Chronos 2

Internal version of the final report

Project within Electronics, software and communication - FFI

Author Katarina Boustedt, editor; project team, contributors

Date 30 October 2020

FFI Fordonsstrategisk Forskning och Innovation

VINNOVA Energimyndigheten TRAFIKVERKET FKG VOLVO SCANIA VOLVO

Table of Contents

1. Summary 3

2. Sammanfattning på svenska 4

3. Background 5

4. Purpose, research questions and method 6

5. Objective 7

6. Results and deliverables 7

WP1 Project Management and Architecture 7

WP2 Simulation Environments 10

WP3 Physical Object Control and Supervision 33

WP4 AR Viewer for Test Planning/Analysis/Audience 63

WP5 Driver in the loop using virtual reality 65

WP6 Test System Communication 69

WP7 C-ITS Test Capability 71

WP8 Virtual injection - VUT hardware 78

7. Dissemination and publications 86

Dissemination 86

Publications 86

8. Conclusions and future research 87

9. Participating parties and contact persons 89

1. Summary

The Chronos 2 project has enabled technical progress which will change the way vehicles are tested in the future. What you see in the virtual environment during a test is taking place in the real world, with a few milliseconds delay. All inputs controlling the various robots originate in the virtual environment, weaving virtual and real world into one. This means that the testing is moving from the physical test track to the virtual, simulated track, and back again, with the virtual system being in charge of controlling the tests.

The project had seven technical work packages, each contributing with building blocks necessary for future implementation of virtual methods in testing of vehicles.

A cloud-based C-ITS test system using 4G/5G cellular network was set up, showing that different C-ITS applications can be tested and evaluated. Close integration between the AstaZero test system infrastructure and the C-ITS system was also shown making it possible to include physical, proxied (*e.g.* radio-controlled cars) and virtual objects in the test and evaluation of C-ITS applications.

The concept of dynamic trajectories developed in Chronos 2 is now also being implemented in ISO-22133. The Chronos 2 project is hereby leading and giving important feedback to the development for this ISO standard.

Dynamic trajectories serve several purposes, for example:

1. The supervisor utilizes dynamic trajectories for calculating safe-way-out trajectories and can stop the running test scenario, in case of an emergency
2. It adds the capability for, and testing of, a subject vehicle with less constraints assuming that the vehicle acts more unpredictably compared to a predefined static path (*e.g.* autonomous vehicles)
3. It is possible to avoid dangerous test scenarios where humans are at risk of injury
4. Complex test scenarios where a simulator generates the trajectories can be carried out
5. Both basic and more complex test scenarios can be made increasingly efficient with regard to time and cost

The project demonstrated how active safety functions can be tested with virtual target injection and developed a standard interface setup to connect vehicles from different manufacturers to the same injection platform. This system enables safe, repeatable, and efficient testing of active safety functions in dangerous scenarios that may be impossible to recreate in a traditional test-track setup. The project attracted the attention of Euro NCAP which is now considering to include virtual injection into its future testing

protocol. The Chronos 2 consortium is planning to continue research in this area with close interaction with Euro NCAP.

Furthermore, the project has shown that it is feasible to use head-mounted displays to include the driver into the virtual injection testing. Hence, driver response can be tested along with the technical aspects. The dynamic trajectories are then a necessary component to account for the non-deterministic behavior of a driver. The ecological validity of this setup's test environment was quantified in some known problematic situations such as speed perception and lane positioning using naïve test persons in user experiments. The outcome of these experiments can be used as guidance when and how this technology should be used.

The work on simulation environment integration and scenarios in WP 2 gave many promising results, such as the testing capabilities enabled by virtual objects and V2X capable objects in the simulation. The same conclusion applies to the implementation of dynamic trajectories. The work on scenarios gave further insights in the scenario standard OpenSCENARIO and showed that other scenario standards from different tools can be executed in diverse environments.

2. Sammanfattning på svenska

Projektet Chronos 2 har gjort tekniska framsteg som kommer att förändra hur fordon testas i framtiden. Det man ser i den virtuella miljön under ett test sker i den verkliga världen, med några millisekunder fördröjning. Alla indata som styr de olika robotarna har sitt ursprung i den virtuella miljön, vilket väver samman den virtuella och den verkliga världen till en. Det innebär att testningen går från den fysiska till den virtuella, simulerade testbanan och tillbaka igen, medan det virtuella systemet ansvarar för att kontrollera testerna.

Projektet hade sju tekniska arbetspaket, som var och en bidrog med nödvändiga byggstenar för framtida provning av fordon med virtuella metoder.

Ett molnbaserat C-ITS-testsystem baserat på 4G/5G-mobilnät sattes upp, där projektet visat att olika C-ITS-program kan testas och utvärderas. I nära integration mellan AstaZeros testsysteminfrastruktur och C-ITS-systemet visades också möjligheten att inkludera fysiska, fjärrstyrda och virtuella objekt i tester och utvärderingar av C-ITS applikationer.

Konceptet dynamiska trajektorier, som utvecklats i Chronos 2, implementeras nu även i ISO-22133. Projektet Chronos 2 leder och ger härmed viktig återkoppling till utvecklingen för denna ISO-standard.

Dynamiska trajektorier tjänar flera syften, till exempel:

1. Supervisorn utnyttjar dynamiska trajektorier för beräkning av s.k. safe-way-out-trajektorier och kan stoppa ett pågående testscenario, i händelse av en nödsituation
2. Det ger möjlighet att använda och testa ett fordon med färre begränsningar, vilket tillåter att fordonet agerar mer oförutsägbart jämfört med en fördefinierad statisk bana (t.ex. autonoma fordon)
3. Man kan undvika farliga testscenarier där människor löper risk att skadas
4. Komplexa testscenarier där en simulator genererar banor kan utföras
5. Både grundläggande och mer komplexa testscenarier kan göras allt effektivare med hänsyn till tid och kostnad

Projektet har visat hur aktiva säkerhetsfunktioner kan testas med virtuell målinjicering, s.k. virtual injection, och utvecklade en standardgränssnittsuppställning för att koppla fordon från olika tillverkare till samma injektionsplattform. Detta system möjliggör säker, repeterbar och effektiv testning av aktiva säkerhetsfunktioner i farliga scenarier som kan vara omöjliga att återskapa i en traditionell uppställning på testbana. Projektet uppmärksammades av Euro NCAP som nu överväger att inkludera virtuell injektion i sitt framtida testprotokoll. Chronos 2-konsortiet planerar att fortsätta forskningen inom detta område i nära samverkan med Euro NCAP.

Vidare har projektet visat att det är möjligt att använda huvudmonterade displayer för att inkludera föraren i den virtuella injektionstestningen. Därav kan förarens gensvar testas, tillsammans med de tekniska aspekterna. De dynamiska trajektorierna är då en nödvändig komponent för att ta hänsyn till den icke-deterministiska beteendet hos en förare. Effekten av denna testmiljö kvantifierades i några kända problematiska situationer, såsom hastighetsuppfattning och körfältspositionering, med hjälp av naiva testpersoner i användarexperiment. Resultatet av dessa experiment kan användas som vägledning kring när och hur denna teknik ska användas.

Arbetet med integration av simuleringsmiljö och scenarier i WP 2 gav många lovande resultat, till exempel de testmöjligheter som möjliggörs av virtuella objekt och V2X-kapabla objekt i simuleringen. Detsamma gäller införandet av dynamiska trajektorier. Arbetet med scenarier gav ytterligare insikter i scenariostandarden OpenSCENARIO och visade att scenariostandarder från andra verktyg kan användas i varierade miljöer.

3. Background

Development of more advanced driver assistance functions (ADAS) and steps towards autonomous driving (AD) vehicles is continuing at a rapid pace. To ensure high performance and safety of ADAS and AD, the development of test technology at the test labs has to maintain at least at the same pace. One new component widely used today in this development by OEMs and TIER1s, is simulations. Using simulation as a part of

development and verification is of vital importance due to the increasing number of required tests for every step towards completely autonomous vehicles. Virtual, or simulator-based testing will increase in importance and these tests need to be validated with tests on a test track. Chronos 1 (dnr 2016-02573) started prototyping a test system supporting efficient, safe and repeatable testing of ADAS, however, further development is needed together with an integration towards simulations in order to support more complex ADAS and AD tests.

One critical component for a test track test system and for more advanced vehicles is connectivity. In Chronos 1 AstaZero's 4G network performance was profiled with respect to the test system communication requirements. In Chronos 2, reconfiguration of the 4G network will be attempted to better handle the test system requirements.

Furthermore, through Ericsson activities in WASP/WARA a 5G proof of concept (5G PoC) network has been installed at AstaZero. It is essential to verify the behavior of upcoming services (*e.g.* C-ITS services) and network concepts in a real-world environment. The AstaZero test track will provide a unique possibility for controlled research, development and verification in a semi real-world cellular environment using the 5G network as all parts of the test can be supervised and logged which is not possible if tests are performed in the streets using commercial networks.

Chronos 2 was coordinated by AstaZero and executed by RISE, Ericsson, Volvo Cars, AB Volvo, VTI, Fengco, ESI, Inceptive and Veoneer. The project has run for two years with a total budget of MSEK 38.5, with public support of MSEK 19.4

4. Purpose, research questions and method

Chronos 2 has investigated integration of simulations with test track technology in several steps:

1. Adding a new mechanism where a simulator more or less in real-time controls objects, *e.g.,* vehicle targets, on the proving ground. The purpose is to allow for dynamic adaption of test scenarios to the vehicle under test (VUT) behavior.
2. Injection of virtual objects, their position, speed, size, *etc.* into the VUT. This will enable verification of VUT behavior in traffic situations with higher efficiency and lower risk.
3. Capability of combining injected virtual objects with real targets on the test track, which will include verification of sensors and algorithms for real targets while using virtual objects as a constraint for VUT behavior.
4. Virtual reality presentation of a simulation to the driver/passenger to investigate human interaction and experienced comfort.

5. Extension of the test system to connected vehicle technologies, with possibility to control the communication environment and test multiple real and virtual connected vehicles.

5. Objective

The objective of Chronos part 2 was the continued development of the Chronos test system, which is critical for the long-term goals of zero traffic casualties and an efficient transportation system. Stepwise development of the Chronos test system secures that e.g. AstaZero can be the testbed required for more advanced ADAS systems, features for autonomous drive and the systems, such as communication and infrastructure, they depend on. The Chronos test system enables efficient verification of these systems on a test track during development, rating and certification.

Specifically, technical objectives of Chronos part 2 were

- Developing test scenarios where real-time objects adapt to tested vehicles
- Support for virtual test objects in VIL test track testing
- "Driver in the loop" where users experience simulated events through AR and/or VR while driving or riding a real car
- C-ITS testbed based on 5G network

These objectives were not changed during the course of the project.

6. Results and deliverables

WP1 Project Management and Architecture

WP1	PM and architecture
Leader	AstaZero
Other participants	All WP leaders and technical leads
Description of contents	Overall project planning and follow-up. This WP will also coordinate the architecture for the test system.
Method/approach (when relevant)	A limited number of forums will be created. At least one weekly project meeting and one technical meeting for overall architecture and interface development. Each WP will work independently with their

	tasks as described below. WP1 will coordinate towards major project deliveries/milestones organized into the feature areas; Simulation integration, Viewers and C-ITS. Coordination of national/international publications, presentations and demos.
Delivery	1.1 Project meeting minutes. 1.2 Progress reports to steering group and FFI. 1.3 Updated planning as needed. 1.4 Architecture including interface descriptions. 1.5 Define test scenarios for demonstration 1.6 Demonstration coordination 1.7 Coordination of interface for virtual object injection (2.1 and 8.1) with German initiatives.

This work package has managed the project, not necessarily in the way described in the original project application in all details. Normal project management procedures have been followed, such as Deliveries 1.1 Project meeting minutes, 1.2 Progress reports, 1.3 Updated planning.

The work packages have been working autonomously, scheduling their own regular meetings. Over time, work packages have started joint thematic efforts, across WPs, to enhance the collaboration between teams and improving the project result, for example C-ITS, Dynamic Trajectories, and Virtual Injection. Work package leaders have met regularly, as needed, to plan the demos and discuss the progress of the work (1.5 Architecture, 1.6 Demonstration coordination). The project manager has kept the participants, and particularly the WP leaders, informed on the upcoming deadlines and administrative requirements, such as financial reporting, invoicing/requisitions, and technical reporting.

Twice per year, full project meetings have been held, at the AstaZero test track. At these meetings, the WPs and thematic groups have presented their status, either as meeting room presentations, posters, video recordings, desktop demonstrations or demonstrations on the track. At these meetings, partner representatives have participated, not limited to persons involved in the daily project work.

The Chronos 2 Management Group (MG) has had one representative per partner. The group has met after the full project meetings and a few times online, as needed. The

MG has handled issues with redistributing budget as scope of effort has changed for some partners.

Delivery 1.7: Coordination of interface for virtual object injection (2.1 and 8.1) with German initiatives.

Over the years, the Chronos 2 project has been in contact with the Pegasus project. There was a collaboration with different participants in the Pegasus project, which became several working groups focusing on various topics. It appeared that the Chronos work on Virtual Injection was among the very first ones who had actually done anything in the field at the time. Therefore, it was mostly Chronos presenting to the other participants. This led to a decision to end this collaboration, since it was considered time consuming and not fruitful to Chronos 2.

Deviation from plan

Weekly project meetings organized by WP1 have not been held, as it has not been deemed necessary and time was better spent on other activities.

WP2 Simulation Environments

WP2	Simulation environments
Leader (role and responsibility)	Fengco (developing the simulation environment based on dSPACE ASM, coordinate the work between Fengco and other participants in WP2)
Other participants (roles and responsibilities)	ESI (provide experience with sensor models based on Pro-SiVIC, develop scenario editor tool and focusing on portability of test cases), Volvo Cars (develop dynamic trajectory generation and interfaces for the Simulation Platform for Active Safety (SPAS) simulator), AB Volvo (provide expertise in V2V communication for trucks)
Description of contents	WP2 extends the capabilities of the Chronos test system by implementing a simulation environment. The simulation environment is able to run on a real-time platform and simulate the complete test scenario including the test track, VUT and its sensors and other traffic objects. This mainly allows for three features (details in footnotes) - Injection of virtual objects into the VUT on object list level[1] - Execution of dynamic scenarios[2] - Run test cases that require V2V-communication with virtual objects[3] In the simulation environment, based on dSPACE ASM and SPAS, the default sensors are ideal. In WP2, more advanced sensor models are also integrated in order to see the possible benefits from having more detailed sensor models. Examples of such sensor models are from ESI Pro-SiVIC and other MIL environments[4] Another aspect of WP2 is focused around the portability of test cases from existing MIL/SIL/HIL test environments at OEMs to the Chronos test system[5].

[1] The sensor models in the simulation environment will let the simulation determine which objects are detected by the VUT and produce an object list that can be injected into the VUT.

[2] The simulation will be able to produce trajectories for the all objects in the simulation. It will also be able to update the simulated positions of all actors by receiving position updates from the objects themselves or the Chronos server. This allows resynchronization of the test case by updating the position of the virtual objects and produced trajectories so that the correct test case is executed even though the position of the VUT (or other real objects) differ slightly from the initially planned route of the test scenario.

[3] V2X communication from the virtual objects will be simulated and sent from the simulation environment.

[4] The interface for connecting sensor models may be based on the FMI or OSI data exchange protocols.

[5] This work has benefited from having a dialogue with the FFI project “Simulation Scenarios”.

	For scenario import to the Chronos test system, a scenario editor tool will be created[6]. The output files from the scenario editor tool will be stored in an open source library, to build a reference base of use cases[7]. Much of the work in WP2 is focused on OpenScenario which is currently defined by a community, WP2 will try to provide OpenScenario add-on to enhance the standard toward the OpenScenario consortium.
Method/ approach (when relevant)	The work on the simulation environment was divided into the following steps: 1. Definition of necessary interfaces between the simulators (SPAS, ASM), the physical object control server, and virtual object injection. 2. Implement a solution where the simulation environment (and Chronos Server) is inside the VUT. This will minimize time delays between the simulation and VUT. This step is synchronized together with WP3 and WP5 since parts of their work will benefit from having minimized time delays. a. The first goal in step 2 is to have a solution for injecting static objects into the VUT. b. When the concept with static object has been proven, the aim will be to also inject dynamic objects. 3. The final step puts the simulation environment (and Chronos server) outside the VUT in order to explore the feasibility to run the simulation remotely and to have more than one VUT.
Delivery	2.1 Definition and implementation of the interface for sending the object list from the simulation environment. 2.2 Software component to read a position update from VUT and real objects, and resynchronize the scenario (virtual objects) according to the real object positions. 2.3 Demonstrations (with WP4 and WP5) of implemented software interfaces to connect to the AR device in WP4 and VR device in WP5. 2.4 Report on how well the simulation can reproduce the object list with virtual targets and sensor simulation compared to reality. 2.5 Demonstration of dynamic trajectory update by controlling an RC Car with trajectories from the simulation environment. 2.6 Demonstration and report on how we can move a test case from an existing MIL/SIL/HIL environment to the test environment developed in Chronos 2. 2.7 Demonstration with WP8 of virtual object injection into the VUT.

[6] The scenario editor tool will make preparatory work on test scenarios so that for example OpenScenario-files and Drive Files can be created and imported to the Chronos test system.

[7] The stored files can be in, *e.g.*, OpenScenario-format or Drive Files. All files in the library are prepared to run in the Chronos test system. For example, the library could contain prepared files for a cut-in scenario.

	2.8 Demonstrate that the simulation environment can run a test case with a mix of virtual and real objects together with the VUT. 2.9 Implementation of interface for sending V2X communication (802.11p) from virtual objects in simulation to VUT. 2.10 Demonstration of the V2X communication (802.11p) in deliverable 2.9 in the form of a platooning use-case. 2.11 Software component that works as add-on for the commercial simulation tool (ASM) that has been used in WP2. The module provides the implemented functionality described in deliverable 2.1-2.3 and 2.9. 2.12 Demonstration of the developed scenario editor tool. It will show the workflow from creation of a typical test scenario to execution of the test scenario in the Chronos test system. 2.13 An open source library of reference test cases that can run in the Chronos test system.

WP2.1 Introduction

Work Package 2 (WP2) focused on simulation environments in the automotive industry and how to integrate them into the Chronos test system. There are already commercial simulation environment tools available on the market suitable for testing on many levels (MIL, SIL, HIL *etc.*). It is also common that different OEMs create their own simulation environments or models with different levels of complexity. Such a simulation environment (both commercial and in-house developed) may simulate a complete test scenario including test track, VUT and other traffic objects (cars, pedestrians *etc.*). One of the main ideas of WP2 was to run tests on the test track by executing a scenario in real-time in a simulation environment. During runtime, the simulation could control traffic objects on the test track and create V2X-capable virtual objects when necessary. One major aspect of this approach was also to let the simulation environment observe what was happening on the real test track in real-time and adapt the simulated scenario to the events at the track.

Having a sophisticated simulation environment as the execution environment for tests allows for an improved testing workflow and higher flexibility of scenarios that can be executed. This meant that a lot of focus of WP2 was put on scenario design and mobility of scenarios for the test system developed in Chronos 2.

The companies and parties involved in Work Package 2 was Fengco Real Time Control AB, ESI Nordics AB, Volvo Car Corporation and AB Volvo.

Fengco was the leader of the Work Package 2. The role was to coordinate the work between the partners in the work package and to integrate and adapt the simulation environment based on Automotive Simulation Models (ASM) into the Chronos test

system. ASM is a best-in-class commercial tool from dSPACE GmbH. Fengco also assisted Master Students doing their theses on Volvo Car Corporation by providing them with the dSPACE ASM tool suite, training, and tool expertise. Finally, Fengco also worked with the company Ainomaly and its AI based solutions in combination with ASM.

The role of ESI in the work package was to bring experience with sensor models based on their tool Pro-SiVIC™, to provide scenario definitions using a Scenario Editor tool developed in parallel with this project and to ensure the portability and interoperability of the use cases. ESI was also involved in the final demonstration at the AstaZero test track.

Volvo Car Corporation (VCC) brought expertise of their simulation environment called Simulation Platform for Active Safety (SPAS) to the work package and participated in the work and discussions about scenario design and mobility. VCC also produced academic work about Dynamic Trajectories, scenario design and control of robotized test objects (both master theses and work on PhD level).

