Safe-AV: A Fault Tolerant Safety Architecture for Autonomous Vehicles

Safe-AV: A Fault Tolerant Safety Architecture for Autonomous Vehicles

By

Syed Asim Shah, B.Eng Mechatronics

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfilment of the Requirements for the Degree

Master of Applied Science

McMaster University

© Copyright by Syed Asim Shah, March 2019

Master of Applied Science (2019) McMaster University

(Software Engineering) Hamilton, Ontario

Title: Safe-AV: A Fault Tolerant Safety Architecture for Autonomous Vehicles

Author: Syed Asim Shah, B.Eng Mechatronics (McMaster University)

Supervisors: Dr. Mark Lawford, Dr. Alan Wassyng

Number of Pages: viii, 166

Abstract

Autonomous Vehicles (AVs) should result in tremendous benefits to safe human transportation. Recent reports indicate a global average of 3,287 road crash related fatalities a day with the blame, in most cases, assigned to the human driver. By replacing the main cause, AVs are predicted to significantly reduce road accidents – some claiming up to a 90% reduction on US roads. However, achieving these numbers is not simple. AVs are expected to assume tasks that human drivers perform both consciously and unconsciously – in some instances, with Machine Learning. AVs incur new levels of complexity that, if handled incorrectly, can result in failures that cause loss of human life and damage to the environment. Accidents involving SAE Level 2 vehicles have highlighted such failures and demonstrated that AVs have a long way to go. The path towards safe AVs includes system architectures that provide effective failure monitoring, detection and mitigation. These architectures must produce AVs that degrade gracefully and remain sufficiently operational in the presence of failures. We introduce Safe-AV, a fault tolerant safety architecture for AVs that is based on the commonly adopted E-Gas 3 Level Monitoring Concept, the Simplex Architecture and guided by a thorough hazard analysis in the form of Systems-Theoretic Process Analysis (STPA). We commenced the architecture design with a review of some modern AV accidents which helped identify the types of failures AVs can present and acted as a first step to our STPA. The hazard analysis was applied to an initial AV archi-

tecture (without safety mechanisms) consisting of components that should be present in a typical AV (based on the literature and our ideas). Our STPA identified the system level accidents, hazards and corresponding loss scenarios that led to well-founded safety requirements which, in turn, evolved the initial architecture into Safe-AV.

Acknowledgements

I would like to start off by thanking my supervisors, Dr. Mark Lawford and Dr. Alan Wassyng for the superb advice, ideas and discussions throughout my stay as a graduate student. They provided me not only guidance but significant opportunity. As part of their team, I was involved in a number of projects that gave me a diverse experience and allowed me to hone a varied skill set.

I appreciate the discussions with friends, fellow graduate students and colleagues that lead to breakthroughs or ideas that would have otherwise not come at all or with much difficulty. A huge thanks to my good friend, Thomas Chowdhury, who got roped into helping me with the hazard analysis. We spent the better part of a year discussing, arguing and "fighting" over our STPA which turned out to be a significant addition to my thesis.

Last but not least,thanks to my parents, Syed Waliullah Shah and Roshan Parveen Abbasi, and my siblings for supporting me throughout this degree. I appreciate them permitting me to shirk certain responsibilities and taking over others while I completed this thesis. Thanks to my older sister, Rafia, for assisting with and advising on the aesthetics of the architecture. Thanks to my younger sister, Bisma, for assisting with and keeping me company when creating the arduous tables in the hazard analysis.

Contents

List of Figures

List of Tables

List of Acronyms

AC Autonomous Control

ACC Adaptive Cruise Control

ADAS Advanced Driver Assistance Systems

AEB Autonomous Emergency Braking

AI Artificial Intelligence

ASIC Application-Specific Integrated Circuit

ASIL Automotive Safety Integrity Level

AV Autonomous Vehicle

BBW Brake-By-Wire

CM Core Module

CPS Cyber-Physical System

CRC Cyclic Redundancy Check

DDD Driver Drowsiness Detection

DMS Driver Monitoring System

DNN Deep Neural Network

DS Decision Switch

ECC Error-Correcting Code

ECU Electronic Control Unit

ETA Event Tree Analysis

FCR Fault Containment Region

FMEA Failure Mode and Effects Analysis

FTA Fault Tree Analysis

GM General Motors

HA Hazard Analysis

HMI Human Machine Interface

HOV High Occupancy Vehicle

I/O Input/Output

IIHS Insurance Institute for Highway Safety

L1 Level One

L1D Level One - Degraded

L1F Level One - Functional

L2 Level Two

L3 Level Three

L3T Level Three - Tester

L3W Level Three - Watchdog

LKAS Lane Keeping Assist System

MC Message Counter

ML Machine Learning

MPC Model Predictive Control

MTW Modelling The World

NHTSA National Highway Traffic Safety Administration

NTSB National Transportation Safety Board

OEM Original Equipment Manufacturer

PAC Planning and Autonomous Control

PMV Process Model Variable

RTOS Real-time Operating System

SBW Steer-By-Wire

SNR Signal-to-noise Ratio

SoC System On Chip

SPA Sense-Plan-Act

STAMP Systems-Theoretic Accident Model and Processes

STPA Systems-Theoretic Process Analysis

TBW Throttle-By-Wire

UCA Unsafe Control Action

V2X Vehicle-To-Everything

VA Vehicle Actuation

WHO World Health Organization

XBW X-By-Wire

Chapter 1

Introduction

AVs have, in recent times, garnered a significant amount of attention. From Automotive Original Equipment Manufacturers (OEMs) to research institutes to the general public, each group has a vested interest in their development, stands to benefit from their successes and will play a role in how the technology shapes society. AVs have the potential to have a transformative impact on a number of industries. The technology will drastically change the transportation of goods and people through increased efficiency and, more importantly, improved safety.

As is the case with any change, the technology will bring with it a host of challenges and concerns. We have to figure out, amongst other things, solutions to problems in the space of Machine Learning (ML), cyber security, functional safety, software design and sensor fusion. We have to find ways to make modern sensors (e.g. radar, LIDAR) cheaper, more effective and robust in the face of different environmental factors. We have to integrate the technology in a manner that does not endanger the lives of current road users, secures public trust and prevents misuse of the technology by over-reliant or malicious actors. The hurdles to the switch from driver operated vehicles to driverless are not only limited to technical ones. Governments, of all levels,

have to figure out ways to modify current infrastructure to support these new (and, potentially, eventually only) vehicles on the road. Legislators have to navigate the complexities of automotive regulation to propose new laws (or revisit old ones) to incorporate and govern AVs.

While the *incomplete* list of issues above will have to be addressed, this thesis will attempt to tackle a specific issue, the functional safety of autonomous vehicles. These vehicles are expected and intended to assume the complex tasks and numerous responsibilities that human drivers perform both consciously and unconsciously – in some instances, with the use of Machine Learning. To that end, they incur new levels of complexity resulting in emergent and, seemingly, non-deterministic behaviour that, if handled incorrectly, can result in catastrophic failures that cause loss of human life and damage to the environment. To develop safe AVs, a rigorous design process and system architecture[1] that make use of the state of the art and best practices in the automotive industry, systems engineering, functional safety and fault tolerance are required. The architecture must take into account how and when these systems can fail and provide a design that allows effective failure monitoring, detection, and mitigation. Ultimately, the architecture must produce AVs that, in the presence of failures, are able to degrade gracefully and remain sufficiently operational (i.e fail-operational) such that the vehicle can reach a safe state that prevents harm. To address these demanding requirements, this work proposes a fault tolerant system architecture in the form of Safe-AV that is designed with an emphasis on functional safety. Henceforth, such an architecture shall be referred to as a *safety architecture.*[2]

[1] An architecture can be considered as a model of system that shows the system's structure, how it behaves, its components and, finally, their interactions.

[2] Definition [1]: *'Set of elements and their interaction to fulfill the safety requirements'*

1.1 Motivation

According to a World Health Organization (WHO) publication [2], every year, there are an average of 1.25 million global fatalities and 50 million global injuries related to road crashes (Figure 1.1). Put in another way, that is an average of 3,287 deaths per day. The report indicates that if no action is taken, these numbers are predicted to increase significantly by 2030 and that the leading causes of these deaths are, in one form or another, the actions of the drivers operating the vehicles. Backing up that view, a National Highway Traffic Safety Administration (NHTSA) publication [3] states that in 94% of the accidents in the US, the critical reason (defined as the last event in the crash causal chain) was assigned to the driver.

The WHO provides multiple recommendations for reducing these numbers. They predict that a 5 % cut in the speed limit in certain areas can result in a 30% reduction in fatalities (Figure 1.2). Strict enforcement of drink-driving laws can further reduce them 20% (Figure 1.3). While they also advocate the design of safer cars, the fact remains, most of these recommendations are behavioural changes on the part of the operators of the vehicles. With an AV, the main cause of these accidents, the human, is either replaced or heavily supported by the autonomous vehicle software. It is proposed that AVs will significantly reduce vehicle related accidents with some claiming there will be as much as a 90% reduction in accidents on US roads [4].

The proposed reduction in accidents can only be a reality if AVs behave correctly and safely. As the technology is still in its infancy, the relevant industry standards, laws and best practices are in their early stages as well. It is not sufficient (but perhaps necessary), for example, to adhere to current safety standards (such as ISO 26262 [1]) as they are more applicable to current non-autonomous vehicles. As such, it becomes unclear if the vehicles touting autonomous functionality in the current

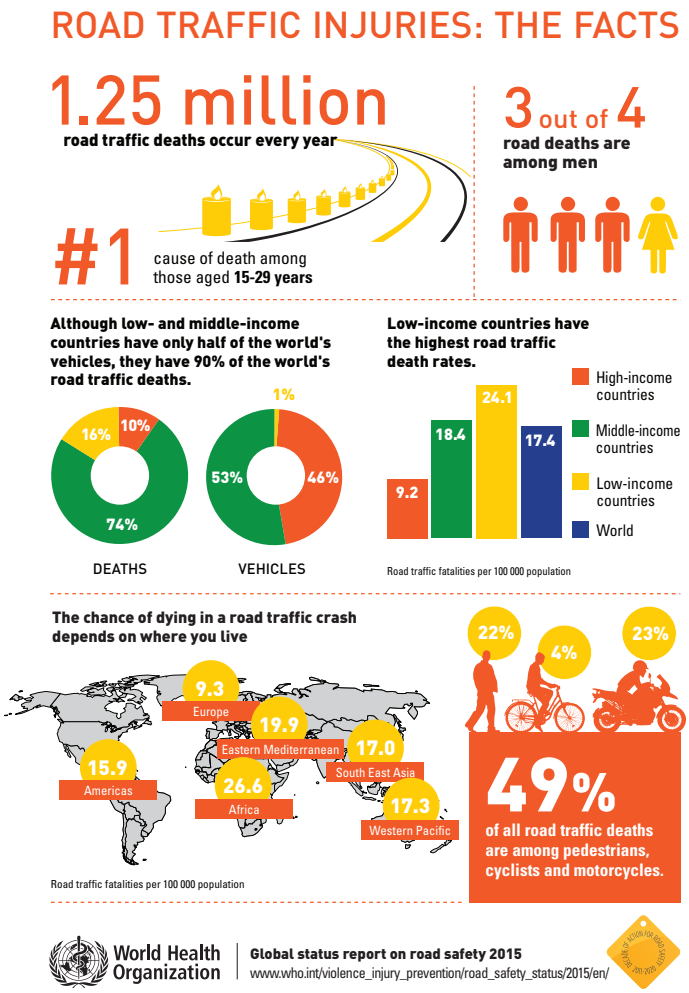

Figure 1.1: WHO's Infographic on Road Traffic Injuries

climate are, in fact, designed sufficiently well and safe. To be clear, current vehicles with Advanced Driver Assistance Systems (ADAS) such as Autonomous Emergency Braking (AEB) have still provided a significant benefit to driver safety and have prevented multiple accidents that would have otherwise occurred (a report [5] by the Insurance Institute for Highway Safety (IIHS) indicates that systems with AEB have reduced rear-end collisions, on average, by 40% on US roads). What is important here, however, is that we do not stop improving them further. Failing that, we may see situations where the AV actually increases the number of accidents. Consider

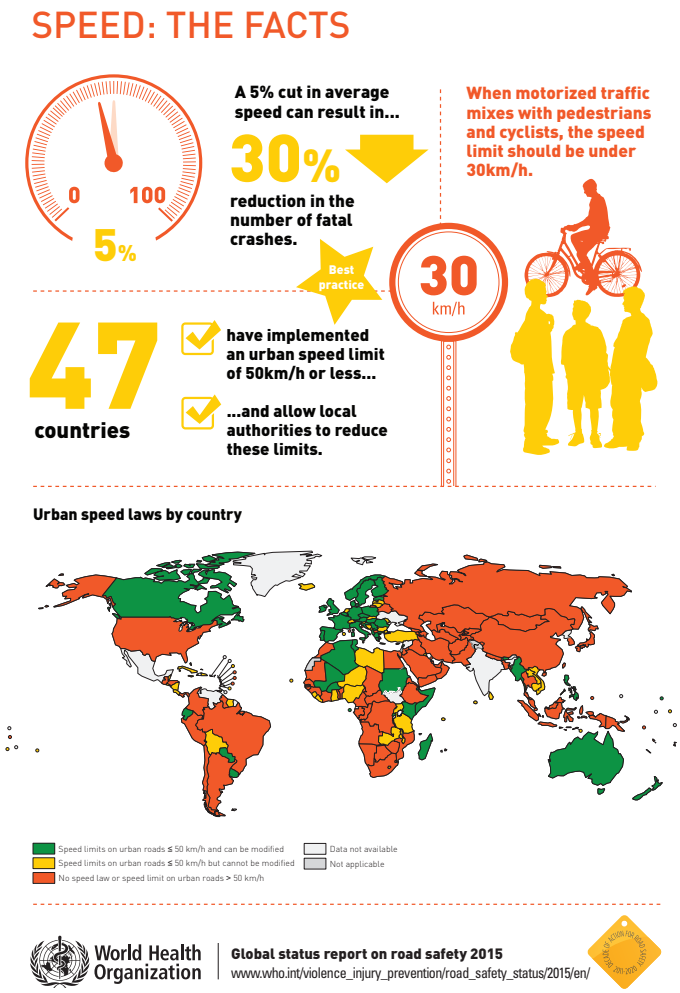

Figure 1.2: WHO's Infographic on Speed

the scenario; you are driving behind a vehicle that proceeds to crash into a wall. A human operated vehicle will most likely react immediately and attempt to avoid the accident. With each subsequent human driven vehicle, the chance of hitting the wall decreases significantly. Now consider a fleet of AVs, all operating on the same software, approaching the accident. If they share a common flaw that prevents them from detecting the imminent collision, we may very well see the whole fleet crashing into the wall resulting in significant harm.

Our proposed solution to this problem is the aforementioned, Safe-AV. To make

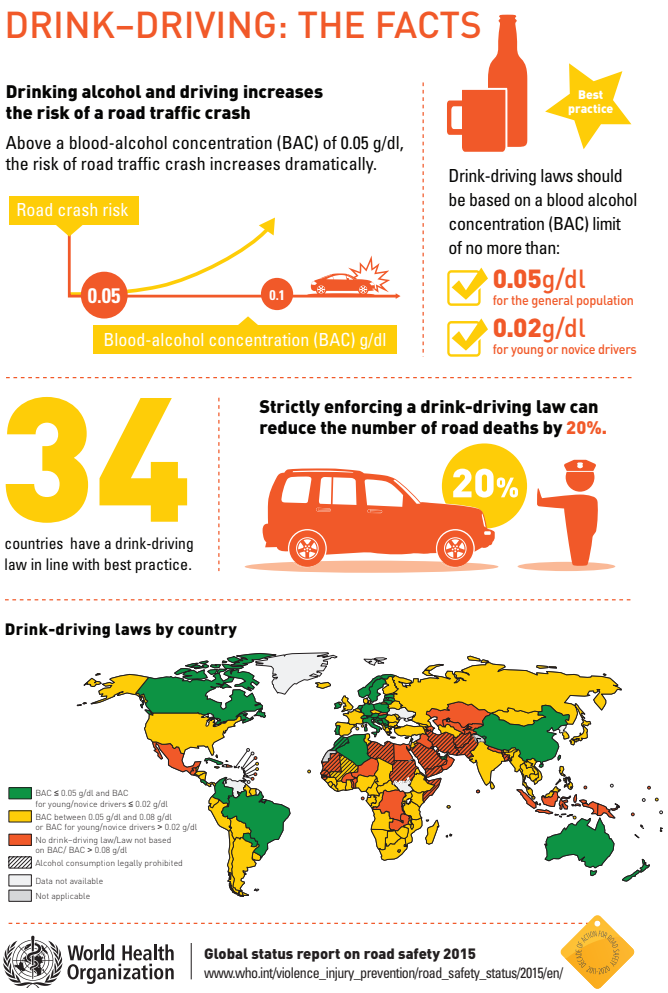

Figure 1.3: WHO's Infographic on Drink-Driving

a case for its necessity, we first look at the shortcomings of current AV system architectures exhibited in the various fatal accidents involving modern Level 2 vehicles in Chapter 3. These accidents could have been, arguably, avoided with architectures that imposed proper design practices, provided monitoring of safety critical systems (ML based and otherwise) to detect failures and allowed fail-operational behaviour to bring the vehicle to a safe state. Even in the literature, we can see that there is lack of focus on functional safety. A review in Section 2.1 of current system architectures for autonomous vehicles shows that most follow a layered approach in which the next

higher layer, in addition to using the lower layer to achieve its main goal, also monitors the health of the layer below. While this type of arrangement provides a measure of improved safety, it does so at the expense of larger and more complex higher layers as they have to meet both their functional (i.e. path planning) and monitoring obligations. This method of enforcing safety violates a fundamental principle of dependable systems - separation of concerns [6] and, in particular, separation of safety and control as a method of dealing with complexity in safety critical systems. Perhaps this lack of focus on functional safety is merely the consequence of the technology being new. It stands to reason that the initial architectures would simply focus on producing an initial functional design instead of focusing on specific views such as safety and reliability.

1.2 Goals

The goal of this thesis is to tackle some key problems in achieving safe autonomous driving systems:

1. There are no standard reference system architectures that help "guarantee", to a reasonable degree, the functional safety of autonomous vehicles.

2. There is no way of observing that the ML based systems (e.g. object detection, path planning) are failing or encountering anything out of the norm.

While we initially hoped to fully address both of the highlighted problems leading to a foundation for autonomous systems safety, we shall focus mainly on providing a solution for the first and a partial solution to the second. Safe-AV is a 3 level safety architecture that adopts the "monitor the monitor" principle of the E-GAS 3 Level Monitoring Concept (Section 2.8) as well as the fail-operational design of the Simplex

Architecture (Section 2.9) to the autonomous setting. The architecture provides a means of monitoring failures of ML based systems while initially treating them as blackboxes.

1.3 Contribution

This work aims to contribute to the domain of functional safety in the automotive industry. Combining engineering best practices in the automotive industry, software engineering and functional safety, we designed a fault tolerant, 3 level safety architecture, Safe-AV. The key contributions of the thesis are as follows:

1. A summary and discussion of a select number of accidents involving current AVs that provides possible causes and potential solutions. The purpose of this contribution is to gain an insight into how current SAE Level 2 autonomous vehicles fail and to provide a starting step for our Hazard Analysis (HA). The eventual goal is to provide mitigations to these accidents, if possible, with Safe-AV.

2. A comprehensive STPA HA for SAE Level 4/5 vehicles that provides a set of safety requirements that (1) partially guide in the development of Safe-AV and (2) support our preexisting ideas on the design of Safe-AV. A review of the literature shows that little work has been in the area of STPAs on autonomous vehicles. Specifically, we found that no work provided a comprehensive set of safety requirements that could be used to guide Safe-AV.

3. A novel architecture that attempts to provide a solution to the current lack of functional safety focus in current autonomous vehicle system architectures by providing:

- A clear separation of safety and control that results in a demonstrable reduction in the effort of the safety argument and an increase in its confidence. This separation also allows Automotive Safety Integrity Level (ASIL) decomposition in a manner that assigns complex components (sometimes consisting of ML algorithms) a low ASIL while their simpler and more deterministic safety systems are assigned the high ASIL needed to satisfy their safety requirements.
- A means to monitor failures in ML based systems while treating them as blackboxes.
- A means for graceful degradation and fail-operational behaviour resulting in increased system reliability and safety.
- A means to monitor systems tasked with ensuring system safety i.e. monitoring the monitor.

1.4 Outline

The thesis is organized as follows: Chapter 2 discusses some of the current literature in the area of autonomous vehicle architectures and HAs. It introduces and summarizes both the E-Gas 3 Level Monitoring Concept and the Simplex Architecture. The chapter also introduces some relevant concepts and provides the necessary background for the rest of the thesis. Chapter 3 presents a summary and discussion of a select number of accidents involving AVs. Chapter 4 presents our STPA on autonomous vehicles and the resulting safety requirements. Chapter 5 introduces and describes the proposed Safe-AV architecture. Chapter 6 provides a discussion on how Safe-AV meets the safety requirements obtained in the HA. It also discusses the architecture's principles, the benefits it brings, how it can be improved and how Safe-AV evolved

during this work. Finally, Chapter 7 concludes the thesis and presents some future research directions in the area of autonomous safety.

Chapter 2

Literature Review & Preliminaries

This chapter will discuss the current state of the literature in AV Architectures. Then, we will introduce and summarize some of the important concepts and background information relevant to the thesis.

2.1 Autonomous Vehicle Architectures

There have been a number of proposed architectures for autonomous vehicles over the past few decades. This section will outline, in a somewhat chronological manner, the literature in this area. We shall discuss the commonalities found across the proposed architectures, their benefits and finally, the gaps that they leave to be filled.

We found architectures targeting autonomous vehicles as early the late 80's [7]. These earlier architectures seem to have been significantly influenced by the then state of the art in robotics. Gat in [8] provides a history of robotics architectures starting with the classic Sense-Plan-Act (SPA). They state,

> "*dominant view in the AI community was that a control system for an autonomous mobile robot should be decomposed into three functional el-*

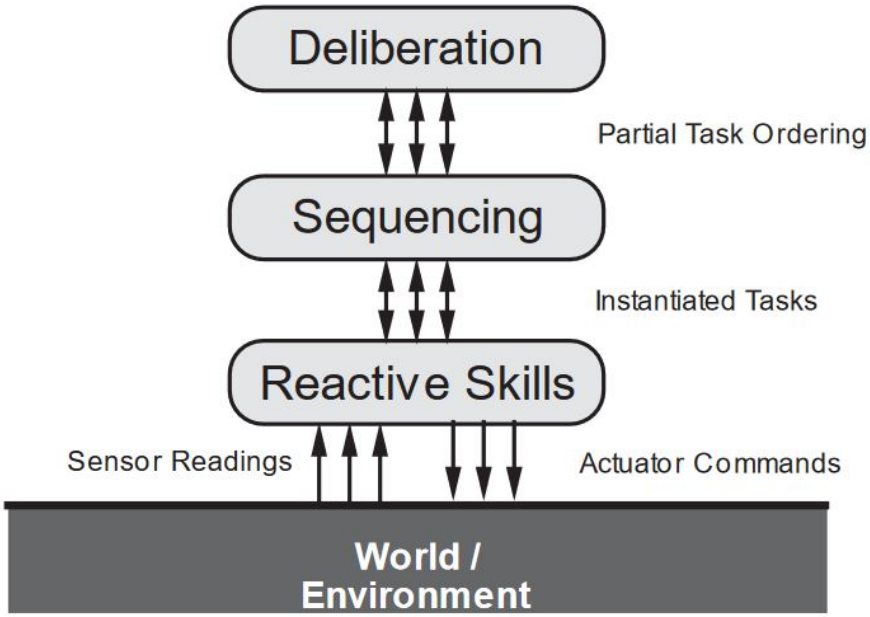

Figure 2.1: 3T - A Three Layered Architecture [9]

ements: a sensing system, a planning system, and an execution system . The job of the sensing system is to translate raw sensor input (usually sonar or vision data) into a world model. The job of the planner is to take the world model and a goal and generate a plan to achieve the goal. The job of the execution system is to take the plan and generate the actions it prescribes."

They continue on to describe SPA's successors including the three layered architectures (3T (Figure 2.1) and ATLANTIS) that became the de-facto standards at the time. These architectures consist of a controller (or skill layer), a sequencer (or sequencing layer) and a deliberator (or planning layer). The controller contains the reactive feedback loop that allows a robot to perform a chosen behaviour. The sequencer is responsible for selecting which behaviour the controller should execute based on current environment conditions. The deliberator is responsible for producing plans for the sequencer to execute or responding to queries from the sequencer.

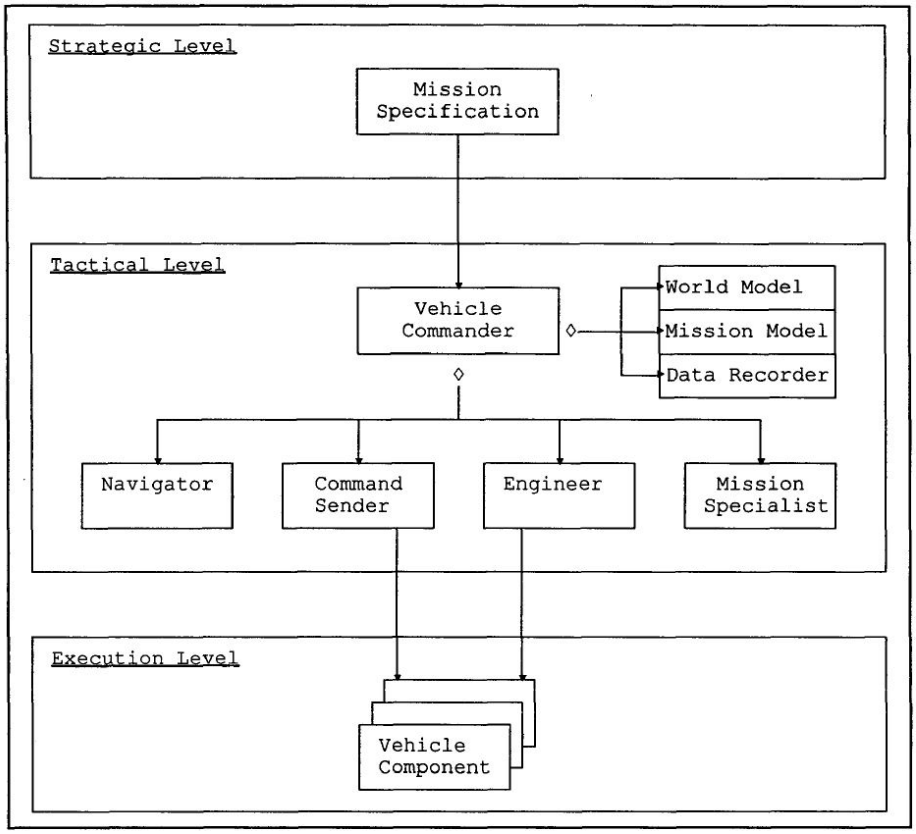

Figure 2.2: STESCA: The Strategic-Tactical-Execution Software Control Architecture [10]

As mentioned earlier, the early (an example is shown in Figure 2.2) autonomous vehicle architectures [11, 12, 13, 10, 14] were either based off of the three layered approach or were heavily influenced by it. They provided a functional view of an AV system that described their own instantiation and/or augmentation of the three layered approach. This functional view, however, did not seem to sufficiently focus on functional safety. As mentioned in the previous chapter, AVs are highly safety critical system that are required to be fail-operational. To achieve this, their architectures have to implement fault tolerance, functional monitoring and provide degraded functionality. Koopman and Wagner [15] express the need for research in monitoring concepts in the context of the automotive standard *ISO 26262 Road Vehicles – Func-*

tional Safety [1]. They also discuss fail-operational system design and how to achieve it. In terms of functional safety, some of the aforementioned architectures do discuss the concept of monitoring wherein a higher layer is able to detect and subsequently, handle failures in lower layers and provide degraded functionality. While this arrangement provides a measure of improved safety, it results in systems that are typically large and complex due to each layer's obligation to meet both its main functional requirements (e.g. path planning) as well as the safety ones. If such monitoring is the main method of enforcing safety, these architectures violate a fundamental principle of dependable systems - separation of concerns [6] and, in particular, separation of safety and control as a method of dealing with complexity in safety critical systems [16]. Wassyng, Lawford and Maibaum [16] state,

> *"We believe that separation of control systems and safety systems in the nuclear power industry is not only a good principle to follow, but that rigorous adherence to this principle should make it possible to analyse the system to an extent where we develop much greater confidence in the safety of the plant .... but the primary reason is that the reduction in complexity allows us to employ techniques that currently would not be possible for more complex systems"*

Koopman and Wagner [15] also share this sentiment as they advocate the "use of a monitor/actuator pair architecture to separate the most complex autonomy functions from simpler safety functions"[1]. The benefits of a less complex safety system as also explored in [17] where Sha argues that the most important consideration for system reliability is not software diversity but the existence of a simple and reliable component that can guarantee critical properties.

[1]The monitor/actuator pair architecture is discussed later in this chapter

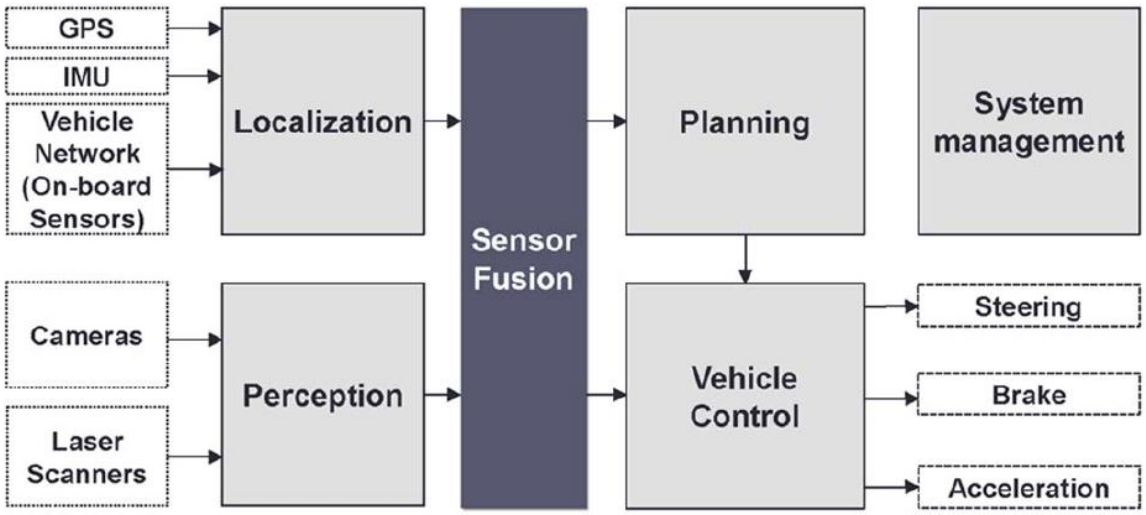

Figure 2.3: Architecture of A1 [18] – Winner of the 2012 Autonomous Vehicle Competition in Korea

We shall now start discussing the modern (post 2000) autonomous vehicle architectures where it seems that the drawbacks mentioned above still apply. These architectures differ from each other to a limited degree as most are still based on the SPA approach or the three layer architecture or a combination of both. We shall divide the modern architectures into those which were designed for the various autonomous vehicle competitions (e.g. DARPA Grand Challenge) or challenges/experiments and those that were not. The architectures from the first category [19, 20, 21, 22, 23, 24, 18] had the benefit of an actual implementation in a vehicle. This allowed them the type of refinement not possible with more abstract architectures such as ours. These architectures also provided an excellent amount of implementation detail and showcased what was currently feasible. This refinement and design, however, would have, for most, been done with the specific goals and requirements of the competition in mind (e.g. specific tracks, specific scenarios). The consequence of this, in the case of some if not all, is that the vehicles were limited to performing in specific scenarios and conditions and this may have been reflected in their architectural design. In other

words, if the objective of the competition is not functional safety, it stands to reason that competition entrants would be less motivated to focus on functional safety further than what is needed for success in the competition. However, we would be remiss if we did not mention that some of these architectures did offer redundancy and fault tolerance. Others [23] benefited from the active safety features (such as AEB) implemented in the base vehicles. Finally, others, such as the one shown in Figure 2.3, implemented some form of a system management/monitoring component that supervised vehicle health and faults and were able to react accordingly. However, the points highlighted above (regarding separation of concerns, simplicity of safety systems, etc.) still apply.

We shall divide the architectures from the second category (architectures not designed for AV competitions) into two categories:

1. those that either do not focus on functional safety at all or do so but still violate the aforementioned principles, and

2. those that do have a focus on functional safety and often times provide system monitoring and reactions that allow for fail-operational behaviour

The first subcategory consists of architectures from [26, 27, 28, 29, 30, 25, 31, 32, 33]. An example can be seen in Figure 2.4 where it is clear that a significant amount of detail is provided from a functional perspective but there is a gap from a functional safety point of view.

