Mediator

# Simulation of Decision Making in the Mediator System

Deliverable D3.2 – WP3 – PU

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 814735.

# Simulation of Decision Making in the Mediator System

## Work package 3, Deliverable D3.2

**Please refer to this report as follows:**
Athmer, C., Spaan, M.T.J., Li, Y., Liu, Y., Bakker, B. (2022).
Simulation of Decision Making in the Mediator System, Deliverable D3.2 of the H2020 project MEDIATOR.

| **Project details:** | |
|---|---|
| **Project start date:**<br>**Duration:**<br>**Project name:** | 01/05/2019<br>48 months<br>MEDIATOR –<br>MEdiating between Driver and Intelligent Automated Transport systems on Our Roads |
| **Coordinator:** | Dr Nicole van Nes \| SWOV – Institute for Road Safety Research<br>Bezuidenhoutseweg 62, 2594 AW, The Hague, The Netherlands |
|  | This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 814735. |

| **Deliverable details:** | |
|---|---|
| **Version:**<br>**Dissemination level:**<br>**Due date:**<br>**Submission date:** | Final<br>PU<br>31/12/2022<br>23/12/2022 |

**Lead contractor for this deliverable:**
Matthijs Spaan – TU Delft

| **Report Author(s):** | Athmer, C., Spaan, M.T.J., Li, Y., Liu, Y. (TU Delft), Bakker, B, (Cygnify BV). The Netherlands |
|---|---|

## Revision history

| Date | Version | Reviewer | Description |
|---|---|---|---|
| **16/11/2022** | Preliminary draft | Matthijs Spaan – TU Delft, The Netherlands | Integration / editing |
| **21/11/2022** | Draft for QA | Canmanie Ponnambalam – SWOV Institute for Road Safety Research, The Netherlands | Internal review |
| **23/12/2022** | Final draft | Matthias Beggiato (Task and WP leader) – TU Chemnitz, Germany, Ernst Verschragen (QA Officer) – SWOV Institute for Road Safety Research, The Netherlands | Final revisions and checks |
| **23/12/2022** | Final deliverable | Michiel Christoph (Technical project coordinator) – SWOV Institute for Road Safety Research, The Netherlands | |

## Legal Disclaimer

# Table of contents

# About MEDIATOR

**MEDIATOR, a 4-year project led by SWOV, started on May 1, 2019.**

**MEDIATOR will develop a mediating system for drivers in semi-automated and highly automated vehicles, resulting in safe, real-time switching between the human driver and automated system based on who is most fit to drive. MEDIATOR pursues a paradigm shift away from a view that prioritises either the driver or the automation, instead integrating the best of both.**

## Vision

Automated transport technology is developing rapidly for all transport modes, with huge safety potential. The transition to full automation, however, brings new risks, such as mode confusion, overreliance, reduced situational awareness and misuse. The driving task changes to a more supervisory role, reducing the task load and potentially leading to degraded human performance. Similarly, the automated system may not (yet) function in all situations. The objective of the mediator system is to intelligently assess the strengths and weaknesses of both the driver and the automation and mediate between them, while also taking into account the driving context.

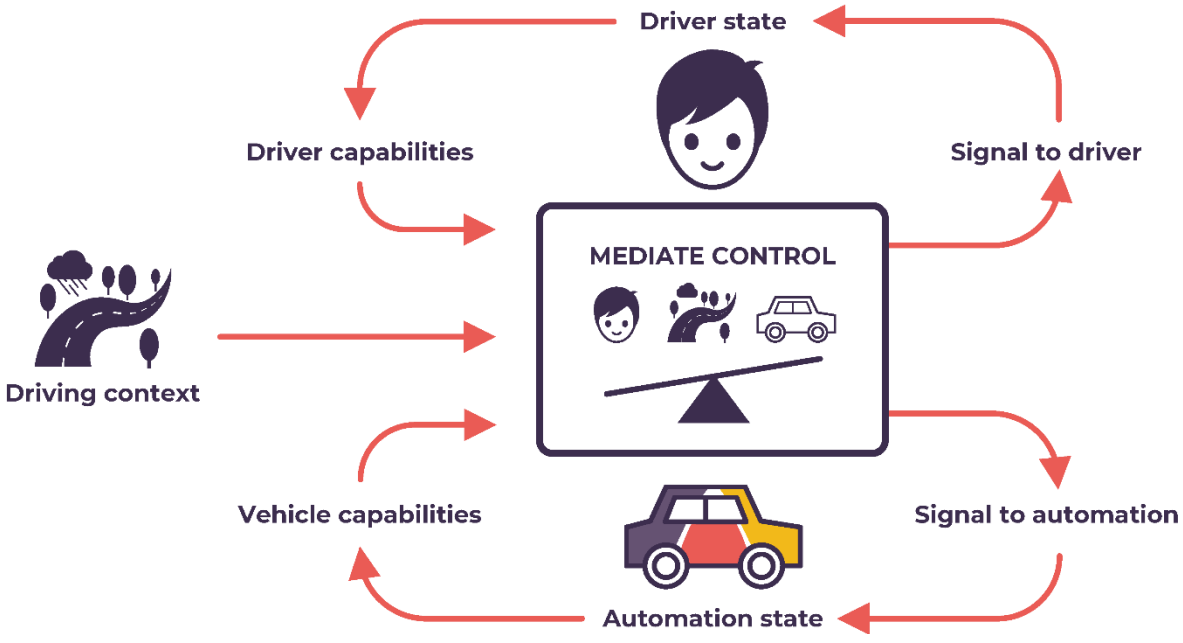


*Figure 1. The Mediator system will constantly weigh driving context, driver state and vehicle automation status, while personalising its technology to the drivers' general competence, characteristics, and preferences.*

MEDIATOR will optimise the safety potential of vehicle automation during the transition to full (level 5) automation. It will reduce risks, such as those caused by driver fatigue or inattention, or on the automation side, imperfect automated driving technology. MEDIATOR will facilitate market exploitation by actively involving the automotive industry during the development process.
To accomplish the development of this support system MEDIATOR will integrate and enhance existing knowledge of human factors and HMI, taking advantage of the of expertise in other transport modes (aviation, rail and maritime). It will develop and adapt available technologies for real-time data collection, storage and analysis and incorporate the latest artificial intelligence techniques, such as deep learning.

## Partners

MEDIATOR will be carried out by a consortium of highly qualified research and industry experts, representing a balanced mix of top universities and research organisations as well as several OEMs and suppliers. The consortium, supported by an international Industrial Advisory Board and a Scientific Advisory Board, will also represent all transport modes, maximising input from, and transferring results to, aviation, maritime and rail (with mode-specific adaptations).

# Executive summary

This deliverable outlines the development of the decision-making system, called the *Decision Logic* (DL), for the MEDIATOR project and a simulator whose main role is to support testing and refining of the aforementioned decision-making system. The system takes input from other components of the Mediator system, such as those estimating the state of the driver and the automation, and aims to take decisions that optimize safety and comfort, based on this input. It does this by using logical trees with a modular structure. Cornerstones of the input are novel concepts relating to the amount of time it takes before an autonomous car becomes (un)able to drive autonomously, and the amount of time it takes before a driver is (un)able to have control over the car. These are called *Time to Automation (Un)Fitness* (TTAF/U) and *Time to Driver (Un)Fitness* (TTDF/U) and are briefly explained in this report. First, the simulation environment is described, followed by the characteristics of the DL, which is the core of this deliverable. Finally, our testing procedures and evaluation results are presented.

The goal of the simulation environment is to simulate all aspects of driving a (semi-)autonomous vehicle on the road that are relevant to the Decision Logic. The relevant aspects mirror the different modules of the Mediator system, developed by various partners in the project, and include:

- Driving context, which encompasses all road-related aspects. This includes the type of road the car is driving on, and various properties of this road such as the speed or roundabouts.
- Automation context, which entails all subjects related to the car that are needed to take correct decisions. This takes information from the automation subsystem to determine available automation levels for the car.
- Driver context, which monitors the distraction and fatigue levels of the driver and estimates the fitness of the driver based on this.
- Actions, which represent the output of the DL and are communicated to the HMI.

Furthermore, real world data is used to enhance the simulator, including a route driven by an actual vehicle in Sweden and by parsing data from the car into the simulator. Effectively, this means replacing the simulated versions of the driving context and automation context with real data. This enables us to test the DL in a more realistic setting.

The DL is used to decide which decisions to take based on input from the environment. The design is modular, such that specific parts can easily be replaced or left out for different use cases. The full system is designed to handle all relevant use cases, which are:

- Automation degradation: this happens when the car is driving in an automation level that will soon become unavailable.
- Automation upgrade: when a higher level of autonomous driving is available, it may be desirable to suggest the driver to switch to that level, depending on the circumstances.
- Distraction: when a driver gets distracted this may need to be dealt with.
- Fatigue: like distraction, but when a driver gets tired instead of distracted.
- Driver request: a driver can make specific requests, which may or may not be granted.

The DL is able to handle each of these situations, and also defines a priority when multiple use cases happen simultaneously. Additionally, it includes a *Window of Opportunity* (WOO) and dynamic action parameters to improve driver safety and comfort. The former allows the HMI to present an action at a moment where the driver is not preoccupied with other driving tasks, while

the latter allows for dynamic periods of time in which an action is active, depending on the situation.

Finally, tests are run to test the performance and evaluate the robustness of the DL. The main conclusion from these tests is that errors in the input can cause errors in the output of the DL, potentially leading to unsafe situations. Therefore, it is of vital importance to handle this. One way to do this is to counteract potential errors with buffers. This indirectly makes the approach of the DL more conservative, which may decrease driver comfort but will increase safety. Real world testing is needed to estimate error margins and devise appropriate buffers based on this.

# 1. Introduction

**A key component of Mediator, the mediating system for autonomous vehicles developed in this project, is the decision-making system. This report focuses on the development and improvement of this system. The decision-making system is called the Decision Logic (DL) and will be referred to as such in this report. It decides which actions to take based on inputs received from other components, and these actions are then passed through to the driver. The DL is built and tested on top of a simulator that is also built in this research, and that has been improved by using data from vehicle provided by partner Veoneer that is driving a set route in Sweden. This report describes the simulator, the DL, the improvements made to both through real world data, and evaluates its performance.**

A cornerstone of the inputs on which decisions are based, are novel concepts called *Time to Automation Fitness* (TTAF), *Time to Automation Unfitness* (TTAU), *Time to Driver Fitness* (TTDF), and *Time to Driver Unfitness* (TTDU). The former two relate to the autonomous vehicle and its (in)ability to use its autonomous function. For instance, when the vehicle is approaching a highway exit that will lead it into a dense urban area (in which automation assistance is significantly reduced), it can estimate how long assisted driving it still available. More precisely, TTAF and TTAU are values that estimate (in seconds) how long it will take before autonomous driving will be available (TTAF) or unavailable (TTAU). These estimates are based on the upcoming road which is assumed to be known.

The latter two relate to the driver and its (in)ability to drive the car, for instance due distraction or fatigue. In other words, TTDF and TTDU estimate (in seconds) how long a driver needs to take over control (TTDF) and how much time remains before a driver will be unable to drive (TTDU). These values are calculated by processing video data of the driver (captured in real-time by a camera in the car monitoring the driver), estimating his fatigue and distraction levels based on the video, and converting these levels into TTDF and TTDU values based on previous research.

The DL takes decisions based on inputs from the environment, which include, but are not limited to, the concepts explained in the previous two paragraphs. To test the DL a simulator is needed that simulates an environment that contains all concepts relevant to the DL. This simulator and DL have been developed in the context of the MEDIATOR project.

Chapter 2 of this report details the main aspects of the simulator, namely how the driving, automation and driver context are being simulated, as well as the actions available to the DL. Veoneer, a partner in the MEDIATOR project, has been doing test drives with a semi-autonomous vehicle in Sweden. From these drives data has been gathered concerning the environment and the car. This data is used to improve the simulator and the DL, to make it more realistic. The data that is used and how it improved the simulator is also discussed in Chapter 2.

Chapter 3 presents the Decision Logic itself, of which several versions have been defined in consultation with project partners. It discusses how different use cases/events are being handled and also introduces some of the enhancements such as endowing actions with a window of opportunity.

Chapter 4 discusses our evaluation setup and results of applying the DL in the simulator. Many simulation runs with different settings have been performed to test the performance of the DL on various *Key Performance Indicators* (KPIs). Goals of these tests include getting a general sense of the DL performance, evaluating the impact of the changes made to the DL, and testing the robustness of the DL.