AB Volvo provided expertise in V2X communication to the work package and participated in the final demo of the project. The company was also involved in the work and discussions about scenario design and mobility.

WP2.2 Integration of Simulation Environment

The simulation tool that was integrated as a simulation environment for the Chronos test system was dSPACE ASM. Integrating ASM extended the capabilities of the Chronos test system significantly. In this case, dSPACE ASM was running on a real-time platform (dSPACE SCALEXIO). The simulation enabled some major testing aspects:

- Injection of virtual objects into the VUT on object list level
- Execution of dynamic scenarios
- Run test cases that require V2X communication with virtual objects

The results for all of these aspects are described separately in chapters WP2.3-WP2.5 within this work package.

The integration of the simulation environment to test system developed in Chronos 2 was designed to be tool agnostic. This means that all interfaces to the simulation environment was well-documented and consisted of non-proprietary signal values. The simulation environment was centralized, with interfaces to necessary infrastructure. Mainly, the simulation environment had two external interfaces, one to the Chronos server (consisting of one TCP channel and one UDP channel) and one to the C-ITS server (a single TCP channel). The Chronos server is the central server that handles, among other things, all communication (except V2X communication) with all physical components (traffic objects on the test track, including the VUT, and viewers such as AR-viewers and VR-viewers). V2X communication is handled by a separate C-ITS server at the test track. In Figure 1 below there is an overview of the implemented architecture, where all blue components are part of WP2, and green components are part of the

infrastructure at the AstaZero test track. The light blue components in the simulation environments corresponds to tools from dSPACE.

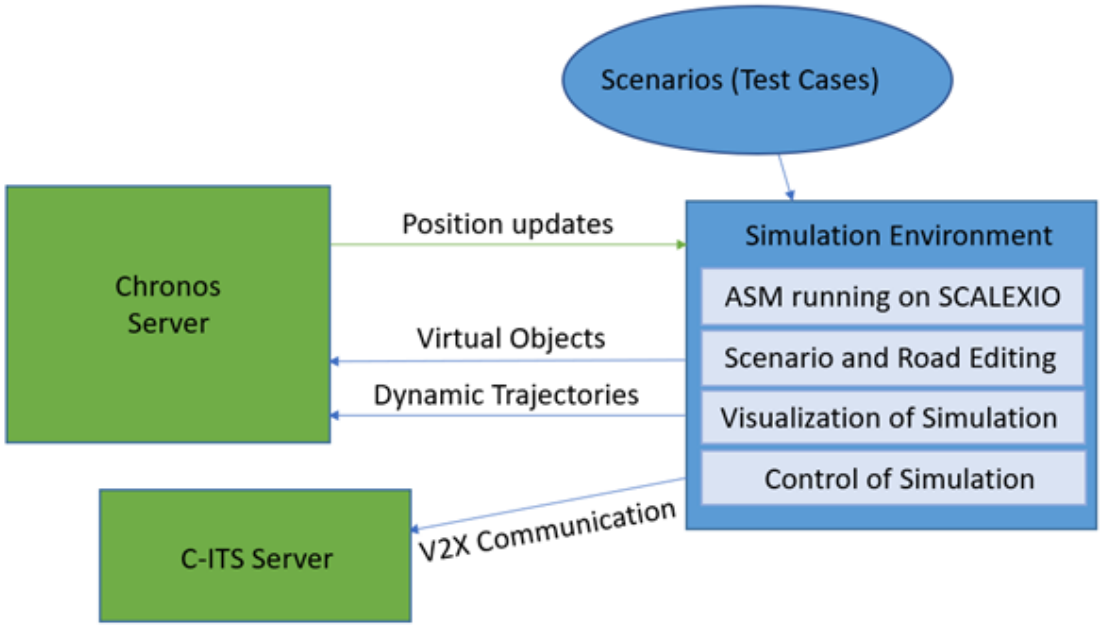


Figure 1. An overview of the implemented architecture.

WP2.2.1 Position Update (Delivery 2.2)

During runtime, the simulation continuously receives information about position, heading, and speed about the VUT and test objects on the test track. This information was used to include traffic objects in the simulation that represented the real objects on the test track. Having traffic objects in the simulation that corresponds to the real objects meant that the simulation could react to actions from the real objects and update the scenario accordingly. This means that the simulation could trigger actions and events depending on what the physical objects on the test track did.

The functionality provided by the position update was crucial for both the use cases with virtual objects and with dynamic trajectories. The reason for this is that the position of other test objects (virtual or dynamically controlled) are highly dependent on the position of the VUT and other externally controlled objects.

WP2.2.2 Software Component (Delivery 2.11)

As a delivery, Fengco produced a software component for all the simulation environment implementation done in this project. That includes the position update mentioned above, and also the capabilities in the following chapters WP2.3-WP2.5, i.e. virtual object injection, dynamic trajectories, V2X communication. Assuming that the off-the-shelf tool chain for dSPACE ASM available, this software component is all that is needed to run the simulation environment together with the Chronos server and C-ITS

server. It comes with an application note which describes the necessary work to be done in the off-the-shelf dSPACE ASM models.

WP2.3 Virtual Object Injection

WP2.3.1 Implementation in Simulation Environment (Delivery 2.1)

All deliveries related to virtual object injection were successfully implemented. When a scenario is running in the simulation, it is possible to set several traffic objects as virtual and send messages about the object(s) to the Chronos server. The messages contain an object list, i.e. information about the type of object (car, pedestrian *etc.*), its position (x, y, z coordinates), its orientation (heading, pitch, roll) and its speed. The object list is continuously updated and streamed over the UDP channel during test execution. The Chronos server distributes the information to the VUT which uses the information to inject the virtual object(s) by overwriting the object list in their Active Safety ECU (Work Package 8). The same object list stream from the simulation is used to send information to the Augmented Reality (AR) viewer and Virtual Reality (VR) headset developed by Work Package 4 and Work Package 5.

WP2.3.2 Demonstrations (Deliveries 2.3, 2.7, 2.8)

During the project, several demos were shown where virtual objects were used. One such example is the final demo of the project. In one part of the demo, a driver was wearing a VR headset in the VUT and was driving into an intersection, followed by a real truck. In this scenario, the simulation environment generated a virtual object that used the position and speed information from the VUT to guarantee a side-on collision in the intersection. The driver could see the virtual object in the VR headset, and the object was injected into the active safety ECU of the VUT in order to activate the emergency brake function. In other demos, the AR viewer was shown to see the same information as in the VR headset.

Figure 2 shows an image from the simulation environment during the final demo. The white car represents the physical Volvo car approaching an intersection with a restricted view where the simulated car (dark red) is simulated in real-time to guarantee a side-on collision.

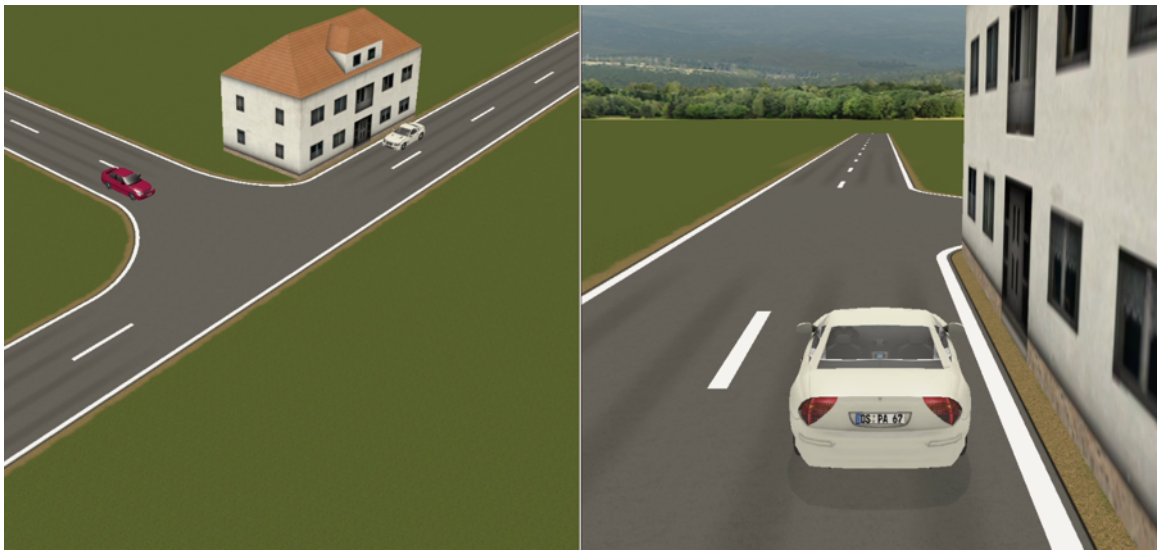


Figure 2. Parts of the simulation of the final demo.

WP2.3.3 Main Takeaways

As seen in the project, virtual objects give some major advantages to the testing capabilities of the test track:

- Virtual objects can be used in dangerous scenarios without risk of harming humans and equipment.
- The infinitely high dynamics of virtual objects makes sure that the object can be precisely where it should be according to the scenario specification.
- Having virtual objects instead of only real objects means that a test setup is much faster since fewer objects must be physically moved to their respective starting position. If many iterations of the same test should be executed in a row, it can save a lot of time.

WP2.4 Dynamic Trajectories

WP2.4.1 Introduction

Even though there was only one delivery regarding Dynamic Trajectories, a lot of work was put here by WP2. In short, dynamic trajectories means that the simulation environment produces trajectories for the test objects on the test track which can be dynamically updated according to predefined rules, i.e. the scenario specification. For example, the trajectories of the test objects could be adapted to match the speed of an autonomous VUT. Dynamic trajectories capability was implemented in the ASM based simulation environment, and there was also a lot of work done by Volvo Cars (both on an academic level and on a practical level on the SPAS platform).

WP2.4.2 Implementation in Simulation Environment

The implementation of dynamic trajectories was realized by letting the simulation continuously generate trajectories during runtime. The trajectories were formatted in a similar way as the static trajectories defined in Chronos 1, often called “Drive Files”. The main difference in the dynamic trajectories being that each produced trajectory, or “chunk”, only covered the trajectory a short time ahead. The simulation then ensured that a new trajectory was generated and sent out before the controlled test object had driven to the end of the previous one. Factors like amount of time between points in the chunk (which translates to the amount of time ahead the chunk covers) and overlap between consecutive chunks were made into parameters in the simulation environment. It was discovered during the project that different test objects had different requirements on these parameters, which meant that it was a big benefit to be able to change this individually for different test objects. The test objects requirements on the parameters were closely related to the control algorithm that the test object used to follow the trajectory. The parameters could also be changed during runtime of a scenario. This meant that the frequency of produced trajectories could be increased in highly dynamic parts of the scenario. For example, if the test object is controlled to follow a straight road and is suddenly triggered to do a lane change, it may be beneficial to send out new trajectories with higher frequency when the lane change should be performed.

Below in Figure 3 is a description of the Drive File format used in Chronos 1. The data stream for dynamic trajectories contained similar data but divided into overlapping chunks that could be updated during runtime by the simulation environment.

Drive File explanation:

TRAJECTORY;<Name>;<Version>;<NbrOfLines>;
LINE;<Time>;<x>;<y>;<z>;<Heading>;<Velocity>;<Acceleration>;<Curvature>;<Mode>;ENDLINE;
LINE;<Time><x>;<y>;<z>;<Heading>;<Velocity>;<Acceleration>;<Curvature>;<Mode>;ENDLINE;
More lines etc…
ENDTRAJECTORY;

Drive File example:

TRAJECTORY;ASTAGarageSquare;0.1;7081;
LINE;0.00;0.000000;0.000000;000.000000;0.000000;0.000000;0.010000;0.000000;0;ENDLINE;
LINE;0.01;0.000000;0.000000;000.000000;0.000000;0.000100;0.020000;0.000000;0;ENDLINE;
LINE;0.02;0.000001;0.000000;000.000000;0.000000;0.000300;0.030000;0.000000;0;ENDLINE;
LINE;0.03;0.000004;0.000000;000.000000;0.000000;0.000600;0.040000;0.000000;0;ENDLINE;
LINE;0.04;0.000010;0.000000;000.000000;0.000000;0.001000;0.050000;0.000000;0;ENDLINE;
LINE;0.05;0.000020;0.000000;000.000000;0.000000;0.001500;0.060000;0.000000;0;ENDLINE;
LINE;0.06;0.000035;0.000000;000.000000;0.000000;0.002100;0.070000;0.000000;0;ENDLINE;
LINE;0.07;0.000056;0.000000;000.000000;0.000000;0.002800;0.080000;0.000000;0;ENDLINE;
LINE;0.08;0.000084;0.000000;000.000000;0.000000;0.003600;0.090000;0.000000;0;ENDLINE;
LINE;0.09;0.000120;0.000000;000.000000;0.000000;0.004500;0.100000;0.000000;0;ENDLINE;
LINE;0.10;0.000165;0.000000;000.000000;0.000000;0.005500;0.110000;0.000000;0;ENDLINE;
LINE;0.11;0.000220;0.000000;000.000000;0.000000;0.006600;0.120000;0.000000;0;ENDLINE;
LINE;0.12;0.000286;0.000000;000.000000;0.000000;0.007800;0.130000;0.000000;0;ENDLINE;
LINE;0.13;0.000364;0.000000;000.000000;0.000000;0.009100;0.140000;0.000000;0;ENDLINE;
LINE;0.14;0.000455;0.000000;000.000000;0.000000;0.010500;0.150000;0.000000;0;ENDLINE;
ENDTRAJECTORY;

Figure 3. A description of the Drive File format used in Chronos 1.

The simulation environment sent out all generated trajectories to the Chronos server, which then distributed the trajectories to the correct test object. Before reaching the test object, the trajectories also passed though the Supervisor developed by Work Package 3.

In the final implementation of dynamic trajectory generation, the simulation was running time synchronized with the objects on the real test track (expect for a couple of milliseconds in order to compensate for network lag). This approach used the fact that the vehicle simulating the complete vehicle dynamics in ASM (typically used to simulate the VUT in a MIL/SIL/HIL use case) produces information about their planned route during runtime. This information was adapted to fit the dynamic trajectory message that the simulation environment sent to the Chronos server. This approach let the simulation react and update the trajectories in real-time depending on the actions of the VUT and other real traffic objects.

The dynamic trajectory implementation in ASM was integrating so called MultiAgent simulation. In a typical simulation setup with ASM, there is only one object simulating complete vehicle dynamics. In MultiAgent simulation, you co-simulate two instances of

interconnected ASM models. This means that two objects with full vehicle dynamics are available in the simulation. This means that trajectories could be produced for two test objects during a scenario. For all demonstration purposes in the project, generating dynamic trajectories for two objects was enough. However, it would of course be possible to extend the MultiAgent simulation to include even more trajectory-generating objects.

WP2.4.3 Demonstrations (Delivery 2.5)

Several demos were shown during the project that showed the capabilities of the dynamic trajectories and controlling the RC Car with dynamic trajectories was done many times. In the final demo of the project one scenario showed that two test objects could be controlled dynamically to ensure consistency in the scenario even though the VUT could freely decide its own speed in the scenario.

In Figure 4 below, a screen shot from the simulation environment during the final demo is shown. All three vehicles are representations of real vehicles on the test track. The brown car is the VUT, with an unknown behavior when the simulation starts. The white and yellow cars are dynamically controlled to box in the VUT if it tries to do an overtake and drive ahead of the white and yellow car. The yellow car stays in the right lane and tries to match the speed of the VUT, while the white car moves into the same lane as the VUT and blocks it. This could be considered a stress test for an autonomous VUT. The important aspect is that the scenario is consistent and automatically executed, even though the speed and time of lane change of the VUT can vary.

Figure 4. A screen shot from the simulation environment during the final demo. The brown car is the VUT.

WP2.4.4 Feasibility of Generated Trajectories

The main issue that WP2 faced when implementing dynamic trajectories on the simulation environment was to ensure that every single trajectory chunk was feasible to drive, especially in the start-up of the scenario when the actors are starting from stand still. In the simulated world, the traffic objects that generated the trajectories could accelerate at a very high rate that resulted in trajectories that were uncomfortable to ride along with in a physical car on the test track. Also, the control algorithms of the car(s) could abort the test if a received trajectory was requesting too high acceleration or too narrow curvature. To work around this in the project, every simulated scenario started with a static segment that made all actors go from standstill to a certain speed. When all dynamically controlled objects had reached a starting speed above ~6 km/h the dynamic generation of trajectories could fully take over control of the vehicles.

There were also some cases where the curvature of a produced trajectory was too high for a test object on the test track, meaning that the object could not turn at the requested rate. To mitigate this phenomena, one solution would be to make sure that the simulated objects have the same vehicle dynamics as the real objects on the test track. In ASM, there is good support for importing this kind of vehicle dynamics parameters. However, this was outside of the scope of this project, and can be considered future work.

WP2.4.5 Main Takeaways

During the Chronos 2 project, it was seen that the dynamic trajectories enabled some major advantages for testing on a test track:

- Dynamic trajectories can ensure consistency in the test case, or scenario, when the behavior of the VUT is unknown, which may be the case if the VUT is an autonomous vehicle.
- The concept of dynamic trajectories can be extended to implement a return-to-start functionality. This would be realized by simply defining a route back to the start position in the scenario definition.

WP2.4.6 Academic Work

Besides the integration of dynamic trajectory capabilities into the Chronos test system, there was also a significant amount of academic work on this topic, mainly from Volvo Car Corporation. Volvo Cars supervised six Master's and one Bachelor's theses directly related to the Chronos 2 project, involving a total of 14 students from Chalmers University of Technology, Gothenburg University, Lund University, and KTH Royal Institute of Technology. The abstracts of the reports supervised by VCC during the duration of the project can be found in Appendix B.

Additionally, an Industrial PhD student at Volvo Cars conducted research about modelling scenarios with dynamic trajectories. Hybrid automata, a modelling paradigm

that combines finite state machines with differential equations, was investigated as an approach to model test scenarios. One of the main results is the proposal to represent the unknown motion of the vehicle-under-test as an additive external disturbance that is confined to some known limits. Tools from robust predictive control were applied to generate dynamic trajectories based on a hybrid automaton description of a scenario. Conference papers are being prepared to disseminate the results.

WP2.5 V2X Communication

WP2.5.1 Implementation in Simulation Environment (Delivery 2.9)

The V2X implementation on the simulation environment allows any simulated object to send out standardized V2X messages. In this case, CAM and DENM as defined by the CAR 2 CAR Communication Consortium and standardized by the European Telecommunications Standards Institute (ETSI). The CAM can be viewed as heartbeat signals where the traffic object continuously sends out information about, *e.g.*, its position to all other traffic objects within reach. DENM can be viewed as the communication of an event. For example, a DENM may be sent out to tell other traffic objects within reach that a certain event has happened. Such an event can be anything from an executed emergency brake to the detection of roadworks.

In the delivery, the communication was specified to be 802.11p which implies that the messages should be sent out via Wi-Fi from the simulation environment. However, due to the architecture of the C-ITS server at the test track, the simulation environment communicated the V2X communication over a local network to the C-ITS server, which distributed the messages to the test objects. Besides, it is worth noting that the V2X communication interface of the simulation environment is a separate one from the interface to the Chronos server. This means that the Chronos server was not involved in the V2X communication.

WP2.5.2 Implementation in Truck

In this project, AB Volvo was able to utilize its expertise in V2X communication for cellular network [LTE-Advanced] instead of short-range communication [802.11p]. The Volvo Trucks C-ITS client had been adapted and integrated with an LTE-Advanced cellular modem that communicates with the C-ITS test network provided at AstaZero test site. At the application protocol layer, the MQTT protocol which is the de-facto standard for Internet-Of-Things was used for the exchange of standardized CAM and DENM messages between the C-ITS client and the C-ITS test server.

WP2.5.3 Demonstrations (Delivery 2.10)

The deliveries regarding V2X-communication were completed, with minor adjustments. Due to re-prioritization of resources, it was not feasible to run a demonstration in the

form of a platooning use case. However, other demos successfully showed the V2X implementation on the simulation environment.

During the project, several V2X demos were presented where the simulation environment produced V2X messages for virtual objects. In one demo, a simulated car was running in front of the real VUT and suddenly executed an emergency brake. The braking triggered a transmission of a DENM which was received by the real VUT. The demo showed that the VUT could react to the received DENM and also brake. In the final demo, the part related to V2X capabilities used the same scenario as the virtual object injection. This demo showed that a simulated virtual car emerging from the side road of an intersection can warn an oncoming VUT about a possible collision by continuously sending out CAM.