The architectures in the second subcategory will be discussed below in a bit more detail. The authors in [34] discuss the concept of a *Surveillance and Safety System* whereby safety components (outlined in red in Figure 2.5) monitor the vehicle's sensor values, heartbeats, cycle times and function outputs to calculate the vehicle's state with respect to a set of performance criteria. This information, combined with heuris-

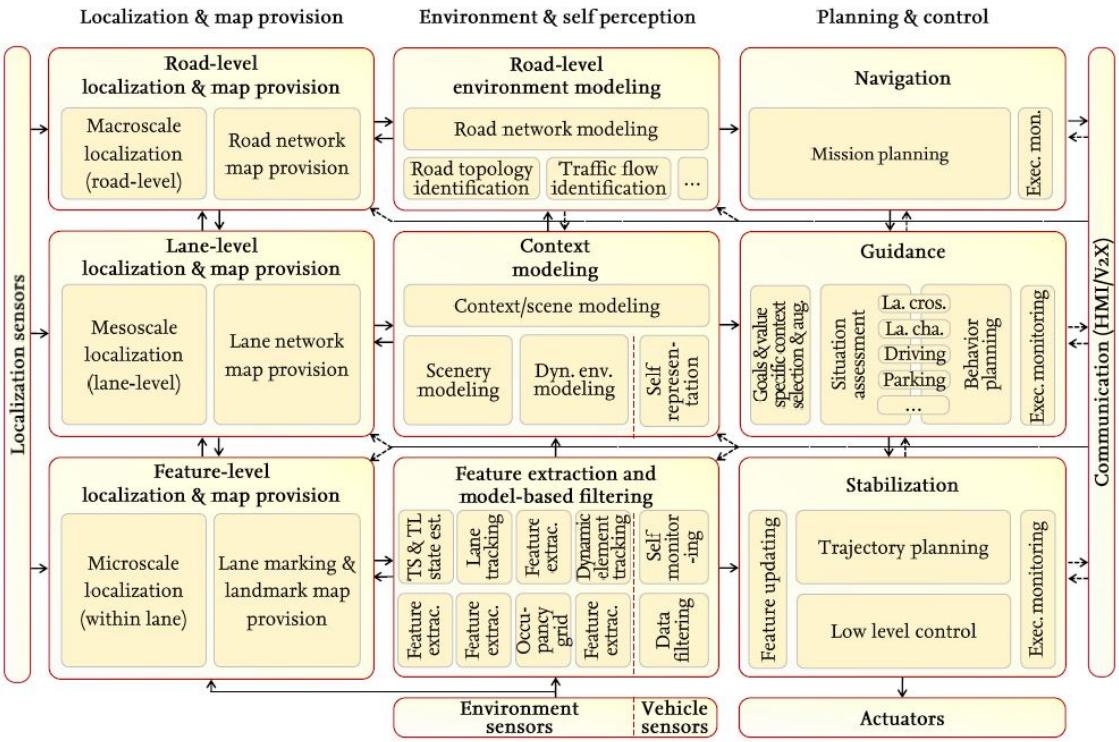

Figure 2.4: Architecture shown in [25]

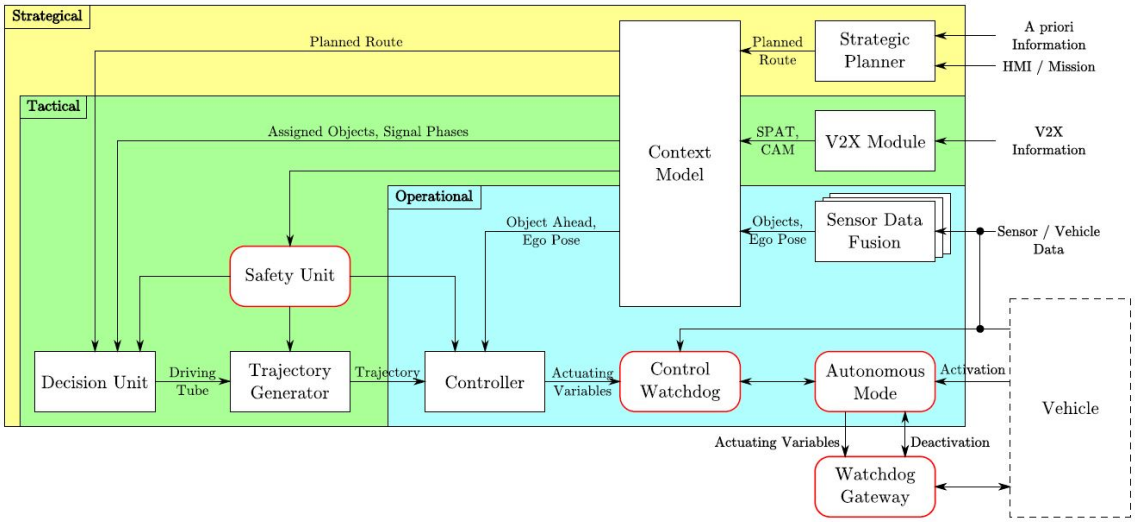

Figure 2.5: Architecture from [34]

tics, is then used to decide which degraded functionality to perform. The architectures in [35, 36, 37] share a similar design with their *Safety Sustainer* component, *Performance Assessment* units and *System and Safety Management* block respectively. It is also shared in [38] though the work does not go into much detail.

While these architectures respect separation of control and safety and also allow function monitoring resulting in a safer architectural design, we note that they do not provide any mechanism to "monitor the monitor". If the required level of reliability and safety is dependent upon the monitors operating correctly and detecting failures in the various functions, failures of the monitor have be detected and should result in a system transition to a safe fail-operational state. This becomes especially relevant when we consider the centralized global monitors seen in [35, 37, 38] that are potential single points of failure.

This issue is addressed in [39] as seen in Figure 2.6. The work is similar to the others wherein they monitor system boundaries during normal operation and transfer the system to a safe state in the case of violated system boundaries. However, where they differ from the others (in addition to "monitoring the monitor") is their clear hardware separation of perception, function (planning) and control which each consist of a local monitor that monitors internal system faults. These local monitors (along with, notably, a plausibility monitor and a function boundary monitor on the perception only [it is not clear though why this is not done for control and planning as well]) feed into a global monitor that is tasked with triggering "action plans" when a system boundary violation is detected. This combined with a monitor that monitors the global monitor for failure is the closest architecture to Safe-AV barring the centralized global monitor approach they have in contrast to our decentralized monitoring approach. The monitoring the monitor concept is also used in [40] where their *safety-bag rules checker* is monitored by their *safety-bag supervisor* and vice-

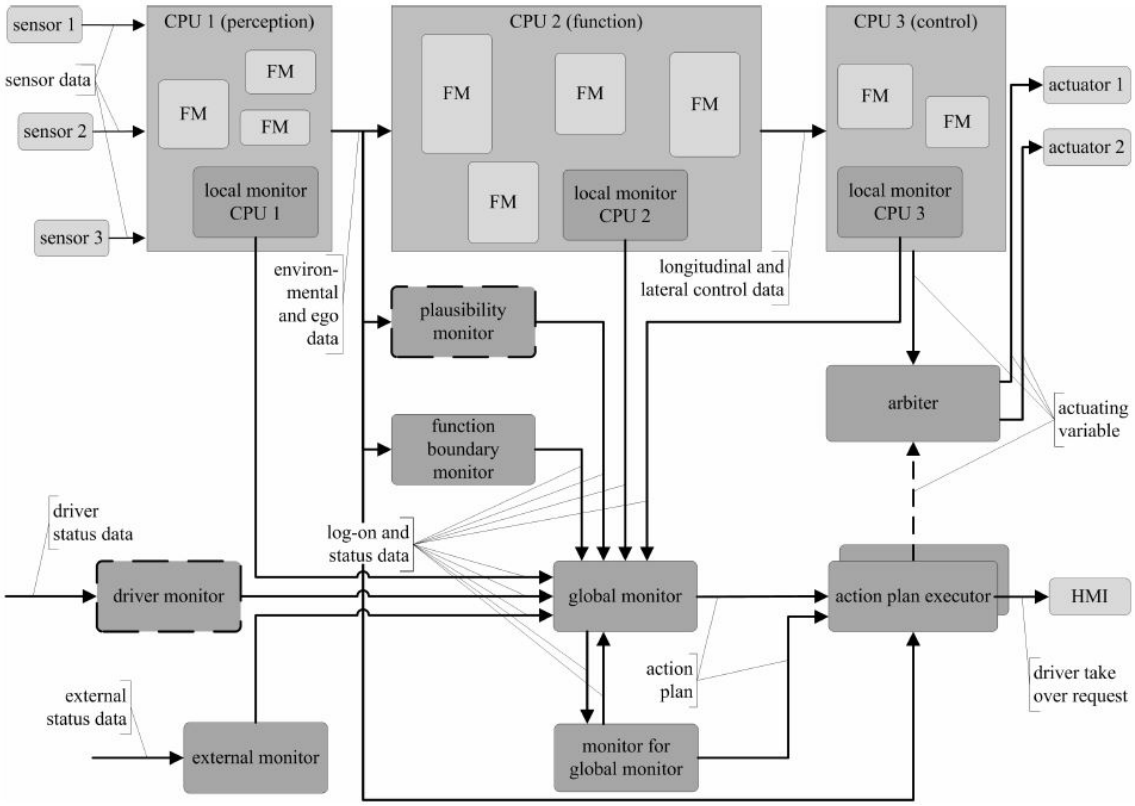

Figure 2.6: Architecture from [39]

versa.

This concludes our literature review of current architectures. As a closing note, we would like to highlight another concern that is not necessarily addressed with AV architectures discussed. Many of the algorithms required for an autonomous vehicle are and will have to be ML based. These types of algorithms are seemingly "non-deterministic" which makes it difficult to make any guarantees around their behaviour. This is an issue when we deal with safety critical systems and certifying them. Koopman and Wagner in [15, 41] discuss the problem of achieving the required level of reliability with ML techniques and issues with the current inductive learning ML systems. They suggest the monitor-actuator pattern as a possible mitigation,

"*An alternative to validating inductive learning systems to high ASIL levels*

would be to pair a low-ASIL inductively-based algorithm that sends commands to an actuator with a high-ASIL deductively-based monitor. This would sidestep the majority of the validation problem for the actuation algorithm, since failures of the inductive algorithm controlling the actuator would be caught by a non-inductive monitor based on a concept such as a deductively-generated safety envelope. ”

Problems and potential solutions for Artificial Intelligence (AI) safety are surveyed in [42]. The problem of AI interpretability - trying to make the reasoning of ML systems more transparent - aims to address concerns about the trustworthiness of ML systems [43, 44, 45]. In [43], a similarity metric is defined that is used to return the k-nearest neighbours in training data for an input as a way of providing a case based explanation of neural network decision. Building on this, [46, 47] define a similarity metric for distance in probabilistic classification and then analyse the nearest neighbour training data to find cases for which the used classifier accuracy is very low and uncertain, even though the predicted class has high probability. There has also been recent related work on characterization of failure modes of ML vision processing systems [48, 49]. These techniques provide an insight into how ML systems operate and may be key to detecting failures in ML systems before or as they are happening .

2.2 Systems Theoretic Process Analysis

STPA [50] is a new hazard analysis technique based on systems theory. Unlike traditional reliability theory based techniques – Fault Tree Analysis (FTA), Failure Mode and Effects Analysis (FMEA), Event Tree Analysis (ETA), etc. – which aim to prevent accidents caused by component failure, STPA is designed to, additionally, address accidents that are a consequence of behaviour that arises when various (not necessar-

ily failing) system components interact i.e. emergent behaviour It was developed to combat the increasing inapplicability of older HA techniques that assumed accidents are only caused by component failure, ignoring unsafe component interactions. While this assumption may sufficiently hold for older and simpler electromechanical systems that were not greater than the sum of their sufficiently independent components, it is inadequate for a modern Cyber-Physical System (CPS) that is significantly more complex.

STPA uses the top-down Systems-Theoretic Accident Model and Processes (STAMP) model in which safety is an emergent system property where proving the safety of all components is not a sufficient condition for total system safety. STPA, instead, takes a more abstract approach where the behaviour of the system is considered and a set of safety constraints are enforced on this behaviour. Safety, then, becomes a matter of enforcing these constraints instead of preventing failures of individual components. STPA considers this a dynamic control issue and aims to prevent the system from reaching hazardous states that arise due to violation of constraints. Consequently, failures are deemed a result of inadequate control.

Figure 2.7 shows the STPA process. We start by first identifying the System, its boundaries, the types of Losses we are trying to prevent and the Hazards. Table 2.1 contains definitions for these terms in the context of STPA.

The reader will notice that we have included the definition of the term 'Accident' in the table. The newer version of STPA replaced the step which listed the accidents (as done in the older version [51]) with a step that lists the losses. Our HA (Chapter 4), which commenced before the release of the newer version, did not follow this change as we found that it did not affect our analysis (we found no new hazards, causal factors or safety requirements).

The second step involves creating a Control Structure (Figure 2.8). This structure

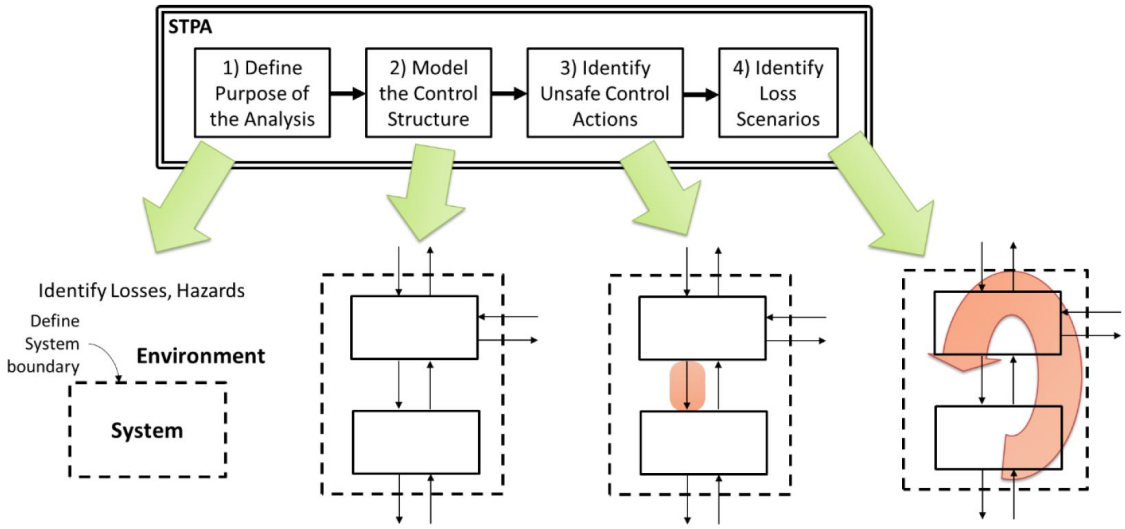

Figure 2.7: The STPA Method [50]

models the system and captures its functional relationships and interactions in the form of a feedback loop. It consists of the controlled process, sensors, a controller (or multiple controllers) and actuators. The controller provides control actions that control the controlled process.

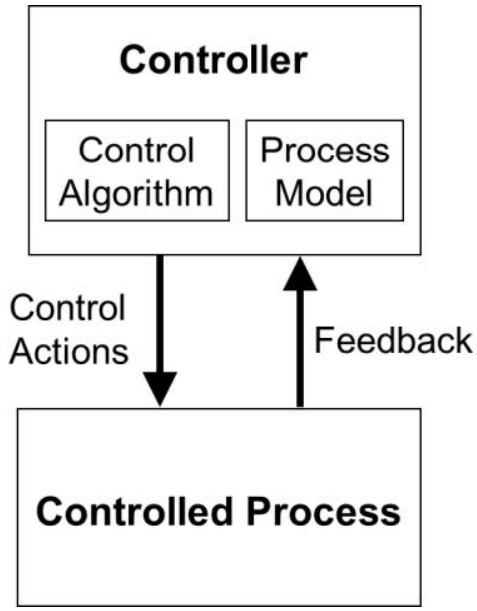

Figure 2.8: A generic Control Structure [50]

Term	Definition
System	A system is a set of components that act together as a whole to achieve some common goal, objective, or end. A system may contain subsystems and may also be part of a larger system
Accident	An accident is an undesired and unplanned event that results in a loss, including a loss of human life or human injury, property damage, environmental pollution, mission loss, financial loss, etc.
Loss	A loss involves something of value to stakeholders. Losses may include a loss of human life or human injury, property damage, environmental pollution, loss of mission, loss of reputation, loss or leak of sensitive information, or any other loss that is unacceptable to the stakeholders
Hazard	A hazard is a system state or set of conditions that, together with a particular set of worst-case environmental conditions, will lead to a loss
Control Structure	A hierarchical control structure is a system model that is composed of feedback control loops. An effective control structure will enforce constraints on the behaviour of the overall system
Unsafe Control Action	An Unsafe Control Action (UCA) is a control action that, in a particular context and worst-case environment, will lead to a hazard
Loss Scenario	A loss scenario describes the causal factors that can lead to the unsafe control actions and to hazards

Table 2.1: Definitions [50]

The third step is to identify the control actions that will lead to a hazard i.e. the UCAs. The final step is to find casual factors for the UCAs with Loss Scenarios which explore (Figure 2.9):

1. *How incorrect feedback, inadequate requirements, design errors, component failures, and other factors could cause unsafe control actions and ultimately lead to losses.*

2. *How safe control actions might be provided but not followed or executed properly, leading to a loss.*

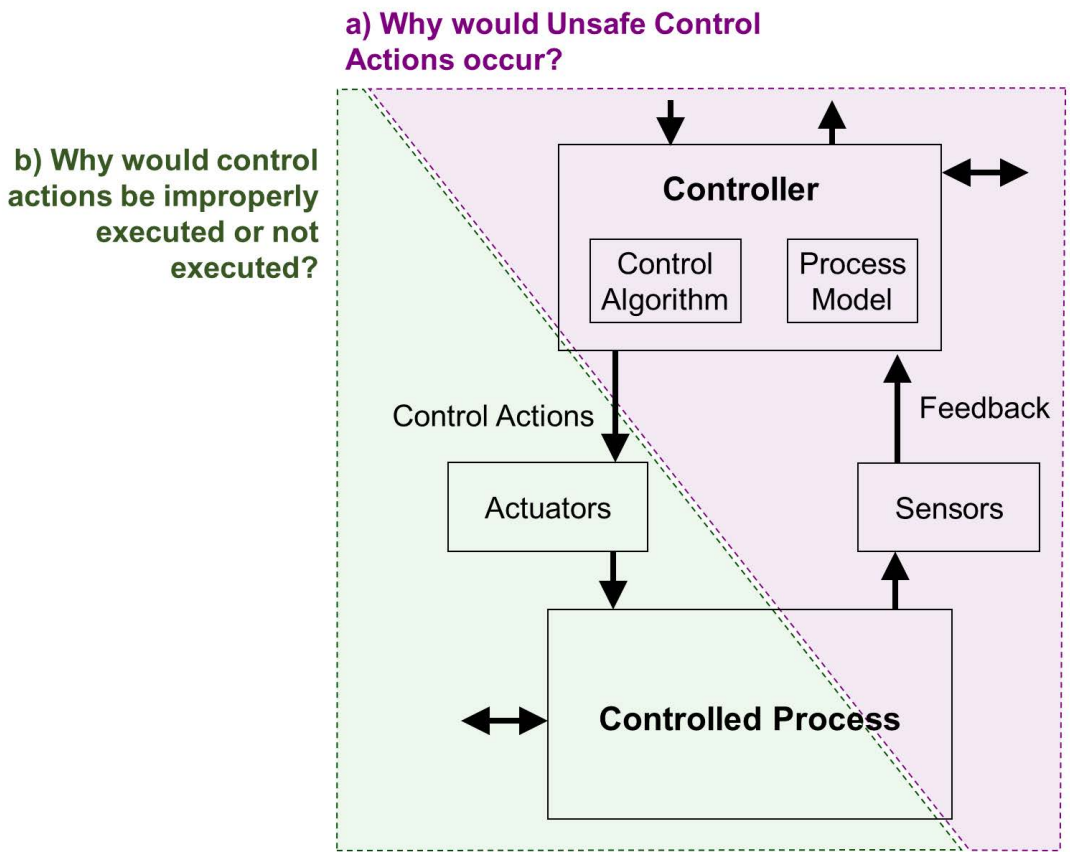

Figure 2.9: Obtaining Loss Scenarios [50]

These scenarios can then be used to obtain the safety requirements that mitigate the unsafe control actions and hazards.

2.3 STPA in Literature

There are currently a number of publications that include the application of STPA to automotive vehicles [52, 53, 54, 55]. However, there are relatively few publications on the application of STPA to *autonomous* vehicles, especially if we are interested in the resulting safety requirements. Further, there are even fewer on a complete autonomous vehicle (as opposed to specific AV functionality such as a Lane Keeping Assist System (LKAS)). In [56], the analysis was focused on Collision Avoidance and Adaptive

Cruise Control as opposed to a completely autonomous system. In [57], the authors perform a safety and security co-analysis using their six-step model. They apply STPA to accumulate safety countermeasures with security countermeasures to provide combined safety and security measures. However, detailed safety requirements were not provided. In [58], the authors apply STPA for "safety-in-use" to detect hazardous interactions in the absence of malfunctions of the systems. They apply STPA to the existing automated driving system at Continental called Cruising Chauffeur® to detect hazardous interactions between the system and other participants of the road. They also did not publish the resulting safety requirements. In [59], STPA was applied to a lane keeping assist system. The authors mentioned that a detailed analysis of other autonomous subsystems would be considered in future works.

2.4 ASIL Decomposition

An ASIL is a risk classification for specifying an element's safety requirements under ISO 26262 [1] and indicates the level of safety measures required for achieving an acceptable residual risk. An ASIL ranges from an ASIL D which represents the most stringent level to an ASIL A which is the least stringent.

Naturally, the increased complexity and safety critical nature of an AV will result in higher ASIL safety requirements for its components compared to a traditional vehicle. This in-turn makes for a very difficult and costly (in terms of effort) safety case for the vehicle. This difficulty can perhaps be reduced via architectural decisions that allow ASIL decomposition. ASIL decomposition is a method of ASIL tailoring that allows the allocation of the ASIL of a safety requirement across several sufficiently independent components that ensure compliance to said safety requirement. The benefit becomes apparent when we consider that the allocation can assign certain

components a lower ASIL than the initial one. The next section will provide an example of this.

2.5 Two Level Safety - Monitor-Actuator Pair Pattern

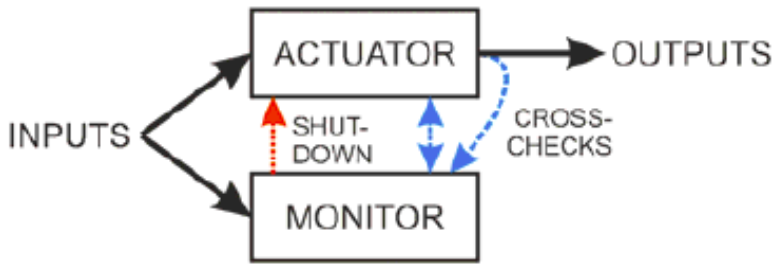

Figure 2.10: The Monitor-Actuator Pair [15]

The monitor-actuator pair (Figure 2.10) is a well established design pattern used in the design of embedded safety critical systems. As implied, it consists of two components, the actuator and the monitor. The former is responsible for fulfilling the functional requirements of the system while the latter is responsible for ensuring that the actuator performs safely and as expected. If the monitor detects that the actuator is not behaving as expected or is exhibiting a fault, it is able to safely shut the system down (or bring the system to an alternative safe state). As the onus of safety is placed on the monitor, we are able to design a complex actuator while the monitor is kept as simple as possible. The benefits of this pattern are threefold:

- It provides a clear separation of hardware between the monitor and the actuator. If we follow best practices, the monitor should be implemented on a separate controller or an Application-Specific Integrated Circuit (ASIC) and should be

developed independently. This removes the single point of failure and reduces common mode failures.

- Using ASIL decomposition, we are able to design the actuator to a lower ASIL while the higher ASIL is assigned to the monitor (provided it designed well enough to detect faults in the actuator). Benefits of this are as follows:
  - The low ASIL actuator incurs a less rigorous development process and an easier safety argument resulting in a cheaper design.
  - The high ASIL monitor can often times be much simpler than its actuator counterpart. This means that we can use a cheaper and smaller ASIC or controller allowing us to offset the cost incurred (both in terms of engineering and safety effort) by the monitor's high ASIL.

  Combined, these benefits make managing the safety and risk of a very complex system more feasible.

- The separation of safety and control combined with the isolation in hardware means that updates or modifications to the actuator can be made much more freely and easily as we do not have to re-certify the monitor with each change.

2.6 Watchdog Pattern

A Watchdog is a cheap mechanism used in embedded systems to ensure system reliability. Unlike the monitor-actuator pattern, it does not perform any sort of monitoring. Instead, at a set interval, it expects to be reset or "petted". If it is not reset, it assumes that there is a fault in the system and triggers a fault reaction. This reaction may be in the form of a simple system reset or, depending on the application, a tran-

sition to a safe state (e.g. a shutdown). The simplest watchdog can trigger a fault reaction if a fault is detected once. More complex ones allow the fault to occur multiple times while triggering a reaction at each occurrence until a threshold is reached. After passing this threshold, they may trigger a major fault reaction such as a system shutdown. An example of this can be seen in the E-Gas Monitoring Concept.

2.7 X-By-Wire Systems

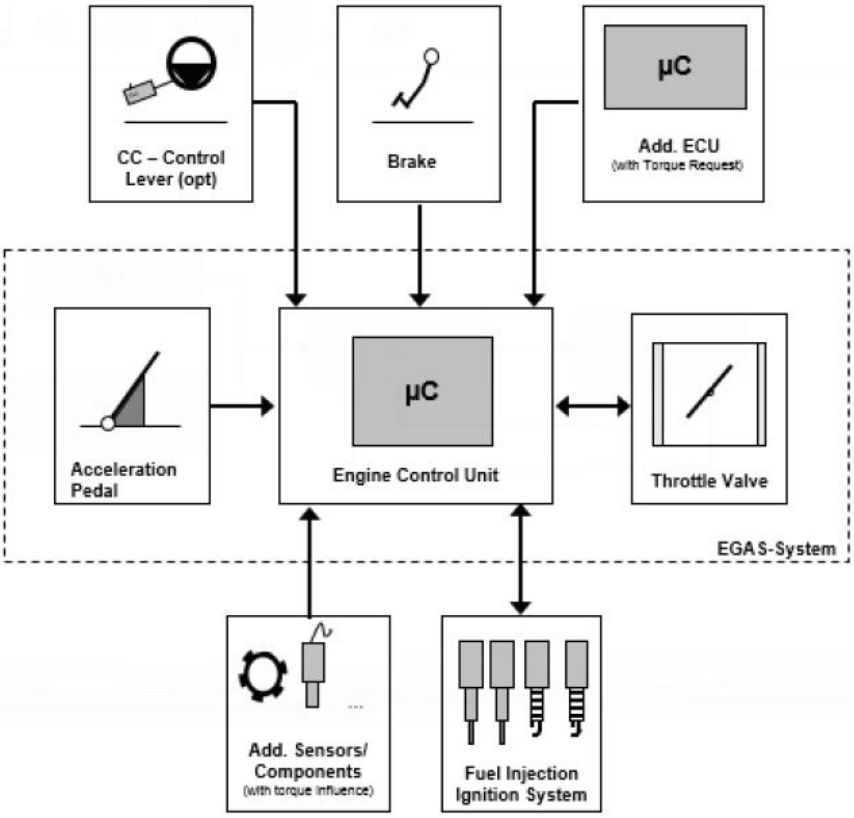

Figure 2.11: Electronic Throttle Control [60]

X-By-Wire systems partially or completely replace the older mechanical mechanisms used to control a car with electric or electromechanical ones. Currently, the most

prevalent example of this is Throttle-By-Wire (TBW). TBW (Figure 2.11) or electronic throttle control is responsible for controlling the driving torque produced by the internal combustion engine (or motor) based on the input of the driver. It consists of three parts:

1. Accelerator Pedal : The accelerator pedal is the source of the driver's interaction with the system. The angle of the pedal (or how much it is depressed) is an indication of the driver's requested torque.

2. Engine Control Unit: The ECU takes the driver's request and based on various other requirements, calculates the actuating signals required to operate the throttle valve.

3. Throttle Valve: The throttle valve controls the air intake of the engine. This in-turn controls how much power is generated by the engine.

Other examples include Brake-By-Wire, Steer-By-Wire and Shift-By-Wire. These, while not as prevalent, are slowly being incorporated into modern vehicles.

2.8 Three Level Safety - E-Gas Monitoring Concept

The E-Gas Monitoring Concept [60] was developed by five German automotive OEMs in an effort to improve the safety and reliability of the drive-by-wire operation of the E-Gas system (i.e TBW). The monitoring concept allows the safety requirements (derived from the ASIL B safety goal – "Prevention of unintended acceleration") that are allocated to the sensors, actuators and the Electronic Control Unit (ECU) of the TBW system to be met. It assumes that an impermissible vehicle acceleration can

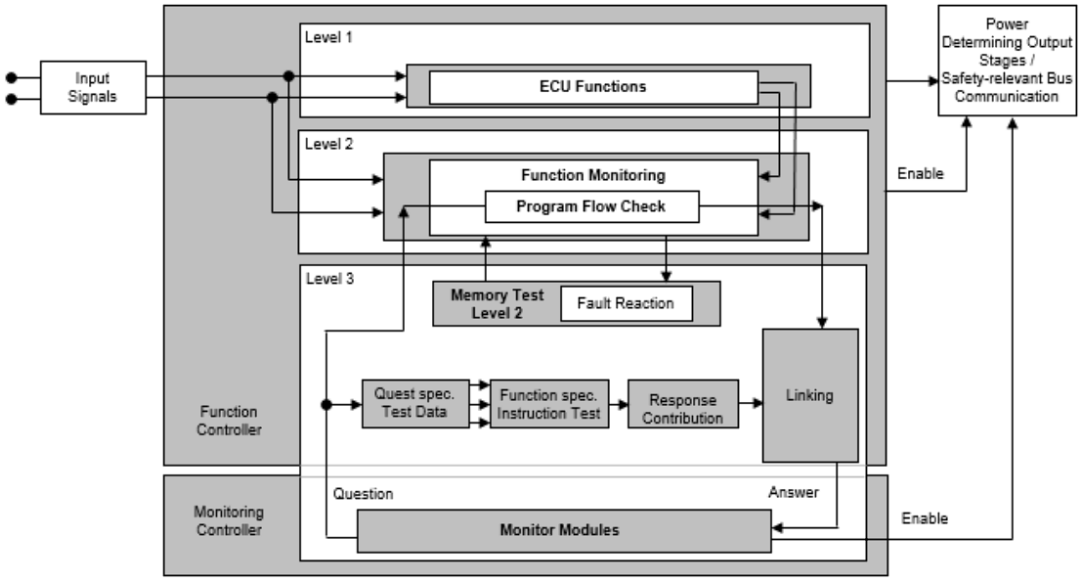

Figure 2.12: Three Level Safety Concept [60]

only be caused by a faulty torque definition/implementation for systems with only one torque source (which is the scope of E-Gas).

The concept (Figure 2.12) is divided into three independently developed levels. These levels exist within two independent controllers, the function controller and the monitoring controller. Levels one and two completely reside on the former whilst level three resides on both. Independent development along with separate hardware reduce the chances of common modes of failure.

2.8.1 Level One - Function Level

The first level, also the most complex, implements almost all of the system functionality within the scope of the E-GAS system. This includes monitoring of components (e.g. throttle valve, brake switch) and signals (e.g. vehicle acceleration, load signal, engine speed signal), fault detection functionality (e.g. redundant pedal value sensor checks), communication, Input/Output (I/O) diagnostics and implementation of the requested engine torque. It also performs output signal regulation. For instance, if,

based on the current driver request via the pedal and the position of the brake, the engine's torque exceeds a maximum controllable limit, level one reduces the output of the engine to a safe limit.

The function level makes available the required information for level two to perform its functionality. This includes all monitored signals and variables. In the case of a detected critical fault, level one can initiate a system reaction that takes the vehicle into a controllable safe state. This safe state may be in the form of a "limp home" mode where certain signals (including the faulty ones) are replaced by set values that are known to be safe.

2.8.2 Level Two - Function Monitoring Level

The second level is responsible for the main monitoring functionality. As the name implies, it monitors the functional software of level one and detects faults in it mainly via validation. Any incoming parameters (e.g. torque requests, spark angle) are first validated via limits. Then, the outputs of the level one software functions (e.g. actual engine-torque, permissible vehicle acceleration) are validated using a comparison against independently calculated (by level two) permissible values. Comparison checks serve two purposes; they ensure that level one works as intended (level one should already be limiting any out of bound requests without interference form level two) and in the case it does not (i.e. level one fails), they ensure that the vehicle still acts within its bounds. Level two is also responsible for the calculations of program flow control and cyclically monitoring its own memory.

