Information and Communication Technologies (ICT) Programme

Project Nº: H2020-ICT-2016-1-732105

D6.4: Smart Travelling Demonstrator (Final version)

Lead Beneficiary: TNO

Work Package: 6

Date: 23/12/2019

Distribution - Confidentiality: [Public]

Abstract:

This document contains the description of the M36 Smart Travelling demonstrator. The description is based on the skeleton defined in D6.1 and includes the scope and purpose of the demonstrator, the description of the demonstrator itself and the accomplished results till now.

The CERBERO Consortium is the following:

Num.	**Beneficiary name**	**Acronym**	**Country**
1 (Coord.)	IBM Israel – Science and Technology LTD	IBM	IL
2	Università degli Studi di Sassari	UniSS	IT
3	Thales Alenia Space Espana, SA	TASE	ES
4	Università degli Studi di Cagliari	UniCA	IT
5	Institut National des Sciences Appliques de Rennes	INSA	FR
6	Universidad Politecnica de Madrid	UPM	ES
7	Università della Svizzera italiana	USI	CH
8	Abinsula SRL	AI	IT
9	Ambiesense LTD	AS	UK
10	Nederlandse Organisatie Voor Toegepast Natuurwetenschappelijk Onderzoek TNO	TNO	NL
11	Science and Technology	S&T	NL
12	Centro Ricerche FIAT	CRF	IT

For the CERBERO Consortium, please see the http://cerbero-h2020.eu web-site.

Document Authors

The following list of authors reflects the major contribution to the writing of the document.

Name(s)	**Organization Acronym**
Joost Adriaanse, Coen van Leeuwen, Toon Albers	TNO
Bas Pijnacker Hordijk	S&T
Antonella Toffetti, Francesco Palma and Giovanni Turi	CRF

Document Revision History

Date	**Ver.**	**Contributor (Beneficiary)**	**Summary of main changes**
30/09/2019	0.1	TNO	1st Draft
19/11/2019	0.2	TNO, CRF, S&T, UNISS	2nd Draft
09/12/2019	0.3	TNO	Version for review
11/12/2019	0.3	UNISS	Review
12/12/2019	0.4	TNO	Final version for review
21/12/2019	0.5	TNO	Update after review AS
23/12/2019	1.0	TNO	Final version

Table of contents

1. Executive Summary

This document describes the final Smart Travelling demonstrator of M36, being one of the three CERBERO use-cases.

1.1. Structure of Document

In Section 2 the scope and purpose of the demonstrator is described. Section 3 includes the description of the developed demonstrator. In Section 4 the tests, results and feedback are outlined. Section 5 summarizes our conclusion and Section 6 provides the references.

1.2. Related Documents

The CERBERO deliverables related to this document are:

- D2.1 – Description of Scenarios (Final version)
  - The smart travelling demonstrator is based on the use case scenario as defined in D2.1
- D2.2 – Technical Requirements (Final version)
  - The development of the demonstrator contributes to satisfy and validate the requirements listed in D2.2
- D3.1 – Modelling of KPI (Final version)
  - The addressed KPIs are based on the generic list of KPIs as defined in D3.1
- D3.2 – Models of computation (Final version)
  - The models of computation are based on models defined in D3.2
- D3.3 – Cross-layer Modelling Methodology for CPS (Final version) –
  - The methodology applied for cross layer modelling of the CPS of the smart travelling demonstrator is described in D3.3
- D4.1 – Multi Layer Adaptation (Final version)
  - Overall adaptation scenario for the CERBERO project, as well as its main components, were the Smart Travelling demonstrator represents a relative high layer of adaptation.
- D4.2 – Self Adaptation Manager (Final version)

  - The methodology for CERBERO self-adaptation management as applied in the demonstrator is described in D4.2.
- D5.1 – Holistic Methodology and Integration Interfaces (Final version)
  - The methodology and interface integration used to implement the (M36) demonstrator is described in D5.1
- D5.2 – Framework Components (Final version)
  - The framework used for setting up and interfacing CERBERO tools in the (M36) demonstrator is based on the framework components described in D5.2
- D6.1 – Demonstration Skeleton (Final version)
  - The generic skeleton used to build the smart travelling demonstrator is described in D6.1

2. Scope and purpose

In this document the M36 demonstrator of the Smart Traveling for Electric Vehicles use case is described in order to demonstrate and validate the CERBERO tool-chain and methodology for deployment in a cross-layered and adaptive CPS. The CRF driving simulator is used for validation of user interaction. To extend the simulator with electric vehicle simulation and advanced driver support functionalities, the smart travelling demonstrator was defined. The demonstrator can be seen as extension of the existing CRF demonstrator, able to simulate electric vehicles and equipped with driver support functionality which requires adaptation and multi KPI optimization. The goal of the demonstrator is thus twofold: 1. demonstrate real time system in the loop simulation using electric vehicle models of TNO (battery and motor) and 2. continuous monitoring and self-adaptation by the driver assistant.

CRFs objective was to be able to extend its current driving simulator functionality in a flexible way, where CRF becomes less dependent on AVSIMULATION (the vendor of the used SCANeR software, see: https://www.avsimulation.fr/) but can quickly incorporate new functionality that AVSIMULATION cannot or does not yet provide. This will be especially valuable for the development of future electric and autonomous cars.

The M36 demonstrator provides the second iteration of the demonstrator, where the development was focused on integrating the (M18) demonstrator components in the CRF driving simulator (based on the SCANeR tool, see [SCANeR]). Furthermore, improvements of the software interfaces were made, and the integration of the HMI with the physical simulator was performed.

The goals of demonstration activities have been defined in D2.2. In the following table an excerpt of Table 4 of D2.2 is provided.

Table 2-1 –Requirement validation table

User Requirements	**Technical Requirement(s)**	**Validation within the ST demonstrator**	**Planned Month**
ST1. Develop reconfigurable extendable modular	5, 7 and 8 (see section 4.5 and table 4 in 4.6 in D2.2)	Integrate CERBERO tools (DynAA and MECA) in the car	M18 integration of tools, M36 execution of use case scenarios.

simulation environment for smart travelling driver interfaces.		simulator and show environment can be configured to support defined use case scenarios.	
ST2. Develop integrated “open” toolchain environment for development of simulation modules and their integration with focus on modular integration with existing virtual environment.	2 and 4 (see section 4.5 and table 4 in 4.6 in D2.7)	Integrate DynAA and several simulation modules into the car simulator in such a way that simulation modules can easily be added. In case of smart travelling demonstrator, the DynAA, MECA and SCANeR tools are not open source tools. The partners are however allowed to use the tools within the project.	In M18 the battery and motor models were integrated using DynAA. In M36 full integration of the HMI is added plus the driver state (tiredness) sensors for the human triggered adaptation scenario.
ST3. Development of a self-adaptation methodology with supporting tools.	20 (CERBERO-0020: “CERBERO framework SHALL provide methodology and tools for development of adaptive applications”, see section 4.5 in D2.7)	Using the DynAA and MECA tools efficient support of functional adaptation is shown, according to system (car), human (driver) and environmental triggers.	In M18 the adaptation mechanism was developed. In M36 usage of different adaptation scenarios is added and validated. Integration of HMI made driving experience more realistic for driver.

In Table 4 of D2.2 we mapped user requirements with technical requirements, while we are discussing here how (Validation within the ST demonstrator) and when (Planned Month) the validation took place along the project timeline. As seen above, the coverage of the various needs was distributed between the M18 and M36 demonstrators with significant increment from M18 to M36.

3. Description of the Smart Travelling demonstrator

For the use case Smart Travelling the goals was to enrich the CRF driving simulator with functionalities required by the electric vehicle use cases (see D2.1). For this several CERBERO tools have been used as new software components. To integrate these tools with the CRF driving simulator (based on the SCANeR software tool), a number of interfaces needed to be developed.

As depicted in the following picture, the interfaces developed for the M36 demonstrator involve both interfaces between CERBERO tools (DynAA and MECA), the HMI of the demonstrator and interfaces with the use case specific CRF tool (SCANeR).

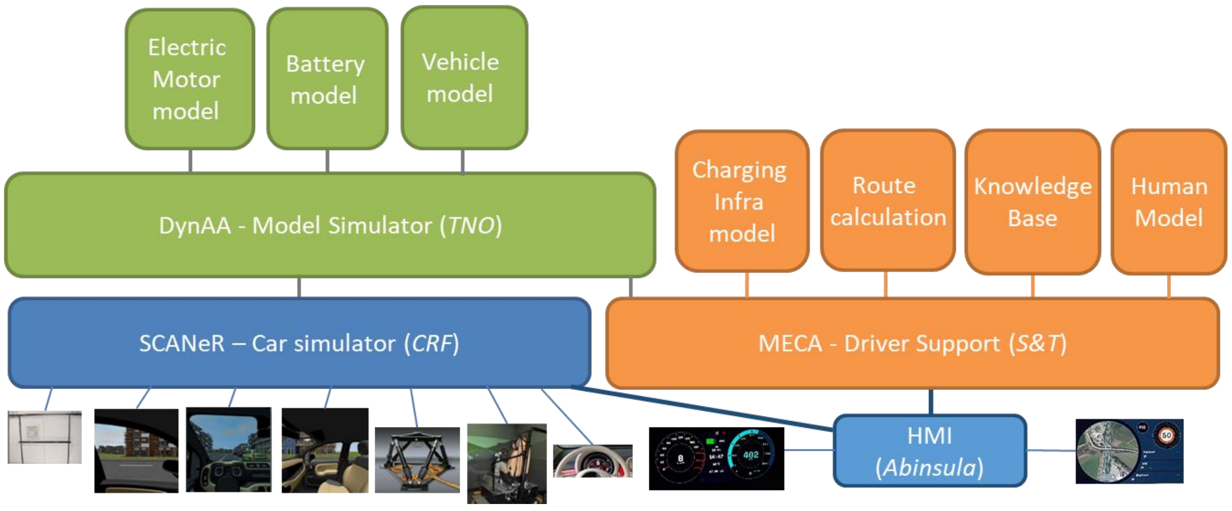


Figure 3.1: Schematic overview of the M36 ST demonstrator components and their interfaces.

As simulator the existing SCANeR based driving simulator of CRF is used, which consists of vehicle and environment simulation software and a physical motion platform and 3D projection screens (see Figure 3.2).

The high level CERBERO requirements related to the Smart Travelling use case are (see D2.2 section 4.6) are:

1. (ST1) Development of **parametric, modular and extendable** cyber-physical co-simulation environment;

2. (ST2) Development of an integrated open-source or commercially available toolchain for design space exploration and co-simulation, with **system-in-the-loop capabilities**;
3. (ST3) Development of **a self-adaptation methodology** with supporting tools.

Derived from these requirements, two main goals define what we accomplished during the development of the M18 and M36 demonstrators (see D2.2 section 2.2):

1. Verify the condition awareness capabilities of the CERBERO framework for **real-time system-in-the-loop simulation**.
2. Continuous system monitoring for **self-adaptation**.

These goals mainly cover the ST2 and ST3 requirements. To comply with the ST1 requirement, the development of the demonstrator has been done in such a way that solution is parametric (avoid hard coded algorithms), modular (so split up in logical modules) and extendable (thus can easily be extended with new functionalities, even beyond CERBERO).

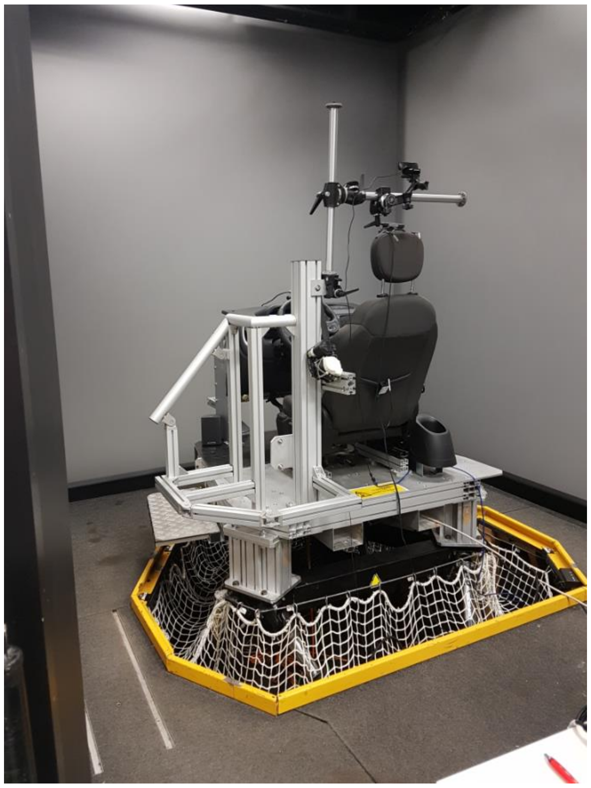


Figure 3.2: Motion platform of CRF driving simulator in Turin

Figure 3.2 shows the seat and motion platform of the CRF driving simulator. The platform was extended with HMI screens for driver interaction (car status screen and driver assistant screen) and special tiredness sensor to detect the driver status (see Figure 3.3).