During the final demo, the emergency brake was exchanged as a DENM between a Volvo test truck and a Volvo test car. The quality and performance of the message transmission over cellular network was evaluated.

WP2.5.4 Main Takeaways

In the project, it was seen that the V2X capability of the simulated object further increased the use cases for the simulation environment. It enabled testing of V2X use cases where it was necessary to use virtual objects (such as dangerous scenarios).

WP2.6 Scenario Design and Mobility

WP2.6.1 Introduction

In the preceding project, Chronos 1, all test executions had to rely on static pre-defined trajectories, so-called drive files. Already in Chronos 1, Fengco developed a scenario engine which used dSPACE ASM to generate drive files from a simulated test case. In Chronos 2, this scenario engine was upgraded to eliminate the need for MATLAB® in the generation of static drive files.

The integration of a simulation environment to the Chronos 2 test system meant that many new types of scenarios, from a wide range of different tools and file formats, could be executed in the Chronos 2 test system. There are already many different commercial and in-house tools used for MIL/SIL/HIL testing, with several different file formats for scenario description. WP2 was therefore investigating the possibilities of moving test cases from commercial and in-house MIL/SIL/HIL tools into the Chronos 2 test system.

Some of the scenario description standards are tool-proprietary and some are standardized description formats. One such file format, which is currently undergoing a standardization is OpenSCENARIO. WP2 investigated in depth the possibilities of OpenSCENARIO, both from the view that OpenSCENARIO could be used in the Chronos 2 test system and from the view that WP2 could provide feedback on the standard to the consortium that handles OpenSCENARIO.

It is of course also interesting to create brand new scenarios, and not only reuse imported ones from various other tools. Therefore, WP2 assisted the development and integration of a scenario editor tool able to produce OpenSCENARIO files.

All deliveries related to scenario design and mobility were successfully fulfilled. These deliveries and their result are listed in the sections below. Delivery 2.4, 2.12 and 2.13 were mainly driven by ESI and Delivery 2.6 was mainly driven by Fengco.

WP2.6.2 OSI (Delivery 2.4)

The initial objective of this task was to replace the idealized sensor models typically used in the simulation environments used by some of the project partners and to support them in their analysis of the benefits of such physics-based sensor models in their platform. Several face to face meetings were held with Volvo Car Corporation and AB Volvo where the ESI solution and sensor model architecture was presented.

In parallel, a European initiative emerged to work on a standardized interface for sensors called OSI (Open Simulation Interface). This initiative was transferred to the ASAM consortium, https://www.asam.net/standards/detail/osi/, to work on the standardization of this approach.

As the main topic of this standard is to define the right level information for the interface between simulation platform and sensor model, it was decided to change the focus of this deliverable to perform a survey and contribution on OSI through the ASAM consortium.

Following this modification, a dedicated presentation was made to the interested project partners (Volvo Car Corporation and AB Volvo) of the OSI initiative and a first proof-of-concept done by ESI on this kind of interface to allow to connect a sensor model to a simulator environment.

The diagram below in Figure 5 illustrates the general architecture for the interface.

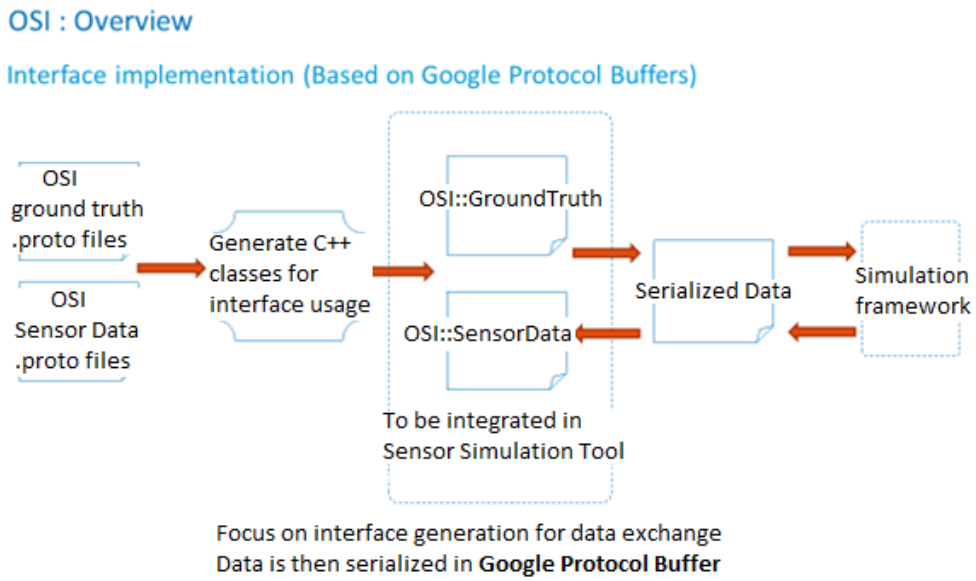


Figure 5. The general architecture of the interface connecting a sensor model to a simulation environment.

Based on this global approach, a first proof-of-concept was made for the lidar sensor model in Pro-SiVIC, Figure 6.

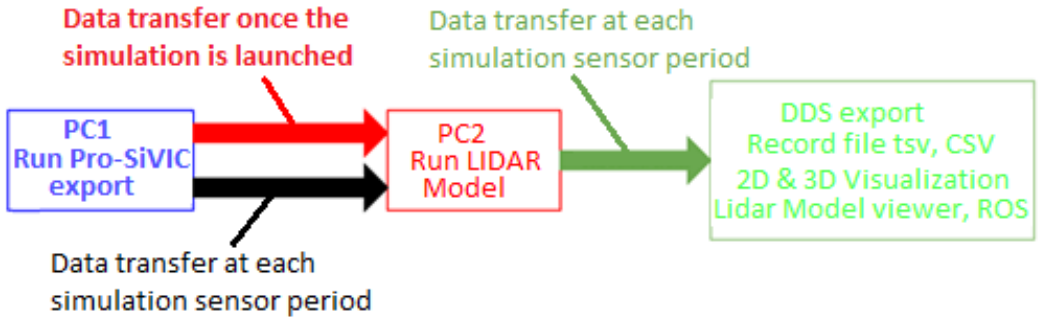


Figure 6. Proof-of-Concept lidar sensor model.

This allowed WP2 to identify in more detail the right level of interface needed for interacting with a physics based lidar sensor model. The following diagram, Figure 7, depicts the variables that must be transferred through the interface to allow an external model to be plugged into a simulation environment.

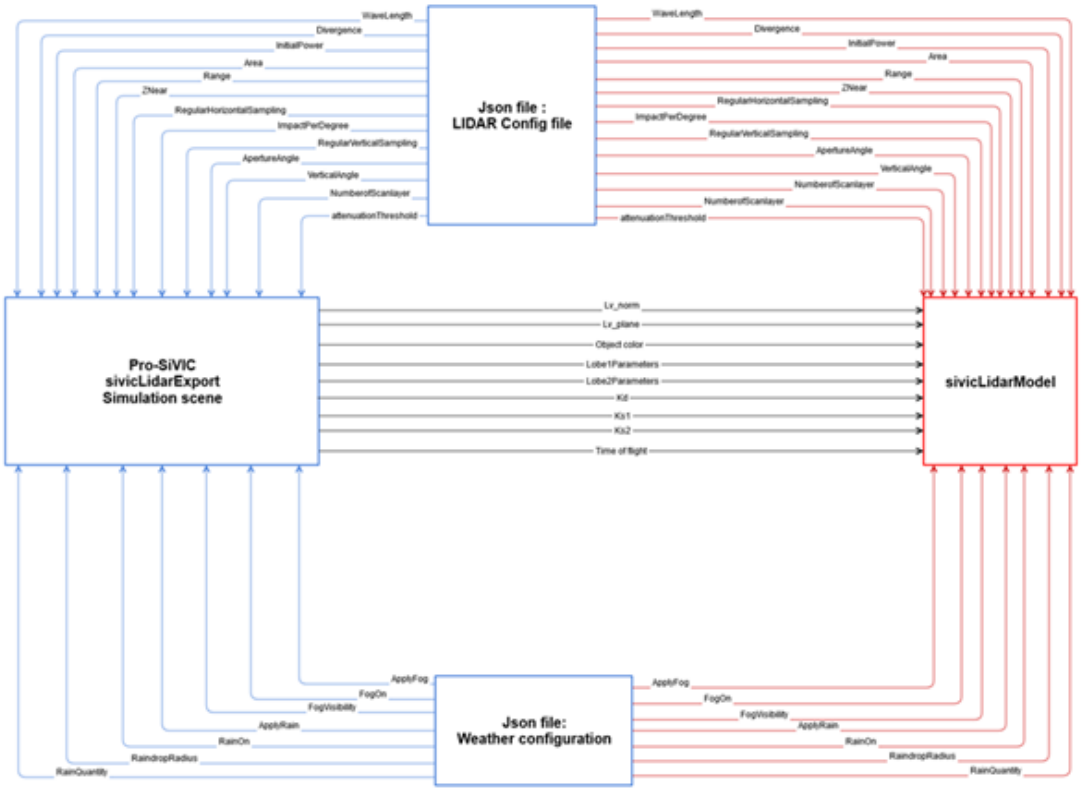


Figure 7. Variables needed for the lidar sensor model.

The first conclusion related to this study is that the OSI is not mature enough yet to allow connecting such a detailed sensor model. However, it remains an interesting approach on which ESI will continue to work.

The early discussions with Volvo Car Corporation and AB Volvo demonstrate the companies' interest in allowing this type of advanced sensor models to connect to their own platform, and this will contribute to define future collaborative work on this topic.

WP2.6.3 Mobility of Scenarios between Simulation Environments (Delivery 2.6)

Since the simulation tool that was integrated into the Chronos test system (dSPACE ASM) already is widely used as a MIL/SIL/HIL environment, this delivery was in one way automatically completed. In the integration of ASM into the Chronos test system, it is possible to use the same interface for reading and executing scenario descriptions as is used in the off-the-shelf ASM tool. It is then of course interesting to explore the different scenario description standards that ASM supports, and perhaps even more interesting to explore how much work is needed to run scenario descriptions from different vendors which are not officially supported. For this reason, scenarios from two different sources were imported into ASM.

The first scenario import into ASM was from OpenSCENARIO files generated by the Scenario Editor Tool developed by Chronos 2 members in parallel with the project. OpenSCENARIO was not yet officially released during this project, so official support for this in ASM is expected sometime in the near future. Within WP2, a small module was implemented as an add-on to ASM which could already now import the OpenSCENARIO files from the Scenario Editor Tool. A similar module was made in order to import two example scenarios from an in-house developed tool from AB Volvo. This shows that the Chronos test system can run imported scenarios from standardized formats such as OpenSCENARIO as well as from some in-house company specific formats.

What was common between both the OpenSCENARIO format and the in-house format from AB Volvo was that they both were using XML to describe the scenarios. In ASM, scenarios are also described in an open XML format, and comes with a user-friendly GUI to create, import, edit and export such XML files. The GUI also comes with an API. This means that it is relatively easy to create import-modules based on scripts, which is the way that both import modules were created.

WP2.6.4 Scenario Editor Tool and Scenario Library (Deliveries 2.12, 2.13)

In parallel with the project, ESI had decided to develop a scenario editor tool. This tool was presented to the project partners and was used to generate the OpenSCENARIO files defined and used in the project. The activity was concomitant to the creation of the OpenSCENARIO database.

The main purpose of the Scenario Editor is to:

1. Simplify the scenario creation stages for Pro-SiVIC users,
   Note that for now, the Scenario Editor is a separate tool that is not integrated in Pro-SiVIC.
2. Help users to define scenarios for test and validation of ADAS functions,
   In this case, the situation(s) that will be presented to vehicles equipped with sensors, detection or control commands functions are precisely described.
3. Manage up to ~ 20 actors
   The number refers to the mobile objects (cars, trucks, motorcycles, pedestrians). This is more related to the simulation goals and is not a strict limitation but rather a context definition.
4. Allow scenario durations of a few minutes.
   Again, this is not a strict limit, but an explanation that a typical scenario is neither a few seconds long nor hours and hours of driving.

The diagram below defines the internal view of the functions supported by the Scenario Editor and the input and output of the tool, Figure 8.

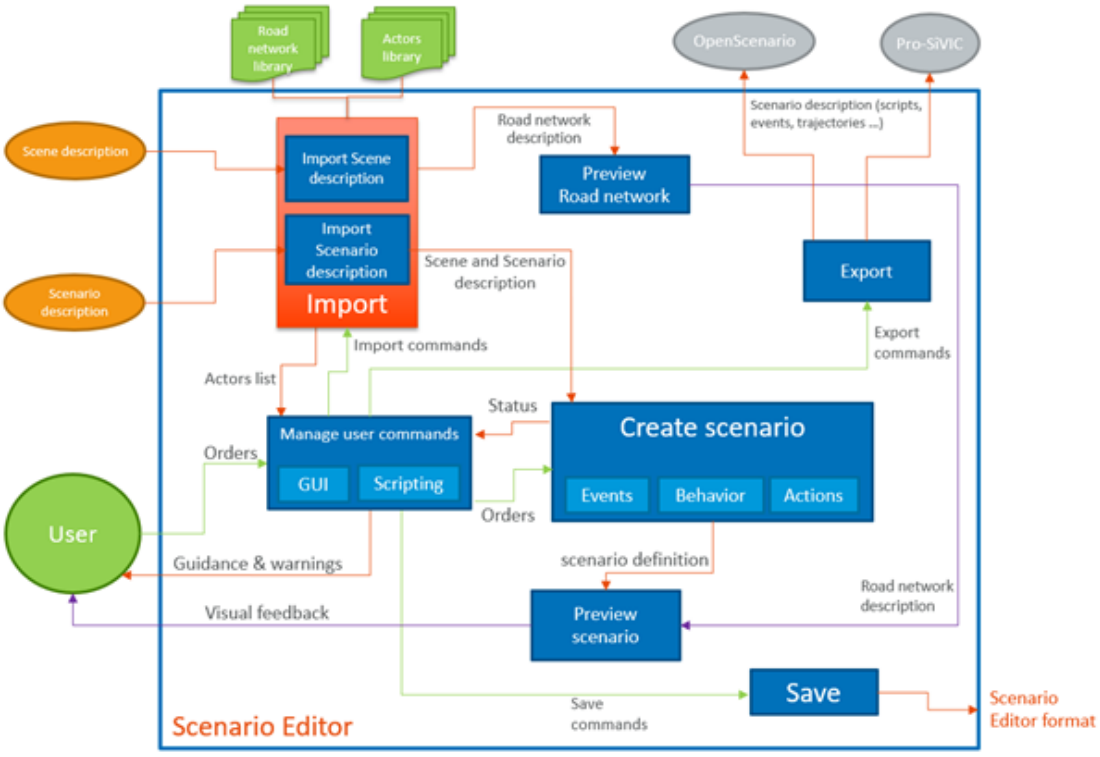


Figure 8. An illustration of functions supported by the Scenario Editor and the input and output of the tool

Thanks to the Chronos 2 project, all the partners share the same understanding of the OpenSCENARIO standard and also ensure that the right level of information is provided, by the way the meta model is embedded in the Scenario Editor.

Several meetings were organized to review results, first the manually created OpenSCENARIO files and then those generated by the Scenario Editor. All these files were shared through the common project file area.

Table 1 below shows the use cases initially identified by the partners of the project.

Table 1. Use cases identified by the project partners.

Use case	Description	Specification
Cut-in		sims status report.pptx
Highway Merge		sims status report.pptx
LTAP-OD		sims status report.pptx
AEB CCRs	on simple track	euro-ncap-aeb-c2c-test-protocol-v201
AEB CCRm	on simple track	euro-ncap-aeb-c2c-test-protocol-v201
AEB CCRb	on simple track	euro-ncap-aeb-c2c-test-protocol-v201

AEB city CCRS		euro-ncap-aeb-c2c-test-protocol-v201
AEB inter-Urban CCRs		euro-ncap-aeb-c2c-test-protocol-v201
AEB inter-Urban CCRm		euro-ncap-aeb-c2c-test-protocol-v201
AEB inter-Urban CCRb		euro-ncap-aeb-c2c-test-protocol-v201
LSS		euro-ncap-lss-test-protocol-v201
CPFA-50		euro-ncap-aeb-vru-test-protocol-v202

Due to the review effort and the test system creation, it was decided to reduce the number of OpenSCENARIO files. The three following scenarios were generated and used by the simulation environment:

- AEB CCRm
- Dynamic trajectory, Figure 9 and Figure 10
- StationaryVehicle_countryRoad, Figure 11 and Figure 12

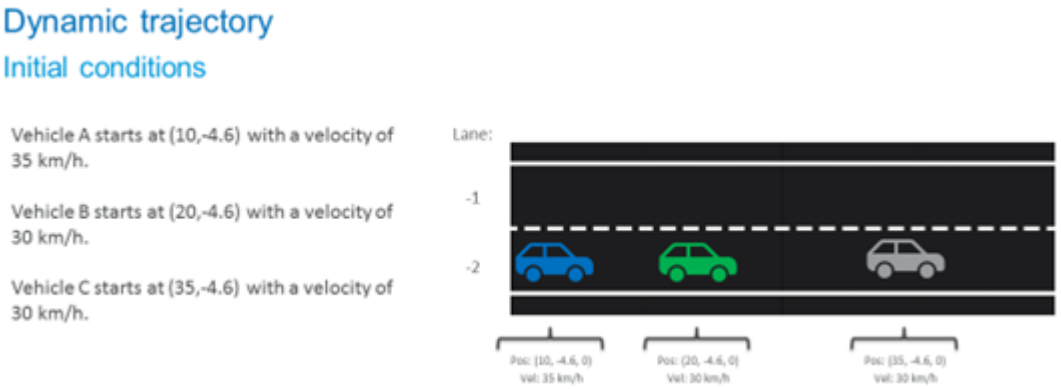


Figure 9. Dynamic trajectory illustration, initial conditions.

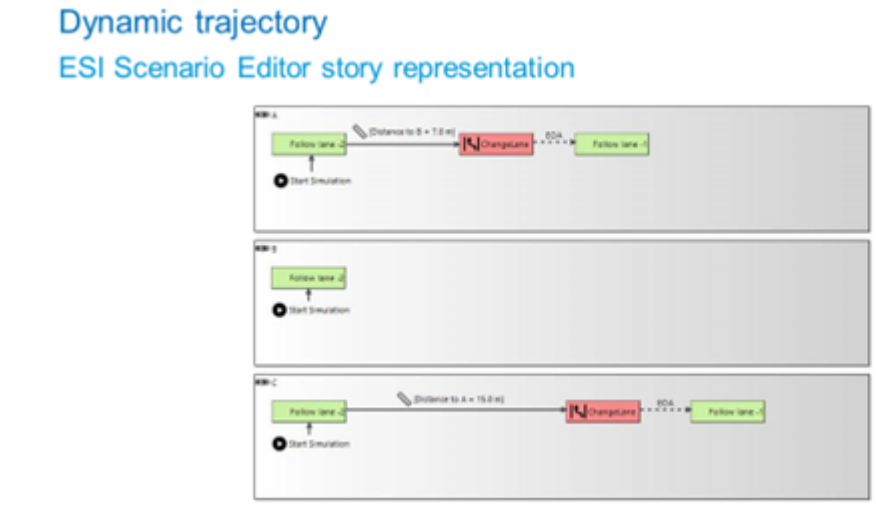


Figure 10. ESI scenario editor representation of dynamic trajectory.