2.8.3 Level Three - Controller Monitoring Level

Level three's main purpose is to monitor the monitor i.e. it monitors the functionality of level two. It is responsible for monitoring both the hardware (e.g. controller core, RAM, ROM) in the function controller as well as the software (e.g. functions, algorithms).

For software monitoring, level three is divided into an independent monitoring controller (L3_MM) that is on separate hardware (with its own clock and cyclically tested RAM/ROM) and a software monitoring module (L3_SW) that is on the function controller. L3_MM cyclically asks L3_SW questions and monitors its answers to test correct execution of level two functions. Each set of questions (i.e test path) has an exactly defined answer that is expected from L3_SW. To prevent any disruption of level two processes, L3_SW has access to a copy (stored in separate RAM or ROM areas) of the validation relevant contents of level two or a comparable instruction set sequence for answering questions. L3_SW transmits these answers to L3_MM which in turn determines if they are correct and if a fault reaction needs to be triggered. L3_MM expects a defined answer from L3_SW within an acceptable period of time. In the case of an incorrect answer or an ill-timed one i.e. a fault, L3_MM starts an error counter (say MM_Count) and repeats the questions. MM_Count is then incremented every time L3_MM provides an incorrect answer. It is decremented, at a much slower rate, when a correct answer is provided. If MM_Count reaches its limit, the monitoring controller triggers a fault reaction (usually in the form of limiting or cutting off the actuator power).

The question-answer process, in addition to detecting faults in the function controller, also allows the ability to detect faults in the monitoring controller. L3_SW expects a new question from the L3_MM within an acceptable time period. This ex-

pectation allows us to test the L3_MM for errors. The test is initiated by L3_SW purposely providing incorrect answers at specific time intervals. The MM_Count, whose value is available to L3_SW, is checked and an error is detected if the MM_Count does not increment as expected for the incorrect answer. In the case that it is not incremented, L3_SW starts its own counter (say SW_Count) and retransmits the incorrect answer and increments/decrements its SW_Count as needed. As before, if the counter reaches its limit, a fault reaction is triggered.

Questions are selected at the discretion of the designer. A set of questions makes up a test path. Questions for a specific test path should be numerous (at least 10 questions are recommended) and of enough quality to allow for comprehensive fault detection. These paths can be separated into two types:

- **Program flow paths**: This type of test path verifies the schedule and sequence of all level two modules.
- **Command set paths**: This type of test path looks for errors in the processor core and the execution of level two functions. In the latter case, as mentioned earlier, any validation input data required for the test path is stored in ROM/RAM areas separate from level two. This is to prevent any interference with level two function. This data is allowed to represent faulty level one states to allow for different types of questions.

An example of a test path is the Shutoff Path Test. The test occurs once per drive cycle and checks the shutoff paths to the power determining output stages. It guarantees a safe shutoff in the case of a fault by only allowing an engine start once the test is successful (for every controller).

2.9 Simplex Architecture

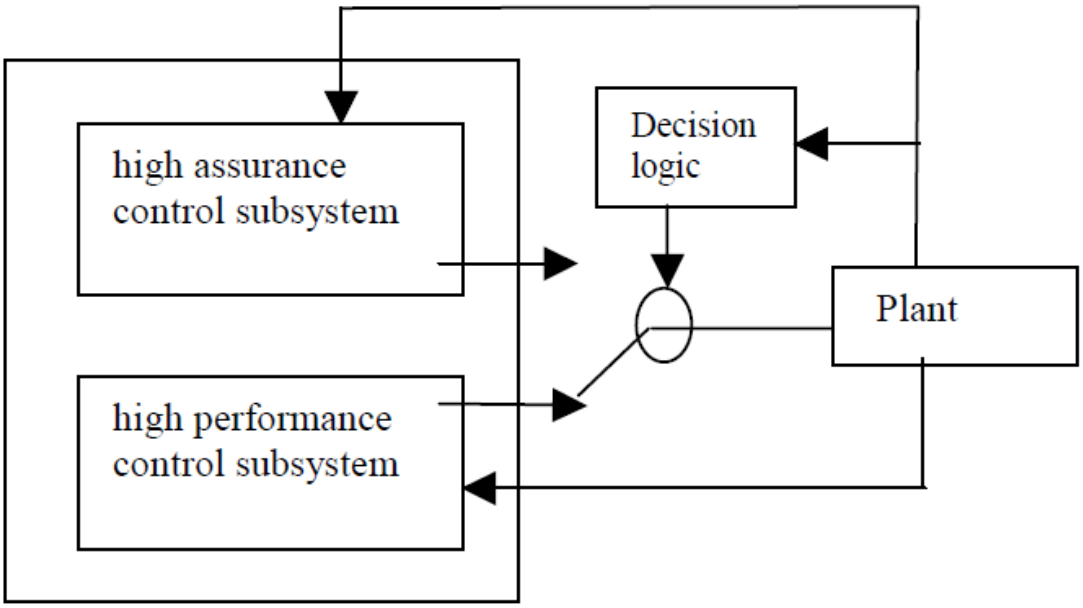

Figure 2.13: The Simplex Architecture [17]

The simplex architecture [17] utilizes the notion of "using simplicity to control complexit" to separate critical properties of a system from its desirable ones. It consists of a high performance controller (Figure 2.13), a high assurance controller and a "switch" controlled by decision logic. The architecture attempts to deal with the unreliability and lack of a safety guarantee that comes with complex systems by using simplicity. The author in [17] states,

> *"The moral of this story is that we can exploit the features and performance of complex software even if we cannot verify them, as long as we can guarantee the critical properties by simple software".*

The high assurance controller consists of well-understood and simple algorithms designed to maximize the system's safety. The high performance controller, on the

other hand, consists of more advanced and complex algorithms that are difficult to verify. Both these components run in parallel and are isolated from each other. Under normal operation, the system is under the control of the high performance controller. The decision logic is tasked with ensuring that control is handed over to the high assurance controller in the presence of faults in the high performance one. This strategy allows a system to provide degraded functionality and remain functional in the presence of failure i.e. fail-operational which is a necessity for safety critical systems such as an AV.

Chapter 3

Accidents Involving Autonomous Vehicles

In this chapter, we shall provide a summary and a discussion of a select number of accidents - both fatal and non-fatal - involving current vehicles with limited autonomous or ADAS functionality. The purpose of this exercise is to highlight some interesting failures that the these systems can exhibit along with their potential causes. When an explicit cause for an accident is unknown, we shall attempt to speculate or make reasonable assumptions. The accidents will be presented in a chronological manner to somewhat exhibit the reactive actions taken (or lack thereof) by some of the involved Automotive OEMs.

These accidents represent the first step of our HA (Chapter 4) as we shall relate the hazards and causes here to the ones obtained in the HA. However, it is important to note that our HA targets SAE Level[1] 4 or 5 AVs whereas all the accidents below involve vehicles that are at most SAE Level 2s.

[1]The term "Level" holds multiple definitions in this thesis. When using it as an AV classification (as done in this chapter), we shall prefix it with "SAE" to separate it from the term's other definitions in Section 2.8 and Chapter 5.

3.1 2016 - January 20th - Fatal - Tesla Model S (China)

Figure 3.1: Moments before the China collision [61]

Short Description

Although unconfirmed at the time[2], this crash [61] resulted in the world's first confirmed fatality involving Tesla's AutoPilot [62]. On the 20th of January, 2016, a Tesla Model S – travelling on the left lane of the G4 Beijing—Hong Kong-–Macau Expressway near the city of Handan, China – crashed into a slow moving (or parked) street sweeper (Figure 3.1) and killed its only occupant (i.e. the driver). An initial investigation by the police concluded that the neither the driver nor the vehicle had attempted any braking or collision avoidance manoeuvres. The vehicle's in-car dash-

[2]Tesla initially stated that the damage to the car made it impossible to transfer log data for confirmation. They later confirmed that AutoPilot was enabled.

board camera footage [63] revealed that the street sweeper was visible early enough such that an attentive driver should have easily been able to react to it (as showcased by the other vehicles moving out of the lane). This hints to the driver being distracted. This theory is further confirmed when we consider the audio of the footage which presents a driver who seems to be singing along to various songs and does not seem to be incapacitated in any form.

Discussion

It is not clear how the vehicle failed. The Model S was equipped with a single forward facing radar, a single forward facing camera and a set of 12 ultrasonic sensors. The camera was used by MobileEye's EyeQ3 computing platform [64, 65, 66] which implemented a Deep Neural Network (DNN) for its object identification and detection. The vehicle was also equipped with Tesla's AEB system which was designed to initiate automatic braking to avoid imminent collisions [67]. The system required agreement between both the camera and the radar before any action was taken. We can surmise that a possible cause for this accident may have been the camera and its ML algorithm failing to detect the object. While it is true that the radar may have also failed, MobileEye's post accident statements seem to support the former scenario. The statements mention that MobileEye's system was not designed to cover all accident scenarios and that Tesla was using it outside of its intended purpose. Regardless of which sensor failed, the agreement strategy would have not triggered the AEB to prevent the accident.

While it is true that the driver, due to his inattention, was deemed responsible for the accident, it is also the responsibility of the system designer to produce a system that is as safe as possible and is marketed such that its limitations are made

very apparent. Despite Tesla's boisterous (yet misleading and potentially dangerous) claims (e.g. "self-driving", "Your Autopilot has arrived"), the Model S was simply not adequate. The vehicle's sensor configuration combined with its AEB policy was optimized for the wrong goal. In an effort to decrease the occurrence of false positives (i.e when the vehicle falsely detects an object and classifies it as a danger when it doesn't exist or exists but is not dangerous [e.g. plastic bag]), they accomplished the opposite for, arguably, a more dangerous case, false negatives (i.e when a vehicle falsely does not detect a dangerous object or detects it but incorrectly classifies it as not dangerous) i.e this accident. Consider a scenario where a dangerous object appears in front of the vehicle; let $P(OD)$ be the probability that a sensor detects and classifies the object correctly. The probability of a false negative for a system with n, independently failing, sensors in series then becomes:

$$P(FN) = 1 - \prod_{i=1}^{n} P(OD)_i$$

As the equation shows, the probability of false negatives **increases** when more sensors are added in series. Hence, for false negatives, the employment of this scheme removes the benefits of redundancy and effectively results in a more dangerous vehicle. Why did Tesla choose this scheme then? We can speculate that the answer to this lies in user experience and acceptance. A ride in a vehicle that triggers frequent false braking events is certainly not enjoyable nor comfortable. Optimizing for comfort helps sell the feature at the unfortunate expense of lulling users into a false sense of safety. However, there may be a positive reason for this scheme as well. A vehicle that offers jerky and erratic braking makes adoption of these (presumably, safety focused) ADAS difficult and may force some to disable them altogether. To prevent this, false braking events might have to be reduced. Regardless, we must consider what the more

preferable scenario is, a passenger in discomfort or one that is exposed to unnecessary harm (or, dead). The Model S could have benefited greatly if more parallel sensors (both homogeneous and heterogeneous), that perhaps allowed the employment of a majority voting strategy, were added. This would have reduced both false positives **and** false negatives. Additionally, better fault detection, diagnosis and monitoring might have detected the failure in the sensors and brought the vehicle to a safe state or given control back to the driver.

Another point to consider is that SAE Level 2 AVs rely heavily on human drivers to take over in situations where the vehicle is unsure of what to do or incapable of performing a task (e.g. performing a braking action). As it will become apparent in most of the accidents discussed in this chapter, the driver's (over)reliance on the AV resulted in an inattentiveness that left them unable to react when required. The result of an SAE Level 2 AV (or, in general, any system that automates a human task) that works great most of the time but fails catastrophically sometimes is the dangerous complacency it gives to its user. This is a strategy for failure and must be dealt with. Driver complacency can somewhat be mitigated with a well designed Human Machine Interface (HMI). The HMI acts as a gateway between the human and the AV. It serves two purposes – informing the driver about the vehicle's status (e.g. current operating mode, engagement and disengagement warnings, etc.) and informing the AV about the driver's status (e.g. drowsiness, attentiveness, alertness, etc.). The HMI must be designed to prevent mode confusion by making it very clear to the driver when autonomous functionality is engaged or disengaged. Additionally, when engaged, the vehicle must also give the driver enough time and sufficient warning to react before a disengagement. Human factors' studies of automation in aerospace have demonstrated that inattentive drivers require time to regain the situational awareness required to respond to failures in the automated systems [68]. Finally, the HMI must

also have some form of a Driver Drowsiness Detection (DDD) system (e.g. Toyota's Driver Monitoring System (DMS)). While autonomous functionality is engaged, the driver **must** be monitored for attentiveness. If the driver is found to be distracted, the vehicle must alert the driver, disengage the autonomous functionality and, potentially, bring the vehicle to a safe state (be that a stop or other alternatives). While Tesla's HMI provides visual and auditory warnings, its shortcomings arise from its inadequately designed (and implemented) DDD combined the lax rules around hands-free operation.

The Tesla DDD monitors driver attentiveness by sensing hands on the steering wheel. With this being the only means of driver monitoring, drivers are allowed to very easily misuse the system while remaining distracted and operating their vehicle hands free for extended periods of time. This results in a strategy that is not sufficient or effective; especially if Tesla's implementation of driver hand monitoring (i.e. a torque sensor) can easily be fooled by the likes of fruits or water bottles (sat on the inner part of the steering wheel). Even if Tesla were to impose more stringent policies around hands free operation and use a different sensing technology (a capacitive sensor would not be as easy to fool), monitoring the driver's hand on the wheel is still not an adequate means of gauging attentiveness as the driver can continue to remain distracted while having their hands on the steering wheel. The NTSB mentions in [69],

> *Because driving is an inherently visual task and a driver may touch the steering wheel without visually assessing the roadway, traffic conditions, or vehicle control system performance, monitoring steering wheel torque provides a poor surrogate means of determining the automated vehicle driver's degree of engagement with the driving task.*

Some automakers such as General Motors (GM) have taken a good step towards a proper DDD by introducing advance driver eye and face tracking systems that make use of in-cabin cameras to monitor driver engagement and attentiveness.

To Tesla's credit, in a September 2016 update, the company reduced hands-free operation time and introduced a strike out system that prevents drivers from re-engaging Autopilot for a period of time if too many warnings are ignored. They also eventually improved their sensor suite with more cameras, radars and ultrasonic sensors that offer more coverage and redundancy. In the author's opinion however, this is still not good enough. The cruxes of the issue – the ineffective DDD, an AEB system in infancy and the misleading marketing – still remain.

3.2 2016 - May 7th - Fatal - Tesla Model S (Florida)

Short Description

On the afternoon of May 7, 2016, a 2015 Tesla Model S with Autopilot engaged, travelling eastbound on a highway in Florida, struck and passed beneath a tractor trailer (Figure 3.2). At the time of the collision, the tractor trailer was making a left turn from the westbound lanes of the highway across the two eastbound travel lanes. After exiting from underneath the tractor trailer, the Tesla left the roadway and collided with a utility pole which broke in half before the car finally came to a stop (Figure 3.3). The Tesla driver died in the crash. It was deemed that the driver was not attentive though a cause for the inattention is unknown.

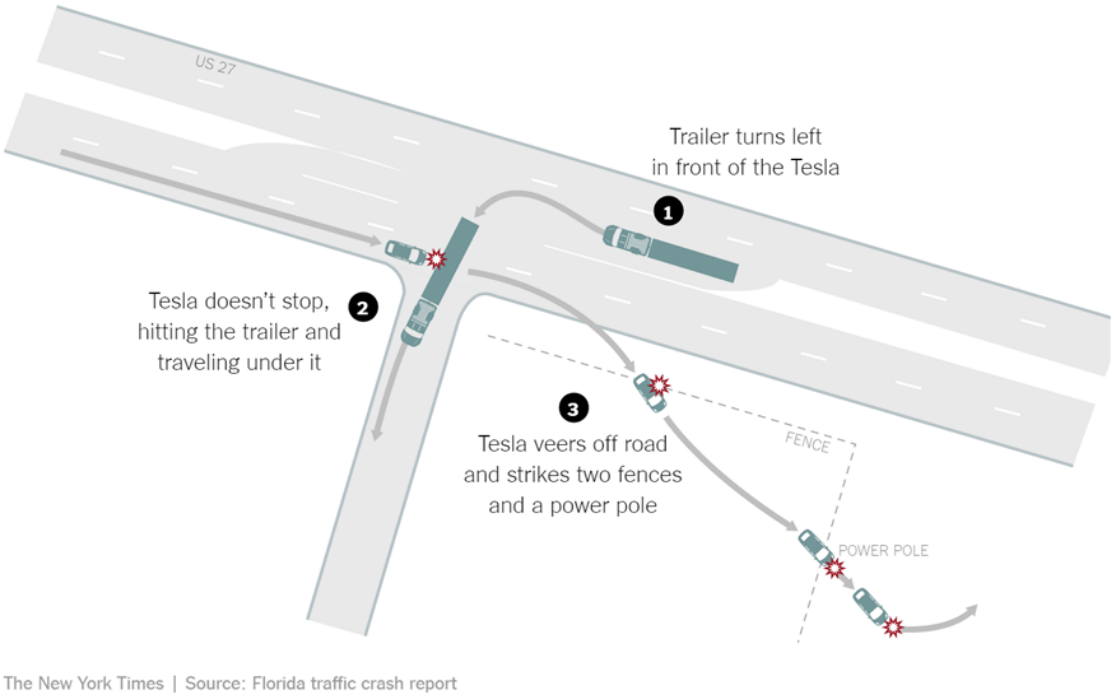

Figure 3.2: The Florida accident explained

Figure 3.3: The aftermath of the accident [70]

Discussion

Both the NTSB and the NHTSA investigated the accident and published their findings in [69] and [67] respectively. The NTSB summarized,

> *The Tesla's automated vehicle control system was not designed to, and did not, identify the truck crossing the car's path or recognize the impending crash; consequently, the Autopilot system did not reduce the car's velocity, the forward collision warning system did not provide an alert, and the automatic emergency braking did not activate.*

The NHTSA offered a similar sentiment,

> *NHTSA's examination did not identify any defects in design or performance of the AEB or Autopilot systems of the subject vehicles nor any incidents in which the systems did not perform as designed. AEB systems used in the automotive industry through MY 2016 are rear-end collision avoidance technologies that are not designed to reliably perform in all crash modes, including crossing path collisions.*

While it is true that the accident may have provided a condition that was out of bounds for the current state of the art in AEB, this does not preclude the system from being better designed. Since this vehicle was the same one as the one in the previous accident, it has the same design flaws mentioned in Section 3.1. Tesla commented that the camera failed to detect the truck due to "white colour against a brightly lit sky" and a "high ride height". They further commented that the radar filtered out the truck as an overhead road sign to prevent false braking. Based on how Tesla's AEB functions, this explains why breaking action was not taken. This accident is also similar to the China accident in the sense that the driver was too reliant on the system

and became complacent. To reiterate what was said in Section 3.1, a more diverse sensor suite (i.e LIDAR) with smarter policies around braking, a more effective DDD and a better fault detection, diagnosis and monitoring system may have prevented the accident.

3.3 2018 - January 22nd - Non-Fatal - Tesla Model S (California)

Figure 3.4: The Model S Fire truck accident – Credit: Culver City Fire Department via Twitter

Short Description

On the 22nd of January, 2018, a Tesla Model S, travelling on a California freeway at approximately 65 mph, crashed into a stopped fire truck (Figure 3.4). It was following another vehicle (a pick-up truck according to some) which swerved out of the lane to

avoid the stopped fire truck The Tesla did not react to the change in the environment and instead **sped up** and crashed into the vehicle. The driver escaped with minor injuries, stated that AutoPilot was engaged and admitted that they were distracted and inattentive to the driving task. The driver also mentioned that they were unable to see ahead of the vehicle in front of them.

Discussion

We speculate that AutoPilot failed to detect the stationary object and, subsequently, failed to bring the vehicle to a safe stop. This highlights a key limitation in Tesla's AutoPilot and perhaps a general limitation of current SAE Level 2 AVs. The Model S' handbook states,

> *Traffic-Aware Cruise Control cannot detect all objects and may not brake/decelerate for stationary vehicles, especially in situations when you are driving over 50 mph (80 km/h) and a vehicle you are following moves out of your driving path and a stationary vehicle or object is in front of you instead.*

This sort of disclaimer is not limited to Tesla. The Volvo equivalent shares similar language,

> *When Pilot Assist follows another vehicle at speeds over approx. 30 km/h (20 mph) and changes target vehicle – from a moving vehicle to a stationary one – Pilot Assist will ignore the stationary vehicle and instead accelerate to the stored speed.*

This seems to be an admission of incorrect requirements. The situation described above, in the author's opinion, is not rare enough that it can be mitigated with

such disclaimers. Automakers should strive to not release products with such known failure modes. Perhaps Tesla's Adaptive Cruise Control (ACC) should mimic the defensive behaviour of a human in situations of uncertainty. A reasonable human driver, when faced with an unknown situation slows down, assess the situation and reacts. When the ACC system sees such a sudden change in environment, instead of immediately speeding up to the stored speed, it should remain at the same speed (or slow down depending on the situation) for a TBD amount of time such the system can properly assess the situation. This solution may, however, not be useful in vehicles that rely on radars that are programmed to avoid stationary objects at high speeds. If a vehicle's ACC system is not equipped to handle stationary objects at high speeds, then giving the vehicle ACC more time to assess the situation may be futile (there are still benefits of not immediately speeding up however). We must then question why radars are programmed in such a manner. The reasons offered by many suggest that stationery objects are purposely ignored to prevent false braking events and that the system would be unable to function otherwise. If indeed this is the case and current technology prevents us from equipping our vehicles to handle scenarios such as this accident, then automotive OEMs should be very careful about making claims about self driving functionality. They should not market their products in a manner that allows their customers to be complacent with SAE Level 2 vehicles pretending to be Level 5s.

To be clear, ADAS systems such as AEB are an excellent and significant benefit to vehicle safety – even in their current infant stage. What the author has issue with is the irresponsible notion of allowing users to be trust their autonomous systems so thoroughly but failing to ensure that they work in most, if not all (easier said than done), conditions. If the ADAS functionality is still in its early stages and requires a human fall-back (as all SAE Level 2s do), then OEMs have to ensure that drivers are

monitored properly to access attentiveness, prevent misuse and reduce over reliance. Part of the solution may also be more sensor fusion with different types of sensors such as LIDAR that result in a more robust sensor suite that may be able to detect stationery objects.

With that being said, a recent IIHS study [71] showed that both Volvo's and Tesla's vehicles are sometimes capable of handling stationery objects. Perhaps then, the cause for this accident may simply have been sensors with a blocked view. It is speculated that the Tesla was behind a pickup truck that blocked the view of the driver. Could the driver being too close to the truck have blocked the view of the sensors? If the follow distance for the ACC allows blocked sensors, then this is a design flaw. If the sensor suite detects that it is unable to perceive the world optimally, the user should be warned and the vehicle should back away. When driving defensively, we typically look at multiple vehicles ahead of us. This should be mimicked by the AV. Another potential solution could come from V2X – the fire truck itself could let the AV know that it is stopped. However, this technology is not really present currently.

3.4 2018 - March 18th - Fatal - Retrofitted Uber Volvo XC90 (Arizona)

Short Description

On March 18th, 2018, a modified Volvo XC90 struck and killed a pedestrian crossing a road at Mill Avenue in Tempe, Arizona. The vehicle was part of the ridesharing/taxi company Uber's Autonomous fleet equipped with a LIDAR unit, forward facing and side facing cameras, radars and Uber's developmental AV software.

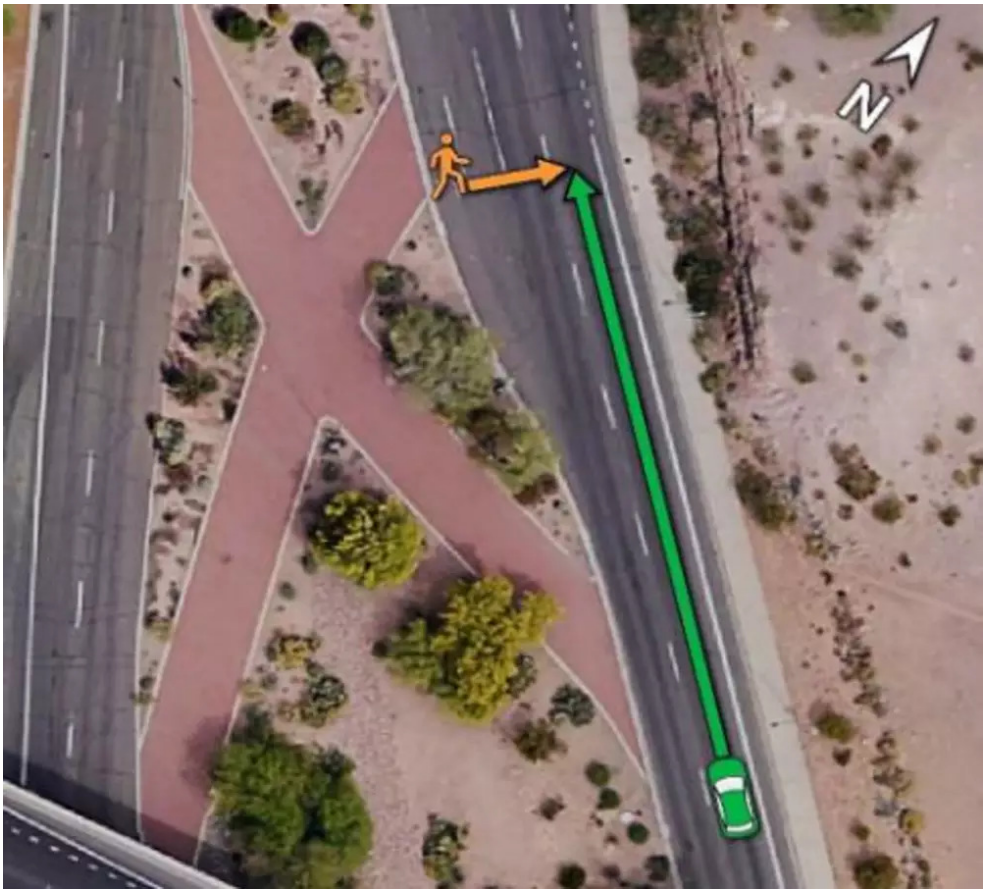

Figure 3.5: NTSB's illustration of the accident [72]

The Volvo was travelling (Figure 3.5) northbound at approximately 69.2 km/h (43 mph) when it collided with the pedestrian who was walking her bike across the road at night.

Discussion

The NTSB's preliminary report [72] indicates that the pedestrian was dressed in dark clothing, crossing a non-illuminated, non-crosswalk region of the road. The investigation states that the Volvo did not brake or attempt to slow down to avoid the collision. The report also reveals that the base vehicle's default safety mechanisms such as AEB were **disabled** by Uber when the developmental AV software software

was active. According to Uber, this was done to reduce the potential for erratic vehicle behaviour. Uber then relied on an attentive back up driver to provide any braking action in the case of an imminent collision. Surprisingly, the company had also **disabled** the driver monitoring system i.e. the DDD. As a result, the final safety mechanism i.e. the distracted (as expected) backup driver was not alerted of the imminent collision. The report further states that the vehicle's sensors detected the pedestrian 6 seconds before the accident; initially detecting her as an unknown object (Figure 3.6), then as a vehicle and finally as a bicycle. 1.3 seconds before impact, the vehicle's software determined that a braking action was required but as AEB was disabled, none was provided.

Figure 3.6: 1.3 seconds before the accident [72] – the yellow lines indicate distance ahead in meters

We can speculate what would have happened if the AEB was not disabled. Figure 3.6 shows that the vehicle was approximately 25 metres away from the victim

at 1.3 seconds before impact. Figure 3.7 shows that, under similar conditions as in the accident, 56 m are required for a vehicle travelling at 70 km/h (the speed of the Volvo) to come to a stop.

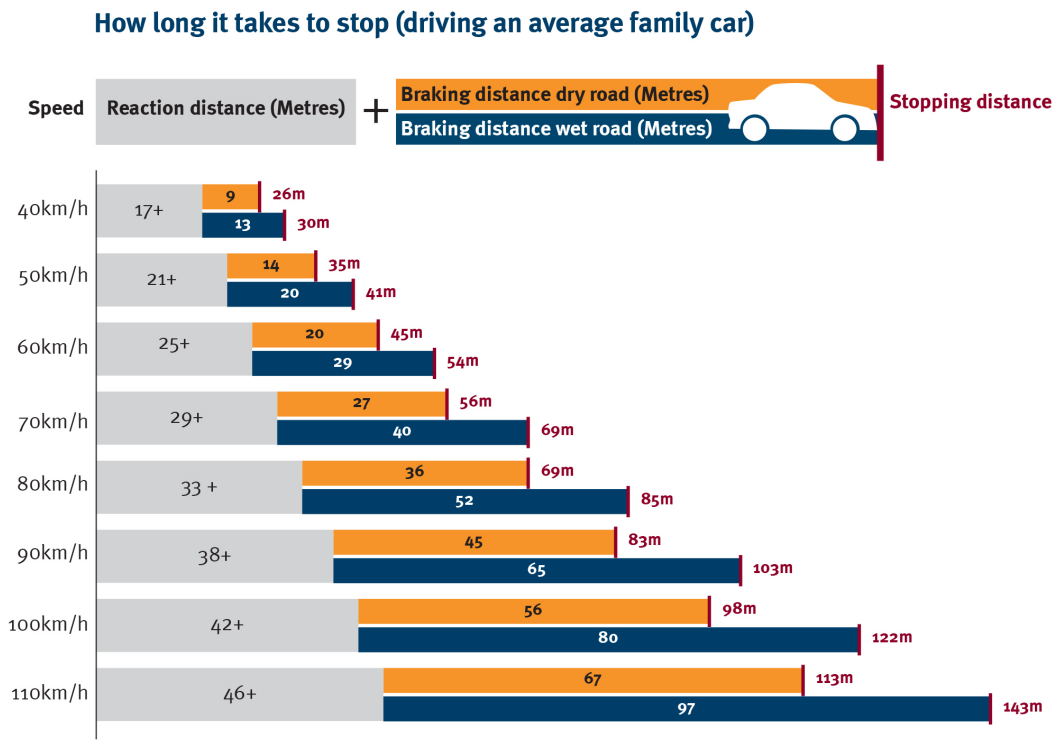

Figure 3.7: Stopping distances for an average human driven vehicle [73]

At this point, had the human driver applied the brakes, the figure implies that the vehicle's speed would approximately been halved. Now consider the AEB which can react significantly faster than a human; say, instead of taking 29m to react, it only takes 5m. Then, the stopping distance becomes 32m. While we can't guarantee that the vehicle would have stopped, we can be confident that AEB would have significantly reduced the vehicle's speed and, subsequently, lessened the impact. It is very likely that the pedestrian would have survived with (perhaps, even minor) injuries.

The lessons here are two fold. Firstly, the emergency braking system should not be disabled. This is an unacceptable compromise, especially if the backup system (i.e.

the driver) is not monitored or alerted. If given no other option, AV experimentation should not be done until a sufficient and competent backup system is implemented. Secondly, it is apparent that Uber failed to account for the human element. As shown throughout this chapter, humans are prone to over-reliance and are easily distracted. The fact that the driver monitoring system was disabled and that no warning was given to the user to take over is both irresponsible and negligent.

A significant piece of the blame falls on Uber. Disabling both the AEB and DDD while providing no other replacements combined with improperly trained drivers shows a lack of a proper safety process. The accident also showcases a failure of legislation. The state of Arizona, in their over willingness, did not properly ensure that the AV experimentation they were allowing was safe; ultimately leading to the death of one of its inhabitants.

3.5 2018 - March 23rd - Fatal - Tesla Model X (California)

Short Description

On the 23rd of March, a Tesla Model X travelling southbound on the US Highway 101 (US-101) in Mountain View, Santa Clara County crashed into a damaged crash attenuator and fatally wounded its driver. As the vehicle was approaching the gore area of an SH-85 exit ramp, it veered out of its previously occupied High Occupancy Vehicle (HOV) lane and into the gore area where it crashed into the crash attenuator at a speed of 71 mph. In the aftermath (Figure 3.8), the driver was transported to a hospital but succumbed to his injuries shortly afterwards. Data from the accident shows that the driver was using the traffic-aware cruise control and autosteer lane-

keeping assistance features collectively referred to as AutoPilot.

