal model to model the possible configurations of an Analyze microservice

urations of the system (e.g. range of values for a variable, value of a constant, property to verify). For instance, it specifies how a threshold shall be set to escalate detected symptoms in an *Analyze* function.

The specification of such values and the relations of context-dependence between the components are originally inherited from the requirements at the design of the system architecture (i.e. during system analysis). Indeed, the use of a MBSE tool is recommended to systematically represent the system, and its components, and possible configurations. The resulting model of the architecture acts for us as a static source of information of the system and can be used as a basis for our ontological reflexive model.

Assessment of the behaviour and the intention of the ADS in the scope of one safety constraint The first goal model details how the adaptations of the ADS may be tuned. Hence, it details the different configuration for the AMs to perform its function (i.e. specific computation or value of constants) and their operating range.

Figure B.11 shows how the possible configurations are specified in the ontology for a given *Analyze* microservice in the AMs. It depicts the relations between the microservice, the values for the configuration and the context in which the microservice and its values are appropriate. We can see that the *Pedestrian_Distance_Analyze* is related to the *Pedestrian_Distance_Analyze_ORange* and specify the restrained operating context as the situation *PEDES_02_01_01*. Besides, the *Pedestrian_Distance_Analyze_MinDist* property extends the *Pedestrian_Distance_Analyze_ORange* and defines the minimum distance to ensure between the car and the pedestrian in this situation.

Metrics for the evaluation of future configuration from the current context

The second goal and adaptation model mainly specifies how the *Analyze* and *Plan* functions need to be performed in the OAM. It provides a sufficient source of knowledge to

determine whether the current managed AMs requires a reconfiguration for the current context.

In the OAM, the *Analyze* function exploits the operating range and the current set of identified *Abstract Context* individuals to identify all the AM microservices defined as relevant. Previous Figure B.11 already illustrates the context-dependence of an *Analyze* microservice that answer to such request.

Then, the *Plan* function evaluates the outputs of the planning services that propose future possible and relevant configurations for the current situation. We design this evaluation around two metrics: distance and semantic coverage of the configuration. They both take the current configuration as origin of comparison. The distance determines the number of changes that need to be performed between the two configurations. It aims to monitor and evaluate the stability of the system regarding the reconfiguration. The semantic coverage serves to identify the number of semantic relations the future reconfiguration is involving. According to the complexity of the decision process and the evaluation of its outputs, some change plans may result to be reckon as an optimum plan (i.e. optimizing all available restrictions), or the most relevant one with the less AM reconfigurations (i.e. optimizing the number of reconfiguration) for example. Those metrics contribute to specify the type of change plan or a sensitivity interval to adopt for the evaluation in the OAM's *Plan* function.

B.6 Service-orientation for shared knowledge and MSBE

In the previous sections, we have presented the different parts of the ontology that constitute the knowledge bases and the levels of adaptation for the framework. They possess refined and semantic models that drive the decision-making to provide more explainability and transparency.

The service-orientation of the framework and the resulting different microservices are initially captured by the model of the architecture in a MBSE tool. It embeds all the specifications for the different microservices, configurations and input/output bindings.

We have seen that the use of an ontology is an additional step based on this existing model to build up a reflexive model of the system. It also provides composable mechanisms and resources to understand and adapt to its environment appropriately. Hence, it would be definitely interesting to streamline the original structure and the changes performed on the initial model in the MBSE tool to the ontology.

Table B.1 presents the origin of the requirements of the particular models of our reference architecture (i.e. from the MBSE tool to the ontology or from the ontology directly). It also summarizes the involvement of the different types of microservices in the models. Finally, Figure B.12 provides a view of the central axioms and the relations that drive the two levels of adaptation at run-time. Straight arrows correspond to an inheritance relation from the child to the ancestor while dot arrows correspond to the specified labelled relation.

Table B.1 Choice of knowledge representation for each levels and types of model in our reference architecture

Types	Grid of microservices (Layer 2)	Semantic Orchestrator (Layer 3)	MAPE run-time usage
Context model	O	O + O	In: M Out: M,A
Configuration model	D to O	D to O	In: M Out: A,P
Capability model	D to O	O	In: None Out: A,P
Plan model	O	O	In: P Out: E
Goal & Adaptation	D to O	O	In: None Out: M, A

O stands for: Ontology, D to O: from Design on MBSE tool to Ontology, M: Monitor, A: Analyze, P: Plan, E: Execute

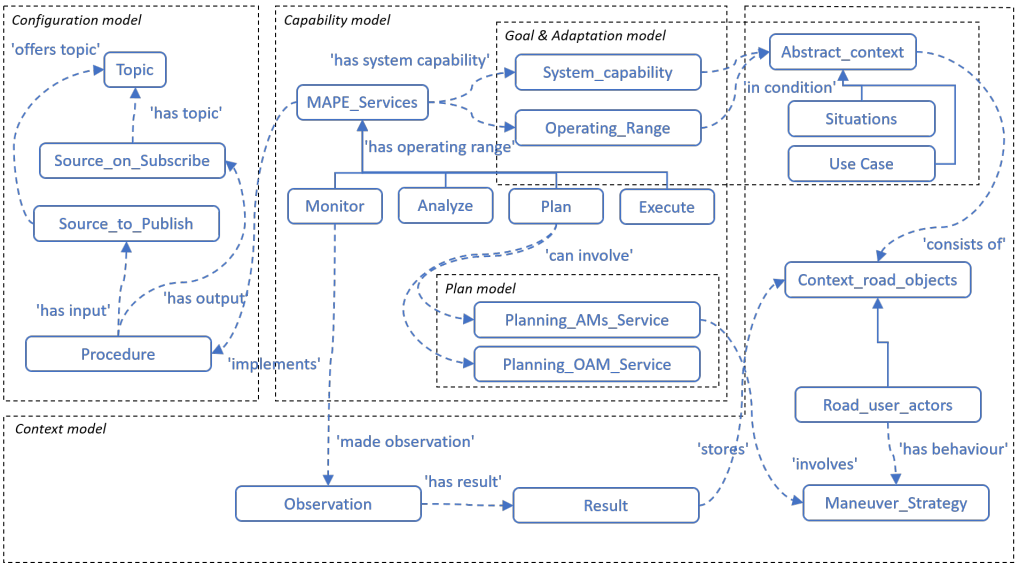


Figure B.12 Overview of the main axioms and the relations of the ontology per model in the framework

ECOLE DOCTORALE :
ÉCOLE DOCTORALE SCIENCES EXACTES ET LEURS APPLICATIONS - ED 211

LABORATOIRE :
Collège STEE – Côte Basque, LIUPPA, Anglet, France

CONTACT

Matthieu CARRE
matthieu.carre@outlook.com