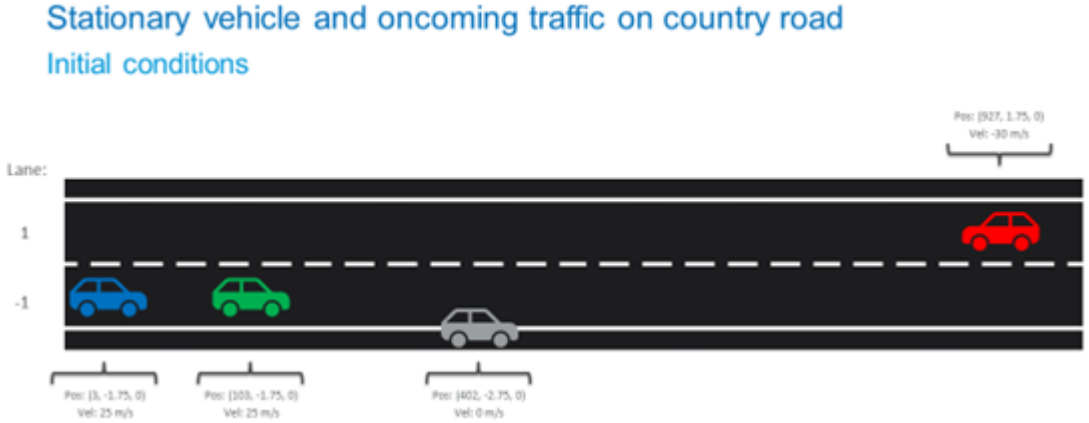


Figure 11. Stationary vehicle on country road illustration, initial conditions.

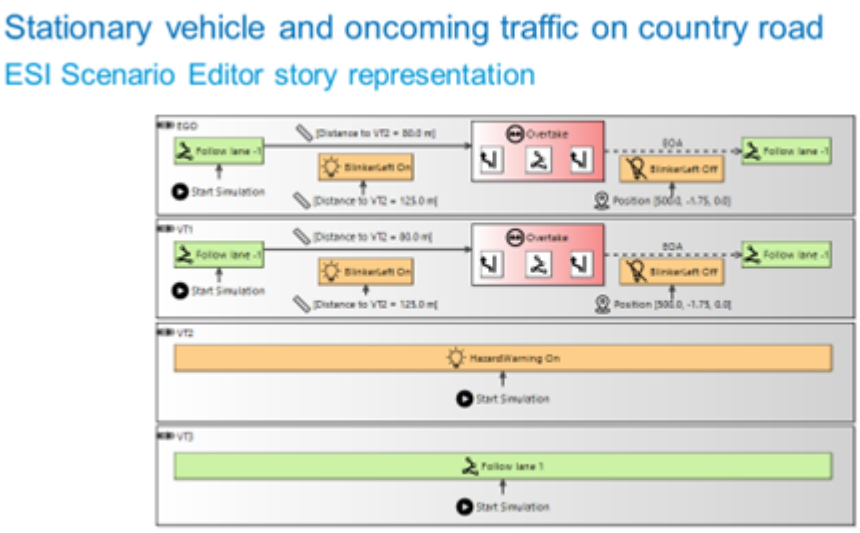


Figure 12. ESI scenario editor representation of stationary vehicle on country road.

WP2.6.5 Demonstrations

In addition to the previously described work, ESI also contributed to the demonstration of the test system server.

The first one, "Demo1", consisted of using the Scenario Editor to define the test scenario and then export the related OpenSCENARIO files to the CHRONOS test system, Figure 13. Pro-SiVIC was then used as a monitor to display, in a virtual environment, the ongoing test on the track.

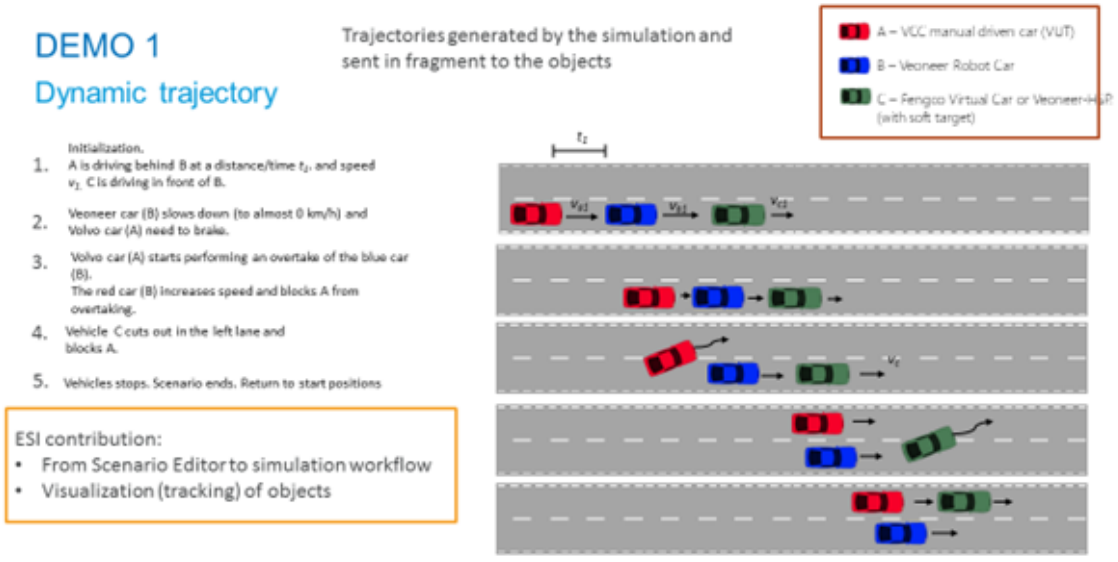


Figure 13. Demo 1 setup.

For the second demonstration case, "Demo 2", Pro-SiVIC was again used as a monitor to display the ongoing test running on the track in a virtual environment, Figure 14.

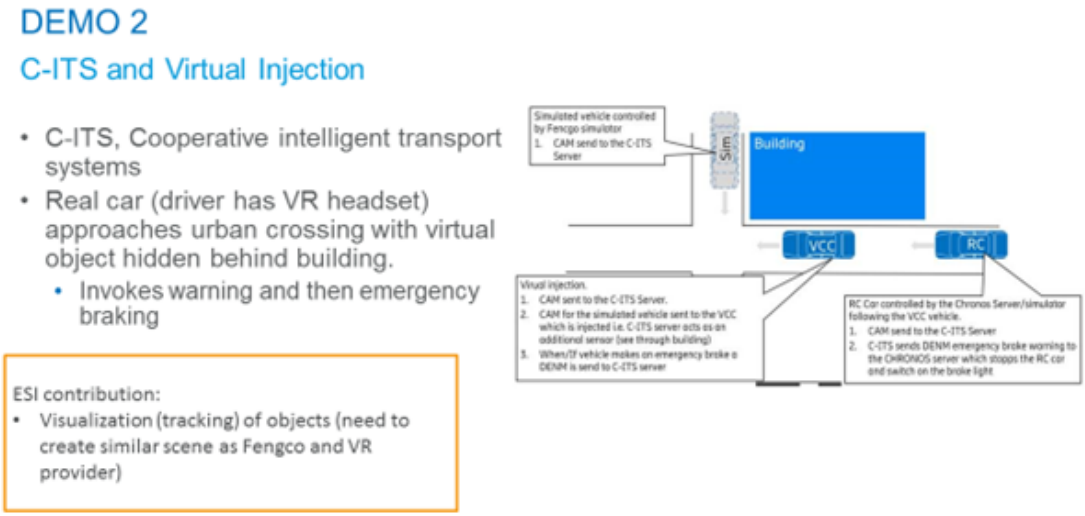


Figure 14. Demo 2 setup.

WP2.6.6 ISO Work

Volvo Trucks has participated as a member of a newly formed ISO group called "Test scenarios of automated driving systems". The formal ISO name is ISO/TC 022/SC 33/WG 09. The main idea with this standard is to facilitate easier use and exchange of scenarios between tools that are used to test automated vehicles and other driving systems.

The group has a broad representation of countries and there is a good mix of OEMs, universities and third-party suppliers.

The work is about halfway, and the due date is 2022. There are no officially released documents yet, but this is a summary of the work packages:

Evaluation of test scenarios for automated driving systems

This group is looking at how to evaluate the quality of scenarios that are created, when they are applicable for a certain level of autonomy. There will be several parts to quantify a given scenario, including its validity, accuracy and effectiveness.

Scenario attributes and categorization

This is about how scenarios can be grouped into different categories. It should help to understand the total test coverage and find the blind spots on the test map.

Taxonomy for operational design domain for automated driving systems

The ODD definition is very important, especially for commercial vehicles. It serves effectively as the contract between the provider and the customer, i.e. during what circumstances and conditions the vehicle should work. It should resolve questions about weather, road conditions, time of day, specific geographic locations *etc.*

Engineering framework and process of scenario-based safety evaluation

This framework is about data base structures and strategies for storage and access of scenarios. The background is that it is expected to be a huge number of scenarios produced, both manually and automatically. Some 3rd party suppliers might specialize in providing libraries of scenarios, hence the need for a standard.

Terms and definitions of test scenarios for automated driving systems

This work group should create a taxonomy of all terms related to this topic. It should help the autonomous industry to describe ideas and concrete scenarios in a harmonized language.

The initial scope of WG9 also included the scenario language to describe the test, but it was decided early to hand over this work to ASAM since this was already ongoing work in this organization. The language will be called OpenScenario 2.0. Nothing has been finalized but the decision is currently leaning towards M-SDL which is developed by Foretellix.

WP2.6.8 Main Takeaways

This project allowed us to share a common understanding on the emerging OpenSCENARIO and OSI standards now led by the ASAM consortium.

Thanks to this understanding we were able to implement the right level of meta model in the Scenario Editor to be able to generate OpenSCENARIO files that can be shared between different tool providers.

The quality of the exchanges between the partners and the complementarity of our skills are already leading us to pursue together on new projects.

WP2.7 AI Integration by Ainomaly

During the project, it was realized that it would be possible to add further work within the budget of WP2. This opened the opportunity to include some collaboration work between Fengco and Ainomaly. The focus was the integration of Ainomaly's tool ROBOTest with dSPACE ASM, i.e. the simulation environment that was integrated with the Chronos Server. The work and results are fully described in Appendix A.

WP3 Physical Object Control and Supervision

WP3	Physical object control and supervision
Leader (role and responsibility)	AstaZero (WP leader, responsible for physical object control server, and overall test server architecture)
Other participants	RISE (safety concepts, supervisor), VCC (Provide scenario specifications, development of steer-by-server control), Veoneer (upgraded control of test objects)
Description of contents	This WP will implement a physical object control server which will steer the physical test targets on the test track, interface with the simulator core, and handle overall test control. The server capabilities should include: 1. Dynamic scenarios support to enable testing of autonomous vehicles 2. Enhanced supervisor to ensure test safety and enable tests of higher risk scenarios. 3. Support and synchronization of virtual and real objects on the test track. 4. Overall test system control, *e.g.*, test server shall be able to operate in a standalone mode for physical object only-scenarios.
Method/approach (when relevant)	The object control server will be based on the Chronos server and will include new features, such as dynamic trajectory updates and supervision. Dynamic trajectories and supervision have been prototyped in the iTRANSIT FFI project (2015-02330) using radio-controlled (RC) cars and the concepts developed will be transferred to Chronos 2. Our approach for the main server functions will be: 1. Simulation to track integration: we will develop an interface to import drive files from the simulation to the physical object control server (coordinated with Simulation Scenarios). 2. Dynamic scenarios: we will implement two tracks: a) additional mechanisms which are pre-planned, “follow that car”, also known as target hunter (*e.g.* object maintains constant position relative to VUT) b) use a simulator core (WP2) to adapt test objects, based on the behavior of the VUT (adaption handled by simulator core). One idea is short drive files. 3. Supervisor: the supervisor service should detect when the VUT or test targets are not following scenario specifications and abort the test if

	safety has been compromised. This will include both a geofencing service, which will define areas interdicted to the actors, and a collision avoidance system, which should send the actors to safe trajectories in case of a possible collision. Two possible implementations of the supervisor will be investigated: a) Supervision logics in the test targets: the test targets will have pre-defined safe trajectories at given critical points in the test and geofencing information to avoid exiting the allowed test area. b) Supervision in the server: the server will analyze the real trajectories of each object and decide when the safety conditions have been compromised. Collision avoidance is based on a reachability analysis for the VUT and a model predictive control (MPC). 4. Synchronization: Both real and virtual actors should be synchronized with scenario specifications. This will require sending the actual position of all real targets to the simulator core. Development of the simulator is described in WP2
Delivery	3.1 Safety concept defined for dynamic trajectories 3.2 Description of drive file format to allow seamless integration of MIL simulations and test track. Coordinated with Simulation Scenarios. 3.3 Implementation of drive file export and replay on the test track with VUT driven by robot. 3.4 Feasibility study dynamic trajectories of objects, including network latency and safety. 3.5 Implementation of dynamic trajectories of objects including “follow that car / target hunter”. Includes update of Veoneer High Speed Platform and driving robot. 3.6 Demonstration of dynamic trajectory update capabilities 3.7 Definition of supervisor logics, characteristics, and architecture 3.8 Implementation of geofencing 3.9 Implementation of supervisor, minimal risk maneuver, based on findings of Delivery 3.7 3.10 Demonstration of server capabilities in at least one of the test scenarios defined in WP1

WP 3.1 Safety concept

3.1.1 Introduction and Scope

The Chronos 2 test system consists of one or several Chronos servers controlling one or several Chronos 2 and/or ISO22133-1-compatible objects. This could be on the same test area or different test areas at the same time. Malfunctioning hardware and/or software in the Chronos server or objects could cause severe injuries and even fatalities. The Chronos 2 test system is a kind of machine, and standards related to machine safety become applicable.

As standalone, the server does not constitute any risk. However, the controlled objects can represent a serious risk to personal safety.

This preliminary safety analysis focuses on the communication interface between the server and the object(s), and especially the messages which are communicated on the two channels: safety (periodic) and control (event). The preliminary safety analysis makes no assumption on the communication media (but it is wireless), but the safety channel uses UDP/IP protocol and the control channel TCP/IP protocol. Default ports are 53240 for UDP and 53241 for TCP, respectively.

3.1.2 Applicable Standards

The following standards are relevant with respect to machine safety and Chronos 2:

- **AFS 2008:3**, Maskiner, Arbetsmiljöverkets föreskrifter om maskiner samt allmänna råd om tillämpningen av föreskrifterna
- **SS-EN ISO 12100:2010**, Safety of machinery – General Principles for Design – Risk assessment and risk reduction
- **SIS-ISO/TR 14121-2:2012**, Safety of machinery – Risk assessment – Part 2: Practical guidance and example of methods
- **SS-EN ISO 13849-1:2016**, Safety of machinery – Safety-related parts of control systems – Part 1: General principles for design
- **IEC 61882:2016**, Hazard and operability studies (HAZOP studies) – Application guide
- **IEC 61025:2006** Fault tree analysis (FTA)
- **SS-EN 61800-5-2**, Adjustable speed electrical power drive systems – Part 5-2: Safety requirements – Functional
- **SS-EN 61784-3**, Industrial communication networks – Profiles – Part 3: Functional safety fieldbuses – General rules and profile definitions
- **IEC 60204-1:2016** Safety of machinery – Electrical equipment of machines – Part 1: General requirements

- **IEC 62745:2017** Safety of machinery – Requirements for cableless control systems of machinery

3.1.3 Hazard Assessment and Risk Analysis

The preliminary risk analysis will focus on the communication between server and objects. It is assumed that no safety mechanisms are present, and that only one failure is present at the same time. It is assumed that the downlink has the following messages:

- Object Setting Message (OSEM)
    - Contains coordinate system (Test Origin)
- Static/Dynamic Trajectory Message (TRAJ)
  also known under its old name: DOPM – Dynamic Object Path Message.
    - Contains the trajectory for one object
- Start (STRT)
    - Starts the test now or at a future point in time
- Stop (STOP) – Signal included in the Heartbeat message (HEAB)
    - Stops the test

and the uplink has the following:

- Monitor (MONR)
    - Contains monitor data (position, heading, velocity, *etc.*) for visualization.

The server cannot be dangerous by itself, but when controlling a heavy or fast-moving object, the server-object combination can be dangerous. It is assumed that persons can be in the vicinity of the object without any preventing barrier in between.

Then the following hazardous events from ISO 12100 become applicable:

- Failure to stop moving parts
- Uncontrolled movements (excluding unintended/unexpected start-up)
- Unintended/unexpected start-up

A **failure to stop** can occur during a test when the object(s) are following its drive file and the operator actively wants to end the test before the drive file is completed or when the object(s) have reached the end of their drive files.

In the case when the operator actively wants to end the test, some kind of device has to enable this action to be carried out in a timely manner, i.e. a stop device with a push-button. However, note that in IEC 60204-1, section 9.2.7, that a "cable-free control panel" (i.e. transmitter) must be equipped with a stop device, which must not be marked as an emergency stop device (red button + yellow background, *etc.*).

Since the overall requirement is contained in the Machinery Directive AFS 2008: 3, last point in section 1.2.1. This is a legal requirement if the plant being controlled is rated as a machine the machine should be stopped automatically if incorrect control signals reach the receiver or in case of communication.

In addition to the above requirements, one must also prepare for what other dangerous errors a remote control may cause in its risk analysis and specifying safety features to reduce the risk of these errors.

Then there is a relevant standard called IEC 62745, which is a bit more detailed. This standard is "stand-alone" from (i.e. not harmonized with) the Machinery Directive. But the standards provide good guidance in implementing the above-mentioned risk analysis.

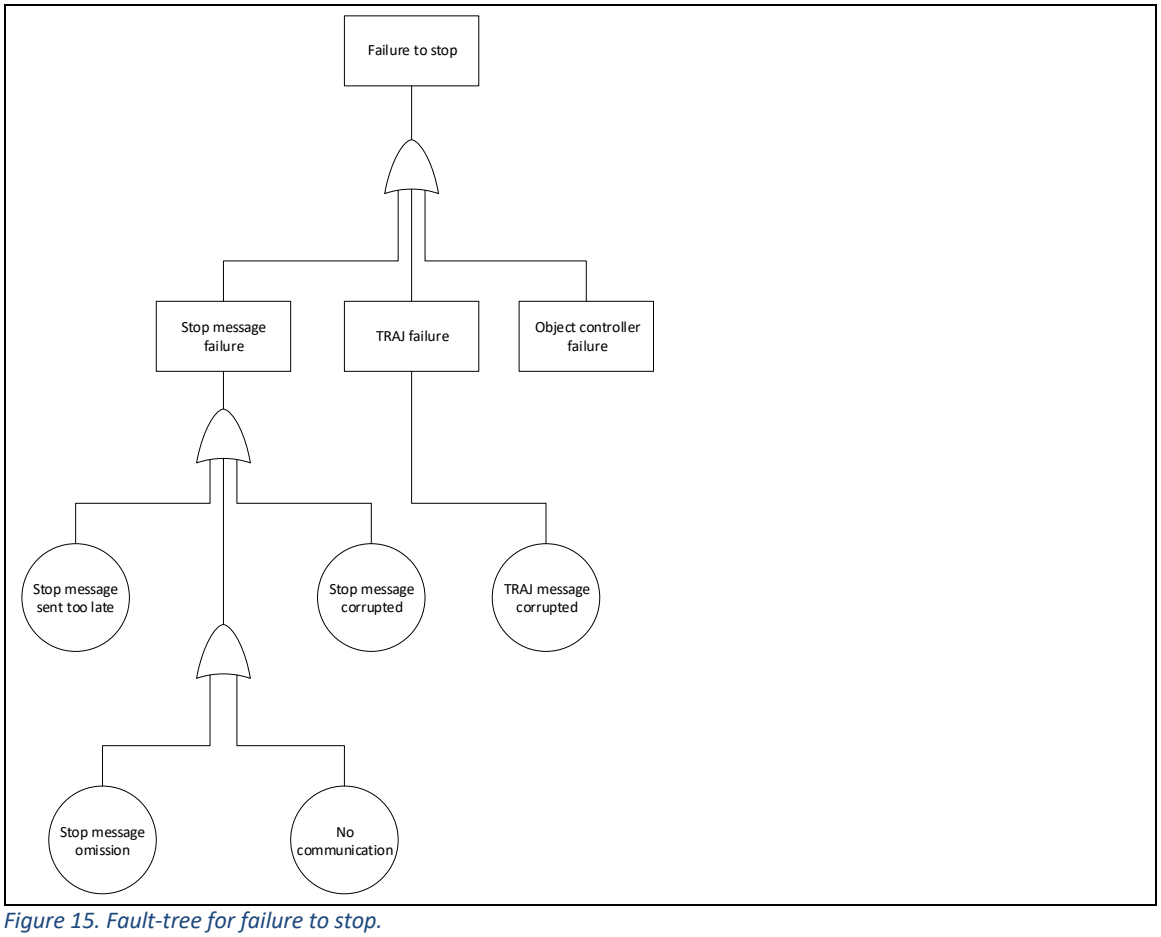


Figure 15. Fault-tree for failure to stop.