Finally, Chapter 5 provides conclusions and suggestions for future work.

# 2. Simulator

**This chapter describes the context within which the decision-making module of MEDIATOR was created and refined. This includes an extensive, highly configurable, simulation environment which realistically mimics (aspects of) a real-world drive and is used to test and improve the decisions made by the decision logic. Furthermore, data from actual drives is used to enhance both the simulator and the decision logic**

## 2.1. Simulation environment

To be able to create, test and evaluate the decision logic module of MEDIATOR an environment is needed that simulates all necessary inputs and outputs of decision logic. Therefore, in this research a simulation environment is created. It enables simulating drives of a vehicle step-by-step, updating all relevant state variables at each step. Events during a drive are determined stochastically. Together with a high degree of customizability this allows for the simulation of a wide range of different scenarios.

To understand the simulator, it is necessary to know what the complete Mediator system consists of. It can be broadly subdivided into five main modules:

- The *driving context module* captures all context relevant information and transforms it into required context variables. The main source of input to the context module is the road the car is driving on, which is crucial to decision-making and therefore a vital part of the simulator. *Section* 2.1.1 provides a high-level overview of the driving context in the simulator.
- The *automation module* refers to the car itself and provides related information to the decision logic, e.g., the speed the car is currently driving at. *Section* 2.1.2 explains how this is simulated.
- The *driver module* deals with all driver-related information, such as the current level of distraction of the driver and passes this information to the decision-making system. *Section 2.1.3* explains its role in the simulator.
- The *human machine interface (HMI)* presents the outputs from the decision logic to the driver. This is simulated through actions, as described in *Section* 2.1.4.
- Finally, the *decision logic module* takes the inputs from the first three modules mentioned above and aims to suggest appropriate actions based on this input. The main goal of this research is to create a well-functioning decision logic module that improves driver safety and comfort, which is why a full chapter is devoted to explaining its internal mechanisms. This is done in *Chapter 3*.

### 2.1.1. Driving context

An important aspect of simulating drives of a (semi-)autonomous vehicle is the road it is driving on, which is captured in the driving context. Since the goal of this simulator is to provide an environment in which decision logic can be tested, only information relevant for this purpose needs to be simulated. To accomplish this the simulator contains road types and road events, which impact the maximum automation level that is available and the speed the car can drive. Based on this information two important inputs to the decision logic are calculated: *Time to Automation Fitness* (TTAF) and *Time to Automation Unfitness* (TTAU), discussed in detail in *Section 2.1.2*.

A road consists of one or more consecutive road parts, and each part has a type. These road types can be defined manually, so there is no set list of available road types and properties. But examples are an urban road, a rural road, or a highway. These may have different speed limits and available automation levels.

On a road, so-called “events” may occur that also impact the available automation levels and the speed the car can drive in. These events can be static or dynamic. Static events are known from the start of the route. Examples of possible static events are a roundabout or a turn. As opposed to static events, dynamic events are only known to the system just before they appear, making it more challenging to anticipate them. Examples are bad weather or heavy traffic. Like road types, road events can be manually defined, which is why there is no set list of road events and properties.

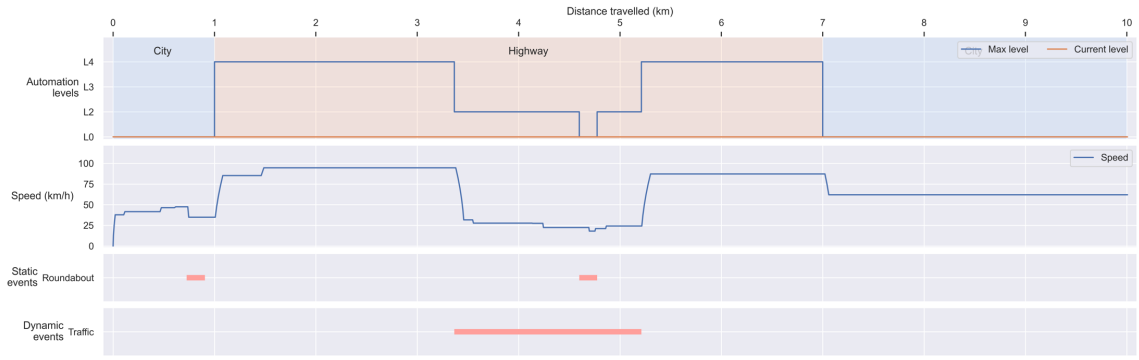


*Figure 2.1* A simple route of 10 km displaying all information generated by the driving context. The image shows four plots that are aligned on the position of the road. The first plot shows the road types, the maximum available automation level at each stage, and the current level the car is driving in. The second plot shows the current speed of the car. The third and fourth plot show when static and dynamic events are active, respectively.

Figure 2.1 shows an example road with two road types (City and Highway), two roundabouts (a static event) and one traffic jam (a dynamic event). It also shows the maximum level available at each part of the route based on the aforementioned information, and the speed of the car, which fluctuates around the target speed of the car. For the configuration used here, the maximum available automation levels are as follows:

- City: L0
- Highway: L4
- Roundabout: L0
- Traffic: L2

### 2.1.2. Automation context

Another key aspect of the simulation environment is the automation context, which inputs to the decision-making module the current automation level the car is driving in, the current position of the car, the speed the car is driving, and the *Time to Automation Fitness* (TTAF) and *Time to Automation Unfitness* (TTAU).

The current automation level the car is driving in is determined by previous actions of the Mediator system and the driver and should never be higher than the maximum available level at that moment. The available level is based on the driving context discussed in the previous section and the type of car. Right now, typically only cars with up to Level 2 automation are commercially

available, but the simulator can also be configured to run with cars that have automation levels L3 and L4 available, since it is assumed that these cars will become available to the general public in the future.

The simulator assumes the exact route the car is driving is known beforehand and as such the position of the car is measured as the distance driven since the start of the route. Based on this position the current speed limit of the car is known, and the speed the car is driving in is simulated to fluctuate around this limit.

Given the information described above, two key inputs to the decision logic can be calculated: TTAF and TTAU. As their names imply, these give an indication of the time (in seconds) before an automation level becomes fit and unfit, respectively. They are calculated for each automation level and are based on the prediction of the maximum available automation levels, the speed limits, and the current position of the car.

Figure 2.2 shows a simple example of TTAF and TTAU on a 3-kilometer road. For simplicity, the road does not contain any road events, only has two road types, and only levels L0 and L2 are available. The speed limits determine the values for TTAF and TTAU and are 50 km/h and 100 km/h for City and Highway road types, respectively. Therefore, the TTAFL2 (which is TTAF for automation level L2) is 72 seconds at the start of the road, since L2 becomes available after 1 kilometer and that is the time it takes to travel this distance when driving 50 km/h. In a similar fashion, the TTAUL2 at kilometer 1 is 36 seconds, given a speed limit of 100 km/h at the highway and the fact that L2 becomes unavailable at kilometer 2.


*Figure 2.2* A short route demonstrating how TTAF and TTAU are calculated. The image shows three plots: the first one shows road types and maximum available automation levels; the second one shows TTAF in seconds for L2; the third one shows TTAU in seconds for L2.

### 2.1.3. Driver context

Besides the driving context and the automation system, the driver context also impacts the decisions made by the decision-making module. A driver can be distracted or get fatigued which may impact his or her ability to drive. This is captured in two variables: *Time to Driver Fitness* (TTDF) and *Time to Driver Unfitness* (TTDU). Finally, a driver can also initiate a request to the car which needs to be handled by the decision logic appropriately.

Distracted driving is known to have a significant impact on crash risk. Whether a driver is distracted and how severe this distraction is, can be monitored by putting a camera on the driver and using artificial intelligence to assess the drivers' distraction level. This is simulated by assigning a

probability that the driver will get distracted during the drive. When distracted behaviour occurs, it can remain at the same level of severity, increase, or spontaneously disappear. All of this is also determined by configurable probabilities.

Another potential for dangerous situations is when drivers get too tired to drive. Fatigue has a more gradual progression as opposed to distraction, which is an acute event. Similarly to distraction, fatigue can also be measured by putting cameras on the driver. Fatigue is simulated by having it gradually increase over time, with a probability of it suddenly increasing significantly. This is done based on the assumption that the system measuring the fatigue is not completely accurate, and that it is hard to predict the rate at which fatigue progresses in each individual.

Based on distraction and fatigue, TTDU and TTDF are calculated. These measures indicate the time (in seconds) it takes before a driver is unfit to drive and the time it takes before a driver is ready to take back control, respectively. The former relates to a situation in which the car is driving in level L0 or L2, in which case the driver is required to be fit to drive. If TTDU gets low, it means the decision logic needs to take appropriate actions to prevent an unsafe situation. *Section* 2.1.4 details how it can do this. The latter refers to a situation in which the car is driving in L3 or L4, in which case the driver is not required to be fit to drive since the car is handling the driving. TTDF estimates how long it takes to prepare the driver to take back control from the car, if necessary.

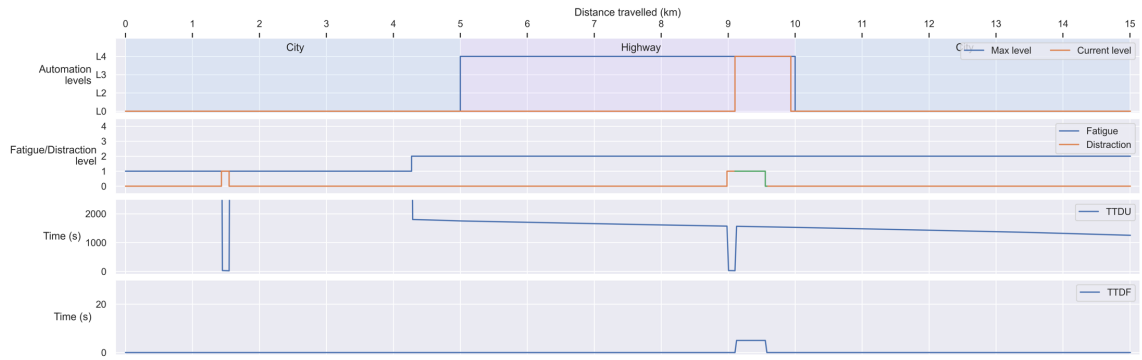


*Figure 2.3* A route demonstrating the interaction between fatigue/distraction and TTDU/TTDF. The first plot shows the current automation level and the maximum automation level available. The second plot shows the level of distraction and fatigue. The third and fourth plot show TTDU and TTDF, respectively.

Figure 2.3 illustrates how fatigue and distraction impact TTDU and TTDF. Halfway between kilometers 1 and 2 the driver gets distracted, which causes the TTDU to drop almost to zero. The distraction is resolved prompty, after which TTDU jumps up again. Just after kilometer 9 another distraction occurs, causing TTDU to drop. Shortly after, the automation level of the car switches to L4, and therefore distraction is no longer an issue, and the distraction line appears in green. This also means TTDU returns to its original level and TTDF increases, since now it would take some time for the driver to be ready to take back control of the car. Between kilometer 4 and 5 there is a sudden increase in fatigue, which causes another drop in TTDU. After that drop, TTDU decreases linearly again.

Finally, a driver can also make a request, which needs to be handled by the decision logic. A driver may have a desire to switch to a particular level, even though the decision logic does not suggest a shift to this level (which can have several reasons, discussed in *Chapter* 3). This is simulated by

having a (small) probability that the driver initiates a request to shift to a different automation level; the request stays active until the decision logic handles it or the driver cancels it.

### 2.1.4. Actions

The *human machine interface* (HMI) displays information from the several Mediator modules to the driver. It may display things such as TTAF, TTAU, upcoming road events, and actions taken by the decision logic. Our goal is to develop the DL, which is why the HMI is simulated by showing the actions it takes.