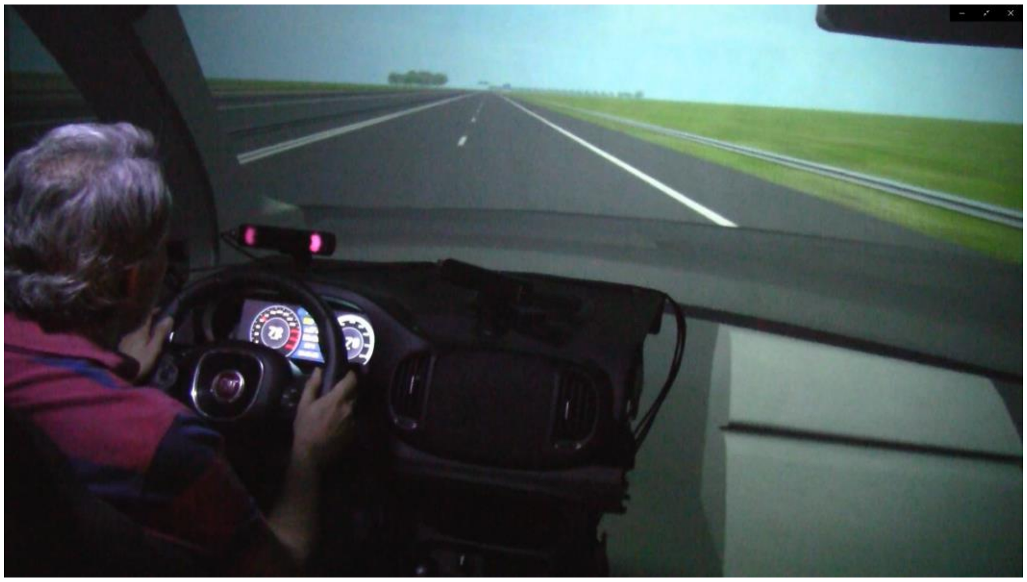


Figure 3.3: Driving simulator with eye detection sensors installed (red lights)

In Figure 3.4 the overall demonstrator setup is shown with machines for DynAA, MECA and the HMI are added during an integration test of the M36 demonstrator. The M18 demonstrator was developed on a test platform in the Netherlands (at TNO and S&T premises), whereas the M36 was integrated in the operational driving simulator of CRF in Turin.

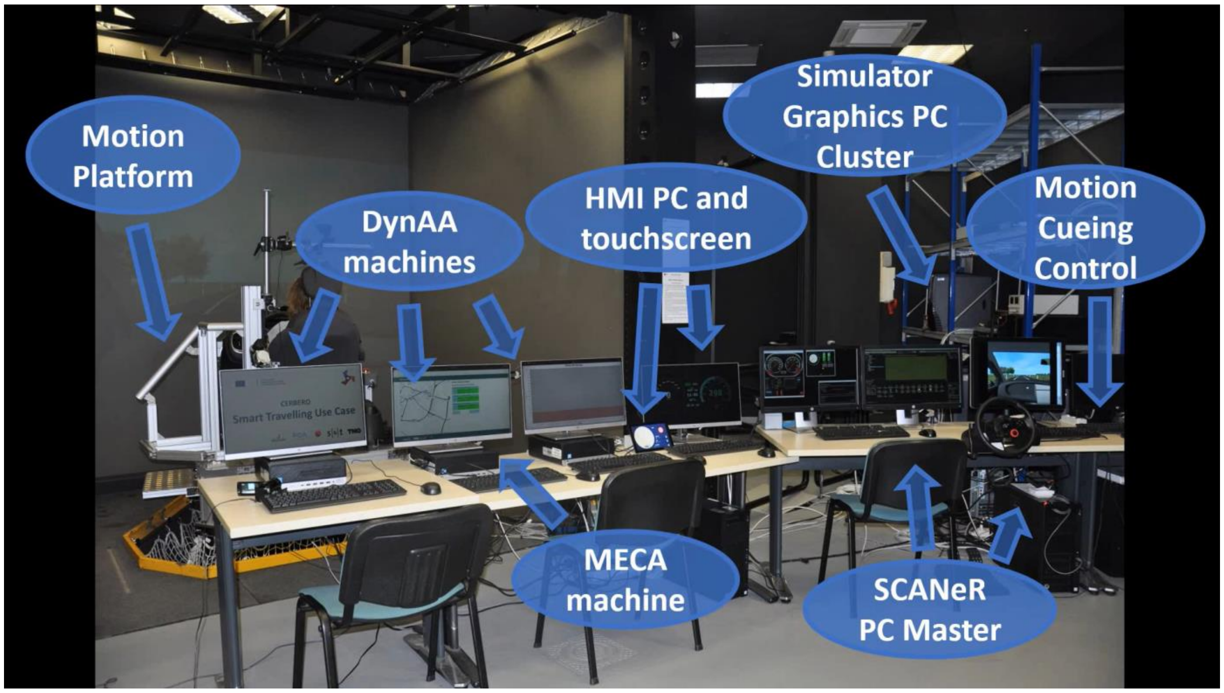


Figure 3.4: Development set-up of the M36 demonstrator at CRF in Turin

As the CERBERO use cases should be developed according to the defined CERBERO demonstration skeleton in D6.1, it was important to be able to identify skeleton parts in the use case, which could help in future reuse of the developed CERBERO tools. In Figure 3.5 below a graphical mapping is done between the use case architecture and the CERBERO skeleton (see also Section 4.1 of D6.1). For distributed simulations in DynAA multiple systems were added to be able to test parallel execution to reduce response time to requests from MECA (being one of the identified KPIs).

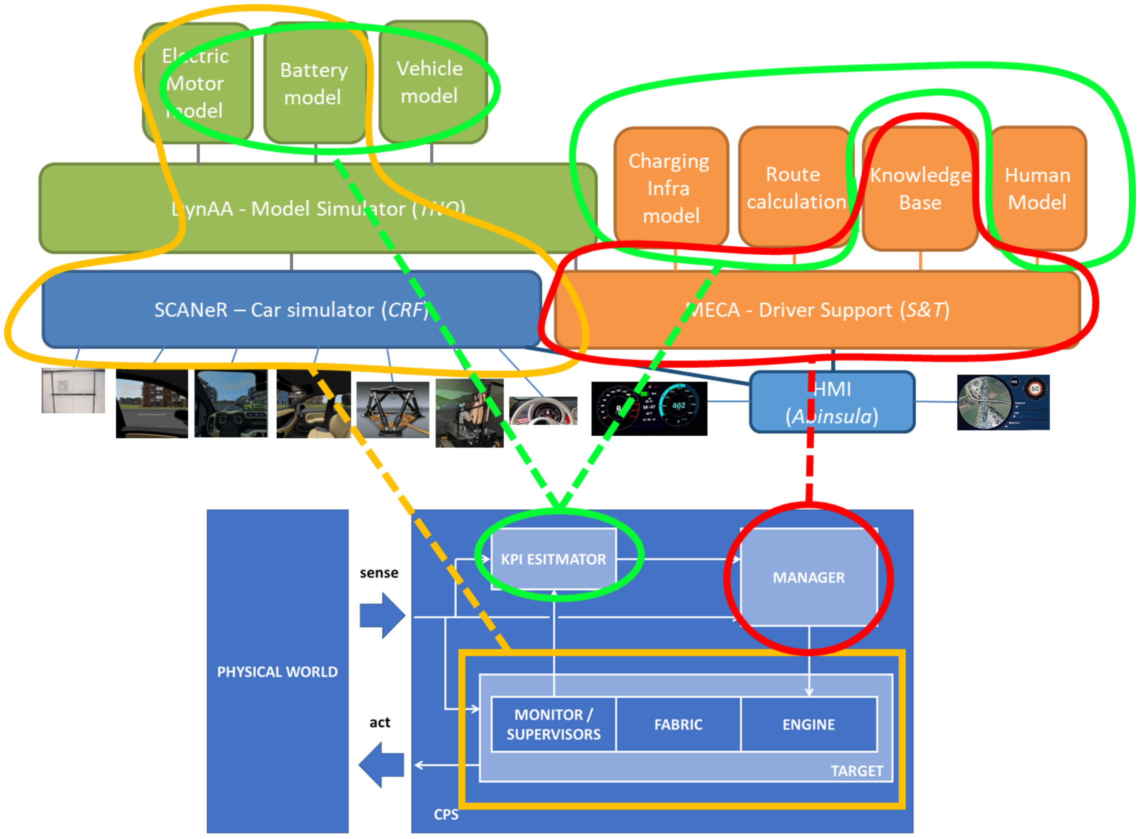


Figure 3.5: Smart Travelling use case - CERBERO skeleton mapping

We can see that the SCANeR car simulator is the main target for adaptation, while the relevant KPI (sensor) data is gathered from the car, the models in DynAA and the external environment (charging infra). MECA acts as the manager responsible for advice to the driver and triggering adaptation based on internal or external signals. In the model DynAA performs two roles, where one is the system in the loop simulation for parts of the car (battery and motor), functioning as part of the target and one role to support MECA in calculating and simulating possible itineraries to take (see D6.1 for a more elaborate explanation of the skeleton). The implemented software architecture can be seen as a cross layer multi agent software framework as addressed in D4.1.

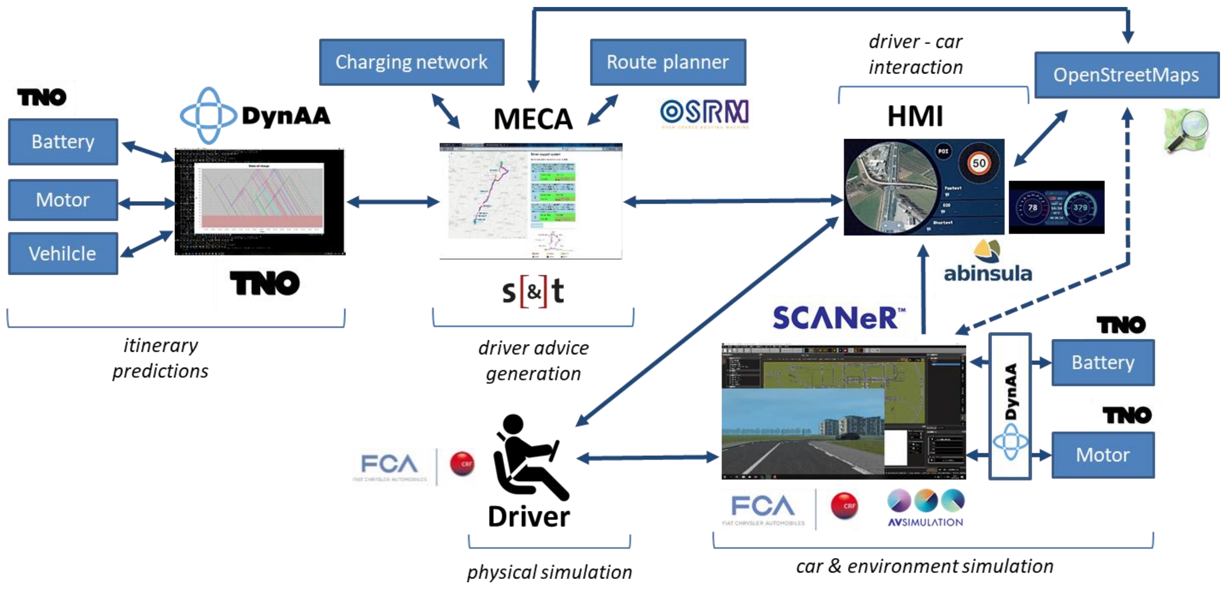


Figure 3.6: Overall component interaction M36 demonstrator

3.1. Functionalities

The implementation of the M36 Smart Travelling demonstrator splits functionalities (according to set goals and requirements in D2.2) into the following four parts:

1. Real time system-in-the-loop simulation (see 3.1.1) – resulting on work on computation models performed within WP3;
2. Self-adaptation (see 3.1.2) – resulting from work on SW agents and self-adaptation performed within WP4;
3. Parallel DynAA (see 3.1.3) – resulting from work performed on computation models within WP3;
4. Development of the data fusion application for CRF logfiles (see 3.1.4) – which was performed within WP4

The actual implementation of the demonstrator itself, which involved integration of results from WP3, WP4 and WP5, was performed within WP6. In the following paragraphs the developed functionalities of different parts of the demonstrator are described in more details.

The overall interaction of the main modules is depicted in Figure 3.6. The interfaces between the modules comprise:

A. **Car info interface between MECA and the HMI**: provides dynamic battery load, GPS, speed and other information from the car which is partly used for presentation in the HMI and forwarded to MECA for monitoring;

B. **User advice interaction interface between HMI and MECA** to:

   a) Propose and select itineraries;

   b) Advice to driver, like how to reduce energy consumption;

C. **Car info interface between HMI and MECA**: providing battery load, GPS data, speed and other dynamic data towards MECA for monitoring;

D. **Itinerary simulation request – response interface**;

E. **Real time Battery & Motor models** as system in the loop simulation;

F. **Scenario control interface**: e.g. to manipulate custom fields inside SCANeR during the simulation, like state of charge.