As can be seen from the fault tree in Figure 15 above, a failure to stop can be caused by:

1. Failure in the stop message (*e.g.* omission)

2. TRAJ failure (*e.g.* the drive file does not end at zero speed)
3. Object controller failure (software or hardware)

Uncontrolled movements, Figure 16**,** occur while a test its running and the object(s) start(s) to deviate too much from the intended paths in the corresponding drive files.

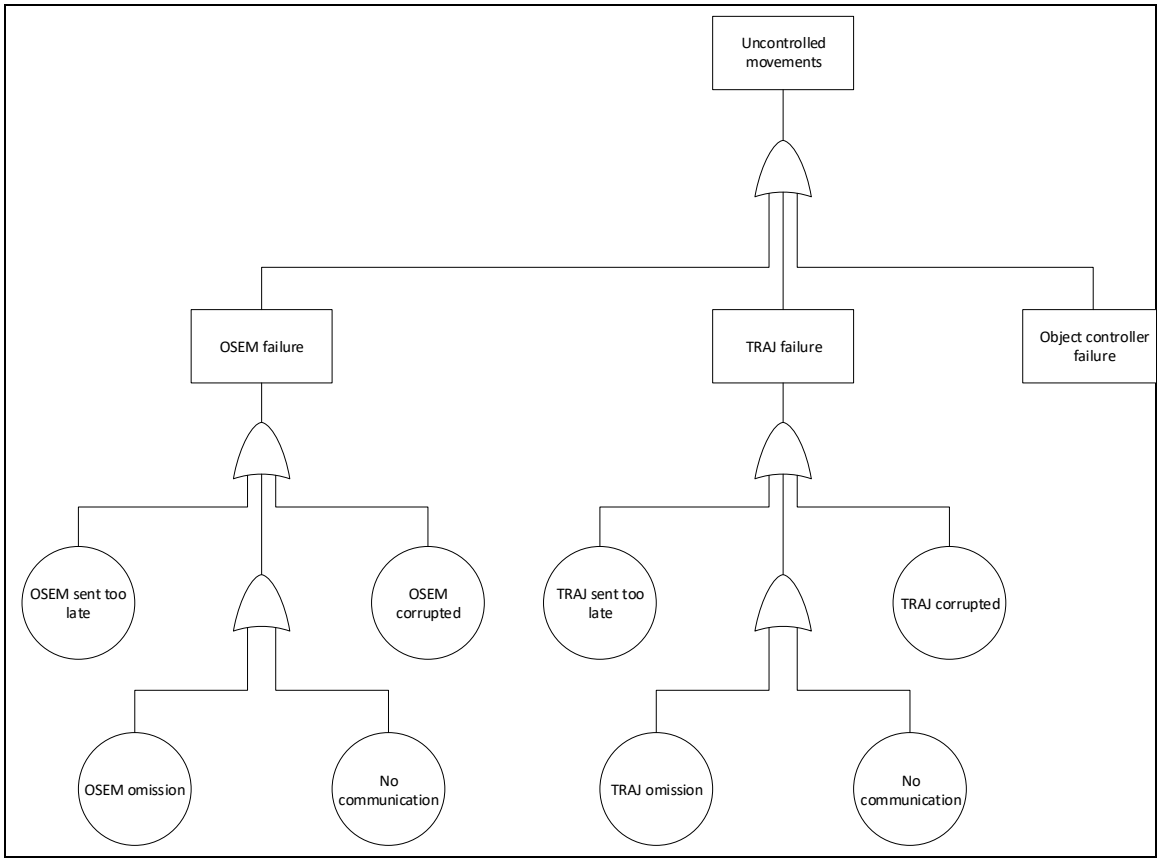


Figure 16. Fault-tree for uncontrolled movements.

Uncontrolled movements can be caused by:

1. OSEM failure (*e.g.* origin configuration error)
2. TRAJ failure (*e.g.* corrupted drive file)
3. Object controller failure (*e.g.* positioning error)

Unexpected start occurs when the object(s) are stationary and starts to move (following its drive file or in random direction) without a corresponding start signal, Figure 17.

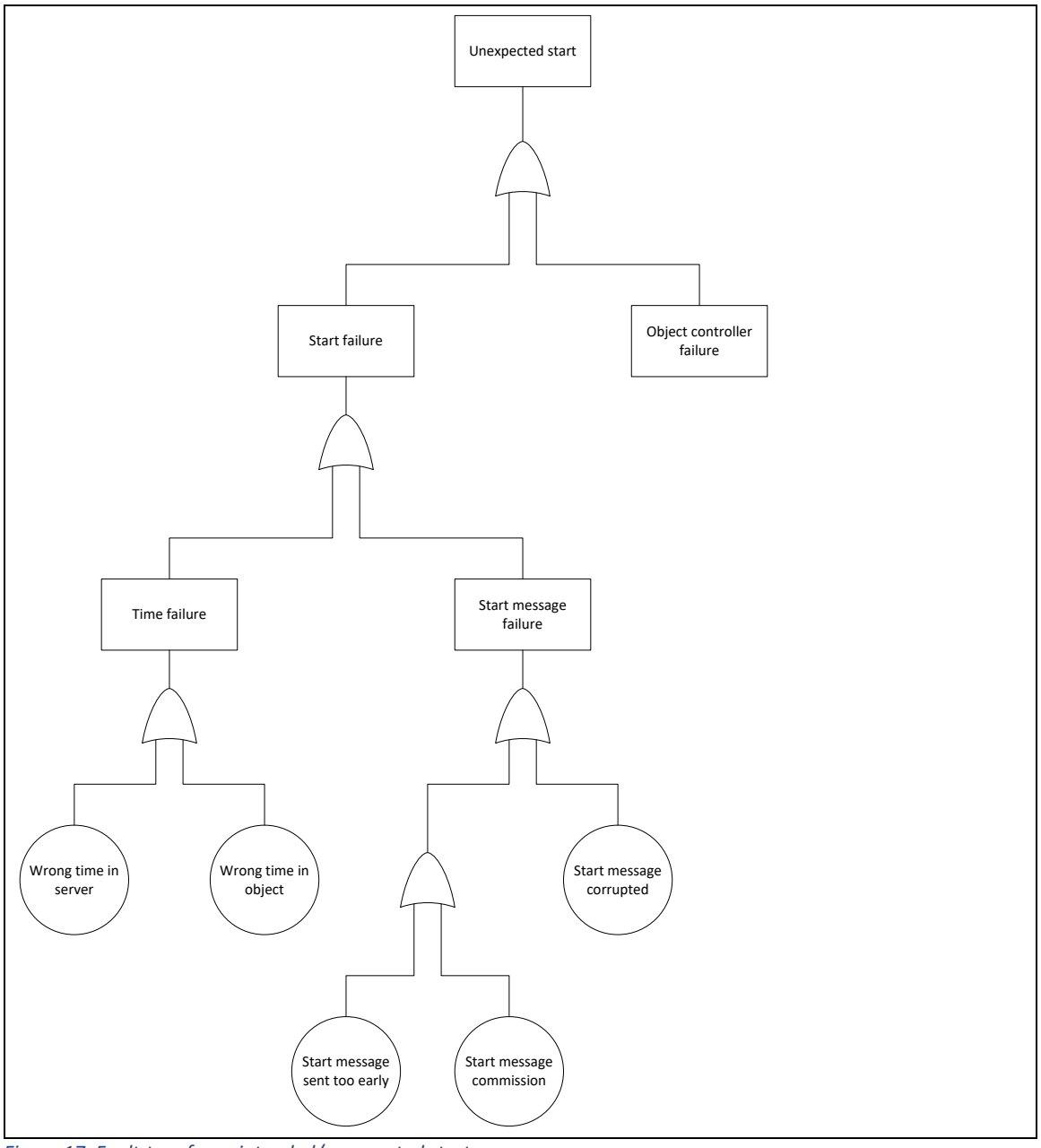


Figure 17. Fault-tree for unintended/unexpected start-up.

3.1.4 Risk Analysis

The risk analysis to determine the required performance level for the three hazardous events above is performed according to Annex A in ISO 13849-1, Table 2.

Table 2. Risk analysis table.

Event	Severity of injury	Frequency and/or exposure to hazard	Possibility of avoiding hazard or limiting harm	Required performance level
Failure to stop	S2	F1	P2	d
Uncontrolled movements	S2	F1	P2	d
Unintended start-up	S2	F2	P2	e

3.1.5 Safety Mechanisms

In order to remove single points of failure and to introduce redundancy, the following safety mechanisms are introduced.

To prevent all three failures:

- Emergency stop functionality based on heartbeat

To prevent uncontrolled movements and unintended start-up:

- Supervision functionality in the server
    - Needs monitor (MONR) messages from the objects
    - Checks that objects are sufficiently close to their trajectories
    - Check that objects are sufficiently separated
    - Checks that objects do not enter forbidden areas

To prevent unintended start-up:

- Arm function (objects need to be in armed state before they accept to start)

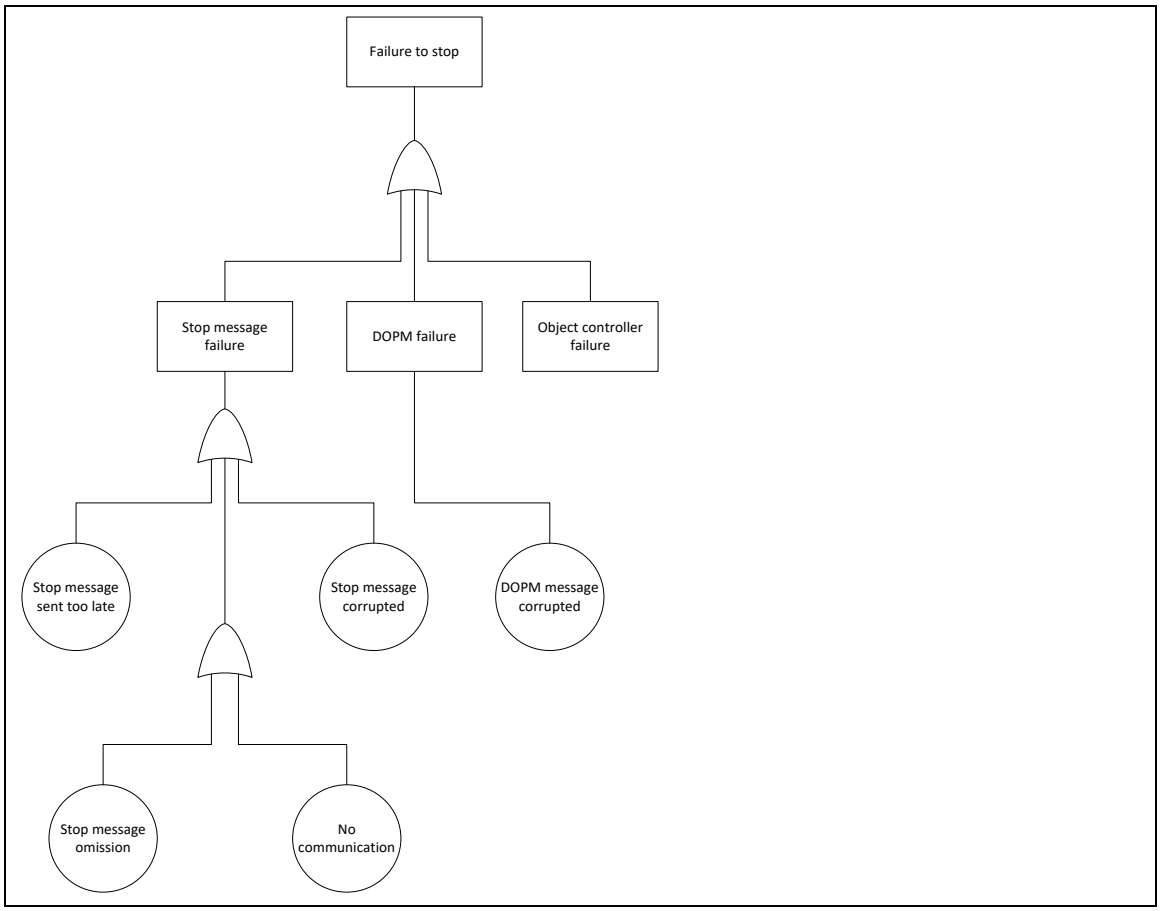


Figure 18. Fault-tree for failure to stop.

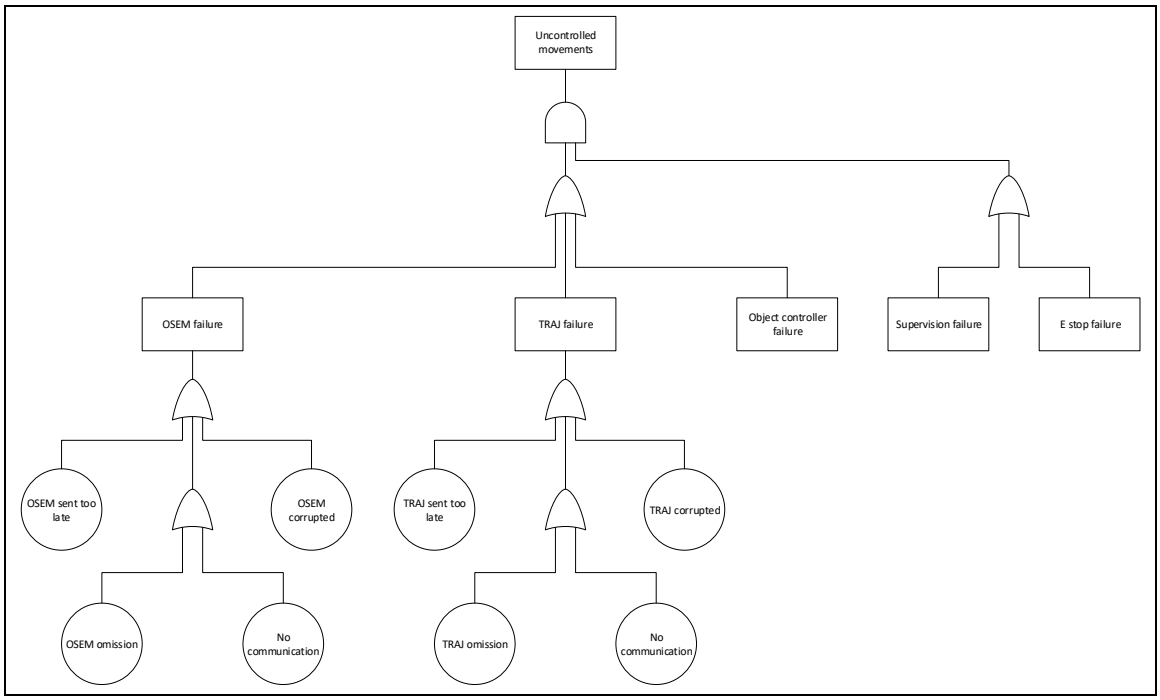


Figure 19. Updated fault-tree for uncontrolled movements.

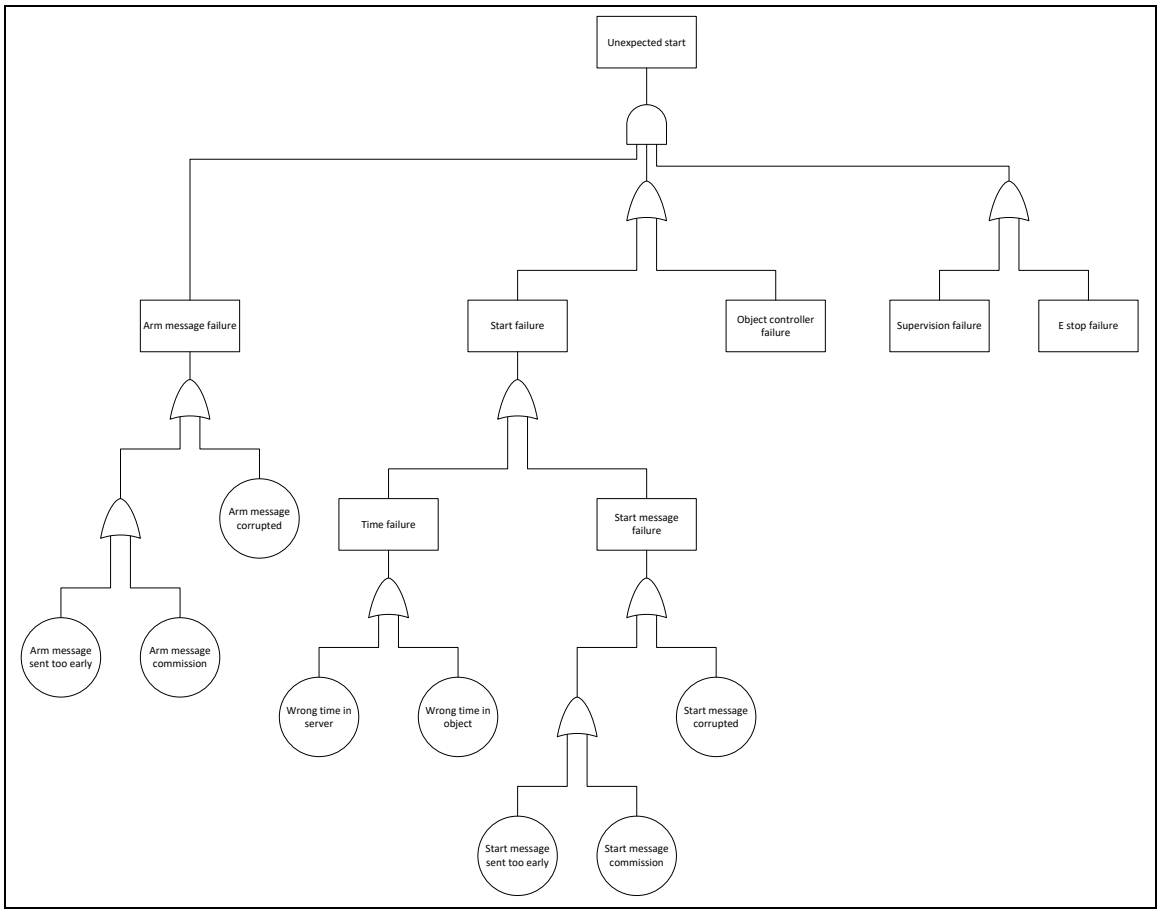


Figure 20. Updated fault-tree for unintended/unexpected start-up.

3.1.6 Stop Strategies

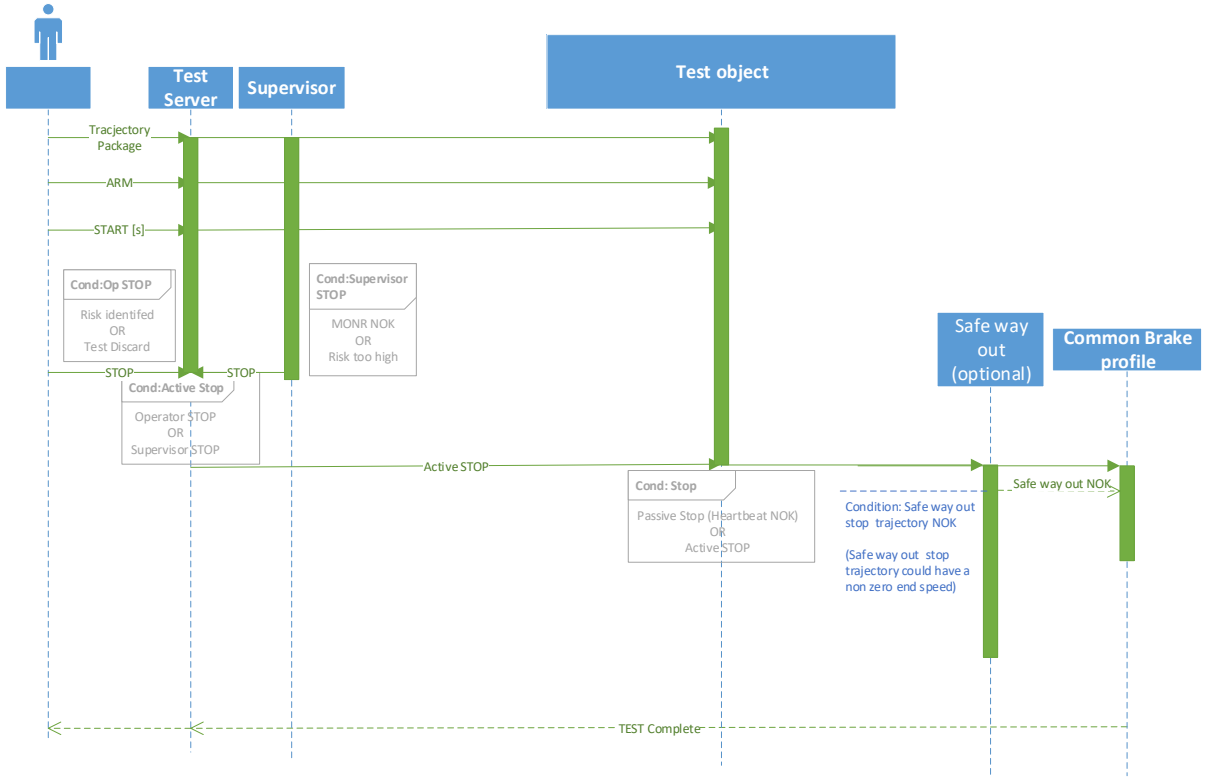


Figure 21. Stop strategies

There are two main strategies to stop (IEC 62745)

1. **active stop** - stop resulting from transmission of a stop signal from the remote station to the base station
2. **passive stop** - safety-related stop resulting from absence of a valid signal at the base station

Where passive stop **must be implemented by the physical objects**, this strategy is the catch-all fallback solution. The active stop initiated a minimal risk maneuvers and is a more intelligent way of stopping and also the preferred way of doing so.