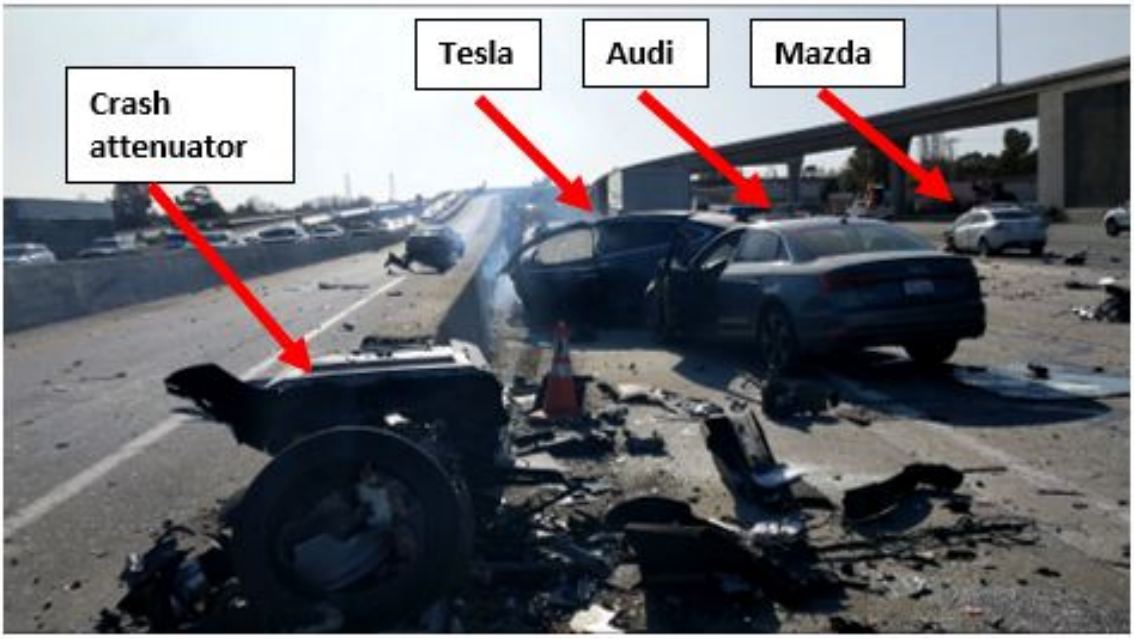

Figure 3.8: The aftermath of the collision [74]

Discussion

The NTSB released a preliminary report [74] which summarizes the events leading up to the accident:

- *During the 60 seconds prior to the crash, the driver's hands were detected on the steering wheel on three separate occasions, for a total of 34 seconds; for the last 6 seconds prior to the crash, the vehicle did not detect the driver's hands on the steering wheel.*
- *At 8 seconds prior to the crash, the Tesla was following a lead vehicle and was traveling about 65 mph.*
- *At 7 seconds prior to the crash, the Tesla began a left steering movement while following a lead vehicle.*

- *At 4 seconds prior to the crash, the Tesla was no longer following a lead vehicle.*
- *At 3 seconds prior to the crash and up to the time of impact with the crash attenuator, the Tesla's speed increased from 62 to 70.8 mph, with no precrash braking or evasive steering movement detected.*

It seems to be clear that the driver was distracted (or for some other reason, unable to react) and this was a case of over reliance on automation (as has been the case in every single accident so far). Tesla stated in [75], "*The driver had about five seconds and 150 meters of unobstructed view of the concrete divider with the crushed crash attenuator, but the vehicle logs show that no action was taken.*"

An interesting fact here is that the driver was aware of this problem beforehand as he had experienced his Tesla veering off of the same lane and into the same divider (he had reported this to Tesla). Other people have also reported similar issues [76] on the same section of the road. This outlines the grim reality of badly designed AVs (not to say that a well designed AV would never exhibit such a problem). If an accident occurs with a conventional vehicle, the vehicles behind the said accident will react and avoid accordingly. But if we have a fleet of AVs with the same faulty software for instance, then, regardless of the accident, all of them might share the same fate.

The NTSB have not concluded their investigation and as such do not have a cause. However, a bird's eye view (Figure 3.9) of the accident location shows an interesting possible cause. ① shows the crash attenuator. ② shows the left lane marker splitting into two.

Figure 3.9: A Google Maps image of the March 23rd accident region

It is not clear in the figure but a more recent frame (Figure 3.10) of the same location [77] shows that the correct side (right side) of the now split lane marker is worn out and the other side (left side) of the split is more solid. We can surmise that this resulted in the vehicle following the more solid of the two lines into the gore area. This combined with it also failing to detect or react to the attenuator is the most likely reason for the collision.

Figure 3.10: A frame of the video showing the faded highway markers [77]

It should be the case that if a dangerous object is detected in front of the vehicle, then regardless of any other directive, a safety short circuit should cause it come to a halt. If detecting stationary objects is out of the realm of the current state of the art (it should be noted that Tesla was not equipped with LIDAR), then this functionality should not be allowed or the hazard at least mitigated. A potential solution could be drivers that are allowed to report hazards [76] to Tesla. This information could then be relayed to all other Tesla vehicles that are on AutoPilot (akin to how Google's Waze reports police camera, accidents, etc.). Finally, the vehicle should have been designed with a proper DDD because, as mentioned previously, all the accidents in this chapter seem to be caused by the vehicle allowing an inattentive operator.

The accident above seems to be similar to another one in California on May 20th, 2018 (Figure 3.11) where a Model S crashed into a parked police vehicle. Fortunately, there were no fatalities. Although the official cause is unknown, a bird's eye view of the road reveals interesting lane markings that may explain the collision.

Figure 3.11: The May 20th Accident [78]

Figure 3.12 shows that before approaching the police vehicle (parked approximately at ①), the lane widens with the right most lane marker leading into the parking area and eventually ending (marked as ②). We speculate that the design may have confused the lane following algorithm causing the vehicle to follow the rightmost lane marker into the parking lane. Alongside improving the lane following algorithm, the vehicle should be designed such that regardless of the lane following algorithm failing, a backup system that detects the stopped object and stops the vehicle should be present.

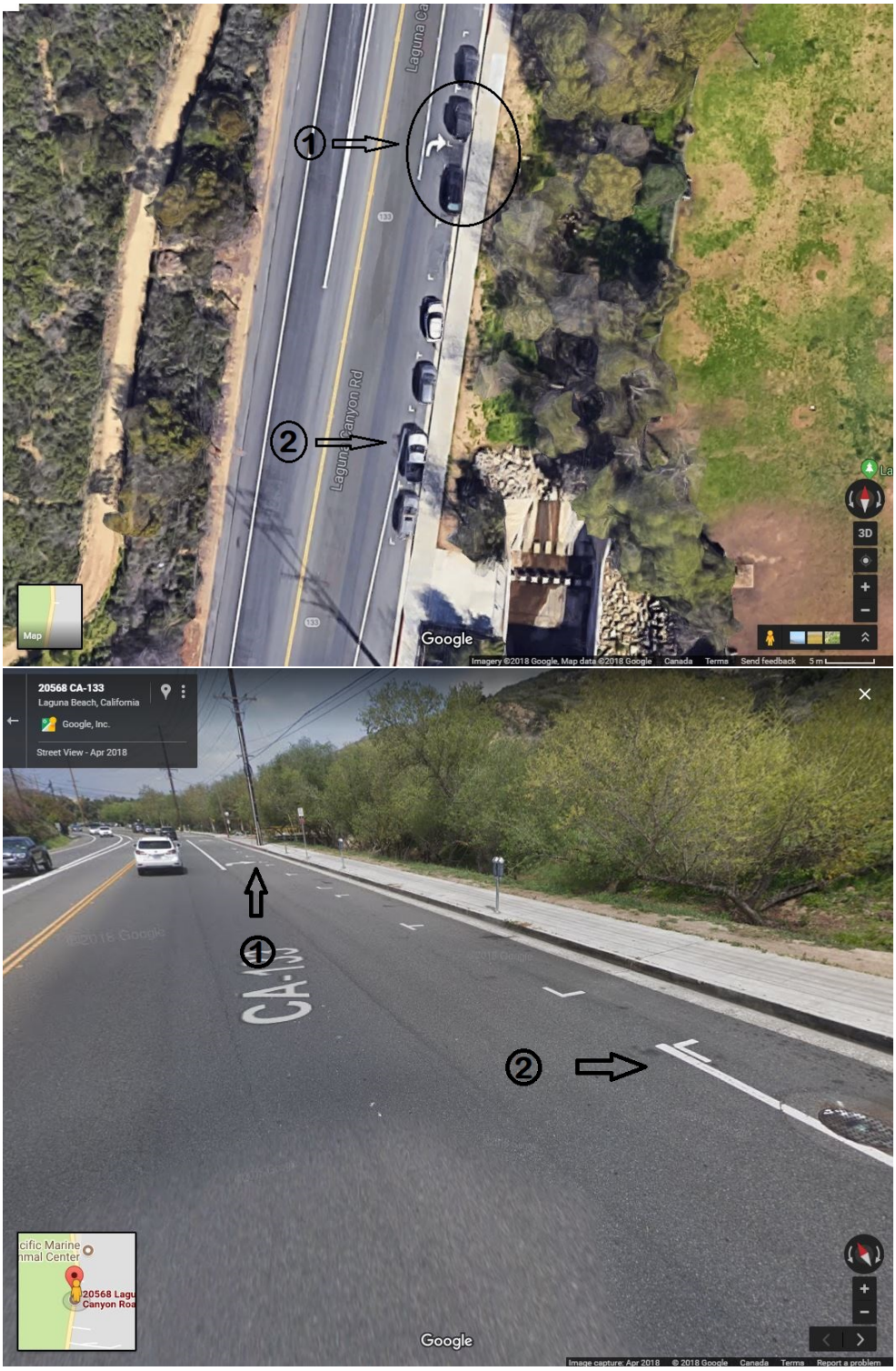

Figure 3.12: Google Maps images of the accident location

3.6 2018 - June - Non-Fatal - Acura MDX (Newfoundland)

Short Description

In early 2018, an owner of a 2016 Acura MDX equipped with Acura's LKAS found that after replacing his vehicle's windshield, the Acura would often attempt to veer off of its lane and sometimes into oncoming traffic [79]. The LKAS system used a front facing camera (highlighted in Figure 3.13) mounted between the windshield and the rear view mirror to detect lane markings. The system was designed to steer the vehicle to the centre of the lane if it detected deviation form the centre line. Unfortunately, when the windshield was replaced, the camera (a crucial component for LKAS) was not recalibrated for the lane keeping functionality. Fortunately, the driver was able to notice this failure before any accidents could occur and remained unharmed.

Figure 3.13: An Acura MDX Windshield

Discussion

Although not as striking as the rest of the entries in this chapter, this incident highlights an important side of AV safety and reveals a potential failure mode. AVs

must be designed to monitor and detect issues with their sensors (or "eyes"). Any autonomous functionality must then take this into account. In this instance, if the camera had performed some form of a self test or self calibration [80, 81], the user could have been warned about the issue and the LKAS system would have, subsequently, never been activated. The vehicle could also benefit from redundant cameras (along with redundant heterogeneous sensors). Apart from increased reliability, the additional cameras would allow easier fault detection, cross-checking and better calibration. If self-calibration is not feasible, a simple workaround would be to implement a system that monitors the windshield and raises a fault code in the event of a displaced windshield. LKAS functionality would then be deactivated until the code is manually cleared after the camera is recalibrated.

Another side to this was user knowledge. The owner in this incident mentioned that he was unaware of the camera or its calibration being crucial to the LKAS system. Although the owner's manual mentions this connection, perhaps the user could be made aware of the camera being used while LKAS is active.

3.7 Conclusion

To conclude, there seems to be a common theme of *permitted* user over reliance in the accidents discussed in this chapter. Some automotive OEMs are responsible of not properly considering the human element and of misrepresenting the capabilities of their SAE Level 2 vehicles. They are responsible of designing their systems to give more weight to avoiding false positives (i.e. falsely detecting an object that does not exist) rather than false negatives (i.e falsely not detecting an object that does exist) in an effort to ensure a good user experience. This, unfortunately, comes at the expense of users gaining a false sense of safety which ultimately costs lives. This is

irresponsibility and negligence on the part of the OEMs. We must realize that AVs are not there yet and that current SAE Level 2 vehicles still need an attentive human driver to take over at all times. To that end, AVs must be designed with proper driver monitoring systems that disallow misuse and only permit autonomous functionality when the driver is not distracted and capable of taking over. Additionally, proper fault detection and monitoring systems must be designed and put in place to allow vehicles to bring themselves to a safe state if optimal autonomous operation cannot be achieved. We hope to contribute to the last point in the in the form of Safe-AV.

Legislation and government also played a role in the accidents. The rigour of safety placed into a production vehicle is correlated to the rigour of laws that govern them. If the government allows lax regulation, OEMs will develop vehicles with lower standards. Governments should not be too eager to introduce AVs onto their roads without proper certification or analysis of the vehicles' safety. Another aspect to consider are industry standards. As the technology is new, these do not exist or are still in their infancy. This leaves OEMs without proper requirements, processes and best practices when developing AVs.

The accidents highlighted in this chapter along with their causes act as a first step for our HA in the next chapter. The safety goals and requirements derived from the hazards, unsafe control actions and causal factors in the HA will then drive the design of Safe-AV.

Chapter 4

Hazard Analysis

This chapter presents our Hazard Analysis for autonomous driving. We start by defining an initial architecture consisting of components that should typically be present in an AV. Then, we determine the system level accidents and hazards. We then apply the STPA hazard analysis technique on the defined system to generate a set of UCAs, their causal factors and finally, safety requirements to mitigate them. For the purposes of this thesis, we shall limit our HA to the highway driving scenario. Although Safe-AV is intended to cover more than just the highway driving scenario, the safety requirements in this chapter will still provide a substantial direction in the architecture's design. In future work, we shall expand the HA to include city driving and other scenarios.

4.1 System Description and Boundary

Taking inspiration from the AV architectures discussed in Chapter 2, we have the following modules:

4.1.1 Sensing

The Sensing module is responsible for interfacing with the various sensors on the vehicle, acquiring data and making it available for the rest of the system. This includes pre-processing and filtering as necessary.

4.1.2 Modelling The World

Modelling The World (MTW) is functionally responsible for Object Detection, Object Tracking, State Estimation and State Prediction. It takes the information made available to it from the Sensing module and forms a model of the world using machine learning algorithms and sensor fusion. The model contains information about, amongst other things, the ego vehicle (e.g. position relative to objects, position relative to a global map, ego vehicle acceleration, ego vehicle speed, etc.) and the environment (e.g. other vehicle positions, other vehicle predicted trajectories, other vehicle speeds and accelerations, other vehicle distances, environment object classification, environment object distance, environment object speeds, road signs, lane markings, etc.).

4.1.3 Planning

Planning acts as the decision maker for the system. At a high level, it is responsible for finding an optimal route based on the inputs provided to it by the user, the current state of the world, any reference map or GPS information and the performance constraints placed upon it by the drive profile/energy requirements. At a lower level, it is responsible for selecting the appropriate vehicle behaviour (e.g. lane change, merge, stop, lane follow) based on the current state of the vehicle, the environment (e.g. other vehicles, pedestrians, road conditions, traffic signals), the planned path

and the rules of the road. This includes reacting to changes in the environment such as a pedestrian suddenly crossing the road or a stopped fire truck. Ultimately, the module outputs a trajectory consisting of the desired position, desired speed, desired acceleration, etc. over time.

4.1.4 Autonomous Control

After a trajectory is generated by the Planing module, Autonomous Control (AC) converts it to inputs for the X-By-Wire (XBW) systems (i.e throttle value, braking value, steering value). It is essentially responsible for executing the generated trajectory and correcting tracking errors. If we take the analogy of a human driver, if Planning is considered the brain, AC can be considered as the driver's body converting the driver's intention to actions for driving.

4.1.5 Vehicle Actuation

Vehicle Actuation (VA) consists of two parts, the XBW actuators and the rest of the vehicle. The former refers to the actuation components of an XBW system (Section 2.7). This consists of the XBW ECU and the electromechanical component it controls (e.g. throttle valve). The latter includes the motors, steering assembly, brake assembly etc. VA deals with the non-autonomous safety features such as stability control, traction control, and anti-lock braking.

4.1.6 System Boundary

It is necessary to define a system boundary before commencing STPA. We can only effect change in systems that we have control over. In this instance, the system boundary is simply the vehicle itself.

4.2 System Level Accidents, Hazards and System Level Constraints

We begin by listing the system level accidents[1] (Table 4.1). While the official definition of Accident (Section 2.2) does include scenarios such as *property damage, environmental pollution, mission loss, financial loss*, we focused on harm to humans as reducing it is the main objective of this work. With respect to the accidents listed, A1 and A2 are self-explanatory. Obvious scenarios that fall under A3 include occupant injuries such as whiplash when the vehicle experiences excessive acceleration (both longitudinal and lateral). Less obvious scenarios include injuries that are an indirect result of erratic or non-conforming vehicle behaviour (e.g. a pedestrian or cyclist is injured when performing an avoidance action after assuming the erratic vehicle is a danger to them).

Ref	System Level Accident
A1	Collision with other vehicles resulting in injury or death
A2	Collision with non-vehicular objects in the environment resulting in injury or death
A3	Injury or death that are not a result of a collision

Table 4.1: System Level Accidents

The corresponding system level hazards are shown in Table 4.2. For hazards H1, H2, H3, H4 and H5, we initially considered using the term "unintended" instead of "incorrect" – keeping in mind its prevalence in the automotive industry. We went with the latter, however, as the downside of the former is that it leaves out scenarios that may be intended and still have the potential to cause harm (e.g. an AV starts

[1]The latest version of STPA [50], which came out after our hazard analysis had already commenced, replaces accidents with losses. The difference in definition between the two is described in Section 2.2. As the change did not affect our existing analysis, we remained with the term "Accidents"

accelerating to an ACC set speed instead of braking before a collision). Indication in H4 includes turn signals, brake lamps and any form of identification light the vehicle provides to inform others of its intention and state (e.g. headlamps, tail lamps, side marker lights). The derived system level constraints for each of the hazards are shown in Table 4.3.

We initially considered three additional hazards that were subsequently removed: RH1 – Vehicle operates in hazardous conditions (e.g. black ice, heavy rain, etc.) [A1, A2, A3], RH2 – Vehicle enters or is in a hazardous zone (e.g. river, desert, flooded roadway, off a cliff, etc.) [A1, A2, A3] and RH3 – Vehicle occupants exposed to health hazards (e.g. fire, whiplash, excessive heat, etc.) [A3]. Their removal from our HA is not an indication of them not causing harm. We deemed that the scenarios in RH1 and RH2 were covered by the remaining hazards presented in Table 4.2. In RH3's case, some instances of the hazard (e.g. whiplash) are covered by the remaining hazards while others are out of the scope of our work (e.g. excessive heat or fire).

Ref	**System Level Hazard**	**Accidents**
H1	Vehicle performs incorrect (uncontrolled, ill-timed, excessive, insufficient) acceleration	A1, A2, A3
H2	Vehicle performs incorrect (uncontrolled, ill-timed, excessive, insufficient) braking	A1, A2, A3
H3	Vehicle performs incorrect (uncontrolled, ill-timed, excessive, insufficient) steering	A1, A2, A3
H4	Vehicle performs incorrect (uncontrolled, ill-timed, excessive, insufficient) indication	A1, A2, A3
H5	Vehicle performs incorrect (uncontrolled, ill-timed) gear selection	A1, A2, A3
H6	Vehicle maintains an inadequate safe distance from vehicles/objects around it	A1, A2, A3

Table 4.2: System Level Hazards

Ref	System Level Constraint	Hazard
SC1	Vehicle must perform correct (controlled, properly-timed, sufficient) acceleration	H1
SC2	Vehicle must perform correct (controlled, properly-timed, sufficient) braking	H2
SC3	Vehicle must perform correct (controlled, properly-timed, sufficient) steering	H3
SC4	Vehicle must perform correct (controlled, properly-timed, sufficient) indication	H4
SC5	Vehicle must perform correct (controlled, properly-timed, sufficient) gear selection	H5
SC6	Vehicle must maintain an adequate safe distance from vehicles/objects around it	H6

Table 4.3: System Level Constraints

4.3 Control Structure

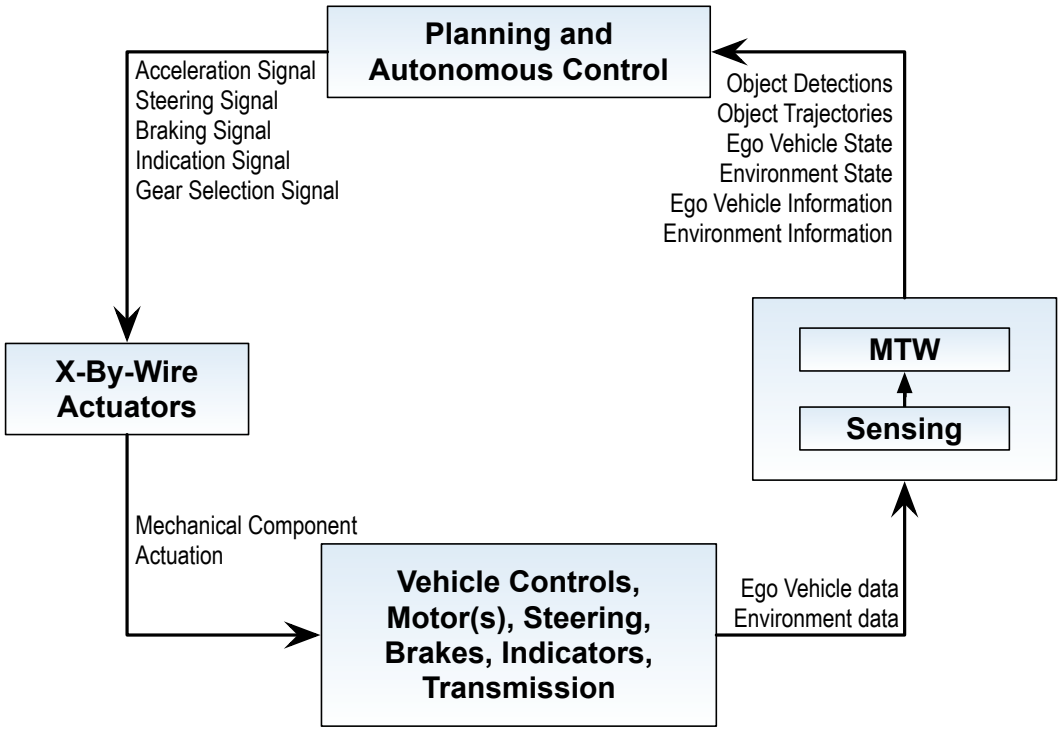

Figure 4.1: Control Structure

Figure 4.1 shows the control structure with Table 4.4 showing the Process Model Variables (PMVs). We consider the Sensing module and MTW to be the "Sensor". The "Controller" is the Planning and AC modules combined i.e. Planning and Autonomous Control (PAC). We decomposed Vehicle Actuation with the XBW actuators becoming the "Actuator" and the rest of the vehicle becoming the "Controlled Process"

Entity	**Variables**
Ego Vehicle	01. Acceleration 02. Speed 03. Position 04. Current Gear 05. Steering Angle 06. Braking Value 07. Indication Value
Environment	01. Object Accelerations 02. Object Velocities 03. Object Positions 04. Object Detections 05. Object Trajectories 06. Road Conditions 07. Road Topology 08. Lane Markers 09. Road Structures

Table 4.4: Process Model Variables

4.4 Unsafe Control Actions and Constraints

This section presents the UCAs for each of the control signals shown in Figure 4.1. Table 4.5 outlines some of the interesting UCAs that were partially formed based on the accidents discussed in Chapter 3. When we began this HA, we found more UCAs then presented here. However, during its course, we found that the additional UCAs did not result in more causal factors or safety requirements. As such, they

were removed (some of the them can be found in Appendix B). As an exercise, we also produced UCAs that were relevant to the city driving scenario. These can also be found in Appendix B. The remainder of the section will outline the UCAs and the corresponding controller constraints.

Accident	**Summary**	**UCAs**
2016 Jan 20th, Section 3.1	Vehicle failed to avoid collision with stopped or slow moving street sweeper	UCA-B1, UCA-A5, UCA-S1
2016 May 7th, Section 3.2	Vehicle failed to avoid collision with a truck crossing its path	UCA-B1, UCA-A5, UCA-S1
2018 Jan 22th, Section 3.3	Vehicle sped up and collided with a stopped fire truck after the vehicle ahead switched lanes	UCA-B1, UCA-B4, UCA-A5, UCA-S1
2018 March 18th, Section 3.4	Vehicle failed to avoid collision with crossing pedestrian	UCA-B1, UCA-S1
2018 March 23th, Section 3.5	Vehicle steered into and collided with a crash attenuator	UCA-B1, UCA-A5, UCA-S1, UCA-S2, UCA-S3
2018 May 20th, Section 3.5	Vehicle steered into and collided with a parked police vehicle	UCA-B1, UCA-A5, UCA-S1, UCA-S2, UCA-S3
2018 June, Section 3.6	Vehicle's LKAS attempted to steer vehicle into oncoming lane	UCA-S2, UCA-S3

Table 4.5: UCAs Mapped to Chapter 3 Accidents

Not Providing Causes Hazard	**Providing Causes Hazard**	**Wrong Timing/Order Causes Hazard**	**Stopping Too Soon/Applying Too Long Causes Hazard**
UCA-B1 Braking signal not provided to avoid collision [H1, H2, H6] ***UCA-B2*** Braking signal not provided as needed when merging or changing into slow lane [H1, H2, H6]	***UCA-B3*** Braking signal provided unexpectedly when vehicle is on a busy lane or merging or changing into a busy lane [H1, H2, H6]	***UCA-B4*** Braking signal provided too late to avoid collision [H1, H2, H6]	***UCA-B5*** Braking signal stopped before coming to a full stop to avoid collision [H1, H2, H6] ***UCA-B6*** Continued braking signal provided after reaching intended speed when slowing down [H1, H2, H6]

Table 4.6: UCA - Braking Signal

Ref	**Controller Constraints**	**UCA**
C-B1	Braking signal must start TBD seconds before detected imminent collision and must continuously be provided as needed until collision is avoided or vehicle comes to a stop	UCA-B1, UCA-B4, UCA-B5
C-B2	Braking signal must start TBD seconds before merge/change into a slow lane and must continuously be provided as needed until merge/lane change is complete	UCA-B2
C-B3	Braking signal must not be provided unexpectedly during vehicle operation	UCA-B3
C-B4	Braking signal must not be provided when vehicle has reached intended speed when slowing down	UCA-B6

Table 4.7: Controller Constraints - Braking Signal

Not Providing Causes Hazard	**Providing Causes Hazard**	**Wrong Timing/Order Causes Hazard**	**Stopping Too Soon/Applying Too Long Causes Hazard**
UCA-I1 Turn signal not provided when merging or changing lane [H4, H6] ***UCA-I2*** Identification light (headlamps, tail lamps, side marker lights) signal not provided during dark or low visibility conditions [H4, H6] ***UCA-I3*** Braking light signal not provided when vehicle is braking [H4, H6]	***UCA-I4*** Left (Right) turn signal provided when merging or changing to the Right (Left) [H4, H6]	***UCA-I5*** Turn signal provided too late when merging or changing lane [H4, H6]	***UCA-I6*** Turn signal not turned off after merge or lane change complete [H4, H6] ***UCA-I7*** Turn signal turned off before merge or lane change complete [H4, H6]

Table 4.8: UCA - Indication Signal

Ref	**Controller Constraints**	**UCA**
C-I1	The correct side's turn signal must be provided TBD seconds before merging/changing lane and must be turned off TBD seconds after merge/lane change is complete	UCA-I1, UCA-I4, UCA-I5, UCA-I6, UCA-I7
C-I2	The braking light signal must only be provided when the vehicle is braking and must be turned off when vehicle is not braking	UCA-I3
C-I3	Identification light (headlamps, tail lamps, side marker lights) signal must be provided during dark or low visibility conditions or as required per government regulations	UCA-I2

Table 4.9: Controller Constraints - Indication Signal

Not Providing Causes Hazard	**Providing Causes Hazard**	**Wrong Timing/Order Causes Hazard**	**Stopping Too Soon/Applying Too Long Causes Hazard**
UCA-A1 Acceleration signal not provided to avoid collision [H1, H6] ***UCA-A2*** Acceleration signal not provided as needed when changing or merging into fast lane [H1, H6] ***UCA-A3*** Acceleration signal not provided to obtain desired speed [H1, H6] ***UCA-A4*** Acceleration signal not provided to leave hazardous zone (flooded roadway, desert) [H1]	***UCA-A5*** Acceleration signal provided when there is a stopped vehicle or object ahead [H1, H6] ***UCA-A6*** Acceleration signal provided when violating minimum safe distance from other vehicle/object [H1, H6]	***UCA-A7*** Acceleration signal provided too late to avoid collision [H1, H6]	***UCA-A8*** Acceleration signal stopped before collision avoidance manoeuvre is complete [H1, H6] ***UCA-A9*** Continued acceleration signal after reaching intended speed with vehicles/objects ahead [H1, H6]

Table 4.10: UCA - Acceleration Signal

Ref	Controller Constraints	UCA
C-A1	Acceleration signal must start TBD seconds before detected imminent collision or hazardous zone and must continuously be provided as needed until collision is avoided or vehicle is safe from hazard	UCA-A1, UCA-A4, UCA-A7, UCA-A8
C-A2	Acceleration signal must start TBD seconds before merge/change into a fast lane and must continuously be provided as needed until merge/lane change is complete	UCA-A2
C-A3	Acceleration signal must not be provided when there is a stopped vehicle or object ahead and/or when minimum safe distance is violated	UCA-B5, UCA-B6
C-A4	Acceleration signal must only be provided until vehicle has reached intended speed when speeding up	UCA-A3, UCA-A9

Table 4.11: Controller Constraints - Acceleration Signal

Ref	Controller Constraints	UCA
C-S1	Steering signal must start TBD seconds before detected imminent collision and must continuously be provided as needed until collision is avoided or vehicle is safe from hazard	UCA-S1, UCA-S6
C-S2	Steering signal must start TBD seconds before merge/change into lane and must continuously be provided as needed until merge/lane change is complete	UCA-S2, UCA-S4, UCA-S5, UCA-S7, UCA-S8, UCA-S9
C-S3	Steering signal value must be maintained such that vehicle continues going straight when intended to go straight	UCA-S3

Table 4.12: Controller Constraints - Steering Signal

Not Providing Causes Hazard	**Providing Causes Hazard**	**Wrong Timing/Order Causes Hazard**	**Stopping Too Soon/Applying Too Long Causes Hazard**
UCA-S1 Steering signal not provided to avoid collision [H3, H6] ***UCA-S2*** Steering signal not provided as needed when merging or changing lane or to follow road curve [H3, H6]	***UCA-S3*** Steering signal provided when attempting to go straight [H3, H6]	***UCA-S4*** Steering signal provided too early when merging or changing lane [H3, H6] ***UCA-S5*** Steering signal provided too late when merging or changing lane [H3, H6] ***UCA-S6*** Steering signal provided too late to avoid collision [H3, H6] ***UCA-S7*** Steering signal provided before acceleration signal when merging ahead of a slow vehicle [H1, H3, H6]	***UCA-S8*** Steering signal stopped before curve following or merging or lane change is complete [H3, H6] ***UCA-S9*** Continued steering signal provided after curve following or merging or lane change is complete [H3, H6]

Table 4.13: UCA - Steering Signal

Not Providing Causes Hazard	Providing Causes Hazard	Wrong Timing/Order Causes Hazard	Stopping Too Soon/Applying Too Long Causes Hazard
	UCA-GS1 Gear selection signal set to reverse or park when vehicle is moving forward [H1, H2, H5, H6] ***UCA-GS2*** Gear selection signal set to forward or reverse when car is intended to be parked [H1, H2, H5, H6]		

Table 4.14: UCA - Gear Selection Signal

Ref	Controller Constraints	UCA
C-GS1	Gear selection signal must be set to forward when vehicle is moving forward	UCA-GS1
C-GS2	Gear selection signal must be set to reverse when vehicle is moving in reverse	UCA-GS2

Table 4.15: Controller Constraints - Acceleration Selection Signal

4.5 Causal Factors and Safety Requirements

Below we shall provide causal factors for some of the UCAs in Section 4.4 (this is done for the sake of brevity; the casual factors for the rest of the UCAs can be found in Appendix A). During the course of the HA, we found that most UCAs, at this level of abstraction, shared similar causes. For the sake of brevity, when a causal factor is shared, we shall only reference the corresponding safety requirements in the first

instance of said causal factor.

The reader will notice that we have discussed the XBW systems at a high level and have generally refrained from providing XBW specific scenarios, causal factors and safety requirements. This is because XBW systems are already present in modern vehicles (in some form or another) and we assume that the safety concepts and mechanisms around ensuring their safety are already implemented and do not conflict with our own. We made this assumption simply to reduce the scope of this research, not because we thought it would be true in practice.