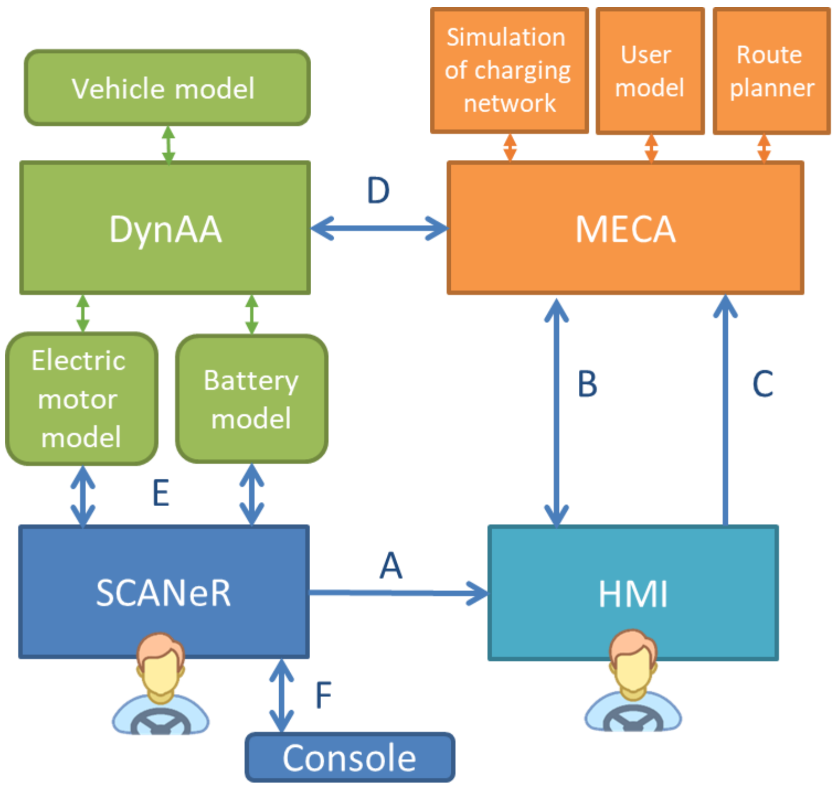


Figure 3.7: Interfaces between the M36 demonstrator components

Apart from these newly developed interfaces, there are existing user interface to SCANeR and MECA, which can be used during the simulation to influence the available functionalities.

3.1.1. Real time system-in-the-loop simulation

In the real time system-in-the-loop simulation DynAA and the TNO battery and electric motor models are used to adapt the existing (combustion car) simulator into an electric vehicle simulator, including an electric motor and a battery (see also vehicle simulation mode in Section 2.3 of D2.1).

The battery and electric motor model implemented in the SCANeR simulator consists of a combination of a battery model based on an electrical equivalent circuit model and an electric motor model based on an efficiency map.

The battery model is constructed to model both the steady-state (constant current, like charging and periods without use), as well as dynamic (typical driving) behaviour. To realize this, it comprises of a set of voltage sources, resistors and capacitors, which are parametrized for varying conditions. By this an accurate calibration can be realized for all occurring conditions during the batteries use (full temperature and SoC range). In this case the model is characterized with laboratory test data generated from Samsung SDI INR21700-48G Lithium-ion battery cells and scaled to full battery pack level. These cells are commercially available and applicable for automotive use, since they are on the top-end of the range in terms of energy density (approx. 240Wh/kg).

The electric motor model is implemented as an efficiency loss in the powertrain, the value of this efficiency is mapped by making use of an efficiency map of CRF's vehicle in the simulator, provided by CRF.

In order to facilitate a cyber-physical co-simulation environment for system-in-the-loop simulations, the existing DynAA simulation framework was extended with a real-time co-simulation module. This module consists of a set of interfaces that describe how real-time sensors or actuators can hook into the DynAA simulation environment. Without making any decisions on how to implement such sensors/actuators, the interfaces ensure that future physical environment or co-simulated systems can be added in a modular way to the framework.

The implementation was performed in the following steps:

- Extend DynAA with real time co-simulation module (see explanation above).
- Update TNO battery and motor models to support real time simulation and integration in DynAA. As DynAA is Java based and the original models were coded using C, some adaptation was needed to execute them within the DynAA environment.
- Integration test (SCANeR and DynAA) in the CRF simulation platform.
- Execute real time simulation scenario in demonstrator.

The co-simulation module requires some external trigger from the external world to synchronize the time. This external trigger should either come from the physical world or as in the Smart Travelling use case, from another simulation. The I/O module makes no assumptions on the type of external world. The external triggers/ events do not have to come at a steady period, however, to progress the simulation one step further, the DynAA simulation must receive some "heartbeat" with a timestamp value, denoting what the external time is. Note, that the external world could either be faster or slower than the real (wall-clock) time, when the external world is another simulation.

Furthermore, there are two interfaces defined by the DynAA-I/O module. One defines how sensor measurements feed into the DynAA simulation, whilst another one defines how actuators can act in the external world. The input and output interfaces are relatively straightforward, it follows the DynAA event-based conceptual design and only defines that a measurement must have some value and a corresponding time, which can be read from the DynAA point of view. The same holds for the interface of the sensor, which only defines that a particular action can be taken at a particular time from the DynAA point of view. *How* the sensor senses from the outside world, and *how* the actuator actuates in the real world is left to implement by a programmer creating a co-simulation.

Now that there is a time-synchronization mechanism and interfaces to sense and act on the external world, there is a library containing some functionality to help the user implement sensors and actuators in a correct way. For reading values from the outside world, there are two possibilities:

- either the external time is behind simulation time, in which case the sensor blocks the thread until the real world has "caught up", or
- the external time is ahead of (DynAA) simulation time in which case the value is stored in a cache for later retrieval.

For the actuation side the two possibilities are slightly different:

- The external time is behind simulation time, in which case the external action is cached until it has caught up, or
- The external time is ahead of simulation time. In this case the designer can choose either to stop simulation since it impossible to recover or accept the delay and perform the action immediately with some lag.

This means that when DynAA only takes input from the external world there is no problem when DynAA simulation is slower. However, in a meaningful and relevant co-simulation where DynAA acts (and reads) on an external world this means that DynAA should be faster than the external time. The only way to overcome such issues is if the remote world also has a time-synchronization method (some real time clock) in order to use DynAA with the real world. The ability of the reaction in this case would be limited by DynAA response time.

In the Smart Travelling use case, we made sure that DynAA did not receive an information overload, which meant it was easily capable of staying in the same pace as the SCANeR simulation.

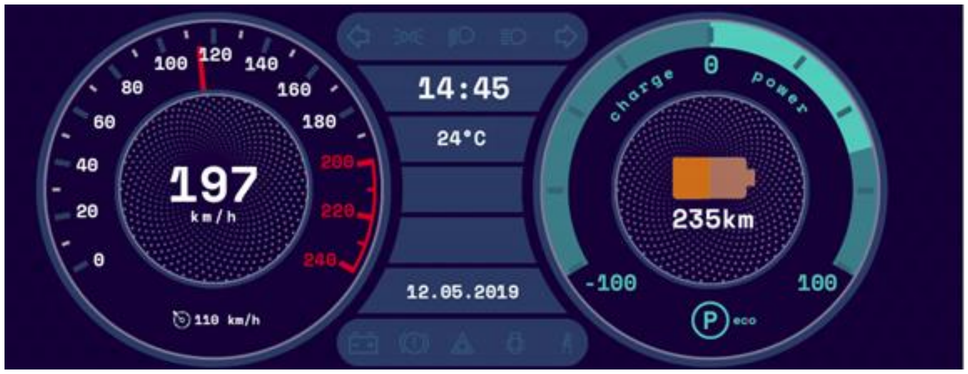


Figure 3.8: The HMI screens showing speed, power and state of charge

In Figure 3.8 the added HMI is shown. This HMI in real-time represents the current speed, the power and the state of charge of the battery (collected from SCANeR). The state of charge and power are provided to SCANeR by the TNO battery and motor models. As CRF decided power instead of voltage in the HMI (more logical to the driver), the HMI and interaction with the models in SCANeR was changed in the last version of the demonstrator to provide and show power as information to the driver. Also, an estimation of kilometre left in the battery is calculated and shown in last version of the HMI.

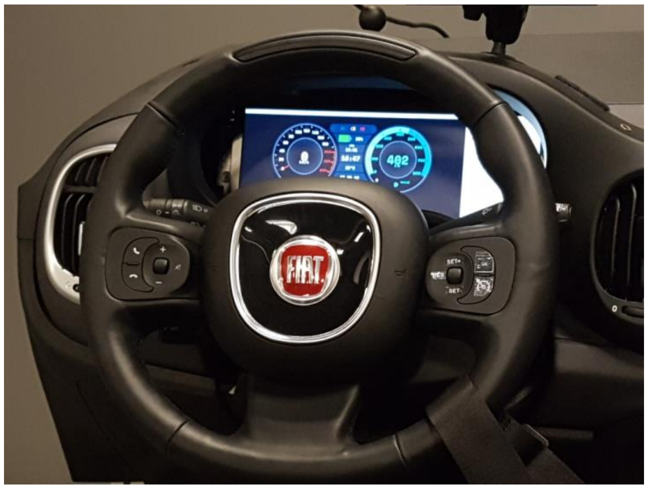


Figure 3.9: The HMI screens integrated in the simulator

The screens are integrated in the cockpit of the driving simulator to create realistic car environment for the driver, as seen in Figure 3.9.

3.1.2. Self-Adaptation

An important part of the self-adaptation functionality is the ability to calculate possible routes to the destination and estimate feasibility, costs, travel time and expected battery load for the given routes. Based on the calculation the user will be provided with a selected and ranked subset of alternatives.

At the highest level two phases are identified:

I. **Planning phase** - where driver provides MECA with starting point and destination as well as desired departure and arrival times and the system returns a limited set of calculated itineraries from which user can select the preferred one;

II. **Adaptation to off-nominal event** - where system will respond to an internal or external trigger and initiate some form of adaptation.

One of the most important activities in the development of self-adaptation was the definition of a format for routes and the design of the process to calculate routes, including the route characteristics and the charging capabilities.

When designing the adaptation mechanism and interfaces for the demonstrator, special attention was paid to the KPIs (defined in D3.1), the adaptation layers and management (as defined in D4.1 and D4.2) as well as existing open standards, specifically the Open Source Routing Machine (OSRM) standard, in order to maximize user acceptance, portability and reusability of our tools.

For the Smart Travelling use case initially the following KPIs were listed:

- *Response time to triggers*: especially important for the real time simulator mode where the TNO battery and motor models are executed and need to provide input at least 100 times a second;
- *Energy*: needed to perform the trip from origin towards destination;
- *Cost*: mainly the cost of energy obtained from the charging points;
- *Travel time*: of the route(s) towards the destination.

When a driver requests the best route to destination, he/she will require the system to reason about almost all the KPIs listed above. An optimal solution has to be provided, which will often be a compromise between travel time, costs and energy. While calculating the best routes, the system will have to ensure that enough energy is available during the drive and that the answer is given within a reasonable timeframe, which will result in requirement for limiting the search space as much as possible.

The generic process of retrieving input from the user, generating and calculating the itineraries and providing advice to the driver is depicted in Figure 3.10.

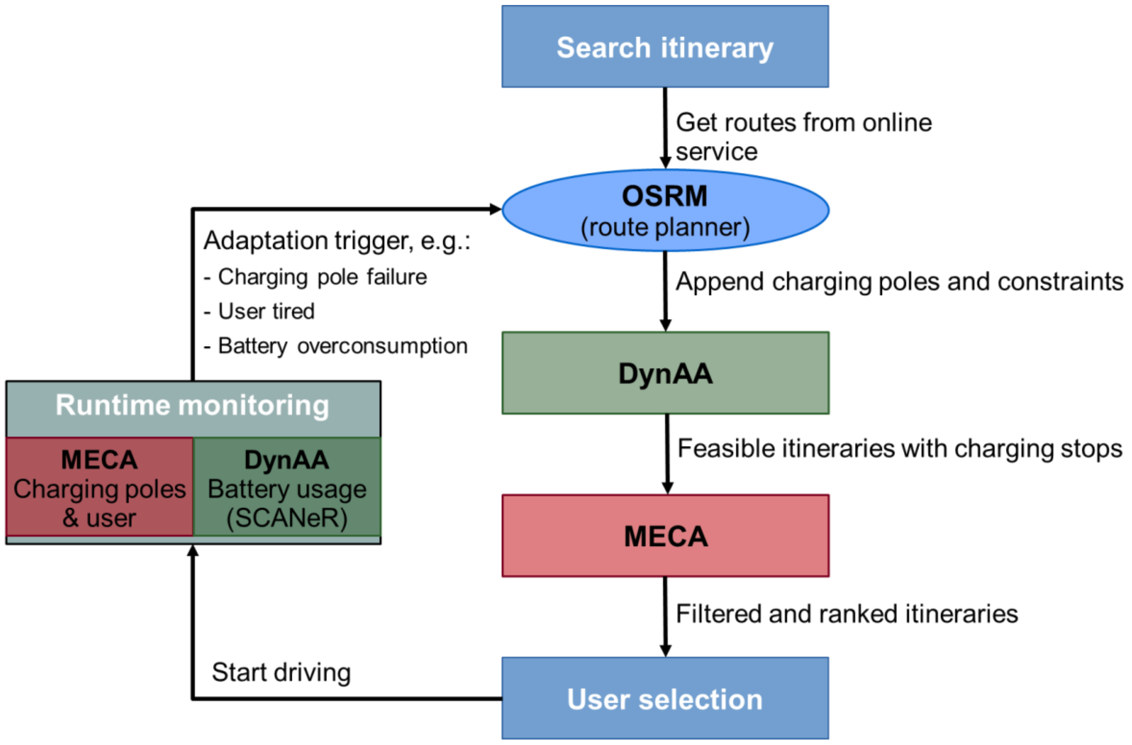

Search itinerary
Get routes from online service
OSRM
(route planner)
Append charging poles and constraints
DynAA
Feasible itineraries with charging stops
MECA
Filtered and ranked itineraries
User selection
Start driving
Runtime monitoring
MECA
Charging poles & user
DynAA
Battery usage (SCANeR)
Adaptation trigger, e.g.:
- Charging pole failure
- User tired
- Battery overconsumption

Figure 3.10: Itineraries generation and validation loop

To design the required functionalities and interfaces, it was important to first define the data structure and components of itineraries. In Figure 3.11 the route is specified, where to each segment or point specific info can be attached (like declination to a segment and opening times to a charging point) which can be used in finding the overall best solution (taking into account all relevant KPIs).