3.1.7 Future work

All identified parameters in the safety analysis has been fully implemented in the Chronos 2 system. For future work and research projects the following parameters needs to be defined more in detail:

- MONR NOK – Supervisor detects MONR absent or more than X ms old (new data, out of order, corruption). -> Action: Active stop and no more heartbeat?

- Heartbeat NOK – Object detects heartbeat absent or more than X ms old (new data, out of order, corruption) -> Action: no more MONR?
- To safeguard against uncontrolled movements, establish criteria for
    - PreRunCheck: OSEM NOK
    - PreRunCheck: TRAJ NOK
- RunTimeCheck: Object navigation NOK = no fixed solution for Y time/Z samples -> Action: no more MONR?
- Risks identified by the operator to be defined in risk analysis for each scenario/test
- Risk too high identified by the supervisor. This is the one research topic, it could be that the supervisor is unable to find a minimal risk maneuver for all objects given the present conditions extrapolated into the future, or that some kind of risk contour surrounding an object will be violated, which is a more general way of solving the issue. Could be applicable on the unpredictable AD-VUT while the first strategy could be applicable for the server-controlled test objects.

WP 3.2 Description of drive file format and coordination with Simulation Scenarios

The drive file, or trajectory file format as it is called, is following the *ISO-22133-1 Test Monitoring and Control* specification. The concept of dynamic trajectories developed in Chronos 2 is now also being implemented in ISO-22133-1. The Chronos 2 project is hereby leading and giving important feedback to the development for this ISO standard.

The FFI funded project *Simulation Scenarios* showed that the Chronos server is also capable of importing an OpenScenario file, recalculating it to an ISO-22133-1 trajectory file and then sent to the physical test object.

WP3.3 Drive file export and replay on test track with robot driven VUT.

A function implemented in the Chronos server makes it possible to export a recorded log file as a trajectory file (TRAJ). This trajectory can later be sent to any Test Object with ISO 22133-1 support such as a robot car or guided soft target platform. In this project this was proved using a RC-car (supporting ISO 22133-1). The car was initially manually driven, and the server recorded the log file. Then the trajectory export function generated a trajectory file which was sent to the RC car. This sequence was repeated over and over again which is shown in the picture below.

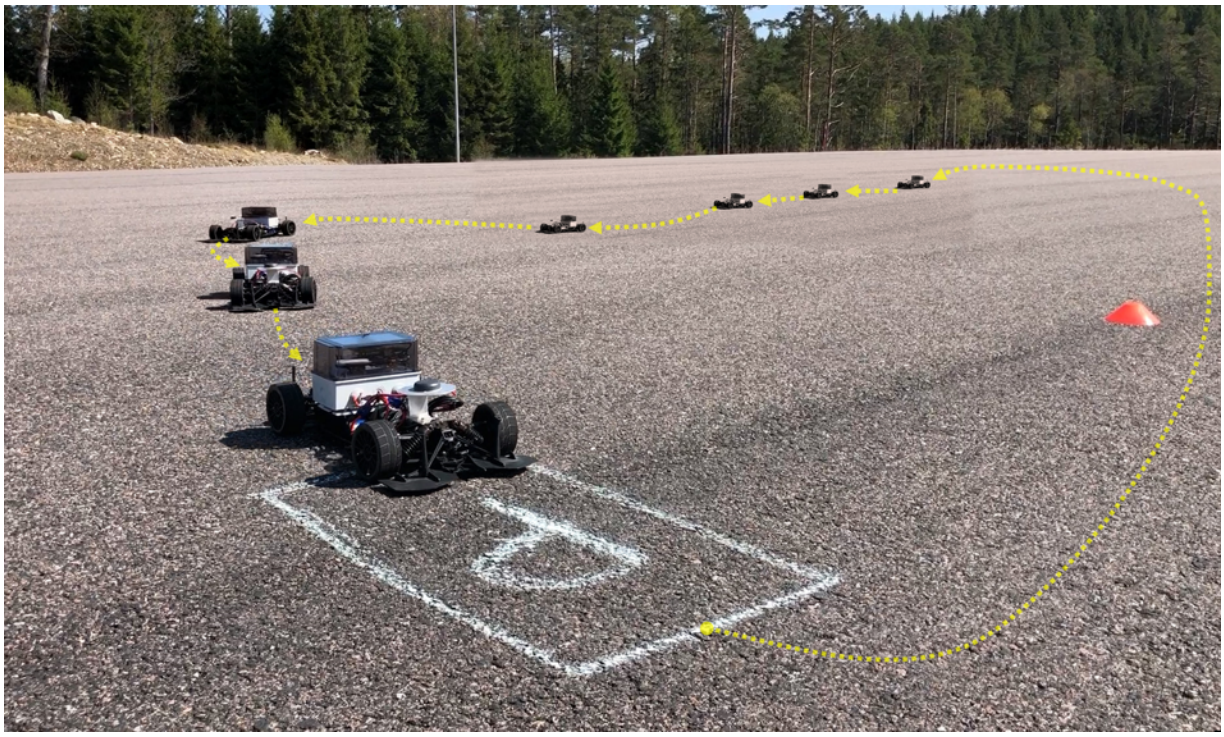


Figure 22. AstaZero RC-car following a replayed log file on the AstaZero test track.

WP 3.4 Feasibility study dynamic trajectories of objects, including network latency and safety.

3.4.1 Use case for the study

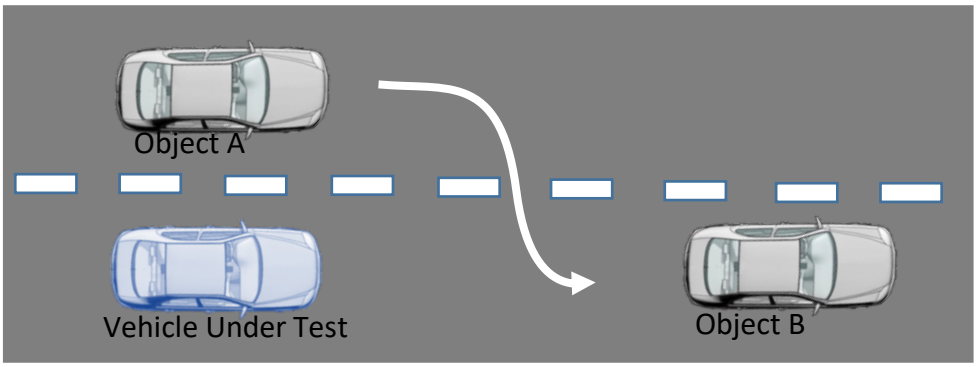


Figure 23. Illustration of the use case for the study.

The Vehicle Under Test (VUT) is following a vehicle (Object B), either in a platoon, with adaptive cruise control, or autonomous drive function, Figure 23. Another vehicle (Object A) cuts in from the left. VUT must then brake or adjust distance without changing lane. The test objects will vary speed, distance from VUT, and cut in maneuver timing, based on scenario requirements and VUT behavior.

3.4.2 Uncertainty factors affecting the test scenario

The project made a study related to safety questions based on four identified areas: Latencies, CPU processing time, vehicle dynamics and positioning accuracy:

- Latency
  - LTE network (based on network measurements in Chronos 1): 15-40 ms
  - LTE network spiked; up to 150 ms
- Processing
  - Commands for robot and lowrider: 20 ms
  - Communication between Chronos server, simulator and supervisor: 0.1-1 ms
  - Generation of trajectories and incl safe-way-out in supervisor: 1-50 ms
  - Generation of safe-way-out: 0.2 ms – 10 ms.
- Dynamics
  - Max deceleration of test objects (lateral and longitudinal)
    - Car: 0.5-0.8 g lat. 0.8 long.
    - Lowrider with dummy: 0.7 lat. 0.25 g long,
- Positioning
  - Accuracy for RTK GNSS: 0.5 m or better.
  - Update rate: 100 Hz

These assumptions give the following min/max table

Table 3. Min/max table based on the above assumptions.

Factor	Min	Max
Total lag (communication + computation + actuation) [ms]	50	100
Max deceleration longitudinal [m/s^2]	6	8
Max acceleration lateral [m/s^2]	2	5
Position accuracy [cm]	2	5

Table 4. Min/max calculations.

Relative speed [km/h]	15	30	50	80	120
Braking distance Min [m] (L=2*pos_acc+V*lag_min+0.5*V^2/along_max)	2.3	6.3	15.2	35.9	77.3
Of which, due to lag [m] (V*lag_min)	0.21	0.42	0.69	1.11	1.67
Braking distance Max [m] (L=2*pos_acc+V*lag_max+0.5*V^2/along_min)	3.8	9.1	20.1	45	94
Of which, due to lag [m] (V*lag_max)	0.42	0.83	1.39	2.22	3.33
Longitudinal distance, coll. avoidance by steering, Min [m] (alat_max)	3.8	8.1	13.5	21.5	32.1
Longitudinal distance, coll. avoidance by steering, Max [m] (alat_min)	6.4	11.8	19.4	31.2	46.3

```
%alat=4.9 % max lateral accelleration m/s2
alat=2.5 % max lateral accelleration m/s2
v=[15.,30.,50.,80.,120.]./3.6% velocity m/s
d=2. % lateral distance in m
r=v.^2./alat
w=alat./v
theta=acos(1-d./r)
t=theta./w
l=r.*sin(theta)
```

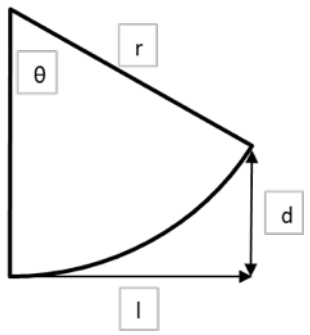


Figure 24. Avoidance by steering along a circular arc, constant velocity and lateral acceleration, assuming lateral distance of 2 m.

WP 3.5 Veoneer implementation of dynamic trajectories

In the first Chronos project a basic implementation of the first Chronos version was added to the Veoneer test objects "Robot Car" (RC) and “Low Rider”. This was based on the first draft of the standard without dynamic trajectories and related functionality. The implementation was added as a separate system outside the regular controller.

During Chronos 2 the implementation has been rewritten and improved according to the new specifications. The test objects now support the latest version of the protocol

and the new Chronos state machine has been fully integrated with the Veoneer software controller.

The handling of drive paths has been replaced to fully support dynamic trajectories. This means that such features as “follow that car” and “target-hunter” are made available using dynamic trajectories.

Fault handling and emergency stop has been upgraded and is now including controlled braking with gyro support for the Low Rider, to avoid skidding and uncontrolled motion during emergency braking.

A third test object called “Balloon Car” has been upgraded to be able to be used together with the “Robot Car” and the “Low Rider” in the Chronos test setup.

WP 3.6 Dynamic Trajectory

3.6.1 Function description

An online planned trajectory is information intended to be used by each Moveable Object to follow a specific trajectory in space and time, and it holds all necessary data for each Moveable Object to control the position and movement. The online planned trajectory is in the Control Center with respect to real-time events.

The Dynamic Trajectory is at start of scenario execution not known by Test Object nor Control Center but calculated during execution of test.

The dynamic trajectory will be sent to the test object during ongoing test in smaller pieces or fragments utilizing the *Trajectory Message (TRAJ)*

Dynamic trajectories serve several purposes, for example:

1. The supervisor utilizes dynamic trajectories for calculating safe-way-out trajectories and can stop the running test scenario, in case of an emergency
2. It adds the capability for, and testing of, a subject vehicle with less constraints assuming that the vehicle acts more unpredictably compared to a predefined static path (*e.g.* autonomous vehicles)
3. It is possible to avoid dangerous test scenarios where humans are at risk of injury
4. Complex test scenarios where a simulator generates the trajectories can be carried out
5. Both basic and more complex test scenarios can be made increasingly efficient with regard to time and cost

3.6.2 Communication flow overview

This section describes the typical data flow between the different systems. The involved parts in this part consists of: A. Simulator, B. Test Server, C. Supervisor, D. Test Object(s) and E. Physical STOP button.

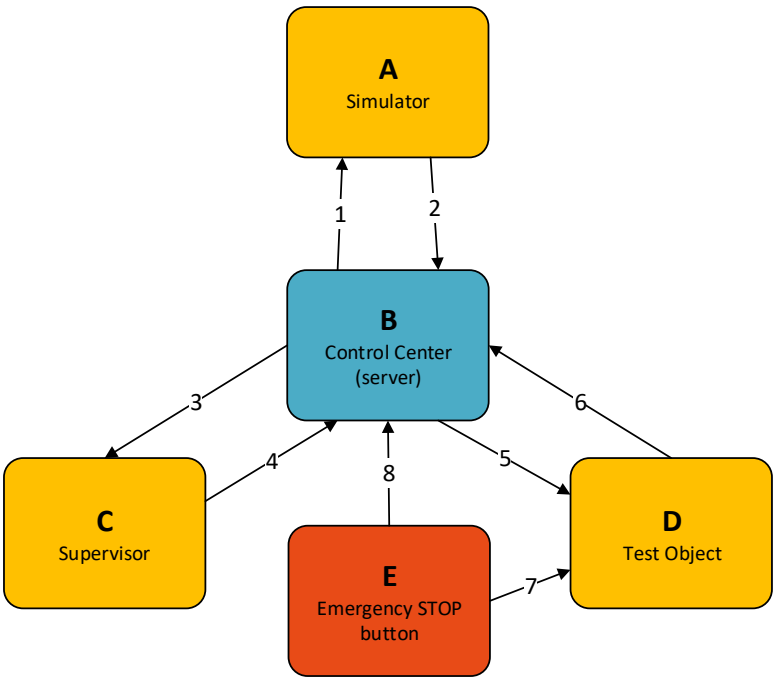


Figure 25. Communication flow overview.

Type of data/message	Data origin	Transmission path
MONR (Monitor Data)	D	6→1, 3
TRAJ (Trajectory)	A (dynamic) / B (static)	2→3→4→5
Initialization message	B	1, 3, 5
OSEM (Object Setting)	B	1, 3, 5
STRT (Start message)	B	1, 3, 5
Normal STOP	B	5, 1
Emergency STOP	B	1, 3, 5
External Emergency STOP	C	4
External Normal STOP	A/C	2, 4
Emergency button	E	8→1, 3, 5 and 7

The recommendation for transmission paths 1/2 and 3/4 and 8 is that they shall be physically connected via ethernet cables, preferably to the same subnet. Transmission path 4/5 (and 7) is typically using a wireless communication interface, *e.g.* a cellular LTE or 5G network.

Definition of STOP-types:

The definition of the different ways of stopping a physical object can be read in the safety strategy section.

3.6.3 Fragment description

A trajectory fragment consists of several positions, way points. Combining several trajectory fragments resulting in a full trajectory

3.6.3.1 Fragment starting point

The starting point of a fragment must be a number of milliseconds in the future. From a "closed-loop"-perspective, a low value is desired.

The positions must be placed in *the future*, meaning that the time specified for the trajectory position must be greater than current time.

In the Chronos 2 project demo, an overlap of 900 ms showed to be sufficient for the use case demonstrated.

3.6.3.2 Fragment start time passed

If the object has passed a position (for example t_{10}) when next fragment is received, starting already at a passed time-location (for example t_8). Then the Test Object must continue to follow the current (old) trajectory and reject the new fragment. If possible, this shall also be reported to the Control Center.

3.6.3.3 Fragment length

There is no minimum length (number of positions) of a trajectory fragment but at least one point is required to be included in the message. Normally several (10-1000) positions are included in a fragment. In the Chronos 2 project demonstration, a message length of 1500 bytes with a frequency of approximately 7 Hz.

3.6.3.4 Fragment ending during execution

If a Test Object is in state RUNNING and the trajectory fragment ends and no new trajectory fragment is received, then then one of the following mechanisms applies:

A. By default, each trajectory is always appended with a "safe way"-stop in the end of each trajectory fragment. (i.e. the ending of trajectory fragment which is sent to the objects contains a deceleration according to the objects brake profile.)

Note: The supervisor is responsible for appending the trajectory fragments with this deceleration in the trajectory.

B. If the object is still moving and the end of a trajectory is reached, the object shall perform an emergency stop and stop as possible.

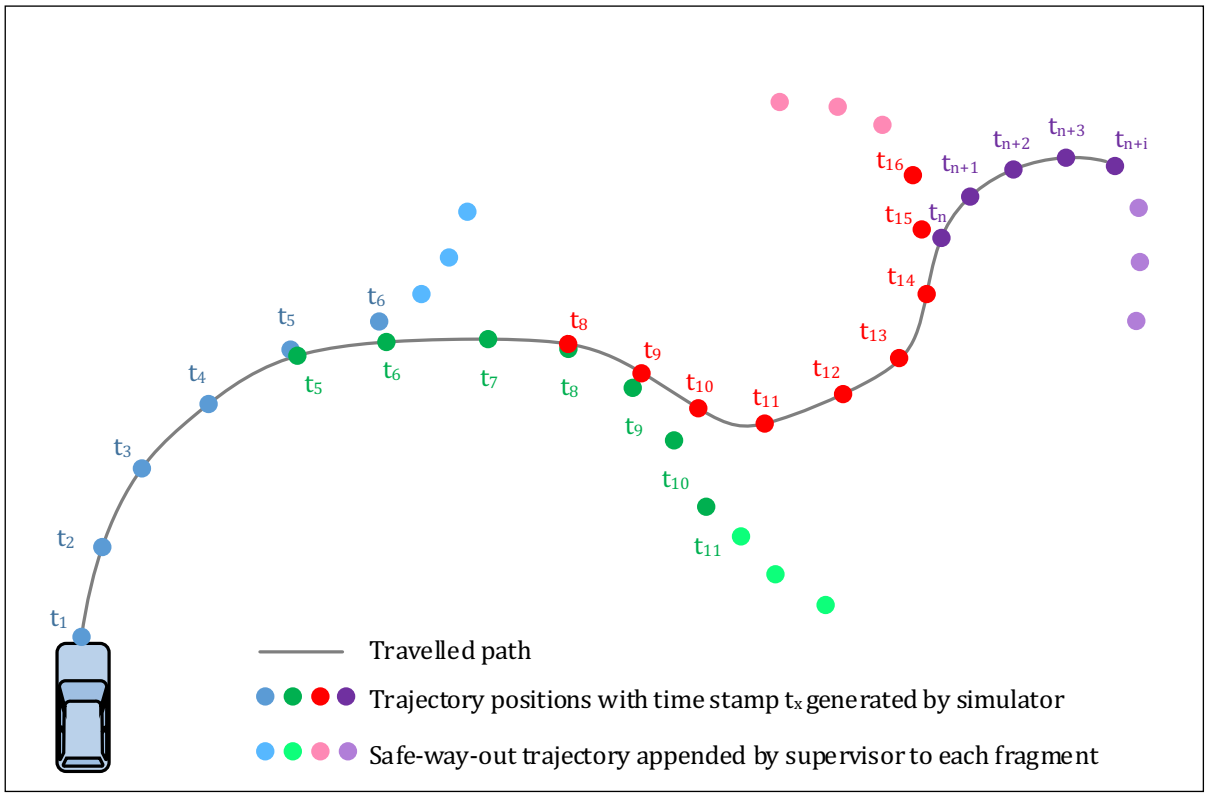


Figure 26. Dynamic trajectory fragment interpretation

Figure 26 above visualizes a schematic view of a hypothetical trajectory with the individual fragments and the corresponding travelled path illustrated. Several trajectory fragments are combined to one longer trajectory. Table 5 below shows the same fragmentation/path but in a tabulated view.