UCA-B1: Braking signal not provided to avoid collision [H1, H2, H6]		
Controller Sends an Incorrect Command		
Scenario	**Causal Factors**	**Safety Requirements**
PAC is not aware of the imminent collision	**CF-1** MTW incorrectly classifies object or environment **CF-2** MTW does not detect hazardous object or environment **CF-3** MTW detects objects but discards it i.e. false negative **CF-4** MTW reports incorrect environment object acceleration, speed or position	**SR-1** MTW must cross check/validate outgoing environment detections (e.g. if detected lane markers match a road map) **SR-2** MTW must cross check/validate incoming data (e.g. if a radar detection matches a LIDAR detection) **SR-3** MTW must report its confidence on each applicable output **SR-4** MTW must monitor its internal functions and report failures, faults or errors **SR-5** Where feasible, Sensing must use redundant (both homogeneous and heterogeneous) sensors **SR-26** Where feasible, MTW and PAC must make use of multiple sources of information when performing their functionality (e.g. MTW, to detect a lane, uses both a camera and a high definition map) **SR-29**: Sensing must report its failures, faults or errors **SR-31**: MTW must take into account error/failure information from Sensing when performing its functionality (e.g. if Sensing reports that a Camera's position has changed, MTW has less confidence on its function) **SR-6** PAC must validate incoming data (e.g. if reported ego vehicle speed shows an impossible change) **SR-7** PAC must monitor its internal functions and report failures, faults or errors **SR-8** PAC must take into account fault info and the confidence of the incoming data when performing its function **SR-9** PAC must validate outgoing values (e.g. if the braking actuation command is within limits) **SR-10** PAC and MTW must provide, where feasible, degraded functionality in the presence of critical failures **SR-11** A backup safety mechanism that makes uses of redundant sensors and processing must exist. This system must bring the vehicle to a safe state if the main control loop fails to keep the vehicle safe

	CF-5 Sensors blocked, broken or tampered with **CF-6** Sensors positioned or calibrated incorrectly	**SR-12** MTW and Sensing must perform relevant sensor diagnostics checks (e.g. calibration check, position check, coverage check, cross checks) at startup and TBD intervals during operation
	CF-7: Delayed, disrupted, missing or not in order communication between MTW and PAC **CF-8** Delayed, disrupted, missing or not in order communication between MTW and Vehicle	**SR-13** There must be redundant communication lines for critical data **SR-14** Communication lines must be protected from noise (high Signal-to-noise Ratio (SNR)) **SR-15** Communication protocols that support a TBD bandwidth and a TBD Baud rate must be used. The protocols shall support fault detection (e.g. Cyclic Redundancy Check (CRC), Message Counter (MC) check), fault tolerance and recovery, arbitration, error reporting **SR-16** Critical data must use separate communication paths from non critical data (e.g. multimedia data is separated from braking signal value)
	CF-9 Corrupted memory or bad processing (e.g. bad scaling, offset, translation, etc.) **CF-39**: Power supply failure or random hardware failures	**SR-17** Diagnostic checks (memory checks, CPU checks, etc.) must be performed at startup and at TBD intervals or based on TBD conditions **SR-27**: System must have sufficient hardware backups to remain operational in the presence of hardware failures **SR-28**: PAC, MTW and their internals must be sufficiently isolated, both in software and hardware to prevent single points of failure and common failure modes **SR-6**, **SR-7**
PAC assumes vehicle is already braking or already stopped	**CF-10** MTW incorrectly reports ego vehicle speed (in this instance, zero) **CF-11** MTW incorrectly reports ego vehicle acceleration **CF-12** MTW incorrectly reports a braking force on ego vehicle **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**	**SR-18** MTW must cross check/validate ego vehicle properties (e.g. if ego vehicle speed is within limits)

PAC assumes that vehicle is unable to brake	**CF-13** MTW incorrectly reports road conditions (e.g. slippery/icy) that prevent braking or acceleration **CF-14** MTW incorrectly reports an object/vehicle near ego vehicle that is not respecting safe braking distance **CF-15** MTW incorrectly reports that ego vehicle's tires are physically compromised (blowout or tread separation) **CF-16** MTW incorrectly reports that ego vehicle's braking components have failed **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-10**, **CF-11**, **CF-12**	
Incorrect Execution		
Brake-By-Wire (BBW) generates an incorrect value or does not generate one	**CF-17**: XBW system incorrectly interprets the signal from PAC **CF-18**: Delayed, disrupted, missing or not in order communication between PAC and XBW	**SR-19** XBW system must validate its output commands (e.g. if the actuation signals are within limits) **SR-20** XBW system must monitor its functions and health and report faults to PAC **SR-13**, **SR-14**, **SR-15**, **SR-16**
BBW provides the correct value but ego vehicle does not brake	**CF-19**: Physical failure in the braking system (e.g. bad calipers) **CF-20**: Ego vehicle tires slip **CF-21**: Delayed, disrupted, missing or not in order communication between XBW and Vehicle	**SR-21** XBW system must validate incoming data (e.g. if the actuation requests from PAC are within limits) **SR-13**, **SR-14**, **SR-15**, **SR-16**, **SR-19**, **SR-20**

Table 4.16: UCA-B1

UCA-B2: Braking signal not provided as needed when merging or changing into slow lane [H1, H2, H6]		
Controller Sends an Incorrect Command		
Scenario	**Causal Factors**	**Safety Requirements**
PAC assumes that there isn't a slower vehicle in the desired lane	**CF-1**, **CF-2**, **CF-2**, **CF-4**, **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-28**, **CF-32**	
PAC assumes that ego vehicle is already braking or already at correct speed for the merge/lane change	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-10**, **CF-11**, **CF-12**, **CF-32**	
PAC assumes that ego vehicle is already in the desired lane	**CF-22**: MTW incorrectly reports the ego vehicle's position **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-28**, **CF-32**	
PAC assumes that ego vehicle's is unable to brake	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-15**, **CF-16**	
Incorrect Execution		
BBW generates an incorrect value or does not generate one	**CF-17**, **CF-18**	
BBW provides the correct value but ego vehicle does not brake or slow down	**CF-19**, **CF-20**, **CF-21**	

Table 4.17: UCA-B2

UCA-B3: Braking signal provided unexpectedly when vehicle is on a busy lane or merging or changing into a busy lane [H1, H2, H6]		
Controller Sends an Incorrect Command		
Scenario	**Causal Factors**	**Safety Requirements**
PAC assumes that a collision is imminent	**CF-23** MTW incorrectly detects an object that isn't there i.e. false positive **CF-1**, **CF-4**, **CF-5**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-14**	
PAC assumes that ego vehicle's speed is too high	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-10**, **CF-25**, **CF-32**	
PAC assumes that ego vehicle's is entering a hazardous zone	**CF-1**, **CF-4**, **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-22**, **CF-23**	
Incorrect Execution		
BBW generates an incorrect value	**CF-17**, **CF-18**, **CF-21**	

Table 4.18: UCA-B3

UCA-B4: Braking signal provided too late to avoid collision [H1, H2, H6]		
Controller Sends an Incorrect Command		
Scenario	**Causal Factors**	**Safety Requirements**
PAC does not react to object in time	**CF-24** Critical tasks are overrun, missing deadlines or there is a priority inversion **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**	**SR-22** A scheduling algorithm that protects against priority inversion and provides a timing guarantee for critical tasks must be used **SR-23** System software must not have more than TBD utilization **SR-24** A High ASIL Real-time Operating System (RTOS) must be used
	CF-40: MTW does not detect object in time or does not identify object as dangerous in time	**SR-30**: PAC must command the vehicle to start slowing down if an object in its path remains unidentified
Refer to **UCA-B1**		
Incorrect Execution		
Refer to **UCA-B1**		

Table 4.19: UCA-B4

Chapter 5

Safe-AV

This chapter provides a detailed description of the Safe-AV architecture. To reiterate what was said in earlier chapters, Safe-AV is a fault-tolerant safety architecture for autonomous vehicles. It incorporates the ideas of fail-operational behaviour, graceful degradation, separation of control and safety and simplicity of safety systems to improve the safety and reliability of AVs. The architecture is a consequence of the safety requirements obtained in Chapter 4 and commonly adopted industry patterns and architectures – specifically, the E-Gas Monitoring Concept (Section 2.8) and the Simplex Architecture (Section 2.9). In its current state, its scope is limited to SAE Level 4 and 5 vehicles. Throughout the chapter, whenever an element of the architecture is presented, we shall attempt to reference all the safety requirements (obtained in Chapter 4) that were relevant in its design. In Chapter 6, a comprehensive mapping of safety requirements to the elements of Safe-AV is provided.

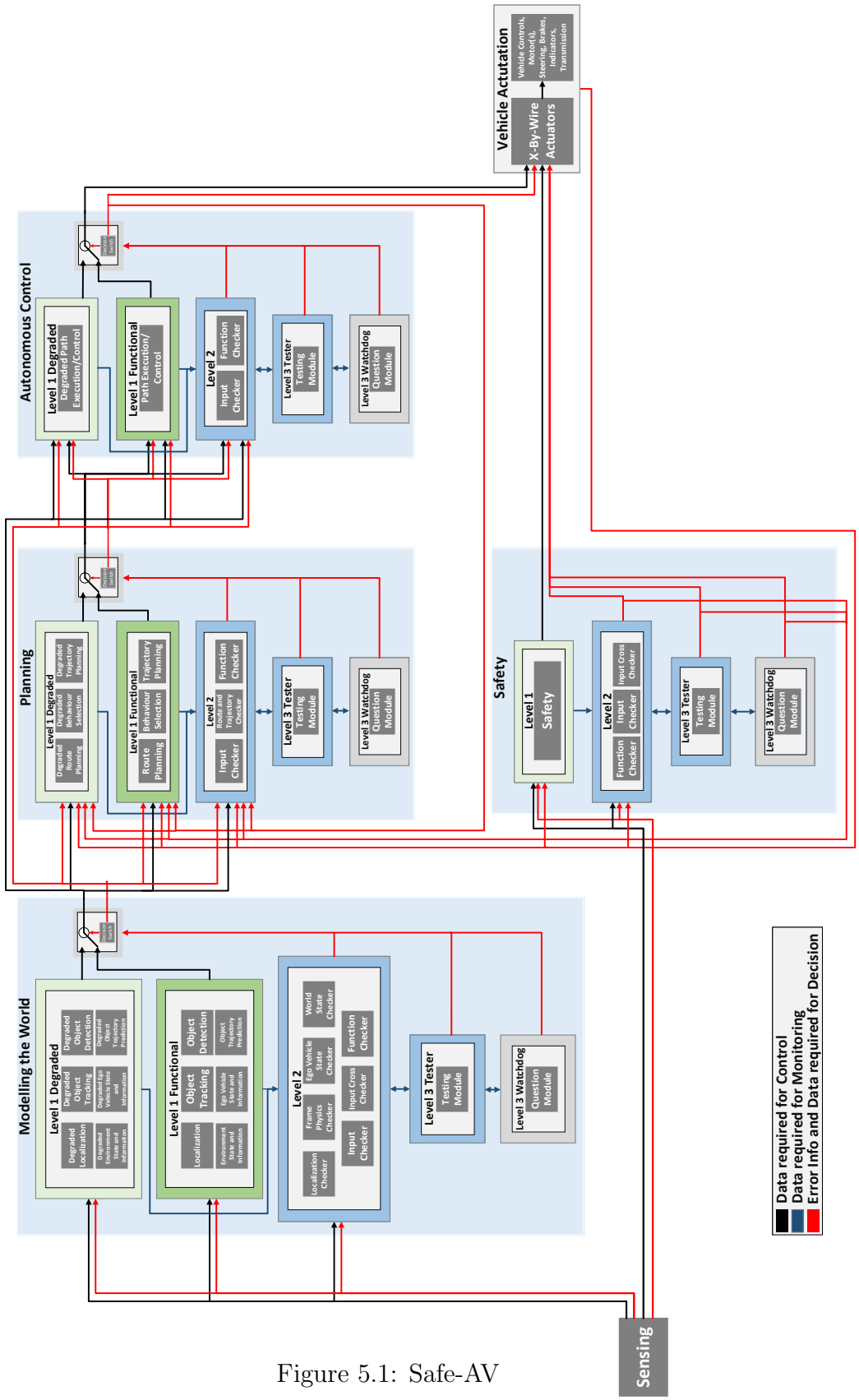

Figure 5.1: Safe-AV

5.1 Overview and General Architectural Features

The Safe-AV architecture (Figure 5.1) consists of six Core Modules (CMs). We use "Core" to indicate that a module is required in an AV. Other modules may exist that interface with the CMs and provide further functionality but this work focuses only on those that must exist. Five CMs are derived from the modules described in Section 4.1 with the final one, Safety, created to fulfill the safety requirement, SR-11: *A backup safety mechanism/short circuit that makes uses of redundant sensors and processing must exist. This system must bring the vehicle to a safe state if the main control loop fails to keep the vehicle safe.* To fulfill the safety requirements SR-13: *There must be redundant communication lines for critical data* and SR-16: *Critical data must use separate communication paths from non critical data*, when CMs communicate with each other, critical data uses redundant and separate paths from non-critical data. Each CM shares some common architectural elements. MTW, Planning, AC and Safety consist of a Level One (L1)[1], a Level Two (L2), a Level Three (L3) and a Decision Switch (DS). We shall begin with a general outline of each element followed by a more specific description of each CM's instance later on.

5.1.1 Level One

Level One consists of the all the core functionality and complexity of a CM. For MTW, Planning and AC, it is decomposed into two components, Level One - Degraded (L1D) and Level One - Functional (L1F). This is done to gain the benefits of the Simplex Architecture as well as to satisfy the safety requirement, SR-10: *PAC and MTW must provide, where feasible, degraded functionality in the presence of critical failures.* The reason for the decomposed components not being present in the other CMs is

[1]These are named to follow the E-Gas Monitoring Concept

explained further.

Level One - Functional consists of all the functionality required when the CM is free of critical and unrecoverable faults i.e. in a functional state. Failing that, Level One - Degraded consists of the minimum required and independently developed functionality for the CM to remain operational in the presence of major system failure or degradation. If a critical part of the CM has failed, L1D provides a path for the vehicle to remain safe and reach a safe state (be it a full stop or a pull over manoeuvre), a major requirement for an AV. The functionality in L1F is much more numerous and complex than L1D. As the latter only provides the minimum required functionality to reach a safe state, it must be made as simple as possible to give us a high assurance of it being less likely to fail. It is important to note, however, that L1F still has fault-tolerant capabilities in that it is able to recover from minor faults and/or provide levels of degraded functionality depending on the type of fault. For instance, if one of the numerous sonar sensors covering the rear of the vehicle fails, MTW's L1F should be able to detect this and provide degraded functionality instead of L1D taking over. The logic that dictates when the former is replaced by the latter will be made clear in the following sections.

To meet SR-17: *Diagnostic checks (memory checks, CPU checks, etc.) must be performed at startup and at TBD intervals or based on TBD conditions*, both L1F and L1D (or just L1 in the Safety CM's case) perform diagnostic checks at system startup and at specific intervals. These checks may include memory tests, CPU tests, transmitter tests, receiver tests, CM specific tests, etc. Both L1F and L1D outputs go to the module's DS and L2 via redundant communication lines. The decomposed components also report their error/fault information (including the results of the diagnostic checks) and a confidence on their output to L2 and DS.

5.1.2 Level Two

Level Two is an independently developed (from L1), high ASIL and low complexity monitor that is responsible for the main functional monitoring of L1. It, together with L1, forms a Monitor-Actuator pair (Section 2.5) that serves to detect faults in both the inputs and outputs of L1 and prevent them from propagating (with the help of the DS) to the rest of the system. As is the case in the Monitor-Actuator Pattern, this separation of control and safety allows us to design a complex but low ASIL L1 provided it is monitored by a high ASIL (and preferably, simple) L2.

L2 has access to L1's inputs and outputs. It consists of a set of plausibility checkers that range from simple limit checkers (e.g. ensure requested acceleration is not out of bounds) to, in the case of MTW, more advanced physics checkers that check the plausibility of a tracked object's trajectory. Plausibility checkers can also be in the form of function checkers that calculate the outputs of L1 functions using algorithms that are different and/or of a lower fidelity in comparison to their L1 counterparts. L2 sends out two types of information to the DS. The first being a signal sending out relevant error information (depending on the CM). The second being a decision signal that indicates if L2 trusts the outputs of L1. This decision signal is based on the result of the various plausibility checkers as well as the error information L2 receives from the components it monitors. Finally, as done in Section 5.1.1, L2 also performs diagnostic checks at system startup and at specific intervals or once certain conditions.

5.1.3 Level Three

L2 provided the means to ensure L1 faults are detected and contained. Consider, however, the scenario where the detector and container of faults (i.e. L2) is itself

faulty; the system may end up in a situation where L1 outputs are allowed to propagate to other CMs or situations where L2 unnecessarily claims that L1 is failing. To combat this, a high ASIL monitor, L3, is introduced. It is responsible for monitoring L2 and ensuring that it works as expected and meets its safety requirements. It consists of two components, a Level Three - Tester and a Level Three - Watchdog. Both allow L3 to perform its job with the use of the question and answer mechanism.

To reiterate what was said in Section 2.8, the question and answer process works as follows. During system design, a set of questions that test L2 functionality are created. Each question has an exactly defined answer. Level Three - Watchdog (L3W), which consists of a Question Module, is tasked with periodically asking these questions. Its counterpart, Level Three - Tester (L3T), consisting of a Testing Module, is tasked with finding the answer from L2 and transmitting the result to L3W. The latter component expects the right answer from L3T at the right time. A wrong or an ill-timed answer results in L3W's fault counter incrementing. Every correct answer at the right time results in L3W's counter decrementing, albeit at a slower rate (a wrong answer is weighted more than a correct one). If this counter reaches a threshold, then L3W knows that something is wrong with either L3T or L2.

To test its counterpart, L3T may sometimes send the wrong answer or the correct one at the wrong time on purpose. If L3W is working as intended, the wrong or ill-timed answer would increment L3W's counter. L3T can read this counter and knows that an error has occurred in L3W if said counter does not increment or decrement as expected. If this happens or if L3T does not receive a question form L3W at the expected interval, L3T updates its own counter. Once again, if this counter reaches a threshold, it knows that L3W is faulty. This can works backwards as well; L3W may occasionally not send a timely question or it may not increment its counter to see if L3T notices this change.

Tests involving L2 should not be done in a manner that interferes with L2's actual functionality (we do not want to be exposed to a faulty L1 at test time). One way this can be achieved is by L3T performing its tests on a copy of L2 software (complete with mirrored variables) which represents the current state of L2. Questions typically mimic various possible values or states of the outputs or inputs of L1 (be it correct ones or incorrect ones). For instance, to test the limit checker, L2 (or rather, its copy) is presented an out of bounds L2 output (say for a requested acceleration). If L2 is successful in detecting that the value is out of bounds, L3 knows that L2 is working as expected. Like L2, both L3T and L3W transmit their decision and error signals to the DS. The decision signals reflect L3W's trust on L2 and L3T as well as L3T's trust on L3W. Finally, as in L2, L3W performs diagnostic checks at system startup and at specific intervals or once certain conditions.

5.1.4 Decision Switch

The high ASIL DS acts as a safety gate and decides what a CM transmits. It contains a predicate that is a function of the decision and error signals from L2 and L3. The complexity of this predicate depends upon the specific CM and how it is implemented. It decides if the origin of the values is L1F or L1D. For instance, if L2 indicates that the data from L1F is untrustworthy, it routes data from L1D instead of L1F. Similarly, if L3 reports that L2 is untrustworthy, it routes L1D data.

5.1.5 Hardware View

A discussion of safety is incomplete without discussing a view of the hardware. To fulfill SR-28: *PAC, MTW and their internals must be sufficiently isolated, both in software and hardware to prevent single points of failure and common failure modes*,

within a CM in Figure 5.1, the outer colour of each component indicates how they should be separated. This separation allows us to avoid single points of failure, provides us separate Fault Containment Regions (FCRs) and allows us to freely update one component without adversely affecting another. Consider a situation where major memory corruption in L1F has caused it to fail. As L1D is tasked with taking over L1F in the presence of critical faults, propagation of this failure to L1D cannot occur as otherwise the system would fall-back to an unreliable and faulty component. In a similar vein, as L2 is tasked with monitoring L1, they must be sufficiently separated before we can trust L2 and before the system can gain the benefits of the Monitor-Actuator pair pattern. This separation must be done both in software and hardware. The former has been discussed in the preceding sections. The latter shall be discussed below.

The straightforward approach here would be to assign a separate physical controller for each colour in each CM. While this would have the benefit of complete hardware separation, the increased financial and potential management costs may make this approach infeasible (the financial costs can somewhat be mitigated with the usage of ASICs where applicable). An alternative solution is shared hardware that offers guarantees of isolation. While making an argument for isolation is difficult, Microchip vendors have, in the past few years, engaged in efforts to develop high ASIL certified System On Chips (SoCs) that offer a combination of CPU isolation, memory isolation, I/O isolation, task isolation and proper scheduling. However, before this approach is taken, a through analysis on its feasibility has to be performed; especially in the case of safety critical systems such as an AV. Special consideration would also have to be given to resource sharing between high ASIL (ASIL D) and low ASIL components (ASIL A); something that may very well not be feasible.

In order to fulfill SR-27: *System must have sufficient hardware backups to remain*

operational in the presence of hardware failures, hardware backups (another ingredient of the hardware view) must be provided. Even though some hardware faults are recoverable (e.g. memory corruption recovery using Error-Correcting Code (ECC) memory), before a system can truly be considered fail-operational, hardware backups that mitigate **unrecoverable** hardware failures must be considered. A straightforward approach here would be a hot standby for each hardware controller. Unfortunately, this solution suffers the same drawbacks that were mentioned in the earlier paragraph (mainly, financial costs). A potential alternative to this is the usage of the same SoC but with a separate core or CPU as the standby. Again, as before, this would require an argument in isolation and the same level of analysis for its feasibility.

5.2 Core Modules

With the general descriptions of the common elements complete, the six CMs can now be described in more detail. For ones that are taken from Section 4.1, their purpose and description shall generally not be repeated as this section serves to only describe the safety mechanisms that have been added. As a disclaimer, the sections below describe **our instantiation** of each CM and its specific safety mechanisms. While it is known what each CM does, how it does it (e.g. the algorithms used, etc.) is out of the scope of this work. Examples and details shall still be provided where needed. Finally, when discussing the various L1Ds, the targeted safe state of the vehicle shall be a complete stop.

The reader will notice that only four of the six CMs have an L2 and an L3. This choice is not a comment on how safety critical the remaining two are. It is made on the assumption laid out in Section 4.5 i.e. a discussion of their safety is out of the scope of this work as it is assumed that the safety mechanisms for the XBW systems and the

sensors that are used in modern non-autonomous vehicles are already present. To deal with the new types of sensors that are used for AV functionality, the responsibility of ensuring that they work as intended has been split between MTW and Sensing.

5.2.1 Sensing

Sensing, perhaps the least interesting CM, interfaces with the sensors of the vehicle (e.g. LIDAR, Radar, Cameras, Sonar, IMU, GPS) and provides their outputs to MTW and Safety. To satisfy SR-5: *Where feasible, Sensing must use redundant (both homogeneous and heterogeneous) sensors*, it consists of redundant sensors (both homogeneous and heterogeneous), where feasible, to avoid single points of failure. The specific sensor configuration is, of course, implementation specific but it is assumed that it provides sufficient sensor coverage.

To fulfill SR-12: *MTW and Sensing must perform relevant sensor diagnostics checks (e.g. calibration check, position check, coverage check, cross checks) at startup and TBD intervals during operation*, the Sensing CM performs diagnostic checks at start-up and at decided intervals or conditions to ensure that sensors are properly calibrated, properly positioned and not compromised. To meet SR-29: *Sensing must report its failures, faults or errors*, the results of these checks are sent to the CMs it outputs to. This allows them to react accordingly if there are sensor issues. If the reader recalls the windshield incident from Section 3.6, they will realize this requirement would have forced the involved vehicle's Sensing CM equivalent to report the displaced windshield to the vehicle's Planning equivalent.

5.2.2 Modelling the World

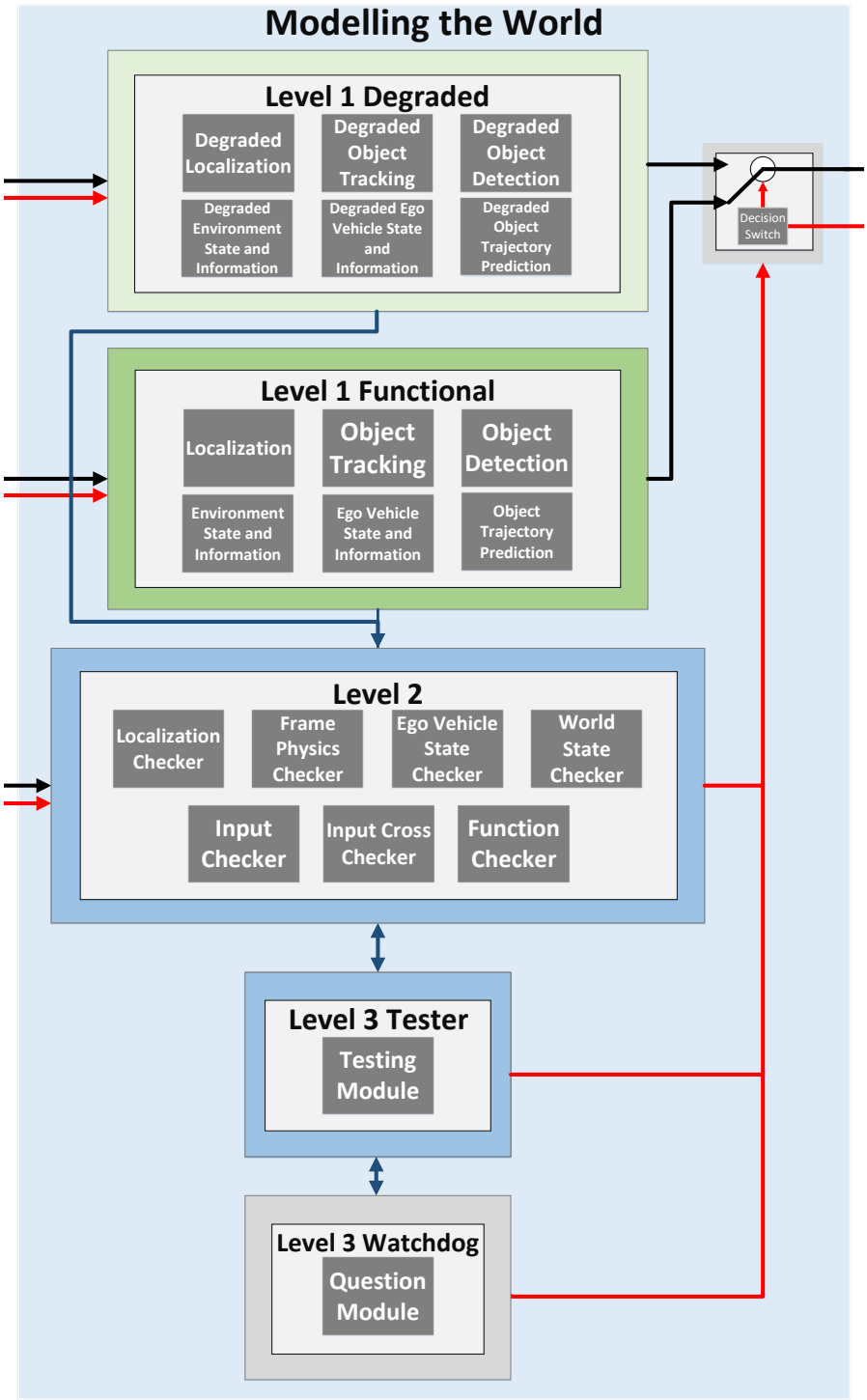

Figure 5.2: Modelling the World

MTW (Figure 5.2) is one of the more complex CMs. To fulfill SR-31: *MTW must take into account error/failure information from Sensing when performing its functionality*, each level takes into account error information from the Sensing CM when performing its functionality. Each level also reports its own error information to DS which in turn reports it to Planning and AC. L1 reports its confidence (via the DS) to Planning and AC to fulfill SR-3: *MTW must report its confidence on each applicable output*. This metric indicates L1's confidence on what it sees, its outputs and how much that information can be trusted. The confidence is based on failing, unsure or uncertain algorithms, sensor reported failures/issues (e.g. camera has not being calibrated after windshield replacement [Section 3.6]) or certain environment conditions that prevent optimal sensor performance (e.g. foggy or dusty conditions affecting LIDAR).

To meet SR-26: *Where feasible, MTW and PAC must make use of multiple sources of information when performing their functionality*, each component makes use of multiple sources of information to perform its functionality and, in the case of L1, to increase or decrease its confidence value. To provide a need for this requirement, consider the fatal accident in Florida (Section 3.2) where Tesla claims that their radar incorrectly detected a truck as an overhead sign. By taking into account some form of a map that indicated the locations of overhead signs, the detection algorithm could have, perhaps, detected the object correctly as a truck or, if it was unable too, at least reported the "overhead sign" with a low confidence. This, in turn, may have allowed the vehicle to act more "cautious" and slow down; potentially preventing the fatality. In a similar view, the fatal accident in Section 3.5 may also have been prevented. The misidentified lane markers, which we speculate to be the cause of the accident, could have been cross referenced against a high density map of the road to prevent the vehicle from following the misidentified lane into the crash attenuator.

Level One

L1F and L1D have been divided into six components. This division is not strictly a separations of concerns as many of the components do intersect in functionality (e.g. Localization can be considered a subset of Ego Vehicle State). Instead, it is an attempt to identify major functionality that MTW should provide. It is assumed that state of the art algorithms will be employed in L1F. A small summary of what each component does is provided:

- **Localization**: Deals with obtaining the ego vehicle's location with respect to the environment. A degraded version may include a simpler algorithm that does not provide as high of an accuracy and/or makes use of less sources of knowledge (assuming this results in less complexity).

- **Object Detection**: Deals with detecting objects in the environment. The degraded variant may choose to make use of deterministic non-ML based algorithms to detect dangerous objects. Alternatively, if need be, it may use simpler ML algorithms that are trained to classify objects into a limited set of classes (e.g. object is a danger or not vs object is a pedestrian or object is a car).

- **Object Tracking**: Deals with tracking objects in the environment across frames. There may not be a degraded variant as the safety mission (i.e. a full stop) may not require object tracking.

- **Object Trajectory Prediction**: Deals with using object tracking information (along with other information) to predict future states of objects in the environment. There may not be a degraded variant as the safe state (i.e. a full stop) may not require object trajectory prediction.

- **Environment State and Information**: Deals with determining the states of various structures and objects in the environment. An example of this is the current state of a traffic light. It also contains information about these objects and structures (e.g. lane markers). An example of this would be the speed limit of the current road. A degraded variant would only need to supply enough information for the vehicle to know that it is not stopping in a dangerous location (e.g. at the middle of an intersection).

- **Ego Vehicle State and Information**: Deals with obtaining the ego vehicle's state and information. This includes the ego vehicle's acceleration, speed, orientation, etc. A degraded variant would only need to supply relevant information to bring the vehicle to a stop (e.g. vehicle speed, requested braking signal, braking value, etc.).

Level Two

To meet SR-1, SR-2, SR-4, SR-12 (partially) and SR-18, we have the following plausibility checkers:

- **Localization Checker**: This serves to detect issues in L1's localization component. It can do so by checking for impossibilities in ego vehicle location and orientation. It can detect, for instance, if the the ego vehicle seemingly teleports from one location to another or is in a location that the checker disagrees with or is in an area of the environment that is inaccessible (e.g. middle of a river).

- **Frame Physics Checker**: This serves to check for impossibilities or inconsistencies in frames. Using knowledge about how objects in the environment can move and their physical limitations, the checker can detect if an object is seemingly moving too fast or is in impossible positions between frames.

- **Input Checker**: This checker performs simple limit checks and plausibility checks on input values.

- **Input Cross Checker**: This checker adds another layer of checking on input sensor values. We can use the fact that certain sensors have an overlap in what they can sense to cross check their outputs and detect if one of them is failing or is unreliable. As an example, radars and LIDARs are capable of detecting if an object is in the ego vehicle's path (assuming ideal conditions for their operation). If the LIDAR is reporting an object that the radar is not and there is no other explanation for the radar failing, then the checker knows that the radar has an issue.

- **Ego Vehicle State Checker**: This checks for inconsistencies in the ego vehicle's reported state. For example, it can check if the reported ego vehicle speed is within its limits or if that speed changes in an impossible manner (i.e. going from 10 km/h to 200 km/h in 1 s).

- **World State Checker**: This checks if objects in the environment exist or behave as expected. For instance, it can check if the reported lane markers match a high definition map of the road. It can also check, with the use of simpler, non-ML based algorithms if L1 is failing to detect objects. In future iterations of Safe-AV, this component can use information from other vehicles or the infrastructure (i.e. Vehicle-To-Everything (V2X)) to do smarter checking (e.g. check if infrastructure reported traffic light state matches MTW reported traffic light state).

- **Function Checker**: A generic placeholder that is used to detect incorrect L1 outputs that were not covered by the other checkers. The functionality here is

ideally deterministic, simple and easy to assure.

If the above checkers report inconsistencies or issues, then L2 is able to detect that L1 is failing and/or the Sensing CM is failing. This information is reported to the DS which in turn reports this to Planning and AC.