The actions taken by the decision logic have an impact on one or more actors in the system and can be used to prevent unsafe situations and/or enhance driver comfort. There are numerous actions available to the decision-making module, including:

- **SSL<X>**, where **<X>** can be 0, 2, 3, or 4. This action suggests to the driver a shift to the defined level. The driver can (1) decline or ignore the suggestion, keeping the car in the current automation level, or (2) accept the suggestion, which results in the car switching to the suggested automation level. In the simulator, probabilities are associated with drivers accepting and rejecting suggestions.
- **ESL<X>**, where **<X>** can be 0, 2, 3, or 4. This action enforces the car to shift to the specified automation level, meaning the driver cannot choose to decline the shift. Note that in practice, it is only desirable for the driver to be enforced to take over control (i.e., ESL0) and never to relinquish control.
- **CF**, Correct Fatigue. This is a corrective action trying to correct fatigue.
- **CD**, Correct Distraction. This is a corrective action trying to correct distraction.
- **PD**, Prepare Driver. This action prepares the driver to take over control when they are distracted while driving in L3 or L4 (e.g., reading a book).
- **CR**, Clear Request. This clears a pending request when it has been dealt with.
- **ES**, Emergency Stop. Forces the car to make an emergency stop, terminating the current drive.

Chapter 3 elaborates on the decision logic and explains in more detail which actions are taken in which situations. The next section discusses how real-world data is incorporated into the simulation environment, and how this improves the simulator and the automation module.

## 2.2. Real world data

The previous section provides a high-level overview of the components of the simulation environment. Ideally, this environment can be used to test different setups of the decision logic before implementing them in the vehicle. To be able to do this it is imperative that the simulations resemble reality as closely as possible. Real data from several modules of MEDIATOR is made available by partners in the project to achieve this.

The data is acquired by Veoneer, a provider of automotive technology based in Sweden and one of the partners in the project. They implemented the modules of MEDIATOR into a (semi-)autonomous vehicle, meaning L2 is available on certain parts of the road. This car is also equipped with sensors and cameras. They let professional drivers drive a predetermined route in Sweden to gather real-world data.

The remainder of this section describes the different data that has been made available, which includes driving context data and automation context data. For each context, an overview of the use cases of the data is given. At the time of writing, no driver context data is available.

### 2.2.1. Driving context data

The road information has been extracted from the actual drives. This is done by manually inspecting the video and sensor data. Based on this, a tabular representation of the route is built which specifies road details for all consecutive segments on the route, each of approximately 10 meters long. This data includes, but is not limited to, speed limits, road types, static road events, and whether L2 is available or not. The latter is directly deduced from the road type and static events.

This tabular route provides the same driving context data that is used in the simulator and is parsed into the simulation environment. This has several uses:

- It helps to improve the road generation: more realistic road types and static events can be generated based on this example. Insights that are obtained based on the tabular route are that one road segment can have different speed limits (e.g., a highway where the speed limit changes), which was not yet incorporated into the road generation algorithm. Furthermore, static events (e.g., turns, roundabouts) are very short and many, much shorter and much more than the events that were randomly generated. These are small differences, but they help in developing a more realistic road generation algorithm.
- Running the simulation on a real route enables us to get a good impression of the behaviour of the DL when implemented in the car. A high degree of realism in the simulator means that the actions shown in the simulator will accurately mimick the actions suggested when driving the car. This can also help to tune the driver preferences in the simulator before testing it on the road, because it shows how different driver preferences impact the behaviour of the DL. Figure 2.4 shows an example of this.

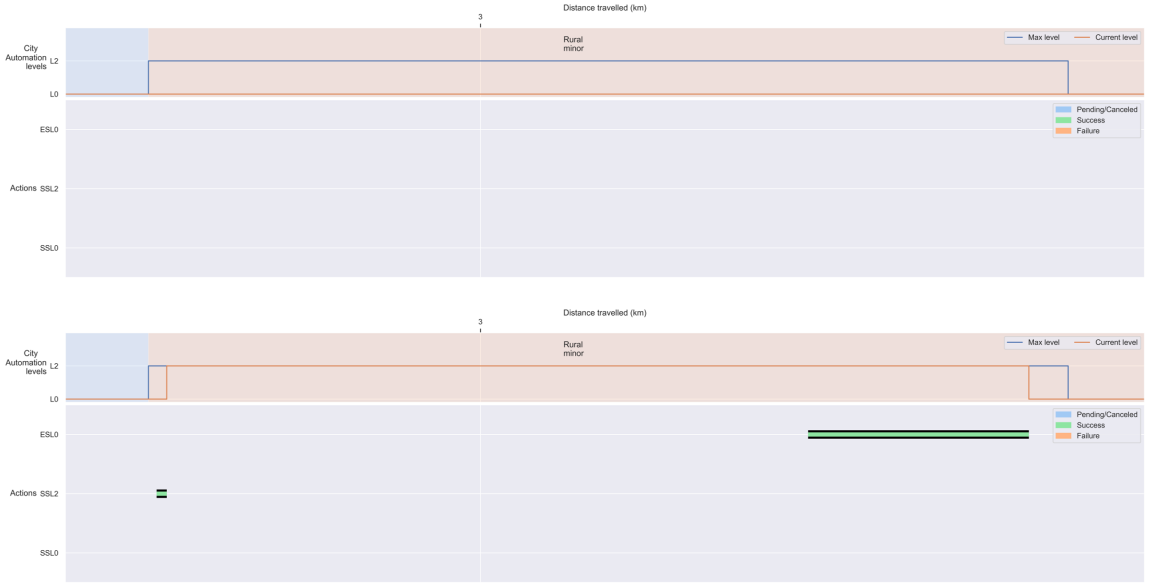


*Figure 2.4* The figures above both show simulation of part of the route driven in Sweden, with different driver preferences. The driver preferences in this case relate to the minimum amount of time the driver wants to be able to drive uninterrupted. In the top simulation, this time is longer than the time that L2 is available, which is why DL does not suggest shifting to L2. In the bottom plot, this time is shorter and thus a shift to L2 is made. This exemplifies how the simulator can be used to tune the driver preferences in such a way that the behaviour of DL changes. By doing this in the simulator, we can create different driver profiles and then assess their comfort in the real world.

### 2.2.2. Automation context data

Besides obtaining data regarding the driving context, data from the automation context is also gathered during the drives in Sweden. This data shows the speed driven by the car and the TTAF and TTAU at each road segment of approximately 10 meters. The automation context module in the simulation environment entails the same datatypes, and therefore this data can be directly parsed into the simulator. This yields various insights and improvements to the simulator.

First, observing the real speed driven allows for an improvement of the simulation of speed. The speed across the drive is quite stable, meaning the car drives at the same velocity for prolonged periods of time. In contrast, the original simulation had much more variance in its speed. Another interesting insight is that the real speed sometimes drops to 0, e.g., when the car is waiting for a traffic light. This was not considered before and has implications for the accuracy of the TTAF and TTAU.

Second, the TTAF and TTAU in the car can be compared with the TTAF and TTAU in the simulator. Since these metrics have a big impact on the decision-making process, it is important that the values in the simulator do not differ too much from the values in the car. When they do, it means the simulator cannot reliably predict actions that will be taken by the decision logic when run in the car. Figure 2.5 shows a comparison of TTAU in the simulator and the car on part of the road. As can be seen in the figure, the differences are negligible.

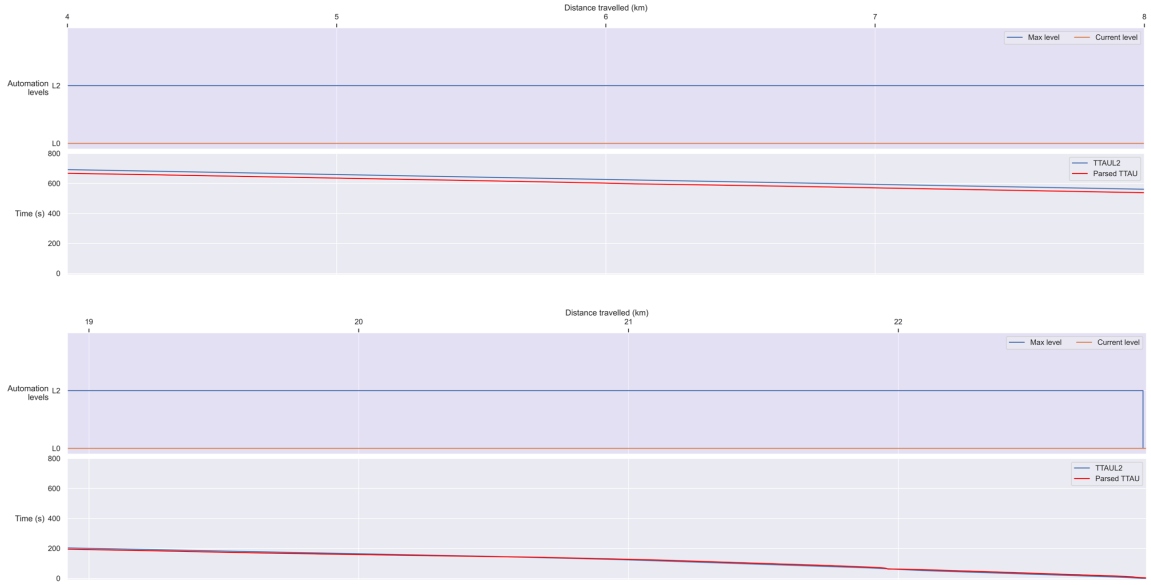


*Figure 2.5* Comparison of the TTAU in the car (in red) and in the simulator (in blue) on the start and end of the longest stretch of road on the route where L2 is continuously available. The simulator gives a slightly higher estimation of TTAU than the car in the beginning, but the difference is small. Towards the end, the estimates are almost the same. These small differences are not expected to have an impact on the behaviour of the decision logic.

Furthermore, comparing the TTAF/U values from the car with the values in the simulator can act as a sanity check for the values used in the car. The software used for these calculations are independently implemented, but the algorithm is the same. Therefore, if there is a mismatch this is an indication that one of the calculations is wrong, allowing us to improve the overall Mediator system.

Finally, the simulator allows for the calculation of the accurate TTAF/U values in retrospect, i.e., how long a segment of road took to drive. With the real speed data from the car these values represent realistic timeframes. These can be used to assess whether the predicted TTAF/U values are accurate; if there is a big mismatch between the predicted values and the real values this may impact DL performance. In Chapter 4 an analysis of the impact of wrong values is given by feeding wrong TTAF/U values into the decision logic. For driver comfort this also plays a role, because when the values shown to the driver are inaccurate this may cause discomfort.

Figure 2.6 displays the comparison of the predicted and real TTAF/U values. A discrepancy exists between the metrics. Further inspection shows this is mainly caused by the car standing still for some time. This cannot reliably be predicted and therefore is not considered an error in the calculation of TTAF/U. Rather, the decision logic should be robust against an underestimation of TTAF/U. In Chapter 4 an extensive analysis is done to show that it is.

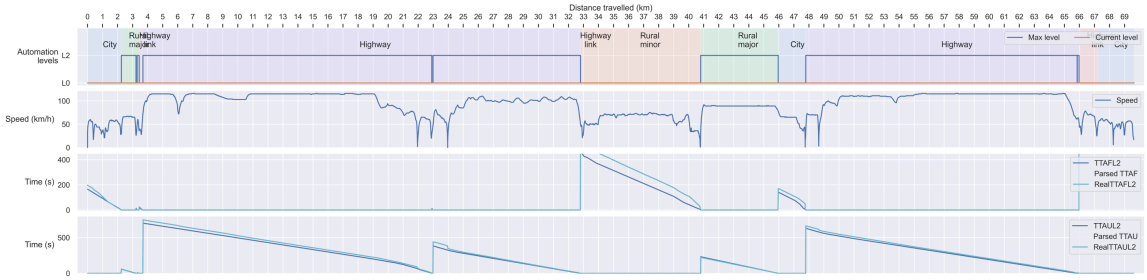


*Figure 2.6* Comparison of the TTAF and TTAU predicted by the simulator (in dark blue) and the real values calculated in retrospect (in light blue). The figure shows a discrepancy between the values; further inspection shows that this is caused by the car standing still.

# 3. Decision Logic

**The previous chapter provides a high-level overview of the simulator that is built and used to test the decision logic that is central to this project. This chapter describes the decision logic (DL) in detail. This logic decides when and what actions to take. It is modelled as decision trees which makes it conceptually easy to understand. For different use cases, different trees are created, making it a very modular design. This way, parts can easily be left out or added depending on the situation or the car.**

The Mediator system collaborates with the DL as follows: at predetermined time intervals (e.g., 1 second) MEDIATOR inputs its current state to the DL. The DL feeds this into the tree and outputs an action. If this action is DN ("Do Nothing") nothing is done. If the returned action is an action that is already pending (because the DL suggested it at a previous timestep and the action has not completed yet) nothing is done. If neither of these cases is true, MEDIATOR initiates the action that is suggested by the DL. If a different action was still active, that is cancelled and replaced by the new action. This way, the DL helps MEDIATOR to quickly adapt to new situations.