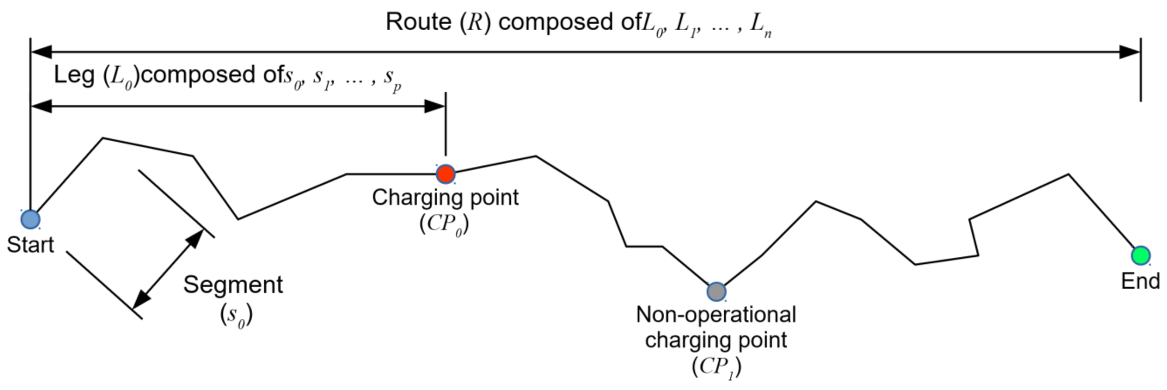

Route (R) composed of L_0, L_1, … , L_n
Leg (L_0) composed of s_0, s_1, … , s_p
Start
Segment (s_0)
Charging point (CP_0)
Non-operational charging point (CP_1)
End

Figure 3.11: Route definition

For some KPIs the system will collect user preferences, which can be used to select or rank specific solutions in order of preference. In the planning phase the flow depicted in Figure 3.12 is executed. Here the interface calls between the different tools are indicated with numbers. An important task for the M18 demonstrator was the development and test of these interfaces between the tools.

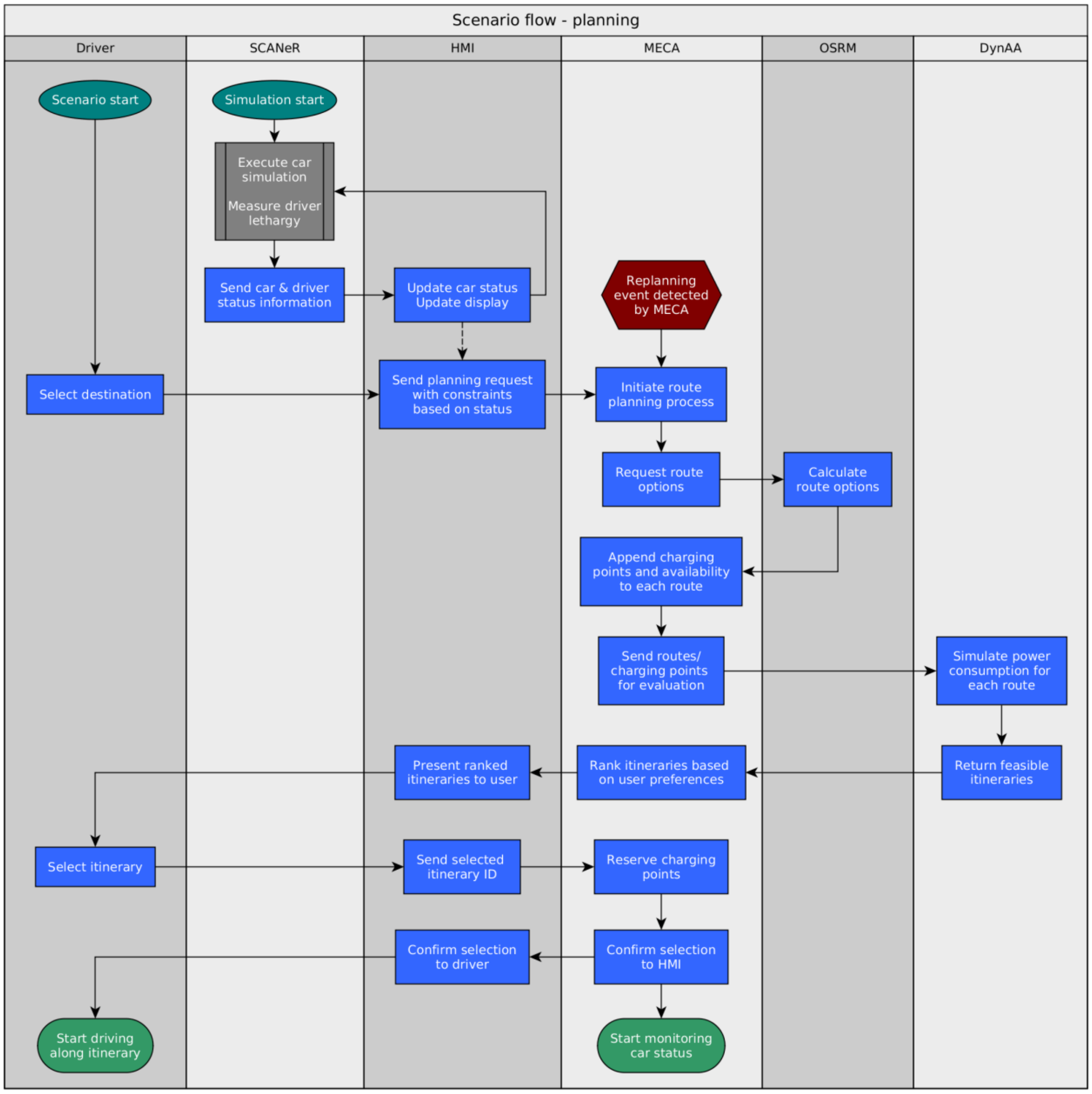

Scenario flow - planning
Driver
SCANeR
HMI
MECA
OSRM
DynAA
Scenario start
Simulation start
Execute car simulation
Measure driver lethargy
Send car & driver status information
Update car status
Update display
Replanning event detected by MECA
Select destination
Send planning request with constraints based on status
Initiate route planning process
Request route options
Calculate route options
Append charging points and availability to each route
Send routes/ charging points for evaluation
Simulate power consumption for each route
Present ranked itineraries to user
Rank itineraries based on user preferences
Return feasible itineraries
Select itinerary
Send selected itinerary ID
Reserve charging points
Confirm selection to driver
Confirm selection to HMI
Start driving along itinerary
Start monitoring car status

Figure 3.12: Phase 1: Planning

The calculation of the itineraries in the planning phase is performed using following transformation:

- **User input** – user defines starting point A, destination B, preferred departure and arrival times;
- **Route planning** – it calculates the fastest routes (R) from A to B comparing the available options, as an example please consider Figure 3.13;

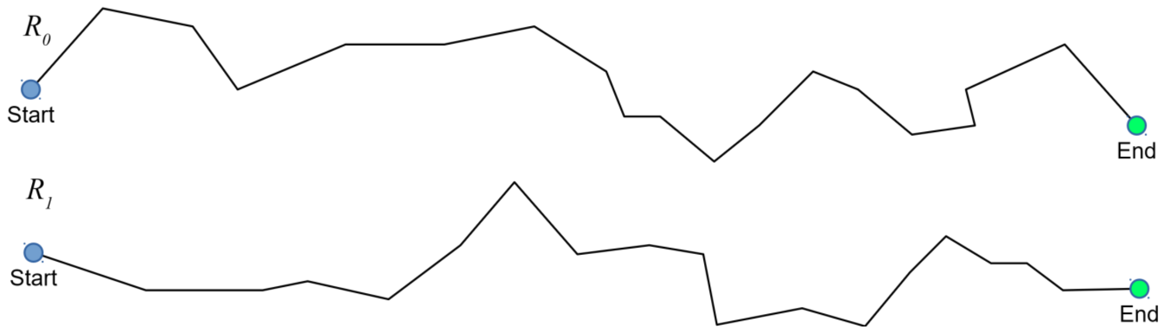


Figure 3.13: Two different routes from A to B

- **Charging point addition -** charging points (CP) along the calculated routes should be connected to points in the route with additional info like availability (opening times) and costs added to the charging points. The calculated routes are called *itineraries* within the use case, as soon as additional info (like charging poles) is added to the route, as an example please consider Figure 3.14

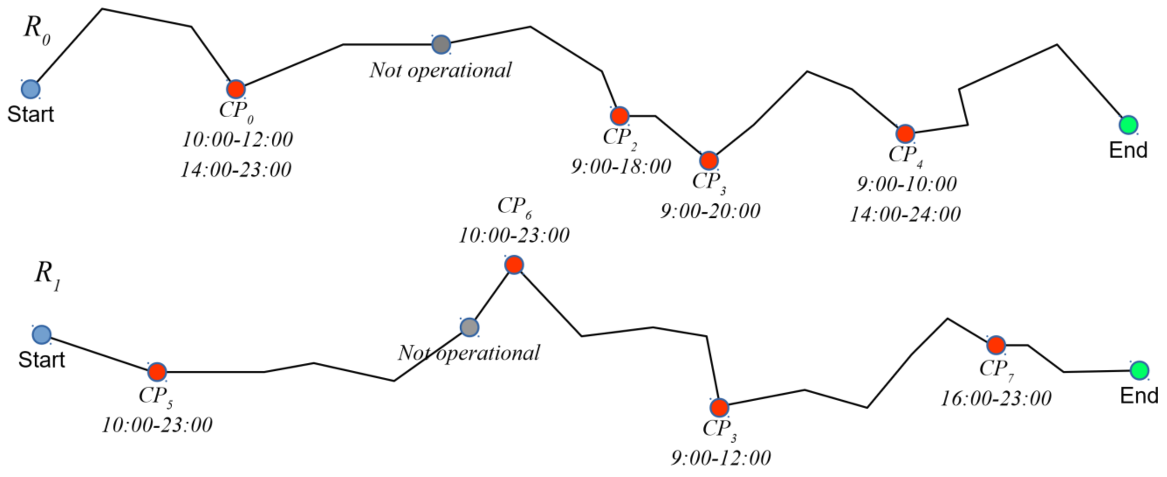


Figure 3.14: Additional of charging points along the itineraries

- **Itinerary computation** – of all the route information (including the charging points and connected information) are sent to DynAA, which simulates all given routes and verify if constraints can be met, based on the current battery level (see Figure 3.15). In case State of Charge (SoC) becomes too low, an itinerary (I) is for example not feasible. The constraints used in the calculation are:
  - Current SoC;

- Minimum SoC on arrival;
- Maximum number of stops;
- Battery capacity of the car;
- Maximum arrival time.