Level Three

The specific question set and, as a result, L3's content is left as a detail of the implementation. In general however, each of the checkers in L2 shall have a set of associated questions that test their ability to detect known issues or inconsistencies with L1. Some examples include:

- Example Question 1
  - **Presented L1 Input**: Image with no dangerous objects
  - **Presented L1 Output**: No dangerous objects detected
  - **Objective**: Test if L2 erroneously fails or correctly passes L1
- Example Question 2
  - **Presented L1 Input**: Image with dangerous objects
  - **Presented L1 Output**: No dangerous objects detected
  - **Objective**: Test if L2 erroneously passes or correctly fails L1
- Example Question 3
  - **Presented L1 Input**: –
  - **Presented L1 Output**: Inconsistent localization information (i.e. vehicle "teleported" from one position to another)

  - **Objective**: Test if L2 correctly detects incorrect localization and fails L1

- Example Question 4
  - **Presented L1 Input**: –
  - **Presented L1 Output**: Impossible changes in ego vehicle speed
  - **Objective**: Test if L2 correctly detects incorrect impossible ego vehicle speed and fails L1

5.2.3 Planning

In general, to meet SR-8: *PAC must take into account fault info and the confidence of the incoming data when performing its function*, the components of Planning take into account the error outputs of AC, MTW, VA and Safety when performing their functionality. They also take into account the confidence values from MTW's L1. These values (indicating MTW's confidence on its outputs) play a part in how "cautious" Planning acts during its operation. Finally, components of Planning report their error information to the DS which in turn reports this to AC.

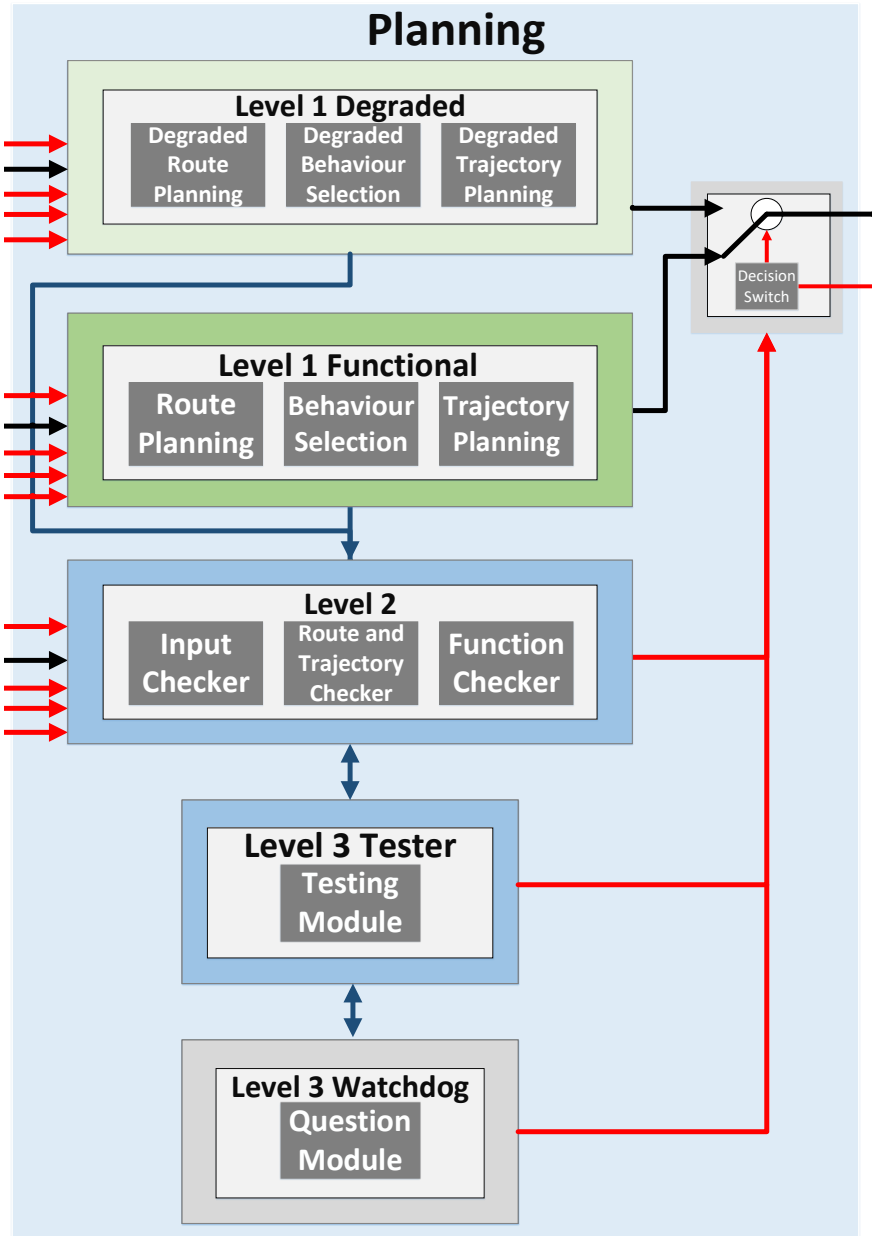

Figure 5.3: Planning

Level One

L1 consists of three components:

- **Route Planning**: This component is responsible for generating a global route

that takes the vehicle from a source to its destination. While the state of the art may attempt to find the optimal safe path based on number of constraints, a degraded variant may just be one that finds a feasible path. If we consider the safe state to be a full stop, the the degraded variant might not be needed.

- **Behaviour Selection**: This is, perhaps, the most complex component in Planning. It is tasked with selecting the optimal behaviours (e.g. lane change, complete stop) that allow the vehicle to safely navigate a path and react to changes in the environment (e.g. a child suddenly appearing on the road). It does this by taking into account information about objects around the vehicle (e.g. object classifications, object locations) as well as their predicted trajectory. The state of the art algorithms here can range from complex state machines that select the appropriate behaviours based on some pre-defined rules and heuristics or, considering the complexity of and the variability in situations that arise when driving, ML based algorithms. A degraded variant of this component could be a simple rule based state machine that is of sufficient fidelity to bring the vehicle to a safe state.

  As a general rule, this component should be "cautious" in its approach and act defensively when selecting behaviours This cautiousness may have prevented the accident in Section 3.3. When the leading vehicle swerved out of the way to avoid the stopped fire truck, the Tesla could have, considering the sudden change in its path, acted cautiously and not sped up immediately. While this may not have necessarily prevented the accident with the fire truck (assuming the sensors still failed to detect it), it may have, in the very least, resulted in a lower speed collision. This approach may also have prevented the accident in Section 3.4 in which the vehicle initially detected the pedestrian as an unknown

object 6s before the collision. Had it acted cautiously and started slowing down at that point ot time, the accident may not have been fatal.

- **Trajectory Planning**: This component takes the selected behaviours and generates an optimal and safe trajectory. In the state of the art, this trajectory may optimize for passenger comfort, fuel economy, drive profile, etc. A degraded variant may simply obtain a feasible trajectory that is aimed at bringing the vehicle to a safe state.

Level Two

To meet SR-6, SR-7 and SR-9, L2 consists of the following checkers:

- **Route and Trajectory Checker**: This component detects issues with L1 by checking that the generated route and trajectories are feasible solutions for a simpler algorithm that only optimizes for a collision free route. The component also checks if the generated trajectories respect the confidence values from MTW. For example, if MTW is unsure that an object is in the vehicle's path, this component would check if L1 appropriately responds with more cautious behaviour.

- **Input Checker**: See Section 5.2.2. To provide an example, this component can check if MTW's reported acceleration is physically possible and within limits.

- **Function Checker**: See Section 5.2.2

Level Three

As in MTW, the question set is an implementation detail and would ideally test each checker in L2. Examples questions include:

- Example Question 1
  - **Presented L1 Input**: –
  - **Presented L1 Output**: Trajectory with imminent collision
  - **Objective**: Test if L2 detects dangerous trajectory and correctly fails L1
- Example Question 2
  - **Presented L1 Input**: –
  - **Presented L1 Output**: Route that is not physically possible
  - **Objective**: Test if L2 detects impossible route and correctly fails L1
- Example Question 3
  - **Presented L1 Input**: Low confidence (e.g. cause by a blocked sensor) MTW detection
  - **Presented L1 Output**: Trajectory that does not respect (e.g. by continuing to accelerate) the low MTW confidence
  - **Objective**: Test if L2 detects L1's failure to respect MTW's low confidence and correctly fails L1

5.2.4 Autonomous Control

In general, to meet SR-8, AC's components take into account error and confidence information from Planning and MTW when performing their functionality. They also report their error information to the DS which in turn reports it to Planning and VA.

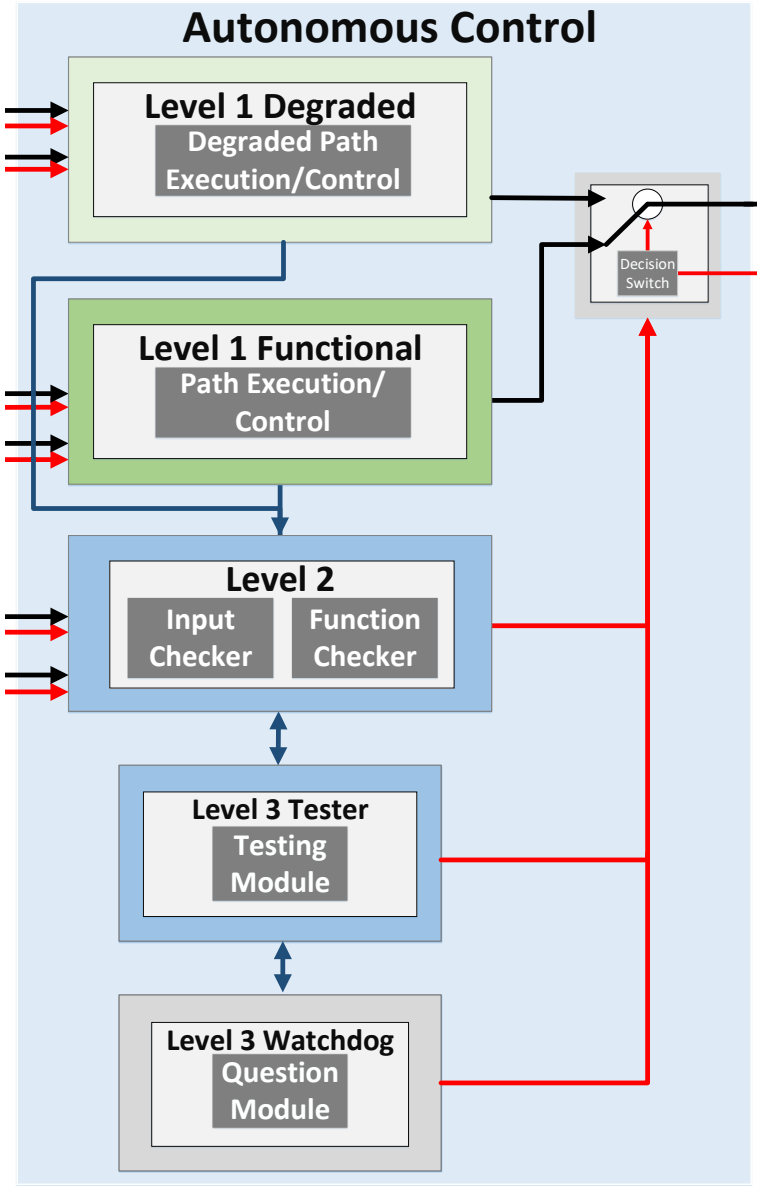

Figure 5.4: Autonomous Control

Level One

L1 consists of a sole component, Path Execution/Control, that executes generated trajectories from Planning and corrects tracking errors. L1F uses state of the art algorithms such as sophisticated Model Predictive Control (MPC) based ones that

can deal with various road conditions (e.g. icy roads). A degraded variant may use simple algorithms such as Pure Pursuit [82] that make assumptions about the road and only allow the vehicle to perform simple manoeuvres (such as drive along a straight road) instead of the more advanced ones.

Level Two

To meet SR-6, SR-7 and SR-9, L2 consists of two checkers:

- **Input Checker**: See Section 5.2.2
- **Function Checker**: See Section 5.2.2. An example check would detect if L1's outputs are feasible (e.g. a braking actuation command that is physically possible or out of limits)

Level Three

As before, the question set would depend on the chosen algorithms and their implementations. Some examples include:

- Example Question 1
  - **Presented L1 Input**: Trajectory that is impossible (e.g. takes vehicle onto opposing lane)
  - **Presented L1 Output**: –
  - **Objective**: Test if L2 detects impossible trajectory from Planning
- Example Question 2
  - **Presented L1 Input**: Trajectory that is impossible (e.g. with impossible ego vehicle speed)

- **Presented L1 Output**: Actuation commands to XBW systems that represent impossible ego vehicle speed
- **Objective**: Test if L2 detects L1 not reacting to impossible requested trajectory correctly fails it

5.2.5 Safety

The Safety CM or the safety short circuit is tasked with keeping the vehicle safe if the main loop (consisting of MTW, Planning and AC) fails to do so. It effectively short circuits the main loop by being directly connected to the Sensing module and VA. This CM may have prevented the accidents in Sections sections 3.1 to 3.5 when each vehicle's Planning equivalent failed to do so. Specifically for the Uber accident, had the vehicle's Safety equivalent (i.e. its AEB) not been disabled, the vehicle, as shown in Section 3.4, would have very likely come to a stop and not hit the pedestrian.

Level One

The reader may notice that this module does not have a decomposed L1. This is because it is assumed that L1 is always a high assurance (using terminology from the Simplex Architecture) component (equivalent to just an L1D). To perform its job, it needs to be simple and of low complexity. It only contains the minimum required functionality to bring the system to a full stop and as such there is no need for a more complex L1F that provides fancier functionality. Due to its low complexity, the module can exist on a faster and smaller processor or ASIC. L1, in its simplest iteration, uses a redundant set of sensors (e.g. LIDAR, Radar, Camera, Sonar, etc.) that are processed using, if feasible, a deterministic non-ML based algorithm to detect dangerous objects. If need be, an alternative would be a simple ML based algorithm

that is trained to only classify objects as dangerous or not dangerous. In an ideal world, the sensors used by this CM would be independent and isolated from the main loop's sensors. However, the additional costs incorrect may be significantly prohibitive to this approach.

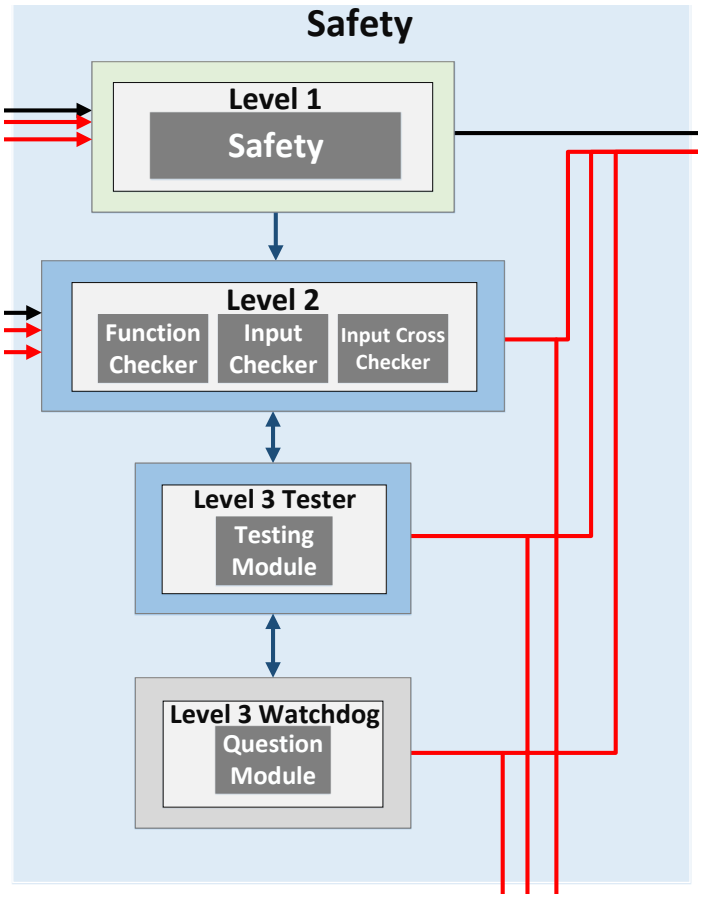

Figure 5.5: Safety

If there is an object in the vehicle's path that is violating or close to violating the safe distance and the main loop fails to react, L1 will take over. If it classifies the object as dangerous or it is unsure of the classification or if its sensors are reporting major failures, L1 commands VA to actuate the brakes and bring the vehicle to a stop.

Level Two

L2 consists of following plausibility checkers:

- **Input Checker**: See Section 5.2.2
- **Input Cross Checker**: See Section 5.2.2
- **Function Checker**: See Section 5.2.2

Level Three

Similar to the previous CM's, the specific question set will depend on the implementation. However, some examples include:

- Example Question 1
  - **Presented L1 Input**: Vehicle not violating safe distance
  - **Presented L1 Output**: Braking command sent to XBW system
  - **Objective**: Test if L2 correctly fails L1
- Example Question 2
  - **Presented L1 Input**: Vehicle is violated safe distance to detected object
  - **Presented L1 Output**: L1 output that does not react to detected objects
  - **Objective**: Test if L2 correctly fails L1

5.2.6 Vehicle Actuation

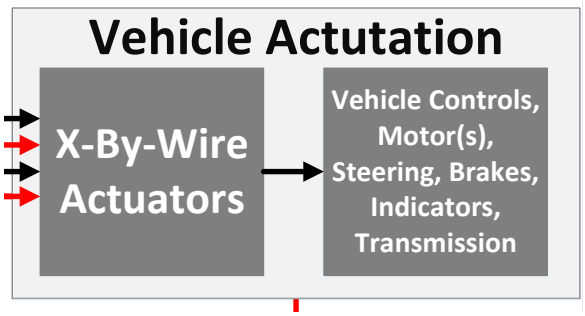

Figure 5.6: Vehicle Actuation

The only requirements placed on this component (under the assumptions made in Section 5.2) are that it must report its errors to Planning and the Safety CM. This satisfies SR-20: *XBW system must monitor its functions and health and report faults to PAC*. It is assumed that existing safety mechanisms (those that are out of the scope of this work) satisfy SR-19: *XBW system must validate its output commands before transmission* and SR-21: *XBW system must validate incoming data*.

This component receives error information from AC and the Safety CM. This allows it to detect issues with both the main loop (consisting of MTW, Planning and AC), the Safety CM and allows it to perform any mitigations. Additionally, if the XBW system satisfies SR-21, it is able to detect out of bounds requests from AC or the Safety CM (we know this is done for TBW if the E-Gas Monitoring Concept is applied). In both instances, it can respond by limiting its output to a known safe value. For instance, if VA deems that the vehicle needs to come to a stop, it can very easily activate the braking actuators to bring the AV to a stop.

Chapter 6

Discussion

We begin this chapter with a mapping of the safety requirements from our STPA in Chapter 4 to Safe-AV components. We then outline the architecture's principles and its merits. This is followed up by a discussion on its evolution over the course of this work. Finally, we discuss the architecture's incompleteness and how it can further be evolved.

6.1 Meeting Safety Requirements

Table 6.1 iterates over all the safety requirements derived in Chapter 4 and indicates if and how Safe-AV meets them. Figure 6.1 more specifically shows how they are allocated to Safe-AV.

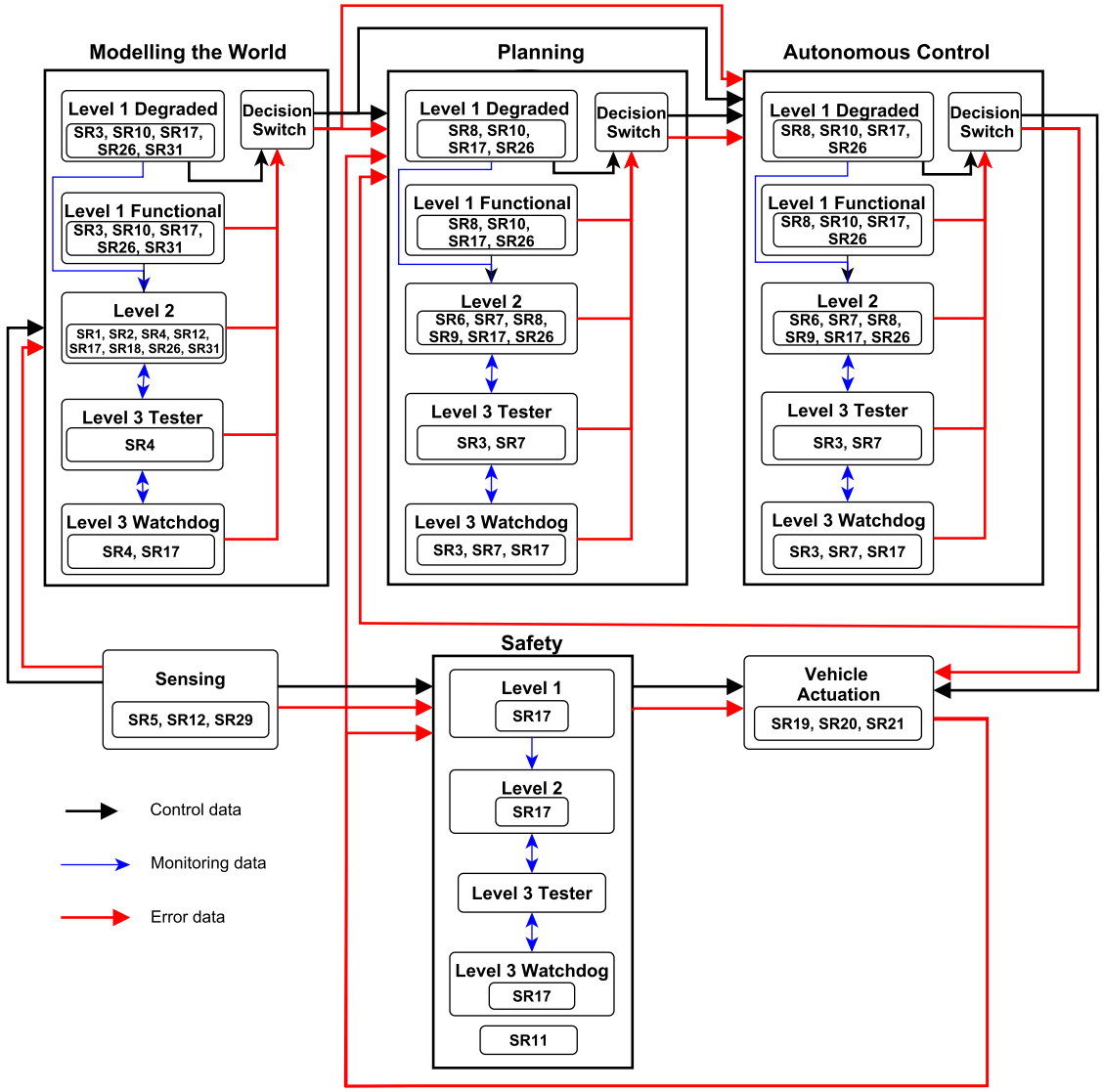

Figure 6.1: Safety Requirements allocated to Safe-AV Components

Ref	Met?	How?
SR-1	✓	MTW's L2 checks outgoing values
SR-2	✓	MTW's L2 checks incoming values
SR-3	✓	MTW's L1F and L1D report their confidence on outgoing values
SR-4	✓	MTW's L2 and L3 report faults and errors found in MTW
SR-5	✓	The Sensing CM uses multiple types of sensors
SR-6	✓	Planning and AC's L2s check incoming values

SR-7	✓	Planning and AC's L2s and L3s report faults and errors found in their respective CMs
SR-8	✓	Planning and AC take into account incoming confidence measures and error information when performing their function
SR-9	✓	Planning and AC's L2s check outgoing values
SR-10	✓	MTW, Planning and AC's L1Fs provide limited degraded functionality. Additionally, their L1Ds provide degraded functionality when the L1Fs fail critically
SR-11	✓	The Safety CM acts as a backup safety system
SR-12	✓	The Sensing CM performs diagnostics checks. Additionally, MTW's L2 validates/cross checks incoming sensor data
SR-13	✓	All critical communication in Safe-AV goes through redundant lines
SR-14	N/A	Requirement placed on implementation
SR-15	N/A	Requirement placed on implementation
SR-16	✓	Critical and non-critical data are separated during communication
SR-17	✓	Each CM's L1, L2 and L3W perform diagnostic checks
SR-18	✓	MTW's L2 checks outgoing values
SR-19	N/A	Requirement placed on implementation of XBW system (out of scope)
SR-20	N/A	Requirement placed on implementation of XBW system (out of scope)
SR-21	N/A	Requirement placed on implementation of XBW system (out of scope)
SR-22	N/A	Requirement placed on implementation
SR-23	N/A	Requirement placed on implementation
SR-24	N/A	Requirement placed on implementation
SR-25	N/A	Requirement placed on implementation
SR-26	✓	MTW, Planning and AC's L1s and L2s use multiple sources of information
SR-27	✓	Safe-AV enforces hardware backups but how it is achieved is left as an implementation detail
SR-28	✓	Safe-AV enforces both hardware and software separation on its CMs and its internals but leaves how hardware separation is achieved as an implementation detail
SR-29	✓	The Sensing CM reports its errors to MTW and the Safety CM
SR-30	N/A	Requirement placed on implementation
SR-31	✓	MTW's L1 and L2 take into account the Sensing CM's error information

Table 6.1: Meeting Safety Requirements

6.2 Principles and Merits

This section will reiterate some of the core principles of Safe-AV and outline their benefits.

6.2.1 Separation of Concerns via Modularity

Separation of concerns is a key concept in the design of system architectures. An architecture that adheres well to this principle yields system components that present a clear separation in functionality, easily identifiable boundaries, high re-usability, simpler development cycles and easier maintainability. Safe-AV, with its modular design, adheres to this principle in a multitude of manners. Starting at the highest level, each CM represents a clear separation in responsibility and functionality. Then, within each CM itself, there is a separation of control and safety with the L1s being responsible for control and the L2s, L3s and DSs being responsible for safety. Further, within each L1 and L2, the overall functionality of the levels is separated, where feasible, and designated to sub-components.

6.2.2 Separation in Hardware

Another manner in which Safe-AV adheres to separation of concerns comes from the rules it enforces on hardware isolation. This isolation is needed to combat single points of failure and reduce common failure modes. The architecture does not explicitly provide a mapping between physical controllers and its CMs (and their sub components); it, instead, places a requirement on which components need to be separated and isolated from each other. How this isolation is then achieved is left as an implementation detail provided a convincing argument of isolation is made.

6.2.3 Separation of Control and Safety

The L1s and L2s of each CM form Monitor-Actuator pairs. This pattern allows each CM to separate its control logic from its safety related logic. The benefits of the pattern are outlined in Section 2.5. To summarize:

- The monitor and actuator provide a natural mapping in hardware and the benefits that come with the potential separation
- With ASIL decomposition, the L2s can be assigned a high ASIL while the more complex L1s can be assigned a low ASIL. This makes managing the safety and risk of a complex L1 cheaper and more feasible
- The separation of control and safety combined with the hardware separation makes updating the L1s a less tedious process as, ideally, the changes should not affect the L2s and there should not be a need to re-certify the latter

6.2.4 Monitoring the Monitor

It is apparent that the Monitor-Actuator Pattern offers significant benefits. However, before making use of such a pattern, we must first properly consider its drawbacks; especially for increasingly complex systems such as AVs and their ML components. While the pattern offers protection against faults in the functional part of the system i.e. L1, it does not provide a means to detect or deal with faults in the component that is responsible for ensuring the safety of the system i.e. L2.

A faulty L2 has the potential to erroneously detect faults in L1 where none exist or, more dangerously, not detect faults in an actually failing L1. While the results of the former may not be that detrimental (consider the system reaching a safe state when it does not have to), it is the latter failure that can potentially cause great harm

or catastrophe. The combination of a failing Object Detector that does not detect an imminent truck in the vehicle's path and an L2 that does not detect this failure might be fatal. To combat this deficiency in the pattern, Safe-AV employs, as is done in the E-Gas Monitoring Concept, L3s that are able to detect failures in the L2s and consequently, provide the AV an opportunity to react to these failures.

6.2.5 Graceful Degradation and Fail-Operational Behaviour

Being operational in the presence of failure is crucial for an AV. As it is a safety critical system, the result of its important components failing can cause great harm or even loss of human life. As such, an AV has to designed in a manner that allows it to either recover from faults or reach a state that prevents the faults from causing harm. Safe-AV does this with its Level One - Functional and Level One - Degraded components. The former is able to deal with recoverable faults and provide degraded functionality while the latter, based on the Simplex Architecture, exists to take over when L1F cannot recover from a fault. As stated in previous chapters, L1D consists of the minimum required functionality to bring the system to a safe state. While its effectiveness depends on the desired safe state, even the simplest safe state (e.g. a complete stop) provides significant value.

6.2.6 Ease of Implementation

A consequence of the architecture being based on well adopted industry concepts and patterns is that the engineering costs associated to its implementation and maintenance are potentially reduced. This is because the experience and expertise with these patterns may already exist within the automotive industry. Suppliers and OEMs might already have the personnel, documentation, processes, etc. required. Further-

more, the architecture's modular nature and its adherence to separation of concerns permits a development process where boundaries are easily identified and responsibilities distributed.

6.3 Evolution of the Architecture

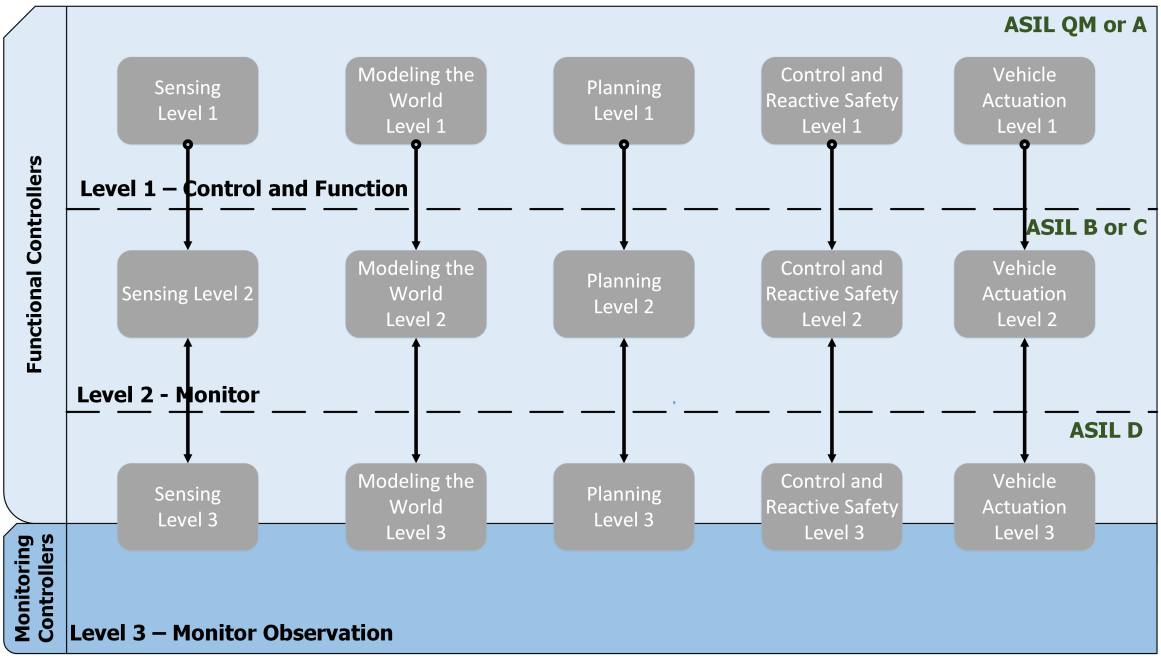

Figure 6.2: Version 0.1

This section showcases the various iterations of the architecture during its development. This is done to provide the reader, what we hope is, an interesting insight into the various factors that affected its evolution. The initial version (Figure 6.2) was a simple extension of the E-Gas Monitoring Concept adopted to autonomous driving. In this version, AV functionality, based on the author's understanding and current literature in functional architectures, was divided up into five modules. The reader will notice that *Reactive Safety* (now called Safety) and *Control* (now called Autonomous Control) were one in the same.