To understand the different modules that are developed for the DL it is vital to know the use cases that it needs to handle. Depending on the use case, a module is selected by the top-level tree. The priorities with which submodules are selected is discussed in Section 3.1. The following use cases exist:

- Automation degradation: this means the car will soon be unfit to drive, meaning it is currently driving in an automation level that will soon be unavailable. When this happens, appropriate actions need to be taken to prevent an unsafe situation. These are discussed in Section 3.2.1.
- Automation upgrade: when a car is not driving in the highest available automation level, you might want to switch to a higher level depending on the situation. This is detailed in Section 3.2.2.
- Driver degradation: this means the driver will soon be unfit to drive, which can be caused by either distraction or fatigue. When this happens, appropriate actions need to be taken to prevent an unsafe situation. These are discussed in Sections 3.3.1 and 3.3.2.
- Driver request: a driver can also make a request to switch to a level, which needs to be handled. This is explained in Section 3.3.3.

Apart from handling these use cases, the DL module has been enhanced based on received data and feedback from partners. These enhancements are detailed in Section 3.4.

## 3.1. High level logic

The DL consists of several modules (implemented as decision trees) that handle different situations. Multiple use cases can happen simultaneously at which point a decision needs to be made regarding which case to handle first. To do this, a top-level decision tree is created that prioritizes the different modules. Figure 2.1 shows an overview of the logic.

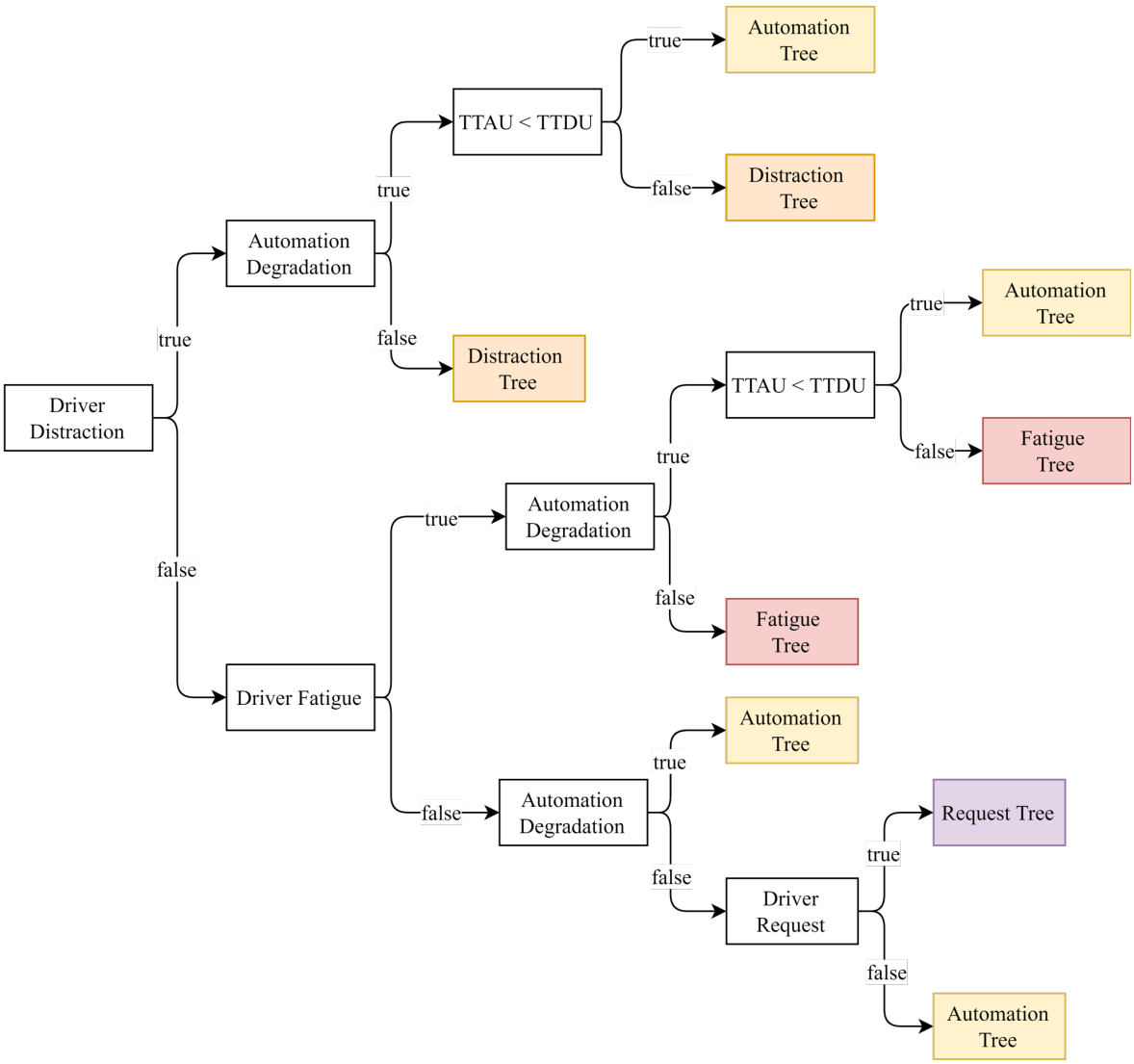


*Figure 3.1* Overview of high-level decision tree that decides which submodule to go into.

The first thing that is checked is if the driver is distracted. This has priority over driver fatigue, since distraction is an acute event; fatigue, on the contrary, increases gradually and can therefore be anticipated long before it becomes critical. When the driver is distracted, the next thing that is checked is whether the current automation level will soon be unavailable (automation degradation). If that is also true, it means two potentially critical events are happening simultaneously.

When that happens, the decision which event to handle first is based on *Time to Automation Unfitness* (TTAU) and *Time to Driver Unfitness* (TTDU) because these indicate how long it will take before the car and driver, respectively, are unfit. The value that is lower is handled first, i.e., if TTAU is lower then the DL moves into the automation tree, the module related to automation. Else, it moves into the distraction tree, the module related to distractions. When there is no automation degradation coming up, the DL also moves into the distraction tree.

If there is no distraction, driver fatigue is checked. If the driver is fatigued, the same logic applies here as with distraction and automation degradation, described in the previous paragraph. Depending on this, the DL moves into the automation tree or the fatigue tree, the module handling driver fatigue.

The remaining branch of the tree is reached when there is no driver degradation. When there is automation degradation, the DL moves into the automation tree to handle it. Else, a check is made to see if there is an active driver request. When there is, the DL moves into the request tree, the module handling driver requests. Else, it moves into the automation tree, which, besides handling automation degradation, also handles automation upgrades. And that concludes an overview of the top-level logic deciding which module to use given a particular situation.

## 3.2. Automation context

One of the submodules of the DL is the automation tree, which handles two use cases: automation degradation and automation upgrade. The former refers to the situation where the car is driving in a level that will soon be unavailable, the latter to the situation in which a higher level is available than the level the car is currently driving in. Figure 3.2 shows an overview of the automation tree. Sections 3.2.1 and 3.2.2 elaborate on the logic relating to automation degradation and automation upgrade, respectively.

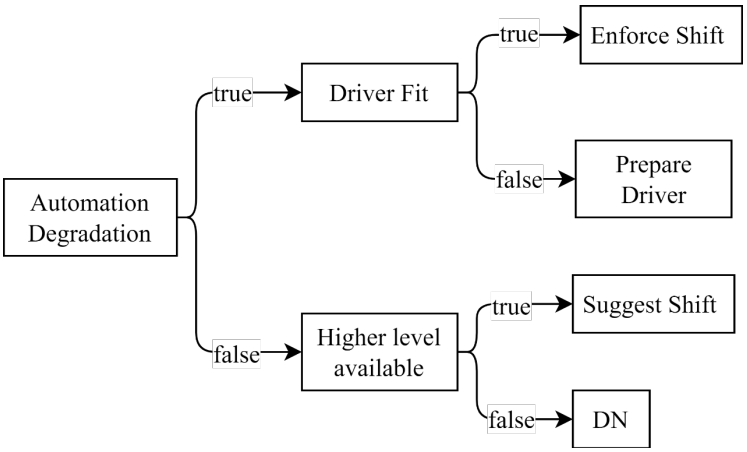


*Figure 3.2* An overview showing the main logic of the automation tree, with some details omitted.

### 3.2.1. Automation degradation

The first thing that the tree evaluates is whether an automation degradation event is coming up, by checking whether the automation level the car is currently driving in will soon become unavailable. If this is not the case, it goes into the branch referring to automation upgrade, detailed in Section 3.2.2.

If it is the case, however, the next thing that is evaluated is whether the driver is fit, i.e., the driver is ready to take over control of the car. When the driver is fit, a shift to a lower level, which will be available, is enforced. The level that will be enforced is the highest level that will still be available. The action that will be taken is *ESL*<*X*>, where <*X*> is the the level that is enforced.

When the driver is not fit, e.g., he is reading a book, the driver needs to be prepared to take over control first. This means warning the driver that they need to take over control soon, which is action *PD* (prepare driver). An extreme case can occur where there is very little time left before automation becomes unavailable, which is expected to not be enough time to prepare the driver for taking over. In such a case, an emergency stop will be made (*ES*).

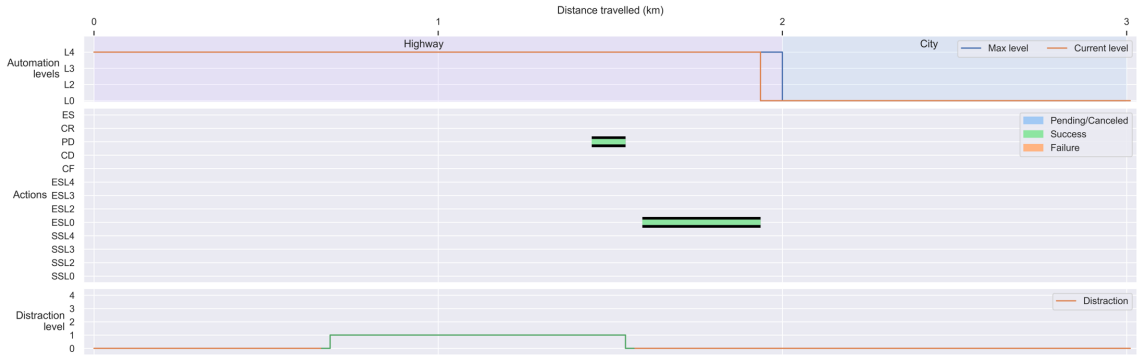


*Figure 3.3* Example of an automation degradation event.

Figure 3.3 shows an example of automation degradation event where the driver first needs to be prepared to take over. The scenario is the car driving on the highway in a fully automated mode, with an urban road (city) coming up, where the car is not able to drive autonomously and the driver needs to take full control. However, the driver is currently distracted (e.g., reading a book), and therefore he first needs to be prepared to be ready to drive. This is done well in advance (since the system knows approximately how long it will take before the city is coming up), after which the driver is enforced to take over control, thereby concluding the handling of the automation degradation event.

### 3.2.2. Automation upgrade

When there is no automation degradation coming up, the tree evaluates whether a higher level is available than the level the car is currently driving in. The idea behind this is that generally it is assumed that driving in a higher automation level is safer and more comfortable. If a higher level is not available, the resulting action is *DN*, do nothing.

However, if a higher level is available, a shift to that level may be suggested. A shift to a higher level is only ever suggested, never enforced, since it is not desirable to enforce a driver to release control of the vehicle. There are some additional constraints that need to be met before a shift to a higher level is suggested, though.

First, the level needs to (estimated to) be available for a longer period. The exact time that it should be available can be adjusted in the settings and may depend on driver preferences. The reason for having this constraint is that it is not deemed comfortable for the driver to have to switch levels in rapid succession.

Second, the driver should not have already declined a suggestion to that level recently. Again, what entails recently can be defined in the settings and may depend on driver preferences. The reasoning behind this is that when a suggestion is made, and the driver rejected it, he apparently did not want to switch levels. In such a case, the DL should not keep suggesting the level because this may cause discomfort with the driver.