Table 5. Table visualization of trajectory fragments.

Timestamp	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}	t_{17}	t_{18}	t_{19}
Fragment 1	t_1	t_2	t_3	t_4	t_5	t_6													
Fragment 2					t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}								
Fragment 3								t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}			
Fragment *n*															t_n	t_{n+1}	t_{n+2}	t_{n+3}	t_{n+4}

3.6.4 Sequence for transferring Dynamic Trajectories

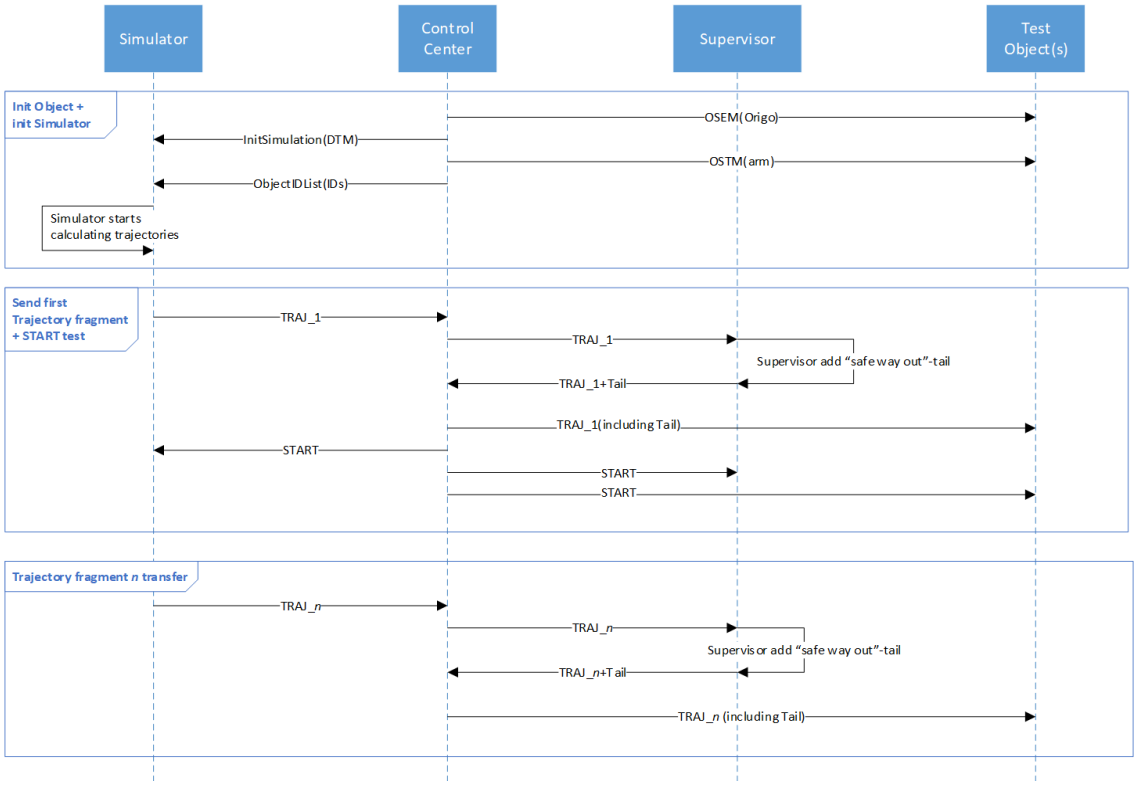


Figure 27. Sequence for transferring dynamic trajectories.

The sequence chart above describes a typical sequence for initialization and start of a *Dynamic Trajectory*-session. The standard use case is described more in detail here.

Preconditions:

- Test Object standstill at its start position.
- TCP/UDP connection established between Test Objects and Control Center. (MONR and HEAB is sent)
- Connection between Simulator and Control Center is established.
- Connection between Supervisor and Control Center is established.
- Supervisor has a map over the test execution area – knowing the boundaries and geofences for the test track.
- Object State = DISARMED

Initialization:

1. Object Properties (OPRO) message is sent to the objects and external supervisor, containing TransmitterId to be used by the objects respectively
2. Object Setting Message (OSEM) including test scenario Origo (lat/long) transferred.
3. Control Center sends Object State Change Request Message (OSTM) *objStateChangeReq(arm(2))* to all Objects.
4. Control Center sends initialization message and ObjectID list to the Simulator.
5. The Simulator starts calculating trajectory (at least the first trajectory fragment)

Main Flow of Events (Start):

1. Simulator is sending the first trajectory fragment to Control Center.
2. Control Center forwards the trajectory fragment to the Supervisor.
3. The Supervisor analyses the fragment and appends a "safe way out" path to the trajectory fragment. And returns this extended fragment to the Control Center.
4. Control Center sends the first trajectory chunk using the Trajectory Message (TRAJ) to the Test Object(s).
5. When at least one trajectory fragment is sent to all connected Test Objects, Control Center sends START-message to Object(s), Supervisor and Simulator. This indicates the start of the test. →Test is RUNNING.
6. Test Object shall now follow the trajectory.

Main Flow of Events (Sending new fragment – during execution):

7. Control Center is sending next trajectory fragment and Object is inserting this fragment at the time position according to the trajectory time information.
8. The Test Object shall always adhere and follow the latest incoming trajectory fragment that contains trajectory positions in the future.

3.6.5 Message structure for Dynamic Trajectories

The same message as for normal trajectories (according to ISO 22133-1) shall be used. The only difference is that the Test Object must be able to accept new trajectories during execution/running.

WP 3.7 – 3.9 Supervision of test scenarios (including geofence)

The supervisor, Figure 28, is a component within the Chronos test server together with the object control unit and the simulator. The supervisor's task is to monitor the test scenario, as it is playing out, for potential dangers. An additional task for the supervisor is to calculate paths that Chronos controlled objects should follow in case the scenario is aborted as result of a safety violation, we call such a path a *safe way out*.

Based on the trajectories produced by the simulator, the MONR messages (containing position information) from objects on the test-track and its own information about geofences, the supervisor can check for dangerous inconsistencies and in that case abort the scenario.

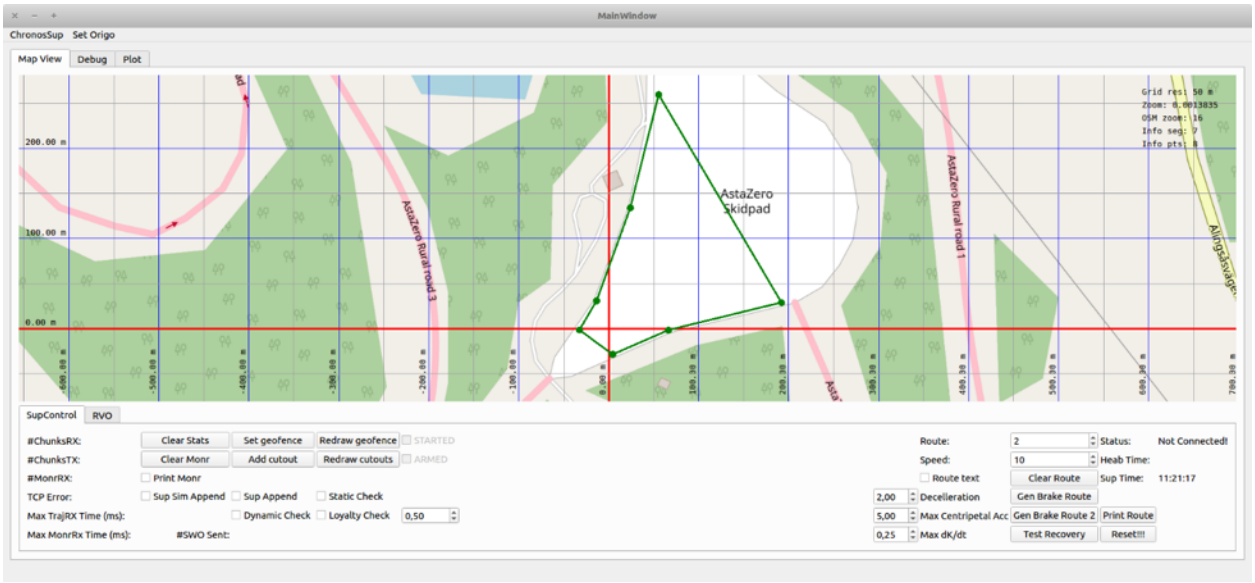


Figure 28. Supervisor graphical user interface with a geofence drawn on the AstaZero high speed area.

Implementing a supervisor for a system like this has turned out to be quite a difficult task. There are numerous constraints in place, and conflicting desires and for this reason the supervisor does currently not know how to handle an autonomous VUT.

The constraints within which the supervisor must operate, fall in the following categories:

- Timing
  - Each Chronos object sends MONR messages at 100Hz. Which means one fresh MONR every 10ms per object.
  - The simulator sends out chunks of trajectory for example every 150 ms, one to each object that is operating on a dynamic trajectory.
- Physical
  - Vehicle dynamics. For example, turning, acceleration and breaking capabilities.
- Knowledge
  - No information is available to the supervisor of what an autonomous VUT is intended to do.
  - Nor is any information about what correct autonomous VUT behavior is.

Out of these constraints the missing knowledge is of course the most important and this is why handling of an autonomous VUT has been pushed to future work. The physical limitations are there and there is nothing to do about that, but the timing constraints

can probably be resolved with a combination of a more powerful and dedicated supervisor machine together with optimization and parallelization. Another change that could improve things when it comes to timing is if the objects didn't send MONR at a fixed rate of 100 Hz, but rather scaled their MONR rate based on the speed with which they are traveling. The criteria could be that a MONR is sent for every change in position that is larger than some distance that we determine beforehand.

Walkthrough of supervisor functionality

The Chronos server connects to the supervisor over TCP and sends an INIT_SUP message. The supervisor takes this as a cue to perform a reinitialization of its internal data structures and get prepared to monitor a test. The Chronos server also sends over a set of object properties messages (OPRO) with information about the objects involved in the test. The OPRO contains IP addresses and object IDs that the supervisor can use to organize data on a per object basis. A plan for future work is that the OPRO should also contain physical properties of the object, a vehicle dynamics profile of some form. A message containing the origin coordinates are also sent to the supervisor at this time to ensure that the server and supervisor agree on where the test is taking place.

Following this setup phase, the server sends an ARM message. Upon reception of ARM the supervisor starts displaying information from the objects MONR messages in the GUI. If MONR messages arrive that does not seem to correspond to an object as identified by an OPRO the supervisor will complain.

When the server issues ARM, the simulator starts to generate some initial "start-up" trajectories. These trajectories also arrive at the supervisor and are displayed in the GUI.

When the system is armed and ready, all we do is wait for the START command and once that happens, the supervisor will continuously perform the following tasks for as long as the scenario is running:

- For each chunk of trajectory from the simulator
  - Calculate a safe way out that ends in speed zero (see Figure 31) while not colliding with any geofence or other trajectory or safe way out (in space and time). See Figure *29*, Figure *30* and Figure 33. Calculating this safe way out trajectory must be done as quickly as possible so that it can be sent out to the objects before they proceed to far on their previous trajectory. An overview of the algorithm used to compute safe way out trajectories is shown further below.
- For each MONR from each object
  - Correlate the object position with that objects trajectory and check if object is adhering to its trajectory within set limits, see Figure 32. These operations must also be done very quickly, with good margins within the 10 ms between each MONR. If the supervisor is unable to keep up with the rate of MONR it would end up in a situation where it is trying to catch up with an ever-growing backlog of

increasingly outdated MONRs to process. Not that this would quickly lead to abortion of the test scenario once the MONR timestamps are too far off.

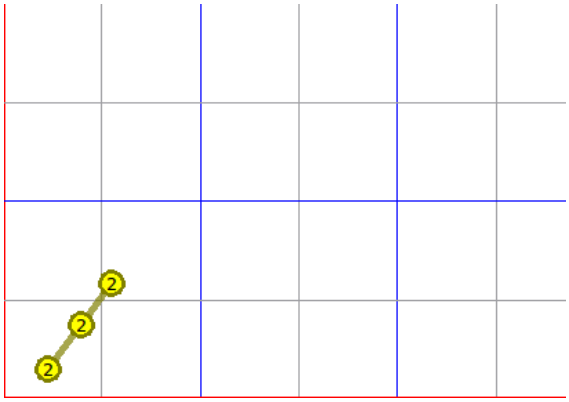


Figure 29. An example chunk of trajectory, 3 points long specifying a speed of 10m/s

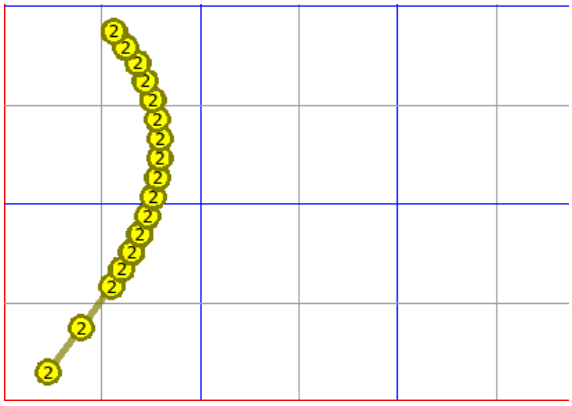


Figure 30. Same trajectory as Figure 29 with a safe way out appended to it.

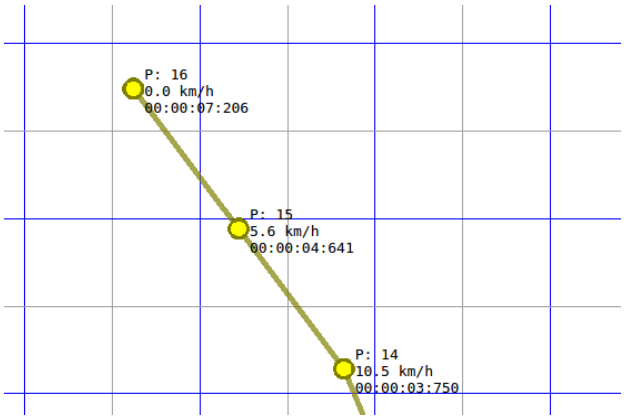


Figure 31. Zoomed in on the end of the trajectory from Figure 30 and with information printing turned on. Shows that trajectory ends in speed 0.

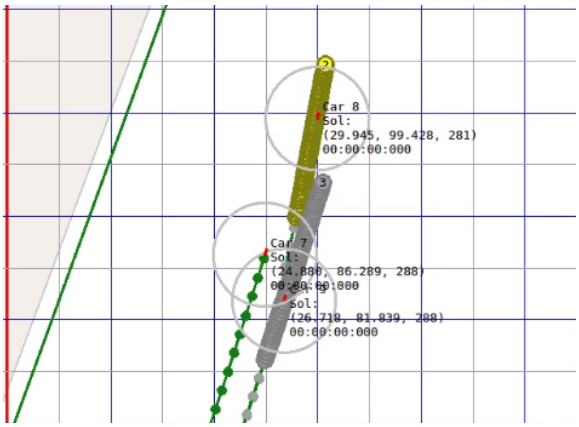


Figure 32. While a test is running, the supervisor GUI is displaying trajectory information and positions of objects.

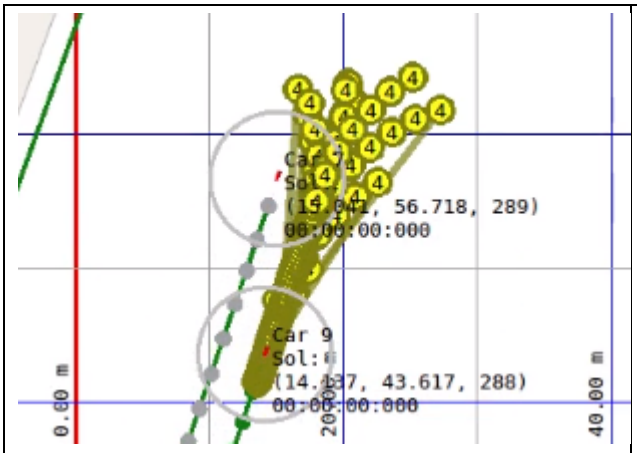


Figure 33. Displaying generated safe way out trajectories for a Chronos object.

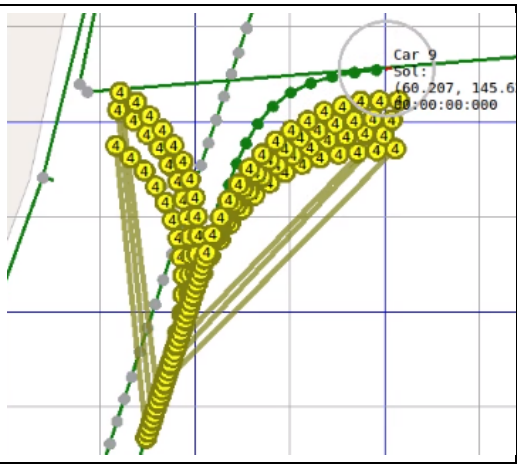


Figure 34. Object is avoiding a geofenced area.

Conditions that lead to abortion of a test scenario

In the current prototype implementation of the supervisor, it is possible to turn each of the checks for abort conditions on or off individually.

1. Chunks of trajectory sent from the simulator can be tested against static geofences. Checking this defends against incorrectly generated trajectories reaching an object.
2. Chunks of trajectory sent from the simulator can be tested against all other trajectories that the supervisor currently knows about. This check protects against incorrect trajectories that lead to a collision. If the test scenario specifies a crash, this information must somehow be communicated to the supervisor beforehand (This is future work).
3. The supervisor tries to append a safe way out to all trajectories it receives. If it is unable to generate a safe way out abort is triggered.
4. The supervisor correlates each received MONR message with the corresponding object's trajectory and if the MONR position or timestamp disagree with the scenario current time or trajectory, the scenario is aborted.

Combining some of these criteria should mean that no collisions of Chronos controlled objects are possible. For example, if criteria 2 and 4 are both activated there should be no collisions.

Still, this setup is not without problems. Imagine if an object within the test is starting to diverge from its given trajectory. If this leads to an abort and all Chronos controlled vehicles start driving along their safe way out. In this case, there is also an object out there that is clearly not following its designated path. Since we know that this object is no longer following trajectory and assuming that it is sending MONR messages that correctly reports its position, speed, heading, some other feature should kick in, perhaps a dynamic geofence surrounding this object.

Safe Way Out generation

The supervisor generates safe way out routes using a series of random attempts. For each attempt to generate a safe way out trajectory it is being checked against geofences and for intersections with already generated trajectories or trajectories received from the simulator. Any violating trajectory is discarded. If no safe way out trajectory can be generated within a fixed number of attempts, the supervisor gives up and aborts the ongoing test.

The capabilities of the vehicles impose a set of constraints on what a correct safe way out trajectory looks like. These constraints are listed below:

1. A maximum deceleration of $2m/s^2$.
2. A maximum centripetal acceleration of $5m/s^2$.
3. A maximum change in curvature of 0.25 per second.

The constraints, 1 – 3, above are a very simplified model of the vehicle capabilities. There are many factors that would influence a vehicle that are not considered here. For example, the road surface texture and whether the road is wet or not would also have significant impact on a vehicle's accelerating, breaking and turning capabilities. So, while adhering to the constraints above may be an acceptable starting point, it does need more thought in the future.

The procedure used to generate a random trajectory randomly tries to apply one of three different route generators. These route generators are given a random number *N* as input that specifies how many trajectory points should be generated using the randomly selected generator. The generators are listed below:

- Turn Left
- Turn Right
- Forward

The **Forward** generator first tries to straighten out the trajectory using the constrain on maximal change in curvature. It then generates a set of *N* points all along the same line.

The different **Turn** generators both apply the strongest turn possible based on constraints 2 and generate a turn consisting of *N* points.