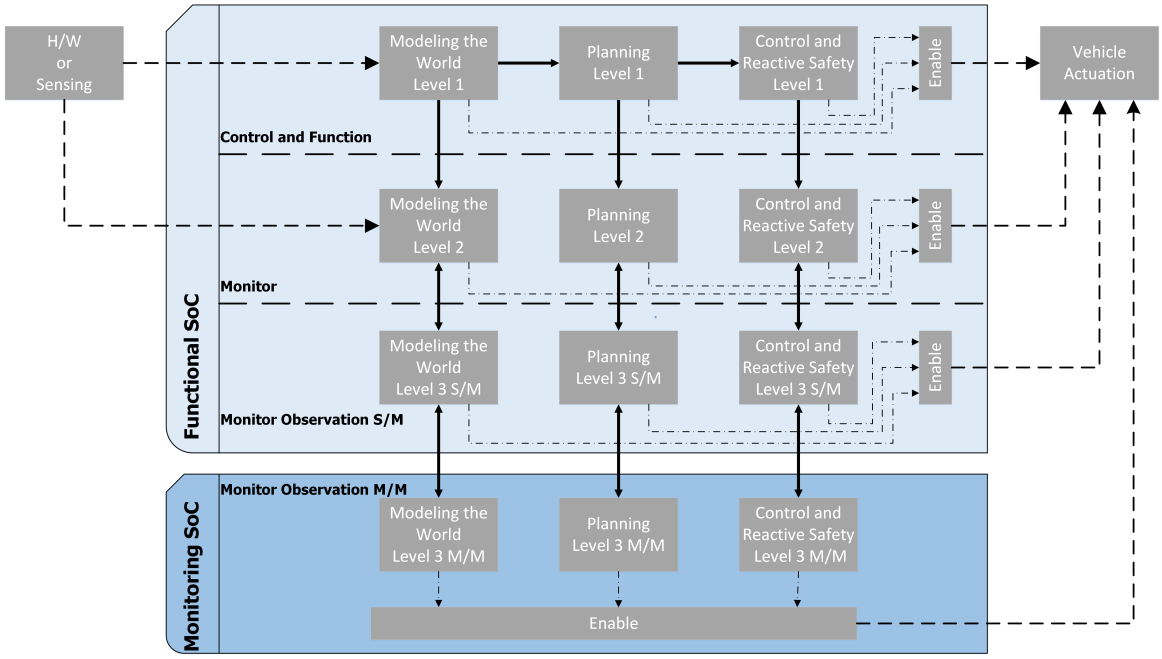

Figure 6.3: Version 0.2

We deemed that the *Sensing* and *Vehicle Actuation* modules were out of the scope of our work (reasons for this were provided in earlier chapters) and as such, Figure 6.3 shows them removed from the three level concept. To further refine the architecture, the flow of data between each module was added. Finally, Level Three was divided into two components to allow for a cleaner separation and a clearer presentation.

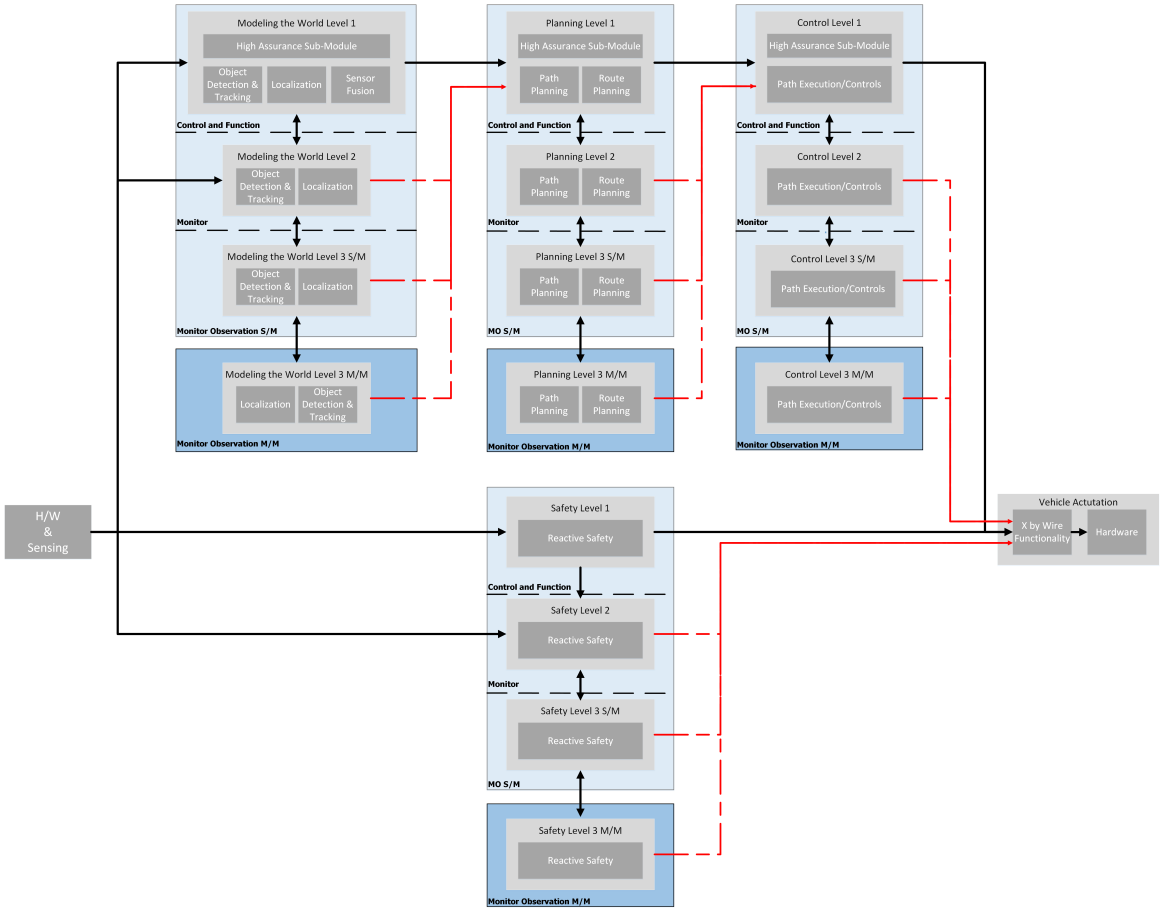

Figure 6.4: Version 0.5

Figure 6.4 shows the next version. Here, more detail was provided for each of the modules, the error flow and the data flow. Additionally, module separation was also made more prominent. To separate the main loop and its back-up safety system, the previously named *Control and Reactive Safety* were split into separate modules. Finally, to allow for fail-operational behaviour and graceful degradation, we adopted the Simplex Architecture and introduced high assurance sub-modules for each module.

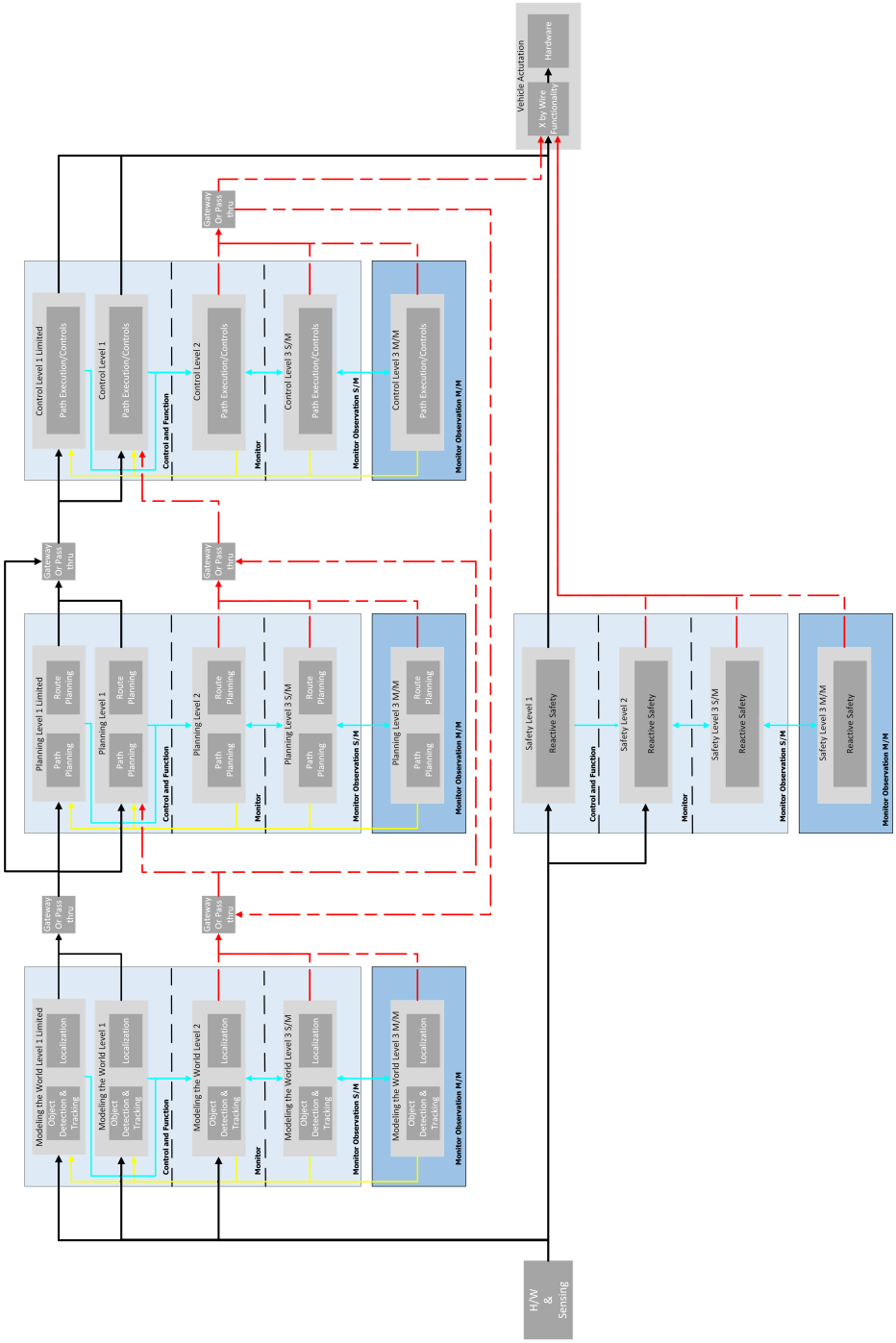

Figure 6.5: Version 0.8

This version (Figure 6.5) further refined Level One by visually separating its high assurance sub-module (renamed to *Level 1 Limited* at the time) and its high per-

formance sub-module (renamed to simply *Level 1* at the time). This was done to emphasize the importance of them being isolated form each other. Additionally, the data flow from the L1 modules was refined.

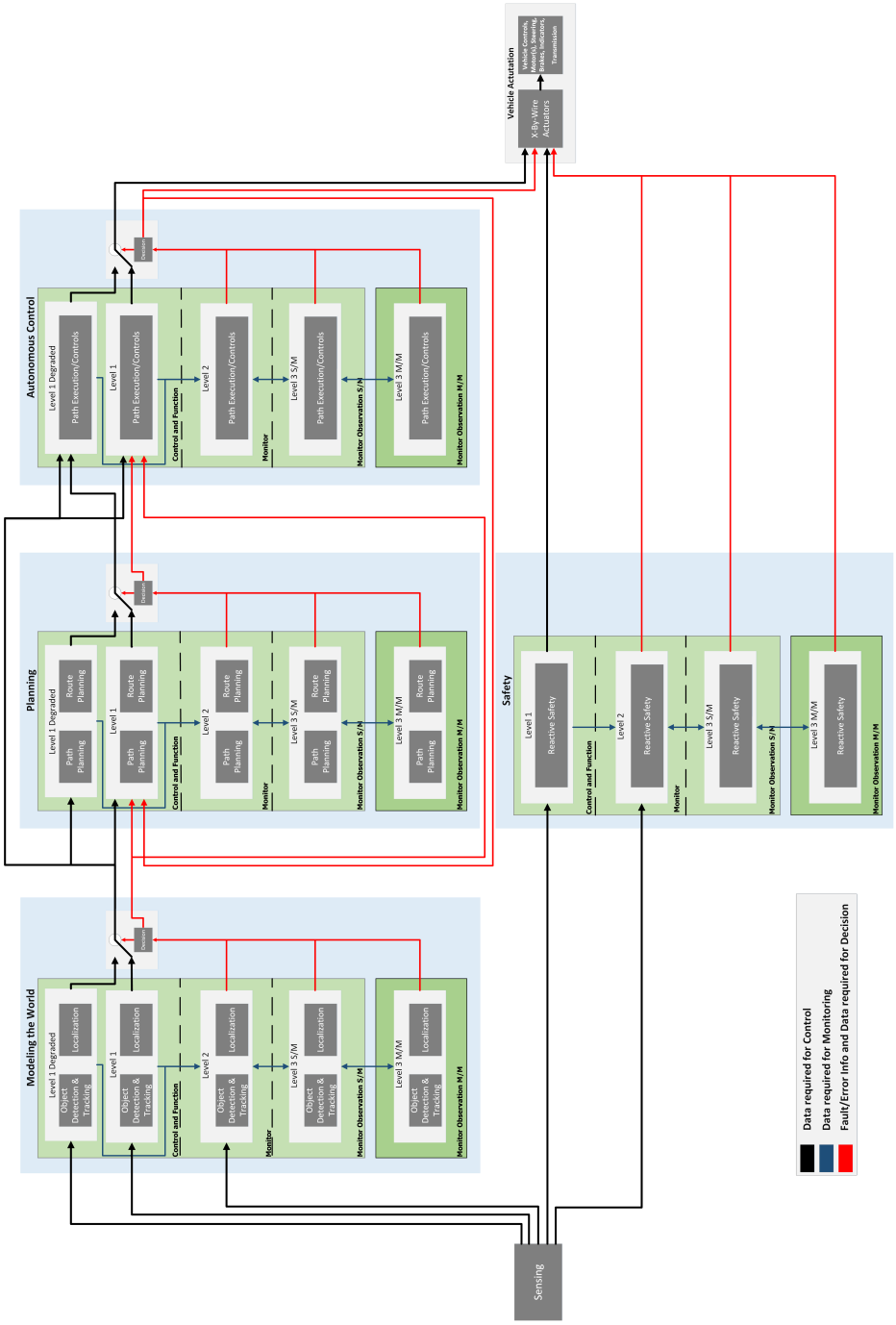

Figure 6.6: Version 1

Finally, Figure 6.6 shows the version preceding the most current one. This iteration removed redundant data flow lines. It also added the *Decision Switch* component to act as a safety gate that decided if L1F or L1D data was allowed through. The version was created before the Hazard Analysis in Chapter 4 was completed. The transition from this version to the latest (Figure 5.1) was a partially a result of the completed HA. Besides removing redundant labels and a general clean up, the latest version added significant detail for each CM and their levels. The completed HA played a part in allowing us to refine each CM. Its results also provided the detail required for the contents of the L2s and partially the L1s.

6.4 Towards Completeness

The latest version of the architecture presented in Chapter 5 represents our first attempt at a safe system architecture for Autonomous Vehicles. As safety is an iterative process, Safe-AV is by no means complete. Just as it evolved in the section above, it must continue evolving before we can be confident of its usefulness. We shall discuss how it can be evolved below.

Further Hazard Analyses

The HA presented in Chapter 4 only covers the highway driving scenario. While the architecture itself is designed to cover other scenarios (e.g. city driving), the HA can be improved by adding UCAs that consider more scenarios. This can potentially result in more safety requirements that augment Safe-AV.

Another major consideration to the HA are the safety mechanisms that have been added to Safe-AV. We must ensure that any introduced elements to the architecture

that are intended to mitigate the initial hazards do not themselves present new hazards that are not identified. As such, the HA presented in Chapter 4 is no longer sufficient and a new one must be performed to identify further hazards and safety requirements. These would then refine the architecture.

Finally, the HA in Chapter 4 is at a high level of abstraction. While this was sufficient for obtaining the current version of the architecture and the initial design for the contents of each CM, we must delve deeper and perform CM specific HAs to obtain more refined CM specific safety requirements. These would, in turn, allow us to refine the L1s, L2s and L3s of each CM.

Further Refinements

In addition to newer and/or refined safety requirements, the architecture can also evolve as follows:

- **Vehicle-To-Everything**: Currently, both the HA and Safe-AV do not consider V2X. As this will be a prominent part of an AV's functionality that will allow much more to be done in the plausibility checkers as well L1 functionality, both the HA and Safe-AV must be updated to consider V2X.

- **SAE Level 1, 2 and 3**: Both Safe-AV and our HA target SAE Level 4/5 vehicles and as such do not consider a driver in the loop. Designing an architecture for Levels 1, 2 and 3 presents its own set of challenges that would include considering an HMI that consists of mechanisms to decide and switch control between the human driver and the AV functionality. From the HA's point of view, this would add another controller in the control structure and result in more UCAs, causal factors, loss scenarios and safety requirements.

Chapter 7

Conclusion and Future Work

In this thesis, we presented a novel safety architecture, Safe-AV, for autonomous systems. Based on engineering best practices and built upon well adopted industry standards and concepts (Simplex Architecture – Section 2.9, E-Gas 3 Level Monitoring Concept – Section 2.8), the fault tolerant architecture aims to be the first step in a foundation for autonomous systems safety.

With any new type of technology, there is a stage in its life-cycle where the recommended best practices are still being explored, the standards still being written or defined and the laws and polices not yet enacted. It becomes apparent that current AVs are at that stage when we consider that some of the publicized accidents (Chapter 3) involving AVs that could have been predicted via modern Hazard Analysis techniques such as STPA (Chapter 4) and mitigated with standardized safety practices and system architectures. Others could have further been avoided with more responsible and non-negligent on-road testing (see Uber accident, Section 3.4).

We reviewed the current literature (Section 2.1) and found that most architectures focused less of safety and more on functionality. Typically being layered, these architecture would achieve a measured amount of safety by having a higher layer that,

in addition to meeting its own functional requirements, would also monitor the layer below it. This increase in the complexity of the higher layer violates a key principal in the design of dependable systems, separation of concerns - specifically, separation of control and safety. Safe-AV avoids this by separating safety and control into separate levels. This separation allows the design of simpler safety systems which, in turn, allow simpler safety arguments. Safe-AV also provides a means to monitor ML based systems. Typically, their complexity and (seemingly) non-deterministic nature makes it difficult to reason about their safety. Safe-AV deals with this by considering them as black-boxes and monitoring them for failures. Finally, it provides a path for graceful system degradation that allows the vehicle to be operational in the presence of critical failures and to bring itself to a safe state to reduce harm to human life and the environment.

In the process of designing Safe-AV, we started by analyzing current AV accidents to gain insight into how they can fail. This acted as a first step for the our STPA hazard analysis (Chapter 4) that was performed on a typical functional autonomous vehicle architecture. The resulting safety requirements partially drove the design of Safe-AV and evolved the initial architecture into the former. Finally, we provided a discussion (Chapter 6) on the merits of the architecture and how it met the various requirements obtained from the hazard analysis.

7.1 Future Work

The first iteration of the Safe-AV was presented in this work. Section 6.4 presented some paths towards the refinement and evolution of the architecture. In this section, we provide some further paths that not only refine the architecture but potentially change it significantly:

- **Implementation**: An implementation of the architecture was out of the scope of this work. However, before stronger claims about Safe-AV's usefulness and effectiveness can be made, an actual implementation is needed. The implementation would allow us to identify issues with the architecture's design that may not have been initially apparent. It would also identify issues that only present themselves during implementation time. Finally, it would test the architecture's feasibility and provide direction for its improvement.

- **Cyber Security**: Among the hurdles that need to be tackled before AVs can safely be deployed on a large scale is cyber security. A "hacked" vehicle with autonomous functionality gives malicious actors significant control that can have catastrophic consequences. We can already see instances of this with the hacked Jeep in [83] and the hacked Tesla in [84]. Safe-AV currently does not consider the cyber security aspect of AVs. As we cannot have a safe AV without it being secure, Safe-AV needs to be augmented from a security perspective.

- **Grey Box ML Observer**: The architecture currently considers ML based systems as black boxes. We do not attempt to reason about the internals of our Object Detection algorithms for instance and as a consequence, are limited to detecting faults **after** they have occurred. Based on the latest advancements in AI interpretability, Safe-AV can be updated to incorporate grey-box techniques (ML observer) that allow us to predict failures as they occur or before they occur.

Appendices

Appendix A

Remaining Causal Factors from Chapter 4

This Appendix contains the remaining UCAs, their causal factors and safety requirements from Chapter 4. As a reminder to the reader, an empty **Safety Requirements** column does not indicate that there are no safety requirements. For the sake of brevity, we have omitted rewriting the safety requirements if they have already been written in a previous table (in the digital version, the reader can click each causal factor to be taken to its safety requirements).

A.1 Braking Signal

UCA-B5: Braking signal stopped before coming to a full stop to avoid collision [H1, H2, H6]			
Controller Sends an Incorrect Command			
Scenario	**Causal Factors**	**Notes**	**Safety Requirements**
PAC assumes that vehicle has already stopped	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-10**, **CF-12**		
PAC assumes that vehicle no longer needs to come to a stop	**CF-1**, **CF-2**, **CF-3**, **CF-4**, **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**		
PAC assumes that vehicle is no longer able to break	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-10**, **CF-11**, **CF-12**, **CF-13**, **CF-14**, **CF-15**, **CF-16**		
Incorrect Execution			
Refer to **UCA-B1**			

Table A.1: UCA-B5

UCA-B6: Continued braking signal provided after reaching intended speed when slowing down [H1, H2, H6]			
Controller Sends an Incorrect Command			
Scenario	**Causal Factors**	**Notes**	**Safety Requirements**
PAC assumes that a collision is imminent	**CF-1**, **CF-4**, **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-22**		
PAC assumes that vehicle has not reached intended speed	**CF-25** MTW incorrectly reports road's speed limit **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-10**, **CF-12**, **CF-32**		
Incorrect Execution			
Refer to **UCA-B3**			

Table A.2: UCA-B6

A.2 Acceleration Signal

UCA-A1: Acceleration signal not provided to avoid collision [H1, H6]			
Controller Sends an Incorrect Command			
Scenario	**Causal Factors**	**Rationale**	**Safety Requirements**
PAC is not aware of the imminent collision	**CF-1**, **CF-2**, **CF-3**, **CF-4**, **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**		
PAC assumes that vehicle is already accelerating or already at the desired speed	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-10**, **CF-11**		
PAC assumes that vehicle is unable to accelerate	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-10**, **CF-12**, **CF-13**, **CF-14**, **CF-15** **CF-26** MTW incorrectly reports that the vehicle's actuators (e.g. motors) have failed		
Incorrect Execution			
TBW system generates an incorrect value or does not generate one	**CF-17**, **CF-18**		**SR-13**, **SR-14**, **SR-15**, **SR-16**, **SR-19**, **SR-20**
TBW system provides the correct value but ego vehicle does not accelerate	**CF-29** Physical failure in the throttle body, transmission or other components required for acceleration **CF-19**, **CF-20**, **CF-21**		**SR-13**, **SR-14**, **SR-15**, **SR-16**, **SR-19**, **SR-20**, **SR-21**

Table A.3: UCA-A1

UCA-A2: Acceleration signal not provided as needed when changing or merging into fast lane [H1, H6]			
Controller Sends an Incorrect Command			
Scenario	**Causal Factors**	**Rationale**	**Safety Requirements**
PAC assumes that vehicle is already accelerating or already at the desired speed	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-10**, **CF-11**		
PAC assumes that vehicle is unable to accelerate	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-10**, **CF-12**, **CF-13**, **CF-14**, **CF-15**, **CF-26**		
PAC assumes that vehicle is already in the desired lane	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-22**, **CF-28**, **CF-32**		
PAC assumes that there isn't a faster vehicle in the desired lane	**CF-1**, **CF-2**, **CF-3**, **CF-4**, **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-28**		
Incorrect Execution			
Refer to **UCA-B1**			

Table A.4: UCA-A2

UCA-A3: Acceleration signal not provided to obtain desired speed [H1, H6]			
Controller Sends an Incorrect Command			
Scenario	**Causal Factors**	**Rationale**	**Safety Requirements**
PAC assumes that vehicle is already accelerating or already at the desired speed	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-10**, **CF-11**		
PAC assumes that vehicle is unable to accelerate	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-10**, **CF-12**, **CF-13**, **CF-14**, **CF-15**, **CF-26**		
Incorrect Execution			
Refer to **UCA-B1**			

Table A.5: UCA-A3

UCA-A4: Acceleration signal not provided to leave hazardous zone (flooded roadway, desert) [H1]			
Controller Sends an Incorrect Command			
Scenario	**Causal Factors**	**Rationale**	**Safety Requirements**
PAC is not aware of the hazardous zone	**CF-1**, **CF-2**, **CF-3**, **CF-4**, **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**		
PAC assumes that vehicle is already accelerating or already at the desired speed	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-10**, **CF-11**		
PAC assumes that vehicle is unable to accelerate	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-10**, **CF-12**, **CF-13**, **CF-14**, **CF-15**, **CF-26**		
PAC assumes that vehicle is already out of the hazardous zone	**CF-1**, **CF-3**, **CF-4**, **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-22**		
Incorrect Execution			
Refer to **UCA-B1**			

Table A.6: UCA-A4

UCA-A5: Acceleration signal provided when there is a stopped vehicle or object ahead [H1, H6]			
Controller Sends an Incorrect Command			
Scenario	**Causal Factors**	**Rationale**	**Safety Requirements**
PAC assumes that stopped traffic has cleared or there is none	**CF-1**, **CF-2**, **CF-3**, **CF-4**, **CF-5**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-23**		**SR-25** PAC must delay an acceleration commands by TBD seconds after an object in its path disappears suddenly
PAC assumes that a collision is imminent	**CF-1**, **CF-4**, **CF-5**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-14**, **CF-23**		
PAC assumes that ego vehicle's is in a hazardous zone	**CF-1**, **CF-4**, **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-22**, **CF-23**		
Incorrect Execution			
TBW generates an incorrect value	Refer to **UCA-A1**		

Table A.7: UCA-A5

UCA-A6: Acceleration signal provided when violating minimum safe distance from other vehicle/object [H1, H6]			
Controller Sends an Incorrect Command			
Scenario	**Causal Factors**	**Rationale**	**Safety Requirements**
PAC assumes that vehicle is not violating safe distance	**CF-1**, **CF-2**, **CF-3**, **CF-4**, **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-40**		
PAC assumes that a collision is imminent	**CF-1**, **CF-2**, **CF-3**, **CF-4**, **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**		
PAC assumes that ego vehicle is in a hazardous zone	**CF-1**, **CF-4**, **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-22**, **CF-23**		
Incorrect Execution			
TBW generates an incorrect value	Refer to **UCA-A1**		

Table A.8: UCA-A6

UCA-A7: Acceleration signal provided too late to avoid collision [H1, H6]			
Controller Sends an Incorrect Command			
Scenario	**Causal Factors**	**Rationale**	**Safety Requirements**
PAC does not react to object in time	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-24**, **CF-40**		
Refer to **UCA-A1**			
Incorrect Execution			
Refer to **UCA-A1**			

Table A.9: UCA-A7

UCA-A8: Acceleration signal stopped before collision avoidance manoeuvre is complete [H1, H6]			
Controller Sends an Incorrect Command			
Scenario	**Causal Factors**	**Rationale**	**Safety Requirements**
PAC assumes that vehicle has already reached the required acceleration to avoid the collision	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-10**, **CF-11**		
PAC assumes that vehicle has already avoided the collision	**CF-1**, **CF-2**, **CF-3**, **CF-4**, **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-10**, **CF-11**, **CF-22**		
PAC assumes that vehicle is no longer able to accelerate	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-10**, **CF-11**, **CF-12**, **CF-13**, **CF-14**, **CF-15**, **CF-26**		
Incorrect Execution			
Refer to **UCA-A1**			

Table A.10: UCA-A8

UCA-A9: Continued acceleration signal after reaching intended speed with vehicles/objects ahead [H1, H6]			
Controller Sends an Incorrect Command			
Scenario	**Causal Factors**	**Rationale**	**Safety Requirements**
PAC assumes that a collision is imminent	**CF-1**, **CF-2**, **CF-3**, **CF-4**, **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**		
PAC assumes that vehicle has not reached intended speed	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-10**, **CF-11**, **CF-25**		
Incorrect Execution			
Refer to **UCA-A6**			

Table A.11: UCA-A9

A.3 Steering Signal

UCA-S1: Steering signal not provided to avoid collision [H3, H6]			
Controller Sends an Incorrect Command			
Scenario	**Causal Factors**	**Rationale**	**Safety Requirements**
PAC is not aware of the imminent collision	**CF-1**, **CF-2**, **CF-3**, **CF-4**, **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**		
PAC assumes that vehicle is already at the correct heading	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-22**		
PAC assumes that vehicle is unable to steer	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-10**, **CF-14**, **CF-15** **CF-27** MTW incorrectly reports that the vehicle's steering mechanisms have failed		
Incorrect Execution			
Steer-By-Wire (SBW) system generates an incorrect value or does not generate one	**CF-17**, **CF-18**		**SR-13**, **SR-14**, **SR-15**, **SR-16**, **SR-19**, **SR-20**
SBW system provides the correct value but ego vehicle's wheels do not turn	**CF-30** Physical failure in the steering components **CF-21**		**SR-13**, **SR-14**, **SR-15**, **SR-16**, **SR-19**, **SR-20**, **SR-21**

Table A.12: UCA-S1

UCA-S2: Steering signal not provided as needed when merging or changing lane or to follow road curve [H3, H6]			
Controller Sends an Incorrect Command			
Scenario	**Causal Factors**	**Rationale**	**Safety Requirements**
PAC assumes that vehicle is already at the correct heading	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-22**		
PAC assumes that the vehicle is already at desired or location lane/location	**CF-28** MTW incorrectly reports lane markers **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-22**, **CF-32**		
PAC assumes that vehicle is unable to steer	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-10**, **CF-14**, **CF-15**, **CF-27**		
Incorrect Execution			
Refer to **UCA-S1**			

Table A.13: UCA-S2

UCA-S3: Steering signal provided when attempting to go straight [H3, H6]			
Controller Sends an Incorrect Command			
Scenario	**Causal Factors**	**Rationale**	**Safety Requirements**
PAC assumes that vehicle is not in the correct position in the lane or is going out of the lane	**CF-4**, **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-22**, **CF-28**		
PAC assumes that a collision is imminent	**CF-1**, **CF-2**, **CF-3**, **CF-4**, **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**		
PAC assumes that vehicle is in a hazardous zone	**CF-1**, **CF-4**, **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-22**, **CF-23**		
Incorrect Execution			
SBW generates an incorrect steering value	**CF-17**, **CF-18**, **CF-30**		

Table A.14: UCA-S3

UCA-S4: Steering signal provided too early when merging or changing lane [H3, H6]			
Controller Sends an Incorrect Command			
Scenario	**Causal Factors**	**Rationale**	**Safety Requirements**
PAC assumes that the desired lane is clear	**CF-1**, **CF-2**, **CF-3**, **CF-4**, **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-28**, **CF-32**		
PAC assumes that a collision is imminent	**CF-1**, **CF-2**, **CF-3**, **CF-4**, **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**		
Incorrect Execution			
Refer to **UCA-S3**			

Table A.15: UCA-S4

UCA-S5: Steering signal provided too late when merging or changing lane [H3, H6]			
Controller Sends an Incorrect Command			
Scenario	**Causal Factors**	**Rationale**	**Safety Requirements**
PAC does not react to lane change in time	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-24**, **CF-28**, **CF-40**		
PAC assumes that there is a vehicle in the desired lane	**CF-1**, **CF-2**, **CF-3**, **CF-4**, **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-23**, **CF-28**, **CF-32**		
Incorrect Execution			
Refer to **UCA-S3**			

Table A.16: UCA-S5

UCA-S6: Steering signal provided too late to avoid collision [H3, H6]			
Controller Sends an Incorrect Command			
Scenario	**Causal Factors**	**Rationale**	**Safety Requirements**
PAC reacts to accident too late	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-24**, **CF-40**		
Refer to **UCA-S1**			
Incorrect Execution			
Refer to **UCA-S1**			

Table A.17: UCA-S6

UCA-S7: Steering signal provided before acceleration signal when merging ahead of a slow vehicle [H3, H6]			
Controller Sends an Incorrect Command			
Scenario	**Causal Factors**	**Rationale**	**Safety Requirements**
PAC assumes that lane is clear	**CF-1**, **CF-2**, **CF-3**,**CF-4**, **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-28**, **CF-40**		
PAC assumes that vehicle is about to enter a hazardous zone (i.e. into a wall)	**CF-1**, **CF-4**, **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-22**, **CF-23**		
Incorrect Execution			
TBW system produces an incorrect value	**CF-17**, **CF-18**, **CF-21**		
SBW system produces an incorrect value	Refer to **UCA-S1**		

Table A.18: UCA-S7

UCA-S8: Steering signal stopped before curve following or merging or lane change is complete [H3, H6]			
Controller Sends an Incorrect Command			
Scenario	**Causal Factors**	**Rationale**	**Safety Requirements**
PAC assumes that the vehicle is already in desired lane or desired position in lane	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-22**, **CF-28**, **CF-32**		
PAC assumes that the vehicle is unable to steer	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-10**, **CF-14**, **CF-15**, **CF-27**		
PAC assumes that a collision is imminent	**CF-1**, **CF-2**, **CF-3**, **CF-4**, **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**		
Incorrect Execution			
Refer to **UCA-S1**			

Table A.19: UCA-S8

UCA-S9: Continued steering signal provided after curve following or merging or lane change is complete [H3, H6]			
Controller Sends an Incorrect Command			
Scenario	**Causal Factors**	**Rationale**	**Safety Requirements**
PAC assumes that vehicle is not in desired lane or desired position in lane	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-22**, **CF-28**, **CF-32**		
PAC assumes that a collision is imminent	**CF-1**, **CF-2**, **CF-3**, **CF-4**, **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**		
Incorrect Execution			
SBW system generates an incorrect value	Refer to **UCA-S1**		