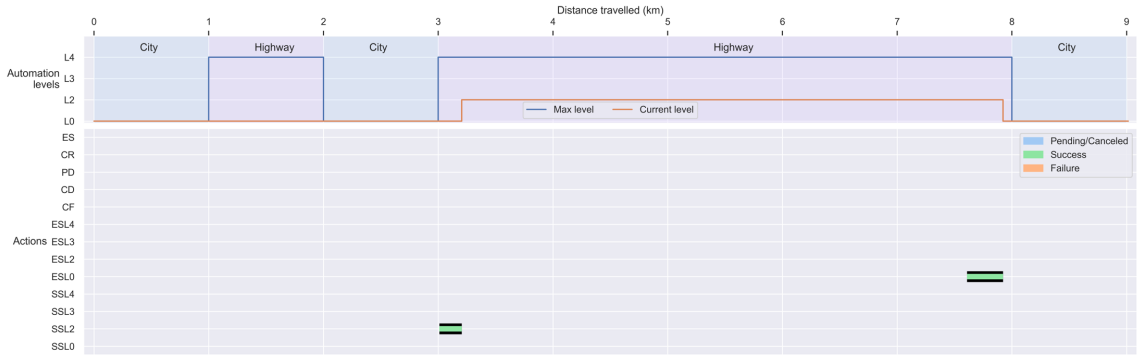


*Figure 3.4* Example of an automation upgrade event.

Figure 3.4 shows behaviour of the decision logic in an automation upgrade use case. The driver in this case prefers to be able to drive for at least 2 minutes in the same level, and his preferred level is L2. The road has two stretches of highway where all levels all available. The first stretch, however, is very short, therefore no suggestion to switch is made there since the driver would have to take back control within the minimum desired time of 2 minutes. The second stretch of highway is longer, and as such a suggestion to shift to L2 is made, which is accepted by the driver.

## 3.3. Driver context

Besides the automation context the other submodules relate to the driving context. For the driving context, several submodules are developed:

- Distraction tree, handling situations in which the driver is distracted.
- Fatigue tree, handling situations in which the driver is fatigued.
- Request tree, handling driver requests.

The remainder of this section illustrates and elaborates on each of the modules.

### 3.3.1. Distraction

Figure 3.5 shows an overview of the distraction tree. As discussed in Section 3.1 this tree is entered whenever the driver is distracted. When the driver is distracted but the car is driving fully automated (i.e., driving in L3 or L4), nothing needs to be done (*DN*) since the driver is not required to be an active participant in the drive.

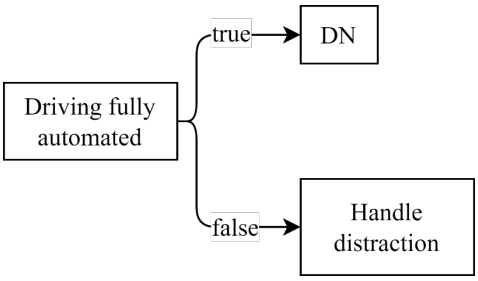


*Figure 3.5* An overview showing the main logic of the distraction tree, omitting details specific to the implementation.

In the situation where the car is not driving fully automated, the distraction needs to be handled. The following actions are considered (in this order) to handle the distraction:

1. If possible, suggest a shift to L4 (*SSL4*). For it to be possible, it must be available, and it must not have already been suggested.
2. If the former is not an option (or the driver declines the suggestion), the next option is to correct the distraction (*CD*), for instance by engaging the HMI to draw the attention of the driver. This can only be done if the estimation is that there is enough time left to correct the driver before the situation becomes unfit.
3. Finally, if correcting the distraction fails the last resort is to make an emergency stop (*ES*).

These three actions summarize the way distractions are handled. Figure 3.6 shows an example in which all three actions described above are executed. The driver is driving manually on the highway (L4 is available) when he gets distracted. First, the DL suggests a shift to L4, but the driver declines. Next, DL attempts to correct the distraction. The first attempt fails, after which another attempt is made. However, this attempt is cancelled midway because the driver is becoming critically unfit, at which point the DL lets the car make an emergency stop.

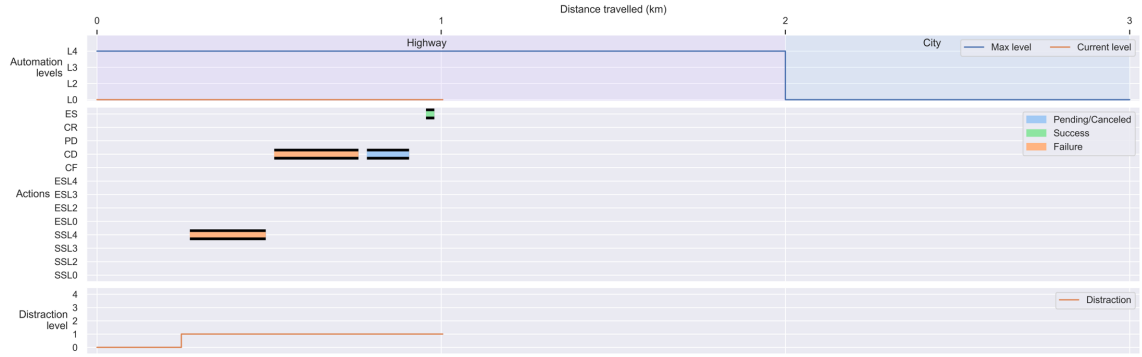


*Figure 3.6* An example of a distraction event.

### 3.3.2. Fatigue

A less acute but nevertheless relevant situation is when the driver gets fatigued, since this may also lead to dangerous situations (e.g., a driver falling asleep). Figure 3.7 depicts a high-level overview of the tree handling fatigue.

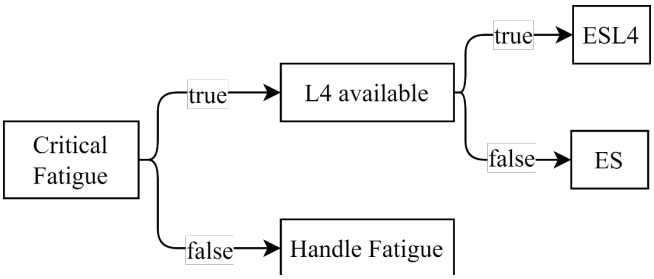


*Figure 3.7* An overview of the main logic relating to coping with driver fatigue, leaving out details irrelevant to this description.

For this discussion, fatigue is defined as either critical or non-critical. The former means that the driver is expected to be so tired that he will soon be too tired to drive, whereas the latter means that the driver is somewhat tired but is still expected to be fine to drive for a prolongued period.

When fatigue is critical, it is assumed to be too late to correct. Instead, the DL aims to make use of the self-driving properties of the car: if L4 is available, a shift to L4 is enforced (*ESL4*). This is the only time that a shift towards a higher level is unforced. This is generally deemed uncomfortable, but safety precedes comfort which justifies it in this case. When L4 is not available, the only remaining option is to make an emergency stop (*ES*).

When fatigue is non-critical, the fatigue is not expected to cause an unsafe situation in the near future. However, the DL preemptively tries to reduce the fatigue, to anticipate on a later stage in which the fatigue might become critical. Fatigue can only be reduced when it is not physiological of nature but rather task-related, meaning the fatigue is caused by an under- or overload of tasks. The only way of knowing whether fatigue is physiological or task-related is by trying to correct it and evaluating if that works.

Therefore, the first thing the DL does is trying to improve the fatigue (*CF*). If that succeeds, it assumes the fatigue was task-related and if future fatigue occurs it may be able to reduce it again. If improving the fatigue fails, the fatigue is assumed to be physiological and in a future stage no more attempts to improve the fatigue are made, since this is assumed to be ineffective.


*Figure 3.8* Illustration of difference between task-related and physiological fatigue.

Figure 3.8 illustrates the difference between physiological and task-related fatigue. The driver starts out with a fatigue level of 1, so the DL tries to correct this (e.g., by giving the driver some artificial tasks). That succeeds, so it is assumed the fatigue was task-related. A little while later, driver fatigue increases again, so the DL tries to correct it again. This time it fails, so it is assumed the fatigue is now physiological (being tired). Therefore, after that the DL no longer tries to correct the fatigue.

### 3.3.3. Driver request

Finally, the last module handles driver requests. This is relatively straightforward: when the driver is fit and the requested level is available, a suggestion to shift to that level is made (*SSL<X>*), which is likely to be accepted by the driver since it was a request. As soon as the switch is made, the request is cleared (*CR*). When the level is not available, the request stays open until the driver cancels it or the level becomes available.

## 3.4. Enhancements

Based on the data and feedback received from partners in the MEDIATOR project, the decision logic is enhanced with features that improve comfort and safety of the driver. Both features are related to the timing of actions. They are a window of opportunity, discussed in Section 3.4.1, and the option to add parameters to actions, discussed in Section 3.4.2.

### 3.4.1. Window of opportunity

The way the DL interacts with the HMI is that it suggests an action that can be taken right now, and the HMI is then expected to execute this action. However, there are situations in which it may not be desirable to immediately execute an action, and instead wait for a better moment. A good example is when suggesting a shift to a different level to the driver, it can be unsafe and uncomfortable when suggesting this while the driver is busy driving (e.g., making a turn). Instead, it would be better to wait until the driver is less occupied with driving such that he has overcapacity to deal with suggestions from the HMI, and then suggest the shift.

This is where the Window of Opportunity (WOO) comes in. It provides additional information to the HMI, informing it of the latest moment the action the DL suggested can still be initiated. This gives the HMI a window from the moment the action is suggested until the latest moment given by the DL in which it can choose the optimal time to initiate the action.

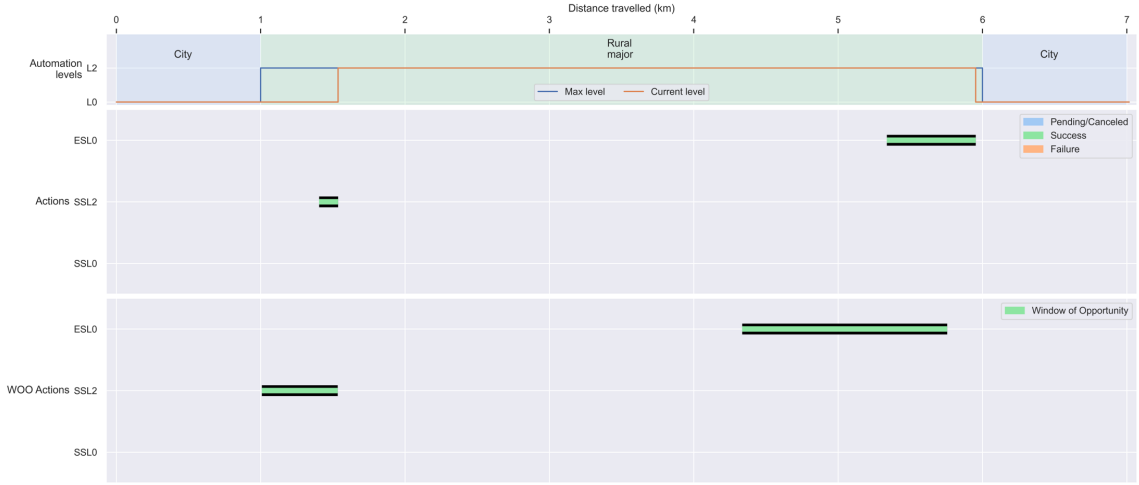


*Figure 3.9* Example of Window of Opportunity (WOO).

Figure 3.9 exemplifies how the WOO works. Level L2 becomes available for a long stretch and as such the DL wants to suggest a shift to L2 to the driver. The WOO (depicted in the third subplot in the figure) defines the window within which the *SSL2* action can be started according to the rules of the DL. Currently, an arbitrary moment within the window is chosen, but in reality this moment should be a moment where the driver is not busy with other driving tasks. Similarly, when enforcing the shift back to L0 at the end of the rural road, another window of opportunity is offered within which *ESL0* can be initiated.

### 3.4.2. Action parameters

Another important aspect of the interaction between DL and the HMI is the total time an action is active. For instance, when you suggest a shift to a different level this can be active for 10 seconds,

after which the suggestion disappears again. This time should also be displayed on the HMI such that the driver knows he must respond within the given timeframe.

However, in some cases it is desirable to have a dynamic timeframe instead of a static timeframe as in the example above. One example is when an automation degradation event is coming up and the DL wants to enforce a shift level on the driver. In such a case, the time the action can be active (which is the maximum time the driver must take over control) depends on the time left before automation becomes unavailable. As such, a dynamic timeframe for the action is preferred.