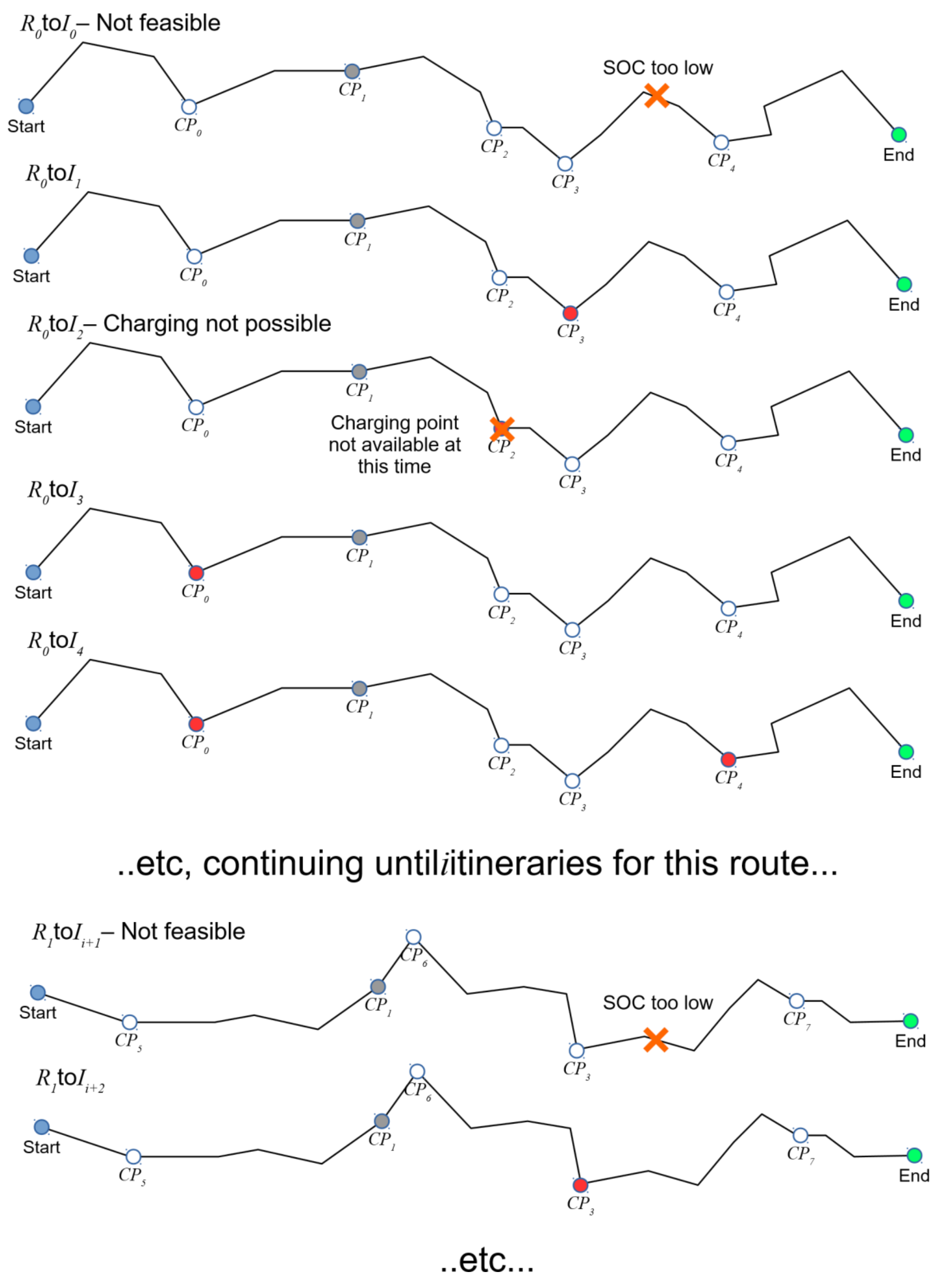

R_0 to I_0 – Not feasible
SOC too low
Start
CP_0
CP_1
CP_2
CP_3
CP_4
End
R_0 to I_1
R_0 to I_2 – Charging not possible
Charging point not available at this time
R_0 to I_3
R_0 to I_4
..etc, continuing until i itineraries for this route...
R_1 to I_{i+1} – Not feasible
CP_5
CP_6
CP_7
R_1 to I_{i+2}
..etc...

Figure 3.15: Computation of KPI results by simulating driving all itineraries

- **Ranking of routes** - based on calculations performed by DynAA and user preferences contained in MECA all the different routes are ranked as shown in Figure 3-16. In Figure 3.23 a scoring graph is shown indicating the level in which specific KPIs are fulfilled;

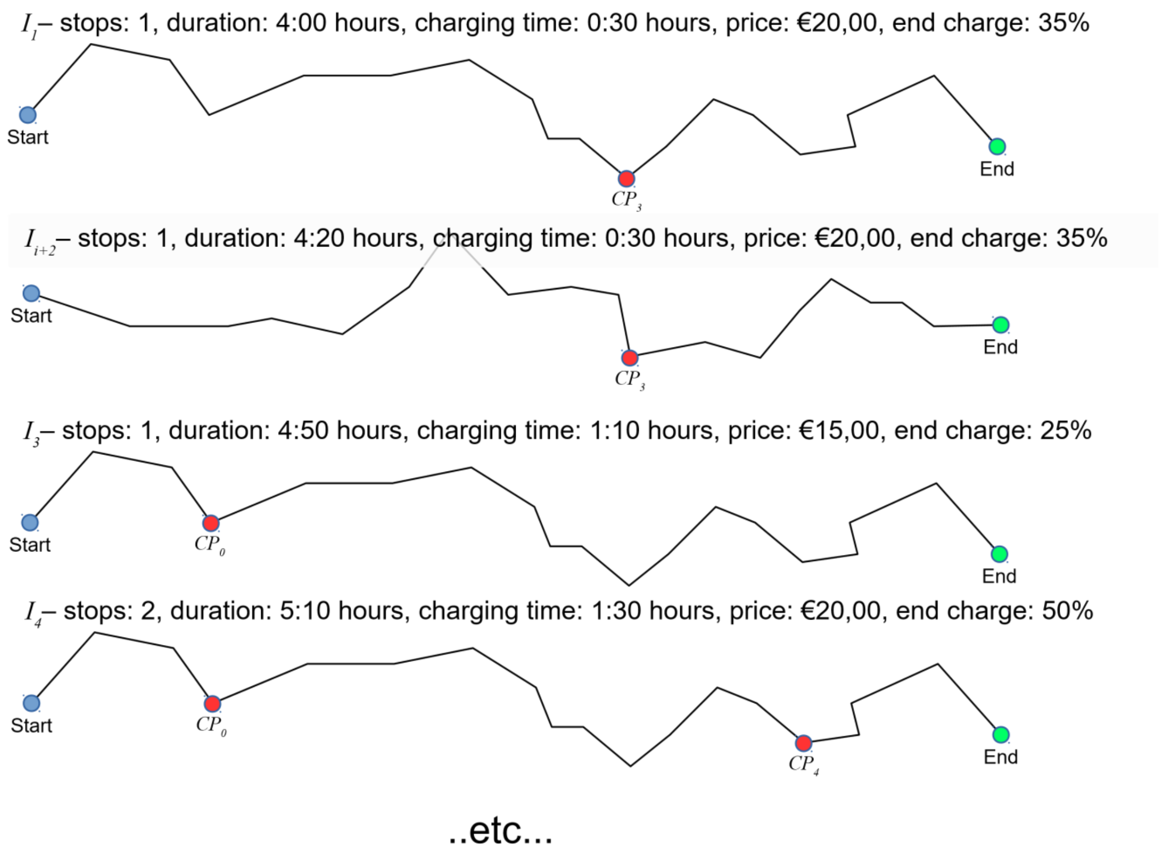

I_1– stops: 1, duration: 4:00 hours, charging time: 0:30 hours, price: €20,00, end charge: 35%
Start
CP_3
End
I_{i+2}– stops: 1, duration: 4:20 hours, charging time: 0:30 hours, price: €20,00, end charge: 35%
Start
CP_3
End
I_3– stops: 1, duration: 4:50 hours, charging time: 1:10 hours, price: €15,00, end charge: 25%
Start
CP_0
End
I_4– stops: 2, duration: 5:10 hours, charging time: 1:30 hours, price: €20,00, end charge: 50%
Start
CP_0
CP_4
End
..etc...

Figure 3-16: Ranked itineraries

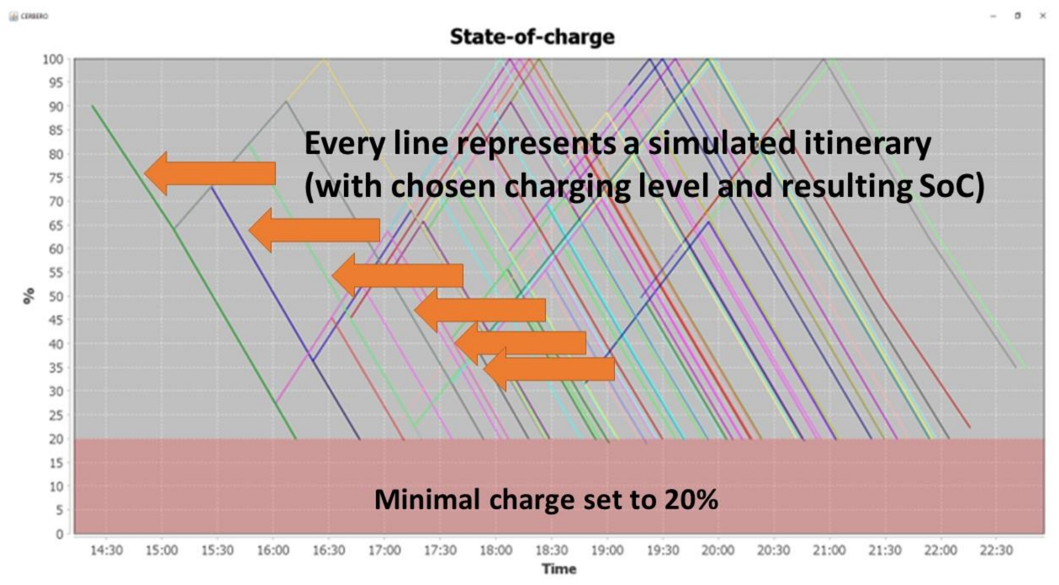


Figure 3.17: Sample of DynAA screen with calculated itineraries

To reduce the time needed to perform itinerary computation, three approaches were considered:

1. Reduce search space by only providing relevant itineraries, where MECA could try to filter all possible itineraries and only provide most likely to DynAA to calculate;
2. Shorten calculation time by splitting the problem in slices and have each slice calculated by separate DynAA instance (each on a different machine or core, see section 3.1.3);
3. Continue calculation of alternatives after the choice of itinerary by performing sensitivity analysis to determine which deviation will trigger changes and already evaluate possible alternatives for some of the cases.

As the first and third solutions would specifically be interesting for the given problem and the second solution would also help TNO to improve the application of DynAA, only the second approach was implemented within the project (see section 3.1.3). It is also expected that these types of calculation for real implementations could well be done in the cloud, where multiple servers would be available to reduce lead times. Approach one could also help to further reduce the calculation time needed.

In phase 2 the system responds to off-nominal events, which could be internal events or external events. MECA will monitor both the execution of the selected itinerary and monitor for internal or external events received from the car, the driver (or sensors monitoring the driver) and the environment. MECA will for example listen to messages received from the charging pole network (operators). In one of the scenarios a “pole out of order” event received from the charging network operator during the trip. MECA will detect that involved pole was on selected itinerary and thus initiate a re-planning of the route based on current position of the car (see Figure 3.18).

Apart from received triggers MECA also monitors the car by checking its GPS position (to see if selected route is used) and the state-of-change. DynAA returns the predicted state-of-charge for every route segment so MECA can monitor the actual state-of-charge during the trip and trigger an unexpected low state-of-charge event in case the real value deviates too much from the estimated value. MECA not only acts as the adaptation manager as defined in the CERBERO skeleton (see D6.1), but also takes a role in the KPI estimation together with DynAA (see Figure 3.5). Based on received information it will determine if a specific KPI is affected and decide to initiate a re-planning using DynAA. To ensure the itinerary is still valid given some changed circumstance, a re-planning will be initiated by MECA, which may provide new states of charge for the possible itineraries. Based on KPI calculation MECA decides if adaptation is required or not. A substantial deviation from the route on the map will for example trigger a re-planning to provide new itineraries to complete the trip. After re-planning via DynAA, MECA will decide if the driver is to be asked for the selection of another itineraries or that current itinerary still provides best solution, in which case no interaction with the driver in necessary.