Table A.20: UCA-S9

A.4 Gear Selection Signal

UCA-GS1: Gear selection signal set to reverse or park when vehicle is moving forward [H1, H2, H5, H6]			
Controller Sends an Incorrect Command			
Scenario	**Causal Factors**	**Rationale**	**Safety Requirements**
PAC assumes that the vehicle is in the incorrect gear so it attempts to set the correct gear	**CF-1**, **CF-4**, **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-22** **CF-31** MTW reports incorrect current vehicle gear selection		
PAC assumes that the vehicle is going backwards or in park	**CF-1**, **CF-4**, **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-10**, **CF-11**, **CF-12 CF-22**, **CF-31**		
PAC assumes that vehicle is intended to go backwards or in park	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-10**, **CF-11**, **CF-12**, **CF-22**, **CF-31** **CF-32** MTW provides an incorrect map of the area		
Incorrect Execution			
SBW system generates an incorrect value	**CF-17**, **CF-18**		**SR-13**, **SR-14**, **SR-15**, **SR-16**, **SR-19**, **SR-20**
SBW systems generates the correct value but the transmission does not switch modes	**CF-37** Physical failure in the shifting related components (e.g. transmission) **CF-21**		**SR-13**, **SR-14**, **SR-15**, **SR-16**, **SR-19**, **SR-20**, **SR-21**

Table A.21: UCA-GS1

UCA-GS2: Gear selection signal set to forward or reverse when vehicle is intended to be parked [H1, H2, H5, H6]			
Controller Sends an Incorrect Command			
Scenario	**Causal Factors**	**Rationale**	**Safety Requirements**
PAC assumes that the vehicle is in the incorrect gear so it attempts to set the correct gear	**CF-1**, **CF-4**, **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-22**, **CF-31**		
PAC assumes that the vehicle is going forwards or in park	**CF-1**, **CF-4**, **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-10**, **CF-11**, **CF-12 CF-22**, **CF-31**		
PAC assumes that vehicle is intended to go forwards or in park	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-10**, **CF-11**, **CF-12**, **CF-22**, **CF-31**, **CF-32**		
Incorrect Execution			
Refer to **UCA-GS1**			

Table A.22: UCA-GS2

A.5 Indication Signal

UCA-I1: Turn signal not provided when merging or changing lane [H4, H6]			
Controller Sends an Incorrect Command			
Scenario	**Causal Factors**	**Rationale**	**Safety Requirements**
PAC assumes that the turn signal is already on	**CF-33** MTW incorrectly reports vehicle's indication signal values **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**		
PAC assumes that ego vehicle is already in the desired lane	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-28**, **CF-32**, **CF-22**		
PAC assumes that ego vehicle's is unable to use its turn signals	**CF-34** MTW incorrectly reports vehicle's indication systems have failed or are physically compromised (e.g. broken bulbs) **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-33**		
Incorrect Execution			
The indication system generates an incorrect value or does not generate one	**CF-17**, **CF-18**		**SR-13**, **SR-14**, **SR-15**, **SR-16**, **SR-19**, **SR-20**
The indication system provides the correct value but ego vehicle does not turn on/off bulb	**CF-35**: Physical failure in the indication system (broken bulbs, faulty wiring, exposed wiring, etc.) **CF-21**		**SR-13**, **SR-14**, **SR-15**, **SR-16**, **SR-19**, **SR-20**, **SR-21**

Table A.23: UCA-I1

UCA-I2: Identification light (headlamps, tail lamps, side marker lights) signal not provided during dark or low visibility conditions [H4, H6]			
Controller Sends an Incorrect Command			
Scenario	**Causal Factors**	**Rationale**	**Safety Requirements**
PAC assumes that the turn identification lights are already on	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-33**		
PAC assumes that dark or low visibility conditions are not present	**CF-36** MTW incorrectly reports the environment's conditions (e.g luminosity, presence of fog, etc.) **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-22**, **CF-32**		
PAC assumes that ego vehicle's is unable to use its indication lights	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-32**, **CF-33**, **CF-34**		
Incorrect Execution			
Refer to **UCA-I1**			

Table A.24: UCA-I2

UCA-I3: Braking light signal not provided when vehicle is braking [H4, H6]			
Controller Sends an Incorrect Command			
Scenario	**Causal Factors**	**Rationale**	**Safety Requirements**
PAC assumes that the braking lights are already on	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-33**		
PAC assumes that vehicle is not braking	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-10**, **CF-11**, **CF-12**, **CF-15**, **CF-16**, **CF-22**		
PAC assumes that ego vehicle's is unable to use its braking lights	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-32**, **CF-33**, **CF-34**		
Incorrect Execution			
Refer to **UCA-I1**			

Table A.25: UCA-I3

UCA-I4: Left (Right) turn signal provided when merging or changing to the Right (Left) [H4, H6]			
Controller Sends an Incorrect Command			
Scenario	**Causal Factors**	**Rationale**	**Safety Requirements**
PAC assumes that vehicle is turning Left(Right) when it is turning Right(Left)	**CF-38** MTW reports incorrect vehicle orientation **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-22**, **CF-33**		
Incorrect Execution			
The indication system provides the correct value but ego vehicle reverses the turn signal bulbs	**CF-21**, **CF-35**		

Table A.26: UCA-I4

UCA-I5: Turn signal provided too late when merging or changing lane [H4, H6]			
Controller Sends an Incorrect Command			
Scenario	**Causal Factors**	**Rationale**	**Safety Requirements**
Refer to **UCA-I1**			
PAC does not react to lane change in time	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-24**, **CF-28**		
PAC assumes that there is a vehicle in the desired lane	**CF-1**, **CF-2**, **CF-3**, **CF-4**, **CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-23**, **CF-28**, **CF-32**		
Incorrect Execution			
Refer to **UCA-I1**			

Table A.27: UCA-I5

UCA-I6: Turn signal not turned off after merge or lane change complete [H4, H6]			
Controller Sends an Incorrect Command			
Scenario	**Causal Factors**	**Rationale**	**Safety Requirements**
PAC assumes that vehicle has not completed the merge/lane change	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-28**, **CF-32**, **CF-22**, **CF-36**		
Incorrect Execution			
Refer to **UCA-I1**			

Table A.28: UCA-I6

UCA-I7: Turn signal turned off before merge or lane change complete [H4, H6]			
Controller Sends an Incorrect Command			
Scenario	**Causal Factors**	**Rationale**	**Safety Requirements**
PAC assumes that the vehicle is already in desired lane	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-28**, **CF-32**, **CF-22**		
PAC assumes that the vehicle is unable to use its turn signals	**CF-5**, **CF-6**, **CF-7**, **CF-8**, **CF-9**, **CF-39**, **CF-33**, **CF-34**		
Incorrect Execution			
Refer to **UCA-I1**			

Table A.29: UCA-I7

Appendix B

Initial Unsafe Control Actions (IUCA)

Not Providing Causes Hazard	**Providing Causes Hazard**	**Wrong Timing/Order Causes Hazard**	**Stopping Too Soon/Applying Too Long Causes Hazard**
IUCA-I1 Turn signal not provided when merging/changing lane or turning [H4, H6]	***IUCA-I5*** Left (Right) turn signal provided when turning or merging/changing to the Right (Left) [H4, H6]	***IUCA-I6*** Turn signal provided too late when merging/changing lane or turning [H4, H6]	***IUCA-I8*** Turn signal not turned off after merge/lane change or turn complete [H4, H6]
IUCA-I2 Headlight signal not provided during [H4, H6]		***IUCA-I7*** Turn signal provided too early when merging/changing lane or turning [H4, H6]	***IUCA-I9*** Turn signal turned off before merge/lane change or turn complete [H4, H6]
IUCA-I3 Headlamps signal not provided when merging/changing lane or turning [H4, H6]			
IUCA-I4 Turn signal not provided when moving into different lane during collision avoidance [H4, H6]			

Table B.1: IUCA - Indication Signal

Not Providing Causes Hazard	**Providing Causes Hazard**	**Wrong Timing/Order Causes Hazard**	**Stopping Too Soon/Applying Too Long Causes Hazard**
IUCA-B1 Braking signal not provided to avoid collision [H1, H2, H6] ***IUCA-B2*** Braking signal not provided as needed when merg-ing/changing into slow lane or turning [H1, H2, H6] ***IUCA-B3*** Braking signal not provided to stop at red light/stop sign [H1, H2, H6]	***IUCA-B4*** Unexpected or unneeded braking signal provided when on road with traffic [H1, H2, H6, RH3] ***IUCA-B5*** Braking signal provided when crossing oncoming lane while turning [H1, H2, H6]	***IUCA-B6*** Braking signal provided too early when approaching a red light/stop sign [H1, H2, H6] ***IUCA-B7*** Braking signal provided too late when stopping at stop sign/red light [H1, H2, H6] ***IUCA-B8*** Braking signal provided too late to avoid collision [H1, H2, H6] ***IUCA-B9*** Reverse Gear command provided before braking signal when switching to reverse [H1, H2, H5] ***IUCA-B10*** Acceleration signal provided before braking when approaching a speed-bump [H1, H2, RH3]	***IUCA-B11*** Braking signal stopped before vehicle comes to a complete stop at a red light/stop sign [H1, H2, H6] ***IUCA-B12*** Braking signal stopped before vehicle comes to a complete stop to avoid collision [H1, H2, H6] ***IUCA-B13*** Continued braking signal provided after reaching intended speed when slowing down [H1, H2, H6]

Table B.2: IUCA - Braking Signal

Not Providing Causes Hazard	**Providing Causes Hazard**	**Wrong Timing/Order Causes Hazard**	**Stopping Too Soon/Applying Too Long Causes Hazard**
IUCA-A1 Acceleration signal not provided to avoid collision [H1, H6]	***IUCA-A5*** Acceleration signal provided during stopped traffic/red light/stop sign [H1, H6]	***IUCA-A9*** Acceleration signal provided before red light turns to green. [H1, H6]	***IUCA-A13*** Acceleration signal stopped before collision avoidance manner is complete. [H1, H6]
IUCA-A2 Acceleration signal not provided as needed when changing/merging into fast lane or turning [H1, H6]	***IUCA-A6*** Acceleration signal provided when making a turn across oncoming lane that is not clear. [H1, H6]	***IUCA-A10*** Acceleration signal provided too late to avoid collision. [H1, H6]	***IUCA-A14*** Continued acceleration signal after reaching speed limit with car/object in front. [H1, H6]
IUCA-A3 Acceleration signal stopped when on a busy road (loss of propulsion). [H1, H6]	***IUCA-A7*** Sudden excessive acceleration signal provided [H1, RH3]	***IUCA-A11*** Acceleration signal provided before braking when making a turn or approaching a red light [H1, H2, H6]	***IUCA-A15*** Continued deceleration after slowing down to intended speed. [H1, H6]
IUCA-A4 Acceleration signal not provided to leave hazardous zone (flooded roadway, desert) [H1, RH2, RH3]	***IUCA-A8*** Acceleration provided when approaching hazardous zone [H1, RH1, RH2]	***IUCA-A12*** Steering signal provided before acceleration signal when merging into an occupied lane [H1, H3, H6]	***IUCA-A16*** Acceleration signal stopped too soon when merging/changing lane or turning [H1, H6]

Table B.3: IUCA - Acceleration Signal

Not Providing Causes Hazard	**Providing Causes Hazard**	**Wrong Timing/Order Causes Hazard**	**Stopping Too Soon/Applying Too Long Causes Hazard**
IUCA-S1 Steering signal not provided to avoid collision [H3, H6]	***IUCA-S4*** Steering signal provided when attempting to go straight [H3, H6]	***IUCA-S6*** Steering signal provided too early when merging/changing lane or turning [H3, H6]	***IUCA-S9*** Steering signal stopped before curve following or turning or merging/lane change is complete [H3, H6]
IUCA-S2 Steering signal not provided to keep vehicle centred in lane (e.g. when vehicle "pulls" to a side or road curves) [H3, H6]	***IUCA-S5*** Steering signal provided into busy oncoming lane [H3, H6]	***IUCA-S7*** Steering signal provided too late when merging/changing lane or turning [H3, H6]	***IUCA-S10*** Continued steering signal provided after curve following or turning or merging/lane change is complete [H3, H6]
IUCA-S3 Steering signal not provided as needed when merging/changing lane or turning [H3, H6]		***IUCA-S8*** Steering signal provided too late to avoid collision [H3, H6]	

Table B.4: IUCA - Steering Signal

Not Providing Causes Hazard	**Providing Causes Hazard**	**Wrong Timing/Order Causes Hazard**	**Stopping Too Soon/Applying Too Long Causes Hazard**
IUCA-GS1 Gear selection signal not provided as needed to avoid collision. [H1, H2, H5, H6, RH1, RH3]	***IUCA-GS3*** Gear selection signal set to forward/park when car is going backwards [H1, H2, H5, H6, RH1, RH3]	—	—
IUCA-GS2 Gear selection signal not set to Park/Drive/Reverse when car is intended to be parked/go forward/go backward. [H1, H2, H5, H6, RH1, RH3]	***IUCA-GS4*** Gear selection signal set to reverse/park when vehicle is moving forward [H1, H2, H5, H6, RH1, RH3]		
	IUCA-GS5 Gear selection signal set to forward/reverse when car is intended to be parked [H1, H2, H5, H6, RH1, RH3]		

Table B.5: IUCA - Gear Selection Signal

Bibliography

[1] *ISO 26262 – Road Vehicles Functional Safety. Road vehicles - Functional safety.* ISO 26262:2011. Standard Document No. International Organization for Standardization/Technical Committee 22 (ISO/TC 22). Geneva, Switzerland, Nov. 2011 (cit. on pp. 2, 3, 14, 25).

[2] World Health Organization. *Global Status Report on Road Safety 2015.* Nonserial Publication. World Health Organization, 2015. ISBN: 9789241565066. URL: http://www.who.int/violence_injury_prevention/road_safety_status/2015/en/ (cit. on p. 3).

[3] S Singh. "Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey (No. DOT HS 812 115)". In: *Google Scholar* (2015). URL: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812115 (visited on 02/08/2018) (cit. on p. 3).

[4] Michele Bertoncello. *Ten ways autonomous driving could redefine the automotive world.* 2015. URL: https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/ten-ways-autonomous-driving-could-redefine-the-automotive-world (visited on 02/08/2018) (cit. on p. 3).

[5] Insurance Institute for Highway Safety. *IIHS Status Report newsletter, Vol. 51, No. 1, January 28, 2016*. https://www.iihs.org/externaldata/srdata/docs/sr5101.pdf. (Accessed on 12/04/2018). Jan. 2016 (cit. on p. 4).

[6] Daniel Jackson and Eunsuk Kang. "Separation of concerns for dependable software design". In: *Proceedings of the FSE/SDP workshop on Future of software engineering research*. ACM. 2010, pp. 173–176 (cit. on pp. 7, 14).

[7] Panos J Antsaklis, Kevin M Passino, and SJ Wang. "Towards intelligent autonomous control systems: Architecture and fundamental issues". In: *Journal of Intelligent and Robotic Systems* 1.4 (1989), pp. 315–342 (cit. on p. 11).

[8] Erann Gat, R Peter Bonnasso, Robin Murphy, et al. "On three-layer architectures". In: *Artificial intelligence and mobile robots* 195 (1998), p. 210 (cit. on p. 11).

[9] R Peter Bonasso et al. "Experiences with an architecture for intelligent, reactive agents". In: *Journal of Experimental & Theoretical Artificial Intelligence* 9.2-3 (1997), pp. 237–256 (cit. on p. 12).

[10] Michael L Nelson. "A design pattern for autonomous vehicle software control architectures". In: *Computer Software and Applications Conference, 1999. COMPSAC'99. Proceedings. The Twenty-Third Annual International*. IEEE. 1999, pp. 172–177 (cit. on p. 13).

[11] SJ Corfield, RJC Fraser, and CJ Harris. "Architecture for real-time intelligent control of autonomous vehicles". In: *Computing & Control Engineering Journal* 2.6 (1991), pp. 254–262 (cit. on p. 13).

[12] Jun Miura, Motokuni Ito, and Yoshiaki Shirai. "A three-level control architecture for autonomous vehicle driving in a dynamic and uncertain traffic environ-

ment". In: *Intelligent Transportation System, 1997. ITSC'97., IEEE Conference on*. IEEE. 1997, pp. 706–711 (cit. on p. 13).

[13] Michelle Bayouth, Illah Nourbakhsh, and Charles Thorpe. "A hybrid human-computer autonomous vehicle architecture". In: *Proceedings, Third ECPD International Conference on Advanced Robotics, Intelligent Automation and Control*. 1997 (cit. on p. 13).

[14] Sasayuki Tsugawa et al. "An architecture for cooperative driving of automated vehicles". In: *Intelligent Transportation Systems, 2000. Proceedings. 2000 IEEE*. IEEE. 2000, pp. 422–427 (cit. on p. 13).

[15] Philip Koopman and Michael Wagner. "Challenges in Autonomous Vehicle Testing and Validation". In: *SAE Int. J. Trans. Safety* 4 (Apr. 2016), pp. 15–24. URL: https://doi.org/10.4271/2016-01-0128 (cit. on pp. 13, 14, 19, 26).

[16] Alan Wassyng, Mark Lawford, and Tom Maibaum. "Separating Safety and Control Systems to Reduce Complexity". In: *Conquering Complexity*. Springer, 2012, pp. 85–102 (cit. on p. 14).

[17] Lui Sha. "Using simplicity to control complexity". en. In: *IEEE Software* 18.4 (July 2001), pp. 20–28. ISSN: 0740-7459. URL: http://ieeexplore.ieee.org/document/936213/ (cit. on pp. 14, 34).

[18] Kichun Jo et al. "Development of autonomous car—Part II: A case study on the implementation of an autonomous driving system based on distributed architecture". In: *IEEE Transactions on Industrial Electronics* 62.8 (2015), pp. 5119–5132 (cit. on p. 15).

[19] Lars B Cremean et al. "Alice: An information-rich autonomous vehicle for high-speed desert navigation". In: *Journal of Field Robotics* 23.9 (2006), pp. 777–810 (cit. on p. 15).

[20] John Leonard et al. "A perception-driven autonomous urban vehicle". In: *Journal of Field Robotics* 25.10 (2008), pp. 727–774 (cit. on p. 15).

[21] Christian Berger and Bernhard Rumpe. "Autonomous driving-5 years after the urban challenge: The anticipatory vehicle as a cyber-physical system". In: *arXiv preprint arXiv:1409.0413* (2014) (cit. on p. 15).

[22] Junsung Kim, Ragunathan Raj Rajkumar, and Markus Jochim. "Towards dependable autonomous driving vehicles: a system-level approach". In: *ACM SIGBED Review* 10.1 (2013), pp. 29–32 (cit. on p. 15).

[23] Julius Ziegler et al. "Making Bertha Drive-An Autonomous Journey on a Historic Route." In: *IEEE Intell. Transport. Syst. Mag.* 6.2 (2014), pp. 8–20 (cit. on pp. 15, 16).

[24] Ömer Şahin Taş et al. "Functional system architectures towards fully automated driving". In: *Intelligent Vehicles Symposium (IV), 2016 IEEE*. IEEE. 2016, pp. 304–309 (cit. on p. 15).

[25] Simon Ulbrich et al. "Towards a functional system architecture for automated vehicles". In: *arXiv preprint arXiv:1703.08557* (2017) (cit. on pp. 16, 17).

[26] Benoit Vanholme et al. "Highly automated driving on highways based on legal safety". In: *IEEE Transactions on Intelligent Transportation Systems* 14.1 (2013), pp. 333–347 (cit. on p. 16).

[27] Felix Lotz. "System architectures for automated vehicle guidance concepts". In: *Automotive Systems Engineering*. Springer, 2013, pp. 39–61 (cit. on p. 16).

[28] Sagar Behere and Martin Torngren. "A functional architecture for autonomous driving". In: *Automotive Software Architecture (WASA), 2015 First International Workshop on*. IEEE. 2015, pp. 3–10 (cit. on p. 16).

[29] Shaoshan Liu et al. "CAAD: Computer Architecture for Autonomous Driving". In: *arXiv preprint arXiv:1702.01894* (2017) (cit. on p. 16).

[30] Assia Belbachir. "An embedded testbed architecture to evaluate autonomous car driving". In: *Intelligent Service Robotics* 10.2 (2017), pp. 109–119 (cit. on p. 16).

[31] Changliu Liu et al. *The robustly-safe automated driving system for enhanced active safety*. Tech. rep. SAE Technical Paper, 2017 (cit. on p. 16).

[32] Wenhao Zong et al. "Architecture Design and Implementation of an Autonomous Vehicle". In: *IEEE Access* 6 (2018), pp. 21956–21970 (cit. on p. 16).

[33] Richard Matthaei and Markus Maurer. "Functional System Architecture for an Autonomous on-Road Motor Vehicle". In: *Automotive Systems Engineering II*. Springer, 2018, pp. 93–120 (cit. on p. 16).

[34] Andreas Reschka et al. "A surveillance and safety system based on performance criteria and functional degradation for an autonomous vehicle". In: *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*. IEEE. 2012, pp. 237–242 (cit. on pp. 16, 17).

[35] Satoshi Otsuka and Kohei Sakurai. *A Safety Concept based on a Safety Sustainer for Highly Automated Driving Systems*. Tech. rep. SAE Technical Paper, 2016 (cit. on p. 18).

[36] Ömer Şahin Taş et al. "Automated vehicle system architecture with performance assessment". In: *Intelligent Transportation Systems (ITSC), 2017 IEEE 20th International Conference on*. IEEE. 2017, pp. 1–8 (cit. on p. 18).

[37] Alexandru Constantin Serban, Erik Poll, and Joost Visser. "A Standard Driven Software Architecture for Fully Autonomous Vehicles". In: (2018) (cit. on p. 18).

[38] *open robinos specification 1.1.0*. Elektrobit. 2017. URL: https://www.elektrobit.com/products/eb-robinos/ (cit. on p. 18).

[39] Markus Hörwick and Karl-Heinz Siedersberger. "Strategy and architecture of a safety concept for fully automatic and autonomous driving assistance systems". In: *Intelligent Vehicles Symposium (IV), 2010 IEEE*. IEEE. 2010, pp. 955–960 (cit. on pp. 18, 19).

[40] Manel Brini et al. "Risk reduction of experimental autonomous vehicles: The Safety-Bag approach". In: *CARS 2016 4th International Workshop on Critical Automotive Applications: Robustness & Safety*. 2016 (cit. on p. 18).

[41] Philip Koopman and Michael D Wagner. "Transportation CPS Safety Challenges". In: *NSF Workshop on Transportation CyberPhysical Systems*. Jan. 2014 (cit. on p. 19).

[42] Dario Amodei et al. "Concrete problems in AI safety". In: *arXiv preprint arXiv:1606.06565* (2016) (cit. on p. 20).

[43] Rich Caruana et al. "Case-based explanation of non-case-based learning methods." In: *Proceedings of the AMIA Symposium*. American Medical Informatics Association. 1999, p. 212 (cit. on p. 20).

[44] Zachary C Lipton et al. "The Mythos of Model Interpretability". In: *IEEE Spectrum* (2016) (cit. on p. 20).

[45] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *arXiv preprint arXiv:1602.04938* (2016) (cit. on p. 20).

[46] Tomas Olsson et al. "Case-based reasoning for explaining probabilistic machine learning". In: *International Journal of Computer Science & Information Technology* 6.2 (2014), p. 87 (cit. on p. 20).

[47] Tomas Olsson et al. "Explaining Probabilistic Fault Diagnosis and Classification Using Case-Based Reasoning". In: *Case-Based Reasoning Research and Development: 22nd International Conference, ICCBR 2014, Cork, Ireland, September 29, 2014 - October 1, 2014. Proceedings*. Cham: Springer International Publishing, 2014, pp. 360–374. ISBN: 978-3-319-11209-1. URL: http://dx.doi.org/10.1007/978-3-319-11209-1_26 (cit. on p. 20).

[48] Peng Zhang et al. "Predicting failures of vision systems". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 3566–3573 (cit. on p. 20).

[49] Aayush Bansal, Ali Farhadi, and Devi Parikh. "Towards transparent systems: Semantic characterization of failure modes". In: *European Conference on Computer Vision*. Springer. 2014, pp. 366–381 (cit. on p. 20).

[50] NG Leveson and John Thomas. *STPA Handbook*. 2018 (cit. on pp. 20, 22–24, 65).

[51] N. G. Leveson. *Engineering a Safer World: Systems Thinking Applied to Safety (Engineering Systems)*. Cambridge, Massachusetts: The MIT Press, 2012. ISBN: 0471846805 (cit. on p. 21).

[52] Qi D Van Eikema Hommes. "Applying System Theoretical Hazard Analysis Method to Complex Automotive Cyber Physical Systems". In: *ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers. 2012, pp. 705–717 (cit. on p. 24).

[53] Archana Mallya et al. "Using STPA in an ISO 26262 compliant process". In: *International Conference on Computer Safety, Reliability, and Security*. Springer. 2016, pp. 117–129 (cit. on p. 24).

[54] Rodrigo Sotomayor Martínez. "System theoretic process analysis of electric power steering for automotive applications". PhD thesis. Massachusetts Institute of Technology, 2015 (cit. on p. 24).

[55] Christopher Becker, Ahmad Nasser, Fouad Attioui, et al. *Functional Safety Assessment of a Generic Accelerator Control System With Electronic Throttle Control in Fuel Cell Hybrid Electric Vehicles*. Tech. rep. United States. Department of Transportation. National Highway Traffic Safety ..., 2018 (cit. on p. 24).

[56] Sardar Muhammad Sulaman et al. "Hazard analysis of collision avoidance system using STPA." In: *ISCRAM*. 2014 (cit. on p. 24).

[57] Giedre Sabaliauskaite, Lin Shen Liew, and Jin Cui. "Integrating autonomous vehicle safety and security analysis using STPA method and the Six-Step model". In: *International Journal on Advances in Security* 11 (2018), pp. 160–169 (cit. on p. 25).

[58] Asim Abdulkhaleq et al. "Missing no Interaction—Using STPA for Identifying Hazardous Interactions of Automated Driving Systems". In: *International Journal of Safety* 2.01 (2018), pp. 115–124 (cit. on p. 25).

[59] Haneet Singh Mahajan, Thomas Bradley, and Sudeep Pasricha. "Application of systems theoretic process analysis to a lane keeping assist system". In: *Reliability Engineering & System Safety* 167 (2017), pp. 177–183 (cit. on p. 25).

[60] EGAS Workgroup. *Standardized E-Gas Monitoring Concept for Gasoline and Diesel Engine Control Units*. 2015 (cit. on pp. 28–30).

[61] The New York Times NEAL E. BOUDETTE. *Autopilot Cited in Death of Chinese Tesla Driver - The New York Times*. https://www.nytimes.com/2016/09/15/business/fatal-tesla-crash-in-china-involved-autopilot-government-tv-says.html. (Accessed on 07/30/2018). Sept. 2016 (cit. on p. 37).

[62] The Tesla Team. *Your Autopilot has arrived — Tesla Canada*. https://www.tesla.com/en_CA/blog/your-autopilot-has-arrived. (Accessed on 08/07/2018). Oct. 2015 (cit. on p. 37).

[63] Netease. *First Tesla Auto-Drive Death Accident in China: Failed to avoid and crash/hit to the front car*. http://money.163.com/16/0914/14/C0UBU2I0002580S6.html. (Accessed on 10/14/2018). Sept. 2016 (cit. on p. 38).

[64] Erez Dagan et al. "Forward collision warning with a single camera". In: *Intelligent Vehicles Symposium, 2004 IEEE*. IEEE. 2004, pp. 37–42 (cit. on p. 38).

[65] Gideon P Stein et al. "A computer vision system on a chip: a case study from the automotive domain". In: *Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*. IEEE. 2005, pp. 130–130 (cit. on p. 38).

[66] Mobileye. *The Evolution of EyeQ*. URL: https://www.mobileye.com/our-technology/evolution-eyeq-chip/ (visited on 02/08/2018) (cit. on p. 38).

[67] NTHSA. *National Highway Traffic Safety Administration Final Report PE 16-007*. https://static.nhtsa.gov/odi/inv/2016/INCLA-PE16007-7876.PDF. (Accessed on 08/08/2018). Jan. 2017 (cit. on pp. 38, 44).

[68] Marilyn Jager Adams, Yvette J Tenney, and Richard W Pew. "Situation awareness and the cognitive management of complex systems". In: *Human Factors: The Journal of the Human Factors and Ergonomics Society* 37.1 (1995), pp. 85–104 (cit. on p. 40).

[69] National Transportation Safety Board. *Collision Between a Car Operating With Automated Vehicle Control Systems and a Tractor-Semitrailer TruckNear Williston, Florida May 7, 2016 - NTSB/HAR-17/02 PB2017-102600*. https://www.ntsb.gov/investigations/AccidentReports/Reports/HAR1702.pdf. (Accessed on 08/09/2018). Sept. 2017 (cit. on pp. 41, 44).

[70] David Shepardson Reuters. *Tesla driver in fatal 'Autopilot' crash got numerous warnings: U.S. government*. https://www.reuters.com/article/us-tesla-crash/tesla-driver-in-fatal-autopilot-crash-got-numerous-warnings-u-s-government-idUSKBN19A2XC. (Accessed on 08/10/2018). June 2017 (cit. on p. 43).

[71] Insurance Institute for Highway Safety. *IIHS Status Report newsletter, Vol. 53, No. 4, August 7, 2018*. https://www.iihs.org/externaldata/srdata/docs/sr5304.pdf. (Accessed on 10/14/2018). Aug. 2018 (cit. on p. 48).

[72] National Transportation Safety Board. *PRELIMINARY REPORT HIGHWAY HWY18MH010*. https://www.ntsb.gov/investigations/AccidentReports/Reports/HWY18MH010-prelim.pdf. (Accessed on 08/13/2018). May 2018 (cit. on pp. 49, 50).

[73] Queensland Government. *Stopping distances: speed and braking*. https://www.qld.gov.au/transport/safety/road-safety/driving-safely/stopping-distances. (Accessed on 02/11/2019) (cit. on p. 51).

[74] National Transportation Safety Board. *HWY18FH011 Preliminary Report: Crash and Post-crash Fire of Electric-powered Passenger Vehicle*. https://www.ntsb.gov/investigations/AccidentReports/Pages/HWY18FH011-preliminary.aspx. (Accessed on 08/01/2018). July 2018 (cit. on p. 53).

[75] The Tesla Team. *An Update on Last Week's Accident*. "https://www.tesla.com/blog/update-last-week\OT1\textquoterights-accident". (Accessed on 08/02/2018). Mar. 2018 (cit. on p. 54).

[76] Alan Ohnsman Forbes. *Fatal Tesla Crash Exposes Gap In Automaker's Use Of Car Data*. (Accessed on 08/03/2018). Apr. 2018 (cit. on pp. 54, 56).

[77] Shantanu Joshi. *Tesla Autopilot Failing (No Warnings) Where There Was A Fatal Car Accident W Model X - YouTube*. https://youtu.be/VVJSjeHDvfY?t=30. (Accessed on 08/03/2018). Apr. 2018 (cit. on pp. 55, 56).

[78] Olivia Solon The Guardian. *Tesla that crashed into police car was in 'autopilot' mode, California official says — Technology — The Guardian*. https://www.theguardian.com/technology/2018/may/29/tesla-crash-autopilot-california-police-car. (Accessed on 08/02/2018). May 2018 (cit. on p. 57).

[79] Yvonne Colbert CBC News. *Camera directed car toward oncoming traffic after windshield replacement, man says*. June 2018. URL: https://keenlab.tencent.com/en/2017/07/27/New-Car-Hacking-Research-2017-Remote-Attack-Tesla-Motors-Again/https://www.cbc.ca/news/canada/nova-scotia/camera-directed-car-toward-oncoming-traffic-after-

windshield-replacement-man-says-1.4715817 (visited on 07/27/2018) (cit. on p. 59).

[80] Javier Civera et al. "Camera self-calibration for sequential bayesian structure from motion". In: *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE. 2009, pp. 403–408 (cit. on p. 60).

[81] Marc Pollefeys, Reinhard Koch, and Luc Van Gool. "Self-calibration and metric reconstruction inspite of varying and unknown intrinsic camera parameters". In: *International Journal of Computer Vision* 32.1 (1999), pp. 7–25 (cit. on p. 60).

[82] Brian Paden et al. "A survey of motion planning and control techniques for self-driving urban vehicles". In: *IEEE Transactions on intelligent vehicles* 1.1 (2016), pp. 33–55 (cit. on p. 105).

[83] Andy Greenberg. *HACKERS REMOTELY KILL A JEEP ON THE HIGHWAY—WITH ME IN IT*. 2015. URL: https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/ (visited on 02/08/2018) (cit. on p. 125).

[84] Keen Security Lab of Tencent. *New Car Hacking Research: 2017, Remote Attack Tesla Motors Again*. 2017. URL: https://keenlab.tencent.com/en/2017/07/27/New-Car-Hacking-Research-2017-Remote-Attack-Tesla-Motors-Again/ (visited on 02/08/2018) (cit. on p. 125).