To this end, the DL now has the option to add parameters to an action, which are then passed onto the HMI. The HMI can then use these parameters however is desired. In the case above, which is currently the only use of parameters, the maximum timeframe for an *ESL* action is calculated by the DL and added as a parameter to the action.

Now that the decision logic has been explained in detail, the next chapter evaluates its performance, by running many different scenarios and measuring key performance indicators. The evaluation also includes a comparison with the decision logic from a previous deliverable, to see if the changes that have been made have improved its performance. Furthermore, its robustness is tested through an extensive sensitivity analysis.

# 4. Testing and evaluation

**To evaluate the performance of the decision logic we run tests on different scenarios in the simulator, and assess the performance through predefined key-performance indicators (KPIs). Tests include a sensitivity analysis that determines how well the system deals with noisy (i.e., wrong) observations, the impact of using optimistic and pessimistic estimates for available automation levels, and a comparison with the performance of the previous version of the DL.**

## 4.1. Setup

Before the actual experiments are discussed, various concepts related to the setup of the experiments are discussed. All experiments are run on a Lenovo P1 Gen 3 with an Intel® Core™ i7-10750H CPU @ 2.6 GHZ (up to 5.0 GHz) processor, with 6 cores and 12 logical processors, and 16GB of RAM. For each analysis, 10,000 runs are done on 5 cores. All analyses take between 5 and 10 minutes to complete with this setup.

Several key-performance indicators (KPIs) are used to evaluate the performance of the DL for the different runs. These are as follows:

- **Percentage of time driven in automation levels** shows the percentage of time that the car drove in each automation level throughout all the 10,000 runs.
- **Average time between actions** shows the average time in seconds between two actions, averaged over all actions in the 10,000 runs.
- **ES**, Emergency stops, shows the total number of emergency stops.
- **Recent switch** means that the driver was forced to shift levels twice within a short timeframe. Short is defined as the minimum time the driver prefers to drive in one level, which is set 120 seconds for all runs in this case. The KPI sums all recent switch events in the 10,000 runs.
- **Quick takeover** means that the driver was forced to take over control (i.e., ESL0) very quickly, which is deemed uncomfortable. This can happen when a dynamic event comes up, not giving the DL and driver enough time to comfortably anticipate. The KPI sums all quick takeovers in the 10,000 runs.
- **Driver unfit** means the driver was in control of the car even though he was deemed unfit (because of fatigue or a persistent distraction), which is considered an unsafe situation. For this KPI, both the count and the sum of the duration of these events is measured. This is done because there is a difference between having an unsafe situation for 1 second or having it for 10 seconds.
- **Car unfit** means the car was driving in an automation level that was not available, which is unsafe. For this KPI, both the count and the sum of the duration of these events is measured, for the same reason as with the driver unfit KPI.

Now that the main parts of the setup and KPIs have been detailed the results can be displayed and analyzed. In the remainder of this chapter the different tests are explained and discussed. This starts with a sensitivity analysis in the next section, exploring the robustness of the decision logic.

## 4.2. Sensitivity analysis

An important notion of the simulator that needs to be addressed is that all values are assumed to be correct. In reality, estimates cannot always be perfectly predicted and may have some noise. This is especially important pertaining to *Time to Automation Fitness* (TTAF), *Time to Automation Unfitness* (TTAU), *Time to Driver Fitness* (TTDF), and *Time to Driver Unfitness* (TTDU), since these are key inputs to the decision logic, and they are likely to have some noise in their calculations and/or the observations that serve as input to their calculations.

Therefore, a sensitivity analysis is conducted to evaluate the impact of wrong observations. This is done by using wrong inputs in the DL for TTAF/TTAU/TTDF/TTDU while the real values are known and are used for evaluation. The noise calculation is deterministic, and different runs are done for several noise levels. The noise runs and evaluation are split into four categories:

- $\Delta$TTAF < 0 and $\Delta$TTAU > 0: in this category a value is subtracted from TTAF (i.e., the input for TTAF in the DL is smaller than the real TTAF) and a value is added to TTAU (i.e., the input for TTAU in the DL is greater than the real TTAU). This is expected to have a negative impact on performance, as explained in the next section.
- $\Delta$TTAF > 0 and $\Delta$TTAU < 0: this category is the opposite of the previous; it adds a value to TTAF and subtracts a value from TTAU.
- $\Delta$TTDF < 0 and $\Delta$TTDU > 0: the previous categories did not add noise for TTDF/TTDU. This category does not add noise for TTAF/TTAU, but instead subtracts a value from TTDF and adds a value to TTDU. Like the first category, this is expected to have a negative impact on performance.
- $\Delta$TTDF > 0 and $\Delta$TTDU < 0: this category is the opposite of the previous one, adding a value TTDF and subtracting a value from TTDU.

For each run with noise there is a subtracted value and an added value. The subtracted and added value are the same within a run and are called the offset from now on. For example, for the first category, an offset of 2 means that 2 seconds is subtracted from the real TTAF and 2 seconds are added to the real TTAU, and these modified values are fed into the decision logic.

### 4.2.1. Automation noise

The first two categories listed above relate to noise within the automation context. This noise will impact decisions made regarding the automation and can therefore lead to a change in the number of car unfit events. This section goes into the detail on the impact of the two different automation noise scenarios, and what this may mean for the DL.

Figure 4.1 and Figure 4.2 show the impact of an underestimation of TTAF and an overestimation of TTAU. As mentioned in the previous section, this is expected to have a negative impact on performance, which it does. The reasons for this are two-fold. First, for TTAF an underestimation means that an automation level is expected to be available before it is available. This may lead the DL to suggesting a level that is (not yet) available. Second, for TTAU an overestimation means that the DL expects an automation level to be available for longer than it will be. Therefore, the DL may wait too long with enforcing a shift to a lower level, creating an unsafe situation.

Both figures show that up to an offset of 2 seconds the impact is minor, but from 3 seconds it has significant impact. The reason for this is that the DL has a small 2 second buffer built in for most branches, which pays off.

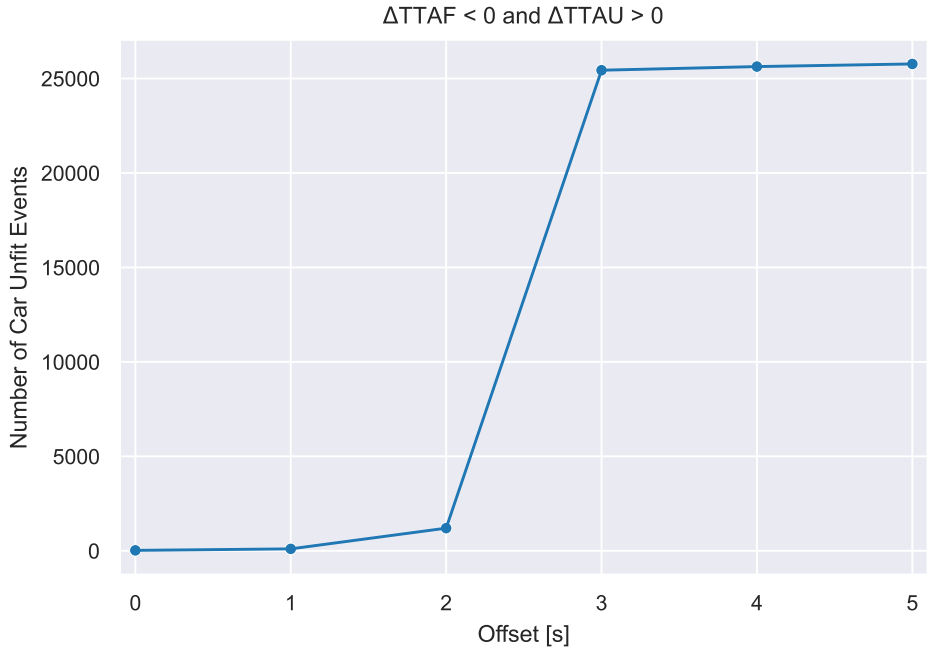


*Figure 4.1* Graph depicting the number of car unfit occurrences with different levels of underestimation of TTAF and overestimation of TTAU (offset).

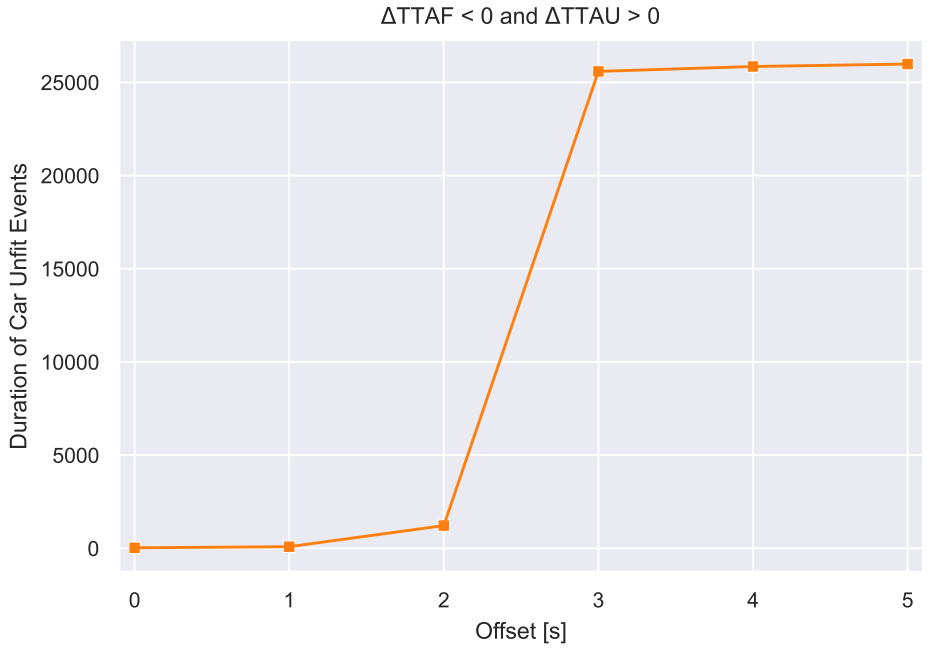


*Figure 4.2* Graph depicting the total duration of car unfit occurrences with different levels of underestimation of TTAF and overestimation of TTAU (offset).

An error in observation of a few seconds does not seem unlikely and therefore these results show that the DL is currently not robust enough to deal with noise. One way to deal with this would be to add buffers within the decision trees to handle this. Another, more simple but possibly also more

robust way, would be to counteract any negative noise by artificially adding time to TTAF and subtracting time from TTAU, respectively.

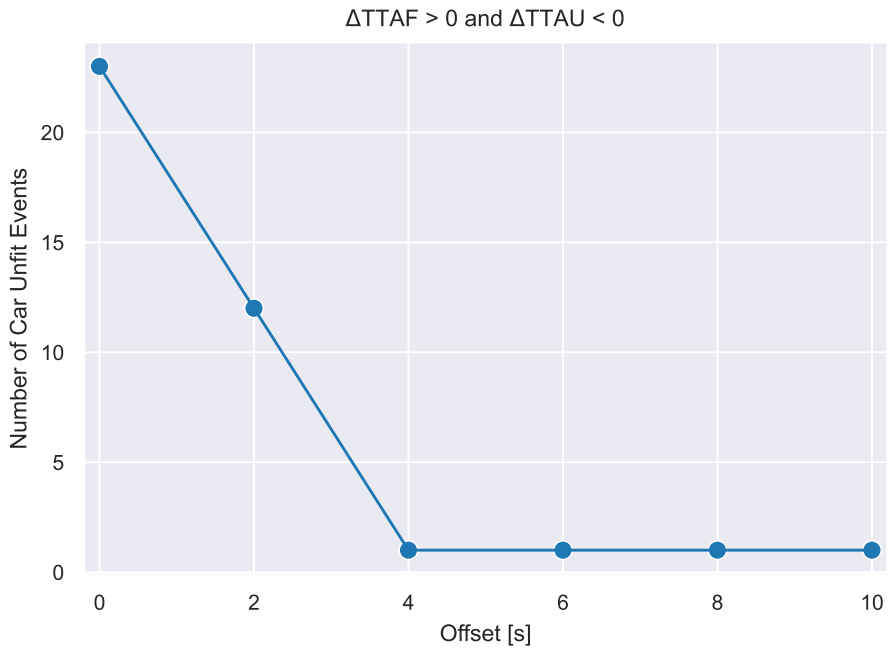


*Figure 4.3* This figure shows the impact of overestimation of TTAF and underestimation of TTAU on the number of car unfit events.