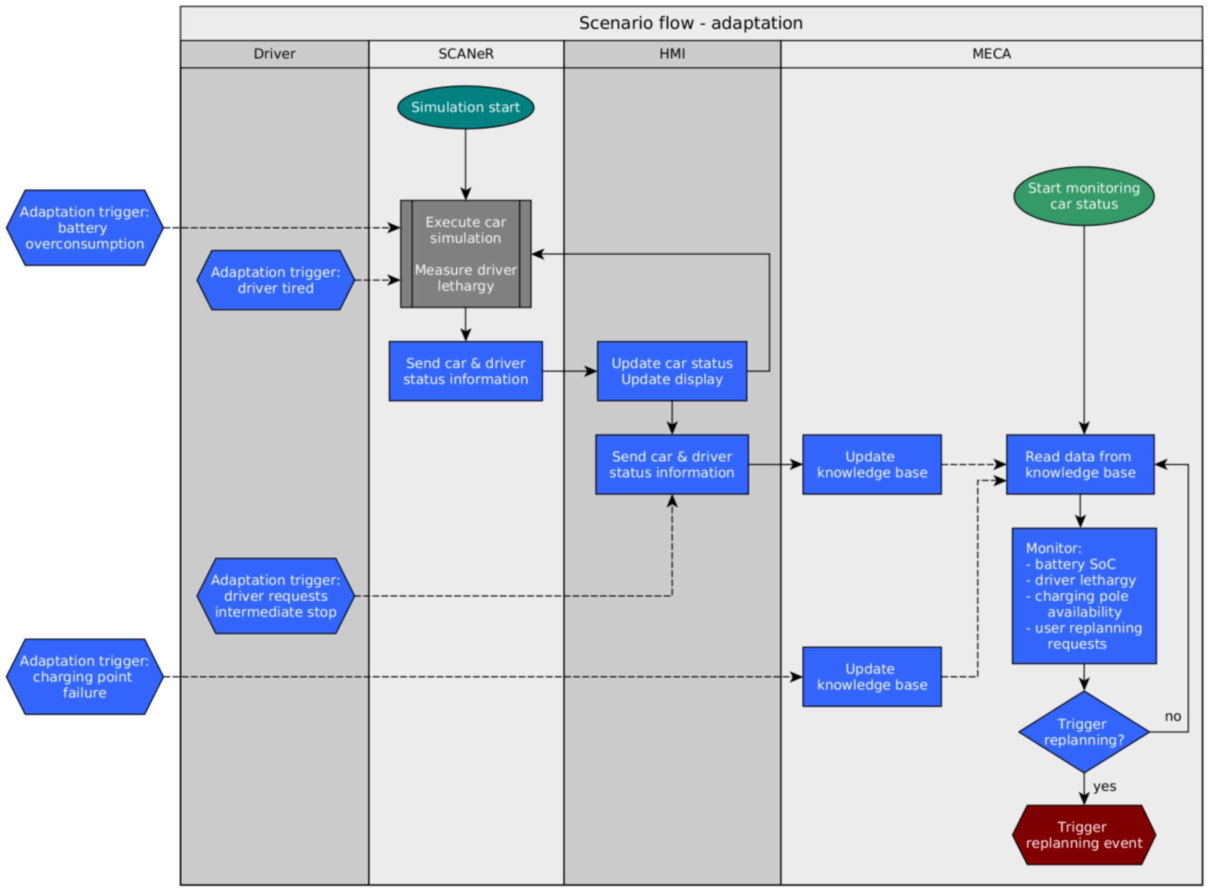

Scenario flow - adaptation
Driver
SCANeR
HMI
MECA
Simulation start
Execute car simulation
Measure driver lethargy
Adaptation trigger: battery overconsumption
Adaptation trigger: driver tired
Send car & driver status information
Update car status Update display
Send car & driver status information
Update knowledge base
Start monitoring car status
Read data from knowledge base
Monitor: - battery SoC - driver lethargy - charging pole availability - user replanning requests
Adaptation trigger: driver requests intermediate stop
Adaptation trigger: charging point failure
Update knowledge base
Trigger replanning?
no
yes
Trigger replanning event

Figure 3.18: Phase 2: Adaptation to off-nominal event

As the actual interaction with the driver of the car is important in the ST demonstrator, specific user interface screens on the HMI (added on the M36 demonstrator) are used. Through them MECA interacts with the driver. Interaction include the advice on best itineraries to use and option to select itinerary by the driver. In case situation is changed and adaptation is required, the HMI provides information on the changes and requests any choices needed by the driver to accomplish the adaptation.

To inform the driver of needed adaptation MECA sends the following status messages towards the HMI:

- an empty status message, indicating that no update is needed (but HMI knows MECA is still supporting / monitoring);
- a status message that indicates that re-planning has been invoked by MECA, and the reason for re-planning;

- a new set of itineraries that has been computed by re-planning, once the new list has been computed.

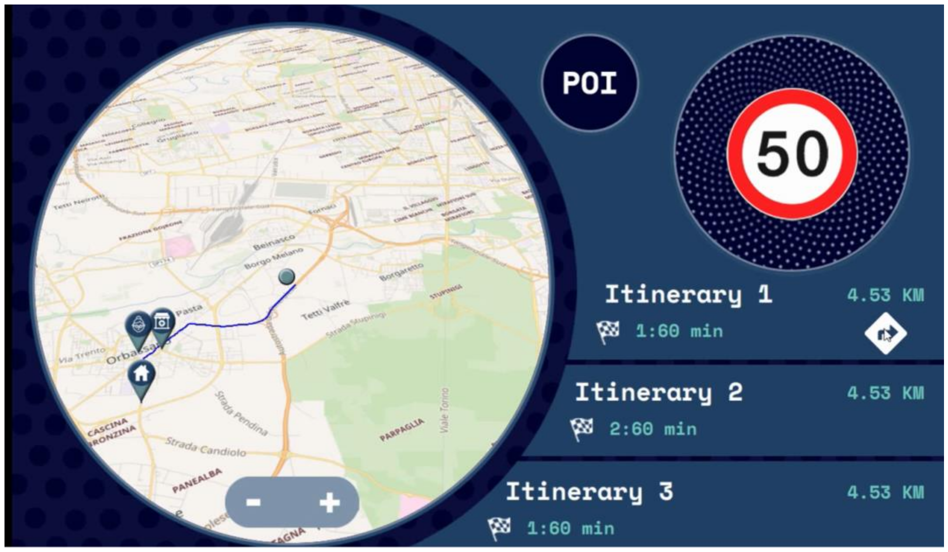
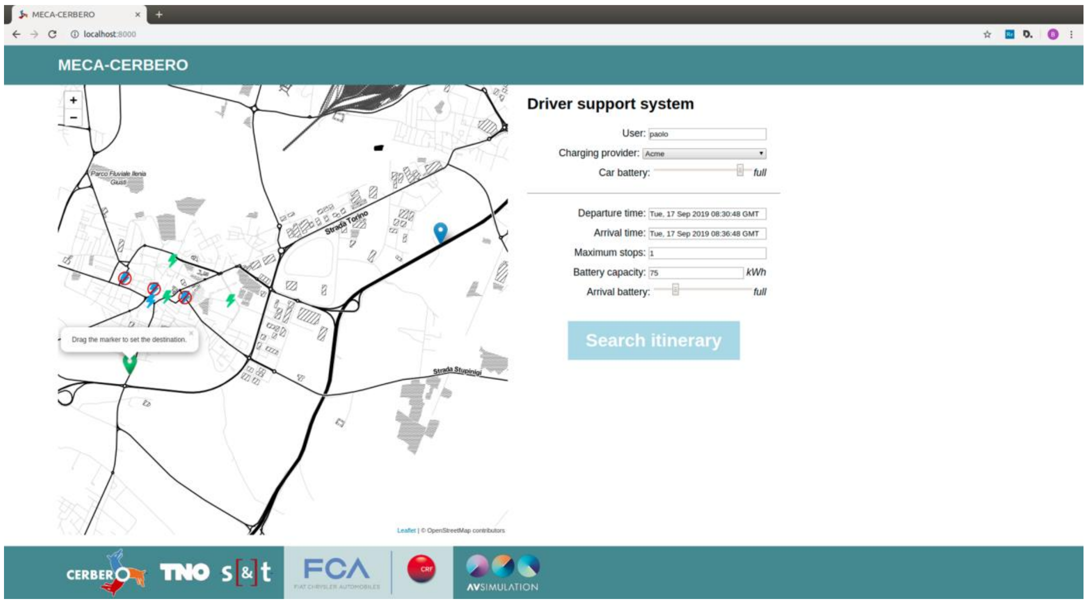


Figure 3.19: HMI interface showing map and the top three itineraries

In following figure (Figure 3.20) the MECA interface screen is shown with the information and interaction options now provide to the driver via the HMI interface.



MECA-CERBERO

Driver support system

User: paolo

Charging provider: Acme

Car battery: full

Departure time: Tue, 17 Sep 2019 08:30:48 GMT

Arrival time: Tue, 17 Sep 2019 08:36:48 GMT

Maximum stops: 1

Battery capacity: 75 kWh

Arrival battery: full

Figure 3.20: Driver support screen

On the system running DynAA the results of state of charge calculations of the itineraries are shown (each itinerary with specific color). Each colored line represents a vehicle simulation run of an itinerary (using an added faster than real time vehicle model) using the battery and motor models, resulting in specific KPI values (e.g. state of charge values through time).

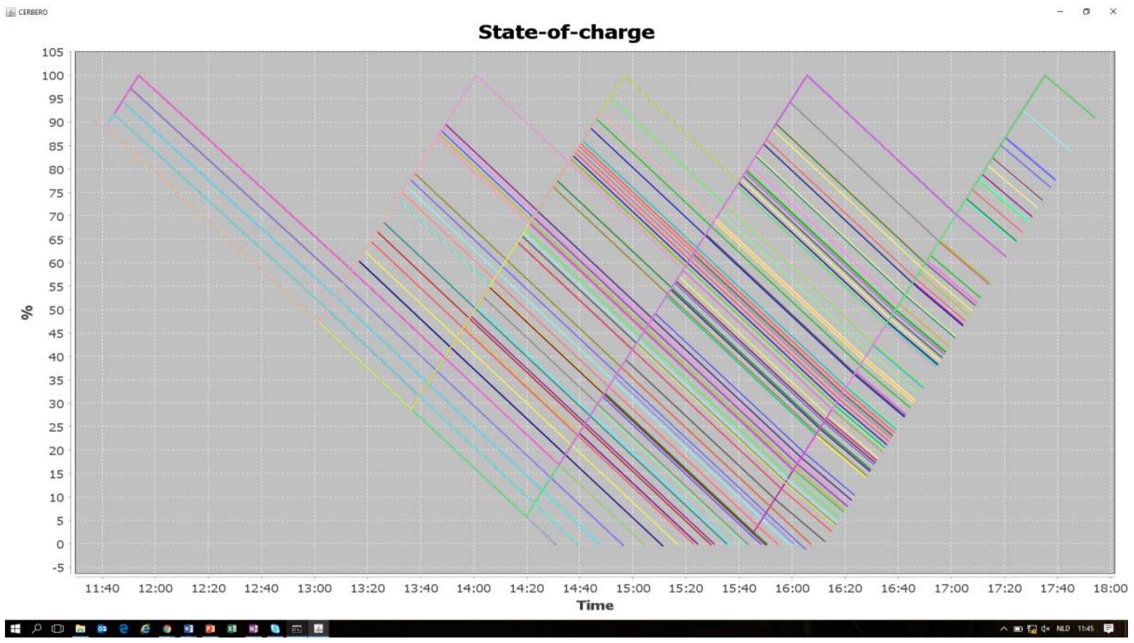
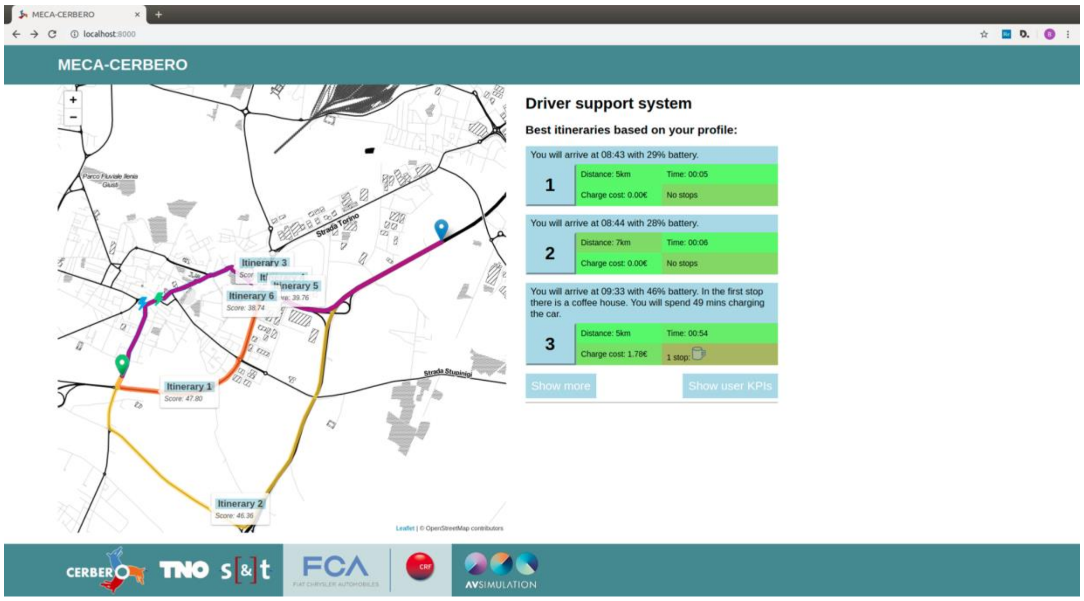


Figure 3.21: DynAA screen with state of charge calculations over time

The results can also contain results from multiple DynAA installations (machines) which helped to solve the request.