Figure 4.3 shows the impact of an overestimation of TTAF and an underestimation of TTAU. Contrary to the opposite noise, this noise has a positive impact on performance of the DL, as can be seen in the figure, since with the increase of noise the number of car unfit events decrease. In this case the noise acts as a buffer. It will cause the DL to always err on the safe side of fitness and unfitness of automation levels. It also shows what a solution of counteracting previous noise configurations can look like, by manually adding time to TTAF and subtracting time from TTAU.

### 4.2.2. Driver noise

The last two categories listed in the introduction of Section 4.2 relate to noise within the driver context. Examples of this are the driver context module wrongly assessing the distraction or fatigue level, or the calculation of the resulting TTDF and TTDU. The values of TTDF and TTDU based on distraction and fatigue levels are a lot more subjective than the calculations of TTAF/U, since it is difficult to predict how long it will take before a driver really becomes unfit to drive because of a distraction. As such, noisy observations are to be expected in this context. The remainder of this section discusses how these noisy observations impact the performance of the DL.

Figure 4.4 and Figure 4.5 show the impact of an underestimation of TTDF and an overestimation of TTDU on the number of times a driver is unfit in 10,000 runs and the sum of the duration of all these occurrences, respectively. For the same reason as for the offsets with TTAF and TTAU this is expected to have a negative effect on safety. Results show that it does, and this negative effect starts from an offset of 2 seconds. Reason for this is that a small 1 second buffer is built into the DL.

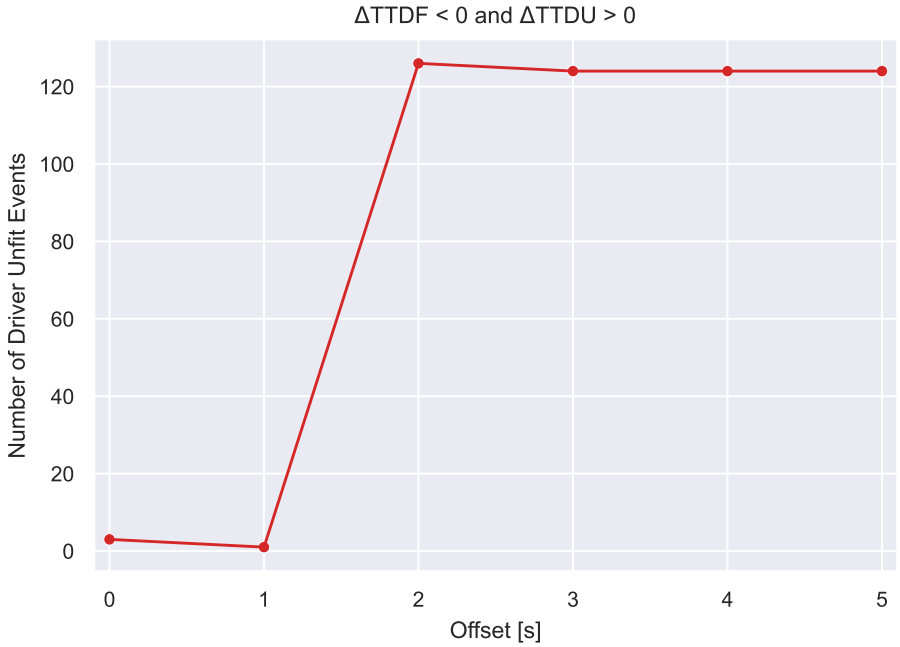


*Figure 4.4* Graph depicting the number of driver unfit occurrences for different levels of underestimation of TTDF and overestimation of TTDU (offset).

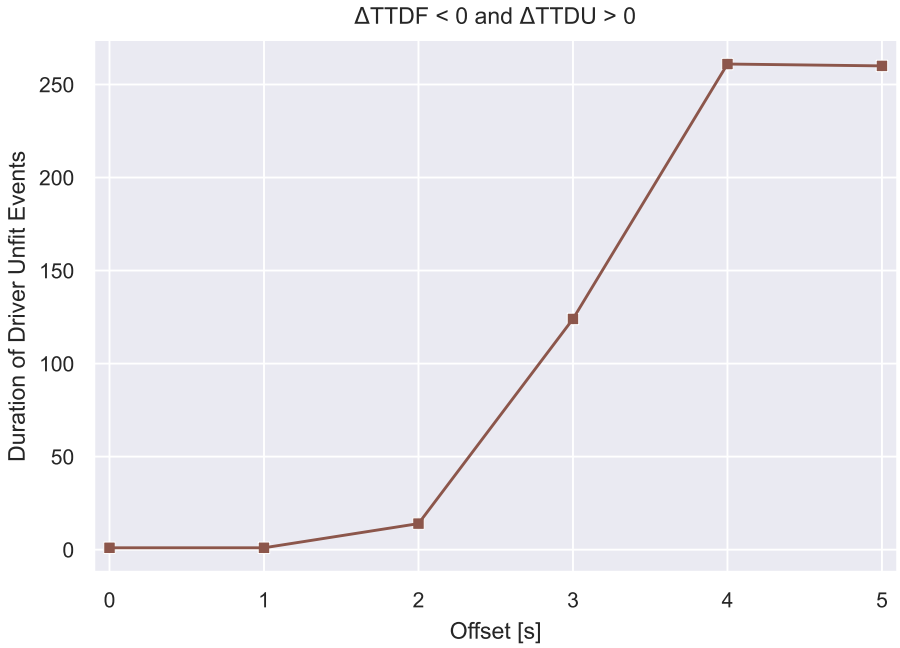


*Figure 4.5* Graph depicting the total duration of car unfit occurrences with different offsets for a negative TTDF delta and a positive TTDU delta.

An offset of 2 seconds for TTDF and TTDU is very small, since it is difficult to predict how long it will take before a driver will really be (un)fit to drive. Realistically, larger prediction errors can be expected. Therefore, it is of paramount importance to err on the safe side and add a large buffer to

counteract the potential prediction errors or make conservative predictions which accomplishes the same result.

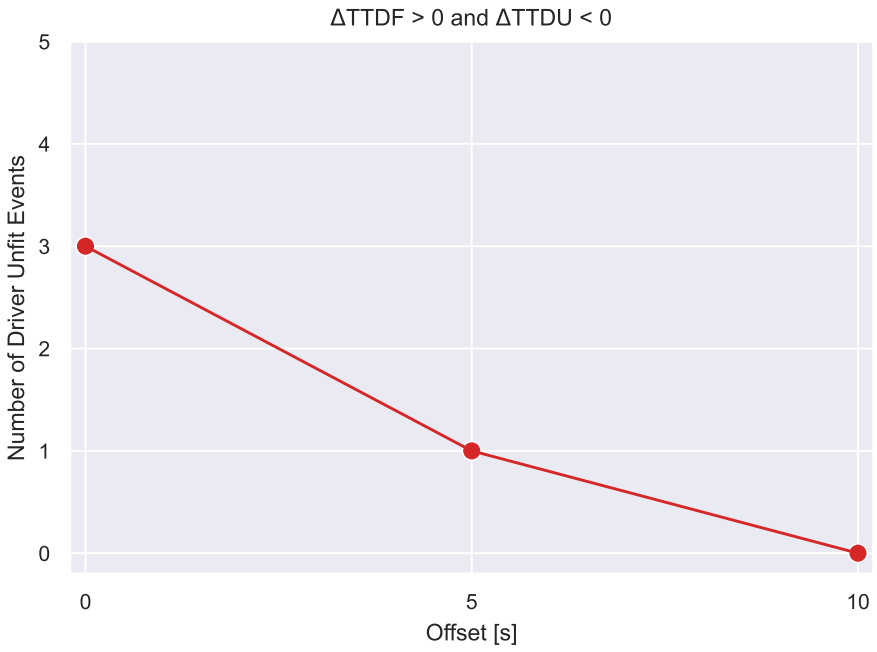


*Figure 4.6* This figure shows the impact of overestimation of TTDF and underestimation of TTDU on the number of car unfit events.

Figure 4.6 shows that an overestimation of TTDF and an underestimation of TTDU results in fewer unsafe situations. This noise can be seen as a buffer or conservative TTD predictions and shows how that favours safety.

The sensitivity analysis shows that noise can lead to an increase in unsafe situations, and one of the goals of MEDIATOR is to increase driving safety, making avoidable unsafe situations unacceptable. Therefore, we suggest using conservative estimates or a buffer for all values related to automation and driver (un)fitness times. A drawback of doing this is that consequently the DL will also act more conservatively. This means it may miss opportunities to increase driver comfort, e.g., it will not suggest a shift to a level up in a conservative approach while realistically the higher level might be available for a long enough period. Ideally, the buffer is as small as possible to reduce this drawback. This requires the estimates to have a low margin of error. More data and testing are needed to determine the current error margin, from which an appropriate buffer value can be deduced.

## 4.3. Decision logic comparisons

The development of the decision logic has seen two major iterations. Based on feedback, the initial version (dubbed *Baseline*) has been improved to result in the final version (dubbed *Default*). This section analyzes how these changes impact the performance of the DL on several KPIs. Furthermore, optimistic and pessimistic estimates for available levels are fed as input to the DL to determine whether this improves performance. Additionally, based on results in the previous section, buffers that are expected to reduce unsafe situations are added to assess whether they

do, in fact, improve safety. Finally, the DL and DL with buffers is run on the real route from which data exists, to get a sense of the performance of the DL in a real-world scenario.

### 4.3.1. Decision logic changes

The results from the previous milestone (*Baseline*) and the current one (*Default*) are compared. Both are run with the same configuration, and Table 4.1 displays the results. The following results stand out:

- Car related safety slightly increased, which can be deduced from the numbers for car unfit events. Consequently, the number of emergency stops (*ES*) also increased. Another consequence is that there is slightly more time driven in L0 now. Both results may be more uncomfortable for the driver but also shows that the DL errs more on the safe side now, since safety is prioritized over comfort.
- The previous version of the simulator and decision logic had one issue with a distraction when it occurred at the same moment the car would switch levels down from L4 to a L2 or L0. In those rare cases the DL would not detect the distraction, and as such, it would go on for a long time. This explains the 4 driver unfit events with a long duration in the baseline version. That issue has been resolved, and as such these occurrences are now instantly handled.

### 4.3.2. Optimistic and pessimistic level estimates

A notion that came back as feedback from the partners about the Mediator system was that they would like to have optimistic and pessimistic estimates for available levels. To comply with this, optimistic and pessimistic levels are implemented in the simulator and their impact on DL performance is evaluated. The way it works is that for road events (e.g., traffic jam) a realistic, optimistic, and pessimistic available level is given. The realistic level is defined by the config file, and the optimistic and pessimistic levels are one higher and lower than the realistic level, respectively. For example, when the realistic expected level for a traffic jam is L2 (i.e., the highest level the car can drive in is L2), the optimistic level is L3 and the pessimistic level is L0. Because of these different levels, different TTAF and TTAU levels are also calculated for the three cases.

In the case of the results for optimistic levels it means that the optimistic levels and corresponding TTAF and TTAU values are used throughout the simulations. However, the assumption is made that the realistic levels are the truly available levels (i.e., the optimistic levels are wrong) and these are used to evaluate the KPIs. The results in Table 4.1 show that there is a big increase in car unfit events when optimistic levels are used. This is as expected, because there are many situations where the DL assumes the car can drive in a level that is not available. Therefore, using optimistic levels is only a viable option if these estimates are guaranteed, but it can be argued that the levels are equivalent to the realistic levels.

In the case of the pessimistic levels, the corresponding TTAF and TTAU values are fed into the DL, while again assuming the realistic levels are the available levels. Since using the pessimistic estimates is a conservative approach, a decrease in unsafe situations related to the car can be seen in the results. Thus, in terms of safety this can be seen as a preferred approach over using the realistic levels. However, one drawback is that the percentage of time driving in L0 is higher than in the default case, which is assumed to be less comfortable for the driver because he has to spend more time driving himself.

| | Percentage of time driven in automation levels | | | | Avg. time between actions (s) | ES | Recent switches | Quick takeovers | Driver unfit events | | Car unfit events | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | L0 | L2 | L3 | L4 | | | | | Count | Duration (s) | Count | Duration (s) |
| **Default** | 63.8 | 14.9 | 13.7 | 7.5 | 154.1 | 220 | 7876 | 3005 | 3 | 1 | 23 | 26 |
| **Baseline** | 62.6 | 14.8 | 15.1 | 7.5 | 131.8 | 178 | 13358 | 2359 | 4 | 1028 | 29 | 27 |
| **Optimistic levels** | 28.7 | 47.5 | 15.2 | 8.7 | 152.3 | 206 | 6604 | 4969 | 6 | 2 | 68958 | 2921159 |
| **Pessimistic levels** | 71.1 | 13.1 | 7.2 | 8.5 | 171.1 | 217 | 6954 | 2451 | 6 | 0 | 16 | 16 |
| **Optimized offsets** | 64.4 | 14.8 | 13.4 | 7.3 | 155.3 | 320 | 7689 | 3391 | 2 | 4 | 1 | 0 |

*Table 4.1* Comparison of different test runs on KPIs for DL.

### 4.3.3. Optimized offsets

From the results of the sensitivity analysis combined offsets for TTAF, TTAU, TTDF, and TTDU can be gathered that are likely to have a positive impact on safety. These offsets are +4 and -4 for TTAF and TTAU, respectively, the turnover point in Figure 4.3 from which there are no car unfit events any longer. For TTDF and TTDU the offsets are +10 and -10, respectively, as Figure 4.6 shows to be the point at which no driver unfit events exist.

The final row in Table 4.1 shows the results for these offsets which can be seen as a buffer. It shows that unsafe events related to the car are almost eliminated. Related to the driver, the total duration of unsafe events increased, which is not as expected. Closer inspection of the two cases reveals that these are rare cases where both an unsafe car and unsafe driver situation is coming up, simultaneously. Without the buffers, the unsafe driver event is handled first, causing an unsafe car situation. With the buffers included, the reverse happens. Therefore, despite the slight increase in unsafe driver situations, we still recommend using buffers for both driver related and car related time values.

### 4.3.4. Sweden route

Finally, the default DL and the DL with built-in buffers (the same as mentioned in Section 4.3.3) are tested on the real route that is parsed into the simulator from the tabular route (as described in Section 2.2). This gives an indication of the performance of the DL in the real world, although results should be verified by test runs in a physicial environment.

Table 4.2 shows the results. It shows that the time driven in L0 and L2 is equally divided in the default case. With buffers (the second row) the time in L0 is a little higher than the time in L2, which can be explained by the fact that the buffers make the DL more conservative. The average time

between actions is almost 5 minutes, which seems like a reasonable time. This metric is also influenced by driver preferences, so if the time between actions is deemed too short this can be indirectly tweaked by using different driver preferences.

The recent switches are similar between both runs. These are mainly caused by dynamic events; if there are no dynamic events, the DL will not suggest a switch to a higher level if it expects to have to switch back soon (which is a recent switch). Dynamic events cannot be predicted and as such recent switches cannot be prevented. Quick takeovers are also mainly caused by dynamic events because these require the driver to quickly take over control. The fact that these are significantly higher in the case of using buffers is odd, and we do not have a good explanation for that at this moment. It is something to investigate further in the future.

Finally, looking at safety, there is not much difference between the two methods. However, both are safe in over 99.9% of cases. The few unsafe situations related to the car have a duration of 0 seconds, meaning they are resolved instantly as they occur. The driver unfit events do have a longer duration, but these cannot be prevented because making an emergency stop is not available in this car. Therefore, if correcting distractions or fatigue is unsuccessful, no actions remain.

All in all, these results show that the DL enables the driver to drive in a (semi-)autonomous mode for 50% percent of the time, while assuring safety. It will be interesting to see if the results in real drives are similar.

| | Percentage in Levels | | Avg. time between actions | Recent switch | Quick takeovers | Driver unfit events | | Car unfit events | |
|---|---|---|---|---|---|---|---|---|---|
| | L0 | L2 | seconds | count | count | count | seconds | count | seconds |
| **Default** | 50 | 50 | 283.9 | 2323 | 2162 | 6 | 15.0 | 3 | 0 |
| **Optimized offsets** | 50.5 | 49.5 | 284.2 | 2315 | 3019 | 5 | 13.0 | 3 | 0 |

*Table 4.2* Results of two test runs on the real route, run in the simulator.

## 4.4. Reinforcement Learning

From a research perspective it is interesting to assess whether Reinforcement Learning (RL) can be used as the basis for a decision-making system, instead of the decision trees that are used now. The main advantage of using RL compared to logical reasoning is its generalizability. With logical reasoning all inputs and outputs need to be known beforehand in order to construct a logical construct for each possible scenario. That also means that a lot of manual work is required to build these constructs when the input and/or output space is large. Furthermore, when the environment changes slightly, the logical reasoning needs to be adapted to handle this.

In theory, RL is better able to handle these issues. An RL agent learns a policy by gaining experience in the environment and getting feedback on its actions, and if the RL system is designed well the agent is able to take the right actions after a period of training. When the environment changes slightly, the RL agent may still be able to take the correct actions, provided that it has been trained under similar circumstances. Logical reasoning, on the other hand, is not

able to handle this by itself because it will not have a rule for any input that was not explicitly previously defined. With these potential advantages in mind, an RL agent was trained and tested for decision-making in MEDIATOR.

The results displayed here are based on work by Yang Liu.[1] It is beyond the scope of this report to go into detail on the background of reinforcement learning and the methods used here. The interested reader can refer to the work of Liu. In this work, an RL agent is trained on a simplified environment in which only driver degradation occurs. The agent is only trained on the distraction use case (meaning the driver never gets fatigued but does get distracted) and tested in an environment where the driver can get both fatigued and distracted. This is compared to a decision tree that only handles distraction (i.e., the tree from Section 3.3.1).

The results show that in 100 runs, RL has **15 unsafe runs (15%)**, while the decision tree that can only handle distraction has **52 unsafe runs (52%)**, which can all be attributed to the tree not being able to handle driver fatigue. This indicates some of the potential RL can have to generalize better, but generalization of RL to severely non-stationary environments is non-trivial and an active area of research.

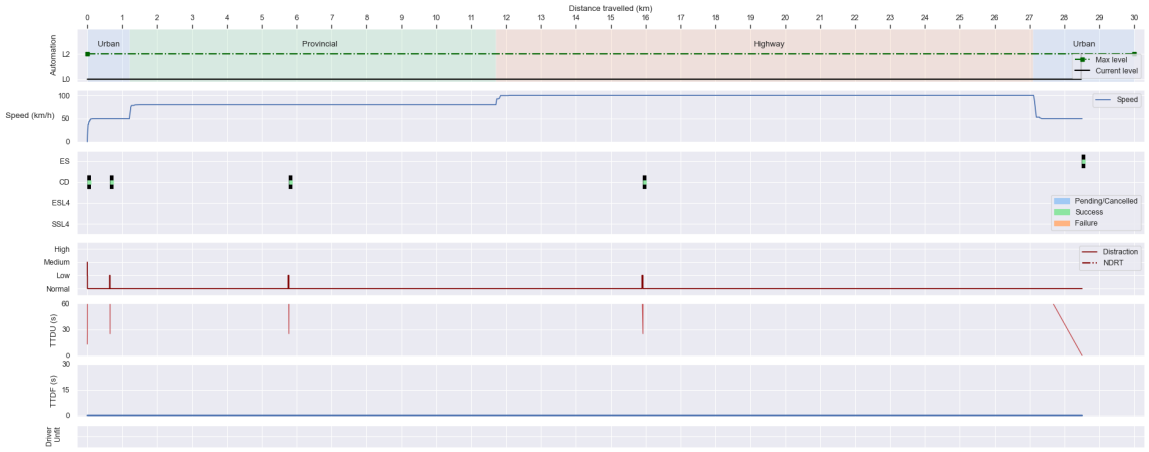


*Figure 4.7* Illustration of the RL agent making an ES when TTDU drops close to 0 because of fatigue.

[1] Liu, Y. (2022). *Hierarchical Clustering Based State Abstraction in Reinforcement Learning*, Delft University of Technology, Delft.

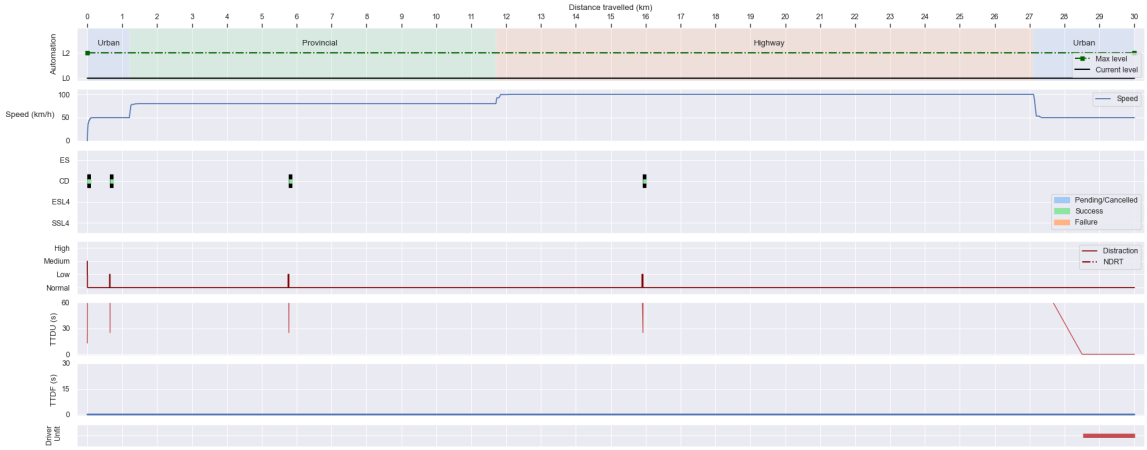


*Figure 4.8* Illustration of the same scenario as the previous figure, but now with the distraction tree as decision-making agent. Shows that the distraction tree is not able to handle TTDU dropping to 0 because it is not caused by a distraction.

Figure 4.7 and Figure 4.8 show an example of the difference in behaviour between the RL agent and the distraction tree. The tree is unable to handle TTDU dropping close to 0 because it is not caused by a distraction, while the agent is able to handle this scenario because it learned to act based on TTDU, regardless of the cause.

The images depicted above show promise for RL. There is a big caveat, however. The RL agent is still unsafe in 15% of runs with a simplified scenario where only driver degradation is considered, compared to decision logic that is unsafe in less than 0.1% of runs in scenarios that are much more complex. That is a large gap to bridge and it will require further development, for instance focusing on safe RL methods, to achieve similar performance to decision trees on complex scenarios.

# 5. Concluding Remarks

This report focuses on the development and evaluation of a simulation environment and a decision-making system, implemented through decision logic (DL), and how it has been improved using data from drives in the real world. In this chapter the main conclusions regarding these topics are listed.

First, the simulation environment is an effective way to test the DL. Many bugs and logical fallacies have been eliminated through extensive tests of the DL in the simulation environment. Furthermore, real world data improved the realism of the simulator concerning driving and automation data. Data regarding the driver is not available at the time of writing. Having this data and using it to improve simulation of driver behaviour would be a good addition to the environment, since simulating the driver is the most difficult and most subjective. Therefore, it is expected that this part of the simulation currently has the largest gap with reality.

Second, through extensive discussions with and feedback from various stakeholders in this project, meaningful additions have been made to the decision-making system. Most notable are the *Window of Opportunity* which allows the *Human-Machine Interface* (HMI) to choose the best moment to communicate an action recommended by the DL to the driver. This is expected to increase driver comfort and safety. Another meaningful addition was the implementation of dynamic parameters for actions, which enables dynamic runtimes for actions, based on the current situation. This is also expected to increase driver comfort. Both features could be tested and refined in the simulation environment, showing the benefits of the simulator.

Third, a sensitivity analysis showed that it is important to think about adding buffers for DL input. Since the DL is reliant on the input it gets, it will make wrong decisions when the input is wrong, which could lead to unsafe situation. Therefore, it is crucial to gather more data about the accuracy of the data provided by the different modules into the Mediator system, and to define appropriate buffers based on the level of accuracy.

Finally, real data enabled us to implement the route that is driven in real life with the Veoneer vehicle into the simulator. This helped to improve the simulator and gives a realistic expectation of how the DL will behave when implemented in the car. This looks promising since the drives were safe and supposedly increase the comfort of the driver. The next step will be to implement the DL into the vehicle and assess its behaviour and how the driver likes it. When this data is then fed back to the simulator it can help to refine it further and make it even more realistic.