After calculation of the KPIs, one of which being state of charge, the itineraries are ranked by MECA (Figure 3.22). MECA also provides simple explanation of the itineraries to support the driver on making a choice. The operators can follow the calculations and options presented to driver via MECA status screens. The actual selection of the itinerary is performed on the HMI and then processed again by MECA, which monitors the itinerary once selected.



Driver support system

Best itineraries based on your profile:

You will arrive at 08:43 with 29% battery.

1	Distance: 5km	Time: 00:05
	Charge cost: 0.00€	No stops

You will arrive at 08:44 with 28% battery.

2	Distance: 7km	Time: 00:06
	Charge cost: 0.00€	No stops

You will arrive at 09:33 with 46% battery. In the first stop there is a coffee house. You will spend 49 mins charging the car.

3	Distance: 5km	Time: 00:54
	Charge cost: 1.78€	1 stop:

Show more

Show user KPIs

Figure 3.22: Screen with calculated itineraries

In Figure 3.23 a radar chart screen of MECA is shown with multiple KPIs scorings of a specific itinerary. The system can use stored user preferences to rate and filter calculated itineraries and rank the feasible itineraries based on the scoring and driver specific weights of the KPIs. In this way the system will only suggest best ranked alternatives to the driver, where the driver should make final choice on which itinerary to select. The KPI scoring itself will not be visible on the HMI screen of the driver. Calculations can however be viewed via the radar chart screen in MECA.

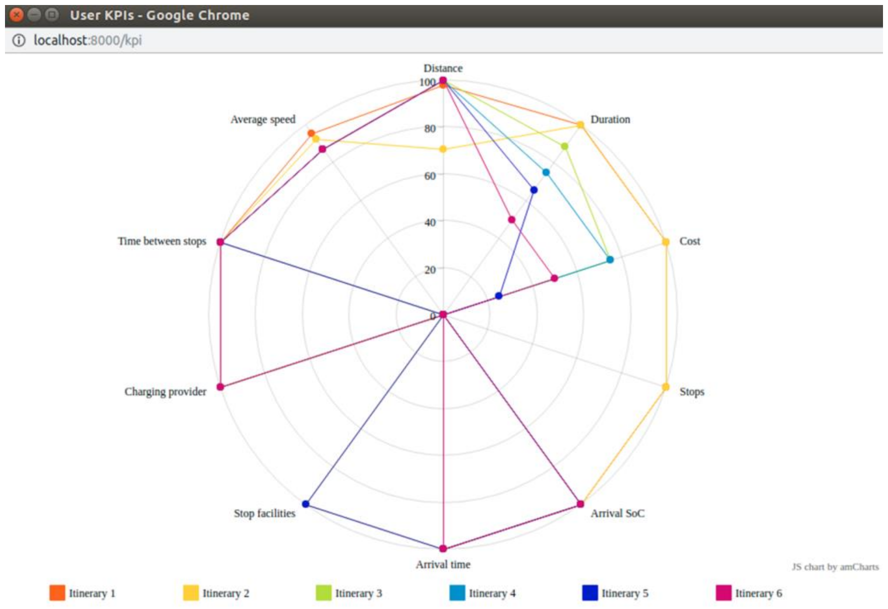


Figure 3.23: Radar chart with multiple KPIs scoring of the specific itinerary

In future iterations of the solution, the system could also be equipped with a learning algorithm to learn the preferences from the driver (e.g. based on performed selections) so it will be better capable to select and rank the best alternatives for specific drivers. In this way the information presented to the driver will be adapted, based on learned preferences and interests of the driver.

3.1.3. Parallel DynAA

DynAA was not originally designed to run computations in parallel. To minimize time needed to provide the itinerary predictions, DynAA was extended with functionality to run the requested simulations in parallel. To minimize the effort Apache Ignite (see: https://ignite.apache.org/) was selected as open source solution to provide parallelism for DynAA. In this case only limited modifications and extensions were needed to run DynAA in a parallel manner. Apache Ignite provided the needed distribution and related messaging functionality.

In Figure 3.24 the design of parallel DynAA is sketched. The machine receiving the request from MECA (the route server) can decide to either run a local simulation or distribute simulation over multiple Apache Ignite tasks.

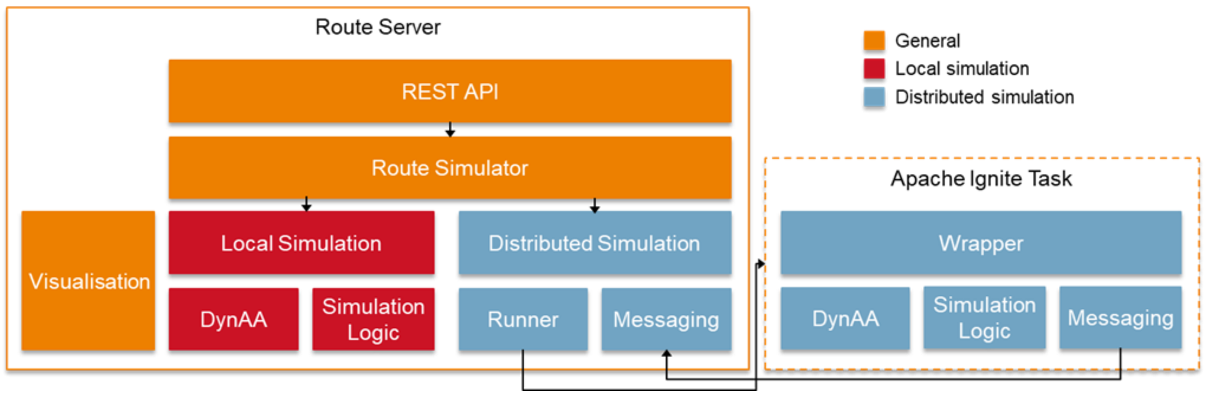


Figure 3.24: Parallel DynAA using Apache Ignite

The route server distributes the simulation requests (in this case from MECA) amongst the different Ignite tasks (each containing a DynAA simulation engine and related simulation logic) and collects the results for visualization (on the route server) and response back to the requestor (MECA). The Apache Ignite performs all necessary wrapping and messaging between the different machines.

Despite being developed to serve the needs of the Smart Travelling demonstrator, the mechanism implemented in DynAA is generic, and it can be applied for all types of simulations requiring fast execution (e.g. because of strict timing constraints or because of the large scale of the simulation).

3.1.4. Development of the data fusion application

As CRF was confronted with non-synchronized logging files from different parts of the simulator, it was very difficult and time consuming to analyze the results of performed test runs. Because of this CRF and TNO decided to develop a separate data fusion application that can be used to fuse and synchronize the data from the existing simulator components and the new logfiles that will be produced by the new CERBERO modules.

The implementation is performed in following steps:

- Implementation of data fusion application at TNO location;
- Test of the application via sample logfiles from CRF and files generated by the new demonstrator components.

It was originally expected that the data fusion component would be deployed in the operational CRF simulator environment directly after the M18 deadline (for the existing CRF components). As during development of the fusion component new formats were introduced by CRF, it was not possible to complete the fusion application for all data sources used by CRF.

As completing the tool was not seen to be an important part of the overall goal of CERBERO tool integration, the software was handed over by TNO to CRF, so additional formats could be added by CRF themselves (outside the scope of the project).

.

3.1.5. Implementation of the demonstrator

The implementation of the M36 demonstrator was performed in following steps:

- Design of Smart Travelling scenarios (see D2.1) and related DynAA (TNO), MECA (S&T) and HMI (Abinsula) functionalities to support the defined test scenario (partly via integration sessions held in Turin in Q1 and Q2 2019 in Turin). The M18 demonstrator was based on a simplified scenario used for simulations at the TNO test environment only. For the M36 demonstrator the complete set of scenarios as defined in D2.1 and requirements defined in D2.2 were used as basis;
- Implement required functionalities for the Smart Travelling scenarios in DynAA:
    - Update interface with MECA to include state of charge of all route segments;
    - Allow dynamic modification of the state of charge while model is running;
    - Improve models to make simulation more realistic;
    - Implement system in the loop module to allow battery and motor models to run in DynAA while connected to SCANeR (and provide more flexibility to CRF);
    - Implement parallelization mechanism (based on Apache Ignite) to allow DynAA to calculate large set of itineraries in parallel, thereby reducing the lead time of the response;
- Implement required functionalities for the Smart Travelling scenarios in MECA:
    - Support all adaptation triggers of scenarios;
    - Replace MECA front end with HMI for the driver;
    - Implement an additional interface between the HMI and MECA;

  - Update the interface between MECA and DynAA;
  - Monitoring of car position during trip;
- Implement required functionalities for the Smart Travelling scenarios in HMI:
  - Design and implement screen layouts, buttons to support scenario functionalities on HMI;
  - Implement specific interaction flows to support scenario;
  - Implement maps with tracking of car and planned itinerary;
- Upgrade of CRF driving simulator:
  - Complete overhaul of actuator of motion platform, motion control computers, motion cueing algorithms and driving mockups;
  - New force feedback system to allow haptic warnings toward driver;
  - Implementation of latest SCANeR software release;
  - Addition of HMI hardware to support driver interaction;
  - Addition of driver tiredness sensors to support related scenario;
- Update of TNO battery and motor models to better support predictions scenarios and allow dynamic state-of-charge modifications;
- Module tests (SCANeR, DynAA, MECA and HMI);
- Preliminary integration test (SCANeR, DynAA and MECA at TNO test platform) to test and validate the new module interfaces (SCANeR-HMI, DynAA-MECA, SCANeR-MECA);
- Explicit definition of itineraries for the different scenarios on the geographic map of Turin and direct surroundings;
- Implement and configure roads in SCANeR (definition of roads using OpenStreetMaps, addition of road information like lanes and crossings and addition of 3D scenery information like buildings and trees);
- Configure scenario data in MECA and HMI (e.g. maps of Turin, charging stations, points of interest, driver information, etc.);
- Integration test via execution of scenarios in the demonstrator on the upgraded CRF driving simulator.

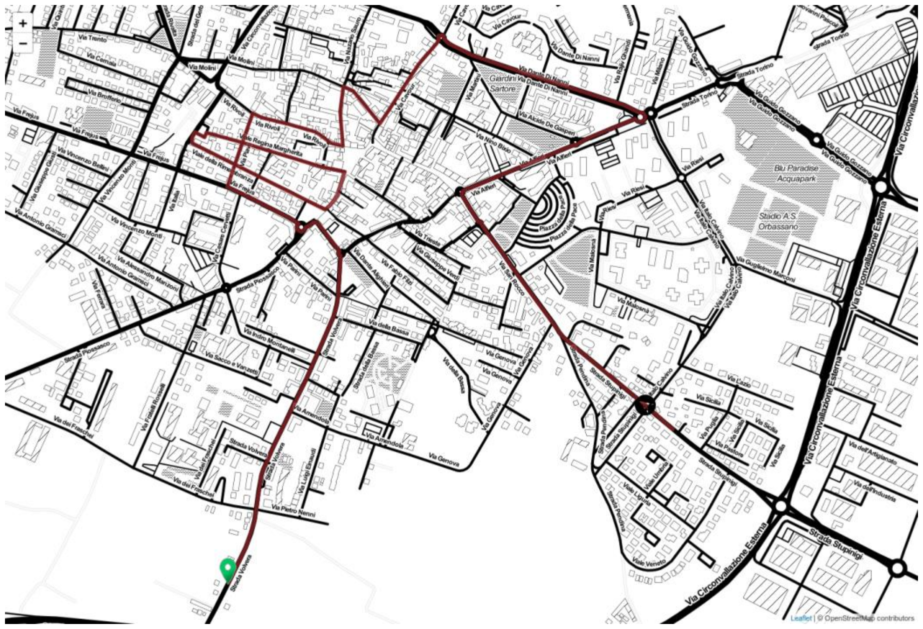


Figure 3.25: Example itinerary for one of the scenarios

3.2. *Integrated and Evolved Tools*

The basis for the demonstrator intended as CRF EV simulator was already in place and within CERBERO no new parts were developed and deployed, other that extensions of the applied CERBERO tools themselves. Therefore, CERBERO approach is mainly used during operation to support reconfiguration.

The CERBERO tools are used in the deployment/operational phase where the tools provide the required real time simulation (to provide electric vehicle models) and adaptation though monitoring and prediction. For the demonstrator the added driver support functionalities forms the heart of the adaptation and reconfiguration functionality (which include user, system and environment driven adaptation).

The CERBERO tools used in the deployment of the M36 demonstrator are DynAA and MECA (also see D2.1 section 2.4). For user realistic interaction with MECA an HMI from Abinsula was added and adapted to be integrated in the CRF driving simulator environment.

For support of the defined use cases the models in DynAA and the driver support functionality in MECA were adapted and extended and new interfaces were developed to interface the tools and to support the monitoring and adaptation loops of the defined scenarios. DynAA was also equipped with generic modules to support system-in-the-loop simulation and parallel simulations. The HMI of Abinsula was adapted to support the defined driver support functionalities. SCANeR and the driver simulator finally were also extended and configured to support the defined scenarios.

3.3. Development and deployment environment

The development environment used for M36 Smart Travelling demonstrator are located at TNO in the Hague – The Netherlands (for DynAA and SCANeR), at S&T in Delft – The Netherlands (for MECA) and at Abinsula (HMI) in Sassari - Italy.

For the SCANeR tool (see [SCANeR]) special hardware was arranged for the development environment as SCANeR required high-end graphical hardware in order to run the real time 3D environment simulations. Other parts of the demonstrator, like DynAA, could be run on already available hardware (Windows based laptops and Linux based cloud servers). For flexibility DynAA can also be containerized using Docker (so it can easily be run on Windows and Linux machine supporting Docker). MECA uses a Docker container to run and can be deployed on any of the (Linux or Windows) machines.

For sharing designs, issues and task status and generated code a Gitlab environment was set up by TNO where all relevant material of the M36 demonstrator was stored. By using a shared Gitlab environment, a central repository could be used to properly manage all developed code. In this way all the developers from TNO, S&T and CRF were able to easily share issues, development plans, documents and code.

The developed and tested demonstrator modules (including the DynAA, MECA and the HMI) were moved to the CRF simulator environment in Turin in the first quarter of 2019 (integration session in Turin on the 16th and 17th of April 2019 and the 26th and 27th of June 2019) to verify if the software was compatible with the software deployed in the real CRF simulator.

4. Tests, Results and Feedback

4.1. Tests

For testing and validating the M36 Smart Travelling demonstrator the scenario as defined in D2.1 were implemented on the components SCANeR, MECA and HMI.

After configuration of all tools the complete end to end scenarios were executed in order to verify developed functions and interfaces.

The development environment in the Netherlands was initially used to develop, implement and test the different modules of the M18 simulators. The environment was also used to perform some initial integration tests of DynAA and MECA for the M36 demonstrator.

Figure 4.26: The test environment in the Netherlands used for development

For the complete integration of the M36 demonstrator the CRF environment in Turin was used. Here also the new HMI (introduced for the M36 demonstrator) was deployed on the real hardware screens of the simulator, allowing complete end-to-end tests. To ensure a realistic driving experience the final HMI implementation was fine-tuned for visualization and MECA interaction to ensure correct information was shown during the scenarios. In order to execute D2.1 scenarios, the actual routes, related 3D sceneries and adaptation triggers needed to be defined and configured in SCANeR and MECA before the final tests were performed.

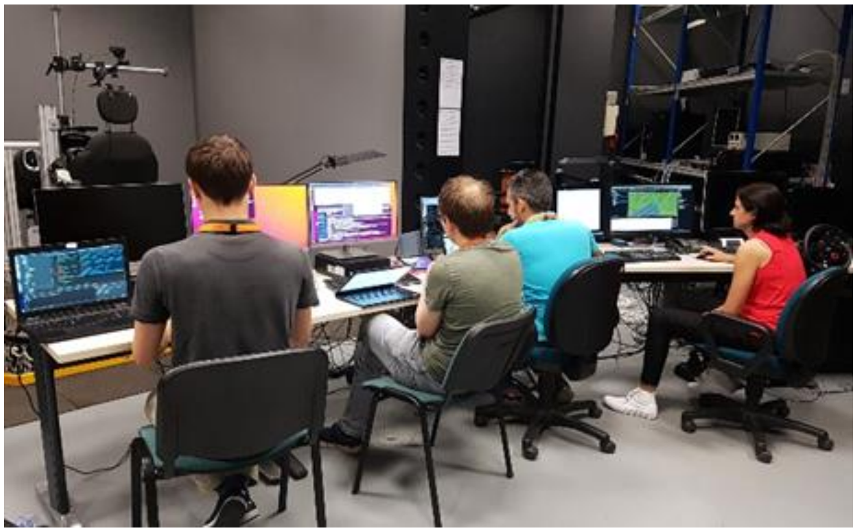


Figure 4.27: The CRF simulator environment in Turin

The tests were focused on the ability to execute the scenarios as defined in D2.1. As the main goal of CRF is to perform driving tests with actual test persons (not expert technicians), it was important to validate that the integrated tools were able to execute the scenarios and perform all advise, monitoring and adaptation functionality, initially using CRF expert as test driver (see Figure 4.28).

The final tests with real test drivers are outside the scope of the CERBERO project and will be executed after completion of the project.

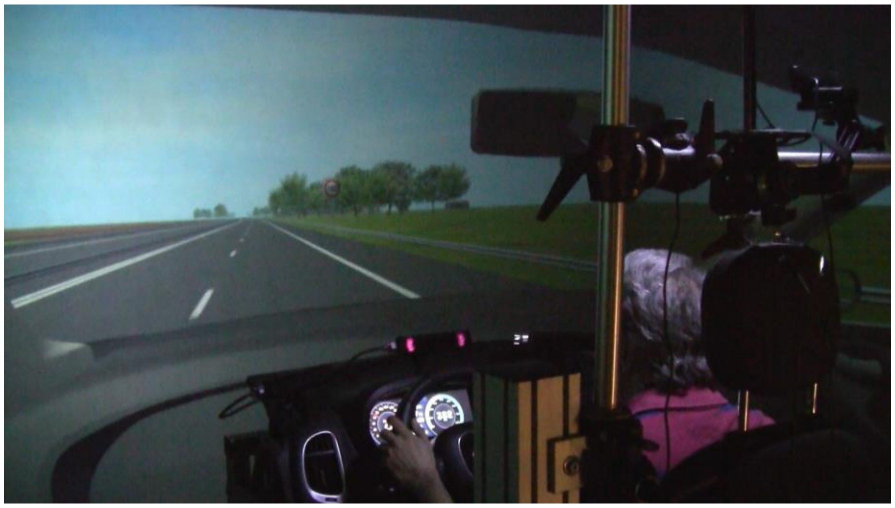


Figure 4.28: CRF expert driving one of the ST scenarios

4.2. Test Results

In the table below the results from the demonstrator development activity were added (in *italic*) to the expected results as specified in D6.1.

Table 4-2 –Smart Travelling Goals and Results

ID	**Goal**	**Results M36 demonstrator**
ST1	Development of parametric, modular and extendable physical-cyber co-simulation environment.	Reduction of costs, increase of reuse of software components in different simulation scenarios. - *Addition of a c0-simulation module to DynAA for system-in-the-loop simulations.* - *Integration of TNO simulation modules into vehicle simulation inside the SCANeR driving simulator.* - *The same TNO models for battery and motor were used in both the driving simulation mode and the planning (route prediction) mode. For route prediction a vehicle model was added to simulate itinerary execution in faster than real time prediction mode.* - *The implemented planning (including prediction) and adaptation phases in MECA and DynAA can be used in all scenarios involving planning and planning adaptation.* - *HMI was added to the simulator capable to interact with the driver and support scenarios involving driver support and related interaction (not possible with existing SCANeR functionality)*
ST2	Development of an integrated open toolchain for design space exploration and co-simulation, with system-in-the-loop capabilities.	Reduce time of development, verification, integration, along with the related costs, exploiting a library of reusable components/metrics integrated by common framework in different levels of abstraction. Incremental prototyping. - *The TNO battery and motor models can be reused in all Electric Vehicle simulations.*

		- *DynAA has been currently integrated with MECA to test user preferences and user-triggered adaptivity, while system-in-the-loop co-simulation extension allows real time simulation of electric vehicle models (battery and motor) and thereby system awareness.*
ST3	Development of self-adaptation.	Efficient support of functional adaptivity, according to system, human and environment triggers. - *Based on MECA and DynAA an adaptation manager was developed, which could trigger adaptation of the generated advice based on any internal or external events.* - *To trigger on driver state (tiredness) additional sensors were added to the simulator to be able to adapt behaviour based on user state.* - *By adding parallel execution advice for adaptation could be generated faster, thereby reducing the response time towards the driver*

4.3. Feedback

To be able to implement the defined scenarios (final version described in D2.1) the team first needed to set up the skeleton of tools to support the defined functionalities.

Based on the high-level functionalities, an initial test scenario was developed for the M18 demonstrator, which contained all the elements of the defined use case scenarios. As CRF does not have separate development or test environments, it was needed to create a separate test platform for development of the demonstrator. As SCANeR is a commercial product, also a separate test license was needed to be able to perform initial test and development work. The project was provided with test license on the condition that a presentation would be given on the created solution on the AVSimulation SCANeR day held in Strasbourg on the 3rd of September 2019 (https://www.avsimulation.fr/scanerday-2019-2/).

As an important part of the SCANeR software is involved with heavy real time 3D environment rendering, specific hardware was acquired by TNO to set-up a small test environment in the Netherlands, to be used for development of the demonstrator by TNO, S&T and Abinsula.

As both S&T and Abinsula used Docker to containerize their software, it was quite simple install and run DynAA, MECA and HMI in the TNO test environment. Furthermore, TNO and S&T were able to closely work together on the demonstrator by using Gitlab as a code sharing environment. By using a direct network connection between the S&T and Abinsula development sites and the TNO test environment, people from the team were able to work on the demonstrator remotely.

After the development of the M18 demonstrator, additional functionality was added, and the structure of the interfaces was simplified in the M36 demonstrator to improve and simplify the integration of the planned HMI functionalities in the CRF simulator. To ensure the demonstrator would be able to run on the operational environment in Turin, three integration sessions were held in Turin in the first half of 2019 to integrate the developed software modules of TNO, S&T and Abinsula in the SCANeR based CRF environment. CRF was able to arrange three separate Linux hardware servers to run DynAA and MECA and special HMI video screens which were installed on the simulator to support the driver interaction.

It was initially an issue to arrange internet connection on the servers to experiment with cloud processing (using data center of TNO) and to retrieving geographic maps of OpenStreetMaps for the HMI. As solution a stand-alone version of OpenStreetMaps (with pre-loaded maps) was installed and multiple servers were arranged to be able to run DynAA in parallel. CRF implemented the scenarios in SCANeR for the scenarios defined in D2.1. This was done in close cooperation with TNO, S&T and Abinsula to ensure all maps, locations and foreseen events were synchronized between the different tools.

After successful installation and integration in Turin in the first quarter of 2019, CRF was able to install updated software modules of MECA, DynAA and the HMI to finalise the M36 demonstrator (without the need for additional physical integration sessions).

During the tests in Turin it showed that the tools were able to perform all defined functionalities within set requirements (D2.2). The solution provided all defined functionalities like predictions, monitoring and adaptivity within set real time requirements (delays).

5. Conclusion

By applying the CERBERO methodology for designing and extending the CRF driving simulator we were able to add adaptive driver assistance functionality which internally used explicit KPIs to optimize the travel experience of the driver. By monitoring KPIs on (simulated) physical level a higher-level adaptation could be developed which used measured KPIs to make predictions and provide advice to the driver. The CERBERO work on modelling of KPIs (D3.1), models of computation (D3.2) and cross layer modelling methodology (D3.3) provided the required basis for the development of demonstrator.

Because CIF implementation was not yet available for operational real time interfaces in the Smart Travelling use case, direct interfaces were developed between the tools. It is expected that once the CIF tool will be able to support operational interfaces it could also be used for DynAA – MECA integration, but this goes anyway beyond the scope of the project. The most time critical interfaces are those between SCANeR and DynAA (the models) and between HMI and SCANeR (for which CIF is probably not appropriate given the strict timing requirements). Prospective usage of CIF for operational integration of the DynAA and MECA tools could positively impact on the flexibility and improved robustness of the interfaces (as described in the holistic methodology and interfaces deliverable (D5.1)).

Integration of the used CERBERO tools DynAA and MECA is described in the framework components deliverable (D5.2). The self-adapting skeleton of the demonstrator (see Figure 3.5) is based on the generic skeleton describe in the demonstration skeleton deliverable (D6.1). The developed skeleton was successfully applied to implement high level adaptation loop in the Smart Travelling use case using MECA and DynAA.

The Smart Travelling use case showed that CERBERO approach and tools were able to construct a high-level adaptation loop on top if the existing CRF driving simulator with minimal development and integration effort. By extending the provided CERERO tools (DynAA and MECA) we were able to implement complex functionalities on top of the driving simulator with maximum reuse of existing functionalities.

Finally, future development can benefit from the implementation of DynAA simulation at the edge (car) and thus optimize simulation algorithms to edge HW using CERBERO tools.

6. References

[CERBERO 2018]	http://www.cerbero-h2020.eu
[D2.1]	CERBERO D2.1 Scenario Description (Final version)
[D2.2]	CERBERO D2.2 Technical Requirements (Final version)
[D3.1]	CERBERO D3.1 Modeling of KPI (Final version)
[D3.3]	CERBERO D3.3 Cross Modelling Methodology for CPS (Final version)
[D4.1]	CERBERO D4.1 Multi-Layer Adaptation (Final version)
[D4.2]	CERBERO D4.2 Self Adaptation Manager (Final version)
[D5.1]	CERBERO D5.1 Holistic Methodology and Integration interfaces (Final version)
[D5.2]	CERBERO D5.2 Framework Components (Final version)
[D6.1]	CERBERO D6.1 Demonstration Skeleton (Final version)
[SCANeR]	Automotive simulator software developed by AVSIMULATION: https://www.avsimulation.fr/solutions/