Each generator slows down the speed for each new point added, while adhering to constraint 1.

These generators are applied for as long as speed remains above 0 and as long as it does not break any rules, such as crossing into a geofenced area or overlapping with another trajectory in space and time. If at one point adding, for example, a left turn fails, the procedure tries one of the other options.

The steps listed above are repeated a certain number of attempts until it runs to completion and returns a trajectory. If all attempts fail, the supervisor gives up and aborts the test.

Evaluation

We have developed a prototype of a supervisor that has some of the properties one may desire but on many fronts it falls short. Some of these shortcomings can be addressed with a more powerful dedicated computer and parallelization and optimization of the software.

There are also shortcomings that come from not knowing enough about of what the correct behavior of the autonomous vehicle, in any given situation, is. If we don't know anything about the autonomous vehicle, the supervisor must be very conservative and that goes against the wishes on what scenarios to test (such as autonomous VUT driving right next a Chronos controlled vehicle). As future work we should explore ways of having the vehicle under test provide more feedback into the system.

Having feedback from the VUT is also very important when it comes to evaluating a test. If the supervisor has access to object lists and positions of objects as seen by the VUT, we can easily spot if there is something the VUT is wrong about or is missing and potentially abort the test case.

In this work we laid some foundations when it comes to geofences. It would be good to expand on these concepts as future work. Maybe by adding dynamic geofences around VUT, that would, for example, expand rightwards if we get knowledge from the VUT that indicates it is going to do a right turn.

3.10 Demonstration of server capabilities in at least one of the test scenarios defined in WP1

The final Chronos 2 demonstration showed the Dynamic Trajectory-functionality and the safe-way-out concept.

The demo was carried out on the High Speed Area at AstaZero test track, Figure 35.

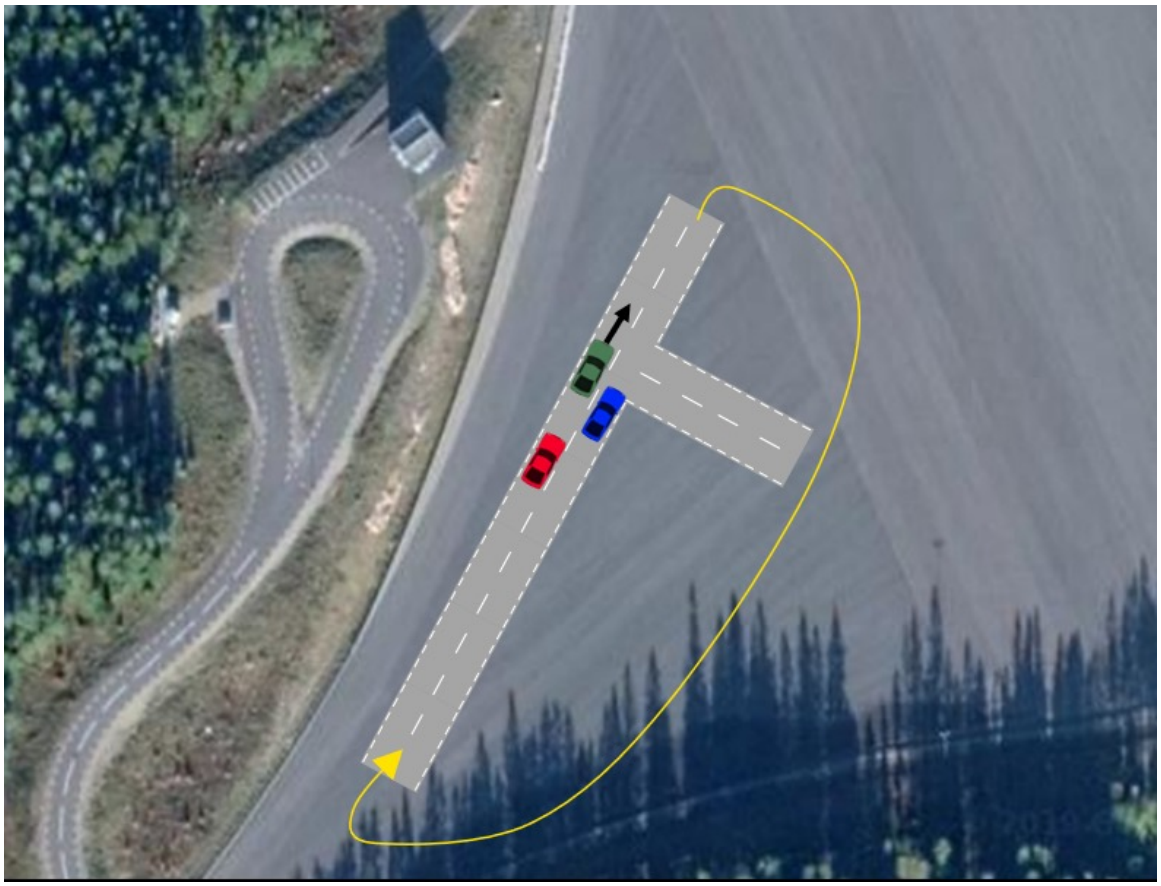
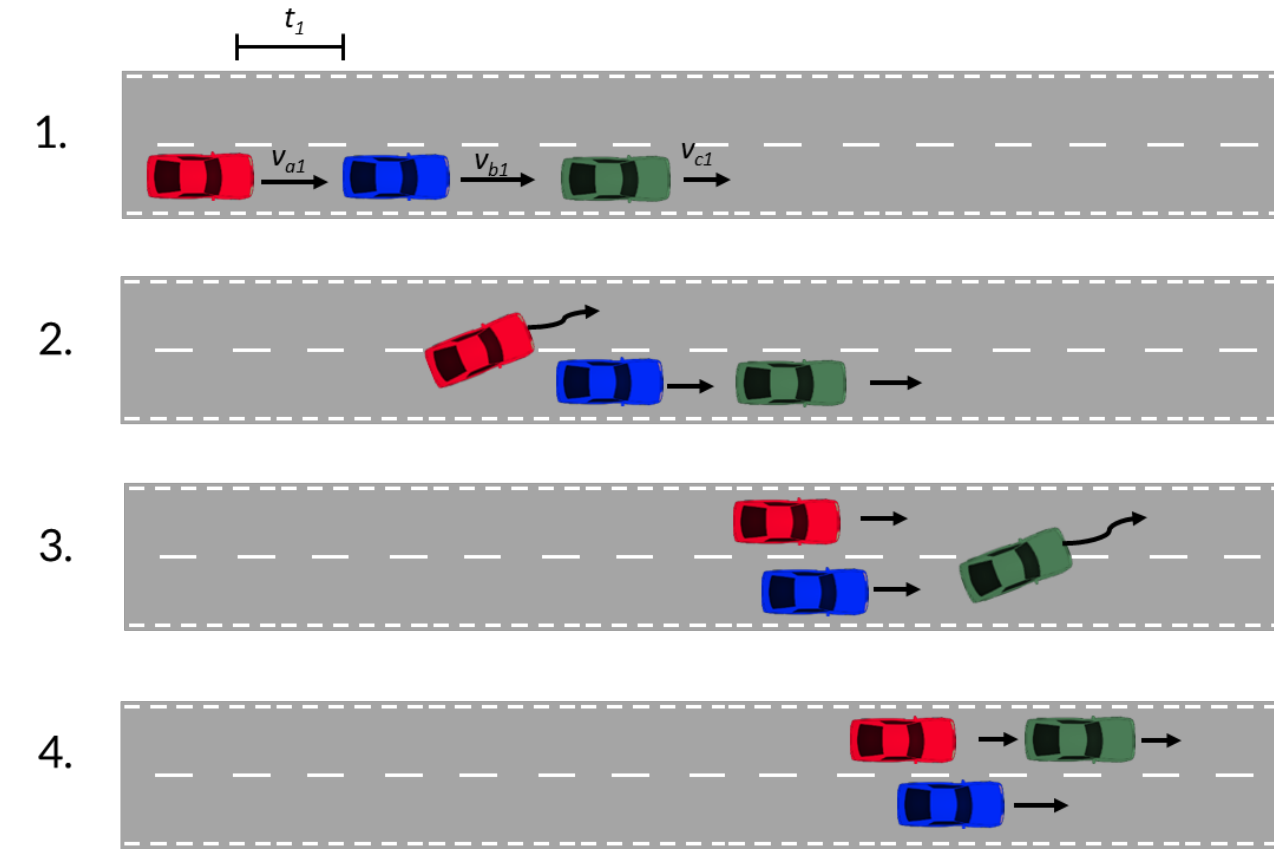


Figure 35. Final Chronos 2 demo scenario location.

Demo description

The demo scenario was chosen to be able to show Dynamic Trajectories, Safe-way-out, Virtual Injection and C-ITS- demo. This illustration shows the Dynamic Trajectory sequence:

1. Initialization. A is driving behind B at a distance/time t_1. and speed v_1.
   C is driving in front of B.
2. The red car (A) starts performing an overtake of the blue car (B).
   The blue car (B) increases speed and blocks A from overtaking.
3. Vehicle C cuts out in the left lane and blocks A.
4. Vehicles stops. Scenario ends. Return to start positions



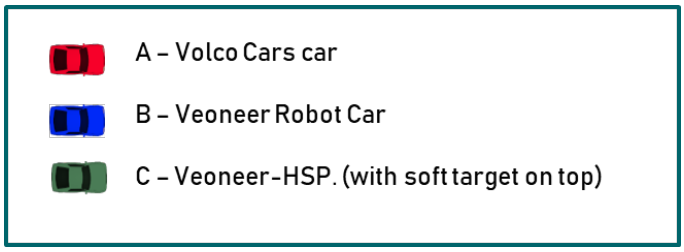


Figure 36. Graphic showing the final demo scenario.

WP4 AR Viewer for Test Planning/Analysis/Audience

WP4	AR viewer for test planning/analysis/audience
Leader	Inceptive
Other participants	Fengco (coordinate definition of software interface between the AR viewer and the simulator core).
Description of contents	AR Vehicle Testing Visualize the various test scenarios using Augmented Reality so that the real test objects can be viewed together with the virtual (injected) objects. Based on recorded test scenarios make it possible to pause the scenario on the proving ground, go in with the AR device, gain insights, and carry out defect analysis on the decision algorithms.
Method/approach (when relevant)	The work was split up into three steps according to the deliveries below.
Delivery	4.1 Definition of an interface between the simulation that handles virtual objects and AR viewer. This interface was defined between the simulation and the AR viewer. All the parameters were discussed, decided, and finally implemented in the server and in the AR viewer app. 4.2 AR Vehicle Test Scenario viewer first utilization. We developed an advanced AR solution for testing in AR (Augmented Reality). The engineer could see virtual test objects on top of the reality, in an Android smartphone. 18 versions of the app were created and tested together with the two servers needed to run the scenarios. The final version is available to the project team here (link requires login to Microsoft Teams). An example of the usage can be seen here:

Digital vehicles can be seen through an Android phone, where positions are sent from the server, through the API.

See the video here:
https://www.youtube.com/watch?v=_MAwKlz1S-Y

4.3 In car AR viewer

Due to unforeseen problems with testing and integrating the interface between the simulator and viewers, networks *etc.* on the track, the time was consumed to the two initial deliveries. A change request was filed to the Management Group of the project, asking for a redistribution of funds from other project partners to this task, but it was declined.

WP5 Driver in the loop using virtual reality

Using dynamic trajectories and virtual injection in this project was mainly targeted towards testing of vehicle systems and autonomous driving vehicles. However, both these new features of testing fit very well into the framework of testing human responses to driving. Dynamic trajectories are necessary components also when a human being is in the loop since the trajectories need to be adjusted for the behavior of the human driver. In driving simulators, the analogy is the scenarios that are event-driven and change according to the response of the driver-in-the-loop. The analogy to virtual injection for driver-in-the-loop testing is the simulation and presentation of cues to the test person.

This part of the project was devoted to implement and extend existing technology for presenting a virtual environment to a human test subject. This is achieved by a head-mounted display that displays a virtual world to the test person, while (s-)he is driving a real vehicle on a test track. Such an environment enables realistic motion feedback and setting, which potentially could be used in both research and product development as a complement to driving simulators. However, the validity of this setup and how reliable the outcome of studies conducted in this environment needs to be understood. This part of the project was set to investigate these questions.

Research questions and challenges

There are several challenges with developing a virtual reality-based driver-in-the-loop platform. The virtual reality display needs to track the user with high accuracy to present a realistic view of the virtual environment. Any error in tracking can lead to simulator sickness and disruption in the sense of presence. These problems can, in turn, affect the results of the tests as the test person changes their behavior. The available virtual reality systems are designed to be used in a static environment. To move them onto a moving vehicle, require alternative strategies for user tracking. There is also a need for high-accuracy vehicle tracking.

There is also the challenge of selecting the correct virtual reality display. Consideration needs to be made regarding display resolution, available field-of-view, and ergonomic comfort. Low display resolution and field-of-view can affect driving behavior, but the selection of a device with higher specifications can negatively change the ergonomic comfort.

Implementation

Two different approaches were implemented using standard off the shelf products. The Unity graphical engine was used to generate the graphics and the Oculus Rift head-mounted display as a display system. The moving car will distort the build-in tracking system of the Oculus, as the sensor sources are a camera system and an inertia

measurement unit (IMU). To account for the motion, one approach disconnected the accelerometer of the Oculus system.

This approach uses the very fast and accurate built-in tracking system and worked out very well for scenarios where the car is driving straight. This approach was used for the project demonstrations, where cross-traffic in an interaction was shown and interaction with the active systems of the car. A simulated car is showing up in the graphical system and is also fed to the cars' auto brake system.

To further understand the driver's behavior when using a driver-in-the-loop system a second approach was taken that could cope with steering and a turning car. This approach was based on a commercially available external tracking system based on cameras only, and with passive markers on the head-mounted display. Neither of the approaches visualized the drivers' hand actively. The steering wheel was however moved according to the steering input.

Outline

The driver interaction work was divided into three parts: a literature review and two user studies. The second user study also included an interview study. All three parts are reported in publications listed in section 7.2.

Literature Review

A literature review was performed, to give a summary of the benefits and challenges of VR in a driving simulator context. Using this knowledge, conference papers published within the simulator community in the last five years were reviewed with the aim of identifying driver studies which could be suitable for virtual reality. 734 papers were reviewed. A total of 203 of these papers concerned interactive driving simulator studies where a test subject was either driving or riding along in a simulated vehicle.

The reviewed studies were categorized depending on the following factors:

- Driving behavior
- Motion sickness
- Interactivity with hardware

Scenarios that would be feasible to transfer into virtual reality without special considerations are scenarios with low lateral motion, and few head-turns. The selected scenarios should also preferably not require the subject to interact with any hardware inside the cabin. Using these criteria, it was estimated that 41% of the reviewed user studies could be moved into a simulation environment that employs a head-mounted display as the primary display.

Simulator Study

A user study was performed in the VTI simulator 3 (see Figure 37) to highlight the differences in driving behavior between projector-based graphics and head-mounted

displays. The study involved five specific driving maneuvers suspected of affecting driving behavior differently depending on the choice of display technology.

Figure 37 The SAAB 9-3 cabin inside the simulator dome.

Results

The results show minor changes in lateral in driver behavior when comparing projectors and a head-mounted display. The most noticeable difference in favor of projectors were seen when the display resolution is critical to the driving task. The speed was perceived as lower in the head-mounted display, which probably relates to the lower available field-of-view.

The choice of display type did not affect simulator sickness, nor the realism rated by the subjects. However, test persons experiencing simulator sickness symptoms connected to the visual impression decreased their head motions.

Test Track Study

To generalize the results from the simulator study, the same types of maneuvers were replicated on a test track. The goal was to investigate how a set of driving maneuvers translates between real driving and the driver- and vehicle-in-the-loop platform. The tests were performed at a test track using an actual vehicle while the drivers wear a head-mounted display.

Results

The results from the test track are very similar to the results from the simulator study, with one exception, the perceived speed. The subject tended to drive faster with the head-mounted display in the simulator compared to the instructed speed. On the test track, the result was opposite as the subjects drove slower with the head-mounted display. A possible explanation may be that the drivers are more hesitant to drive a real

vehicle with an opaque head-mounted display, thus unintentionally reducing their speed.

Interviews

Interviews were performed with three engineers who had experienced the DVIL platform. The questions focused on the technology readiness level of the platform as well as possible future improvements.

Results

All interviewed engineers agreed that the platform was ready to use for a selected set of scenarios. They agreed that the main point was to improve the ease-of-use for the platform. Today the platform is pieced together from many different subsystems that need to be initialized manually and in a certain order. It would be highly beneficial for the usability of the platform to simplify this startup process.

They were asked if high-resolution displays were needed since the visual acuity showed the most significant difference in both the test track study as well in the simulator study. Nevertheless, all of them deemed display resolution as *"nice to have"* but not as a high priority for future improvement.

Publications

The publications listed in section 7.2 were produced through the course of the project within this work package. The first three document the findings described above, while the last summarizes previous work and the present one in a thesis.

WP6 Test System Communication

WP6	Test system communication
Leader	AstaZero
Other participants	Ericsson, VCC (set requirements for closed loop control of targets)
Description of contents	Based on the findings from Chronos 1, the existing LTE network at AstaZero will be optimized for the specific traffic models developed in Chronos 1.
Method/approach (when relevant)	6.1 Updating the communication subsystem to reduce the latency/jitter in the communication with focus on the heartbeat/monitoring signaling, downloading of drive files and uploading of log files. 6.2 Verify the improvements given by the optimizations made in task 1.
Delivery	Communication subsystem optimized for the specific traffic models used in the AstaZero test platform.

Communication subsystem optimized for the specific traffic models used in the AstaZero test platform.

Current network setup between the Chronos server and the LTE-network connected Test Objects does not differentiate between low-prioritized communication such as log file uploading or high-priority messages with high latency-sensitivity such as Heartbeat and Monitor messages, Figure 38.

Figure 38. LTE network before implementing the improved traffic model

From the findings in Chronos 1, a number of possible improvements of the AstaZero LTE-network were proposed, Figure 39.

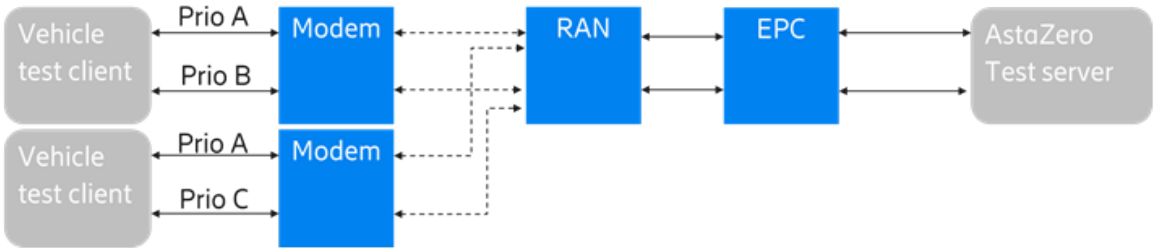


Figure 39 LTE-network after the changes implemented.

- **Priority A:** Small messages (~50-100 bytes of information in a UDP-message) which are latency-sensitive.
  If a message is too old, then it **is no longer relevant and shall be discarded.** To be able to reduce the number of delayed packets of the "priority A"-messages, which are typically Heartbeat and Monitor messages, one solution is to prioritize or discard buffered and delayed packets using specific handling in the network. This handling can be achieved by assigning a dedicated bearer with **high priority QCI and RLC in non ack mode** in the radio. To reduce uplink latency, **1 ms uplink pre-scheduling** can be enabled.

- **Priority B:** Medium sized messages (10k-100k over TCP) that are latency sensitive implying that they must not be delayed, but also not discarded.
  The solution is to assign a dedicated bearer with **high priority QCI with RLC in ack mode** which is default for this LTE-network.

- **Priority C:** Large files, (typically logfiles/video files 1-1000 Mb) which are low priority messages.
  Log file upload should be handled on a bearer with **low priority QCI with RLC in ack mode.**

The proposed changes above are not implemented. The main reason is that a 5G network is now being installed at AstaZero and therefore optimizing the old network is no longer a priority. Future investigations and projects should include analysis of the 5G performance when it comes to this traffic model.

WP7 C-ITS Test Capability

WP7	C-ITS test capability
Leader	Ericsson
Other participants	VCC (Use cases, analysis of network requirements, implementation of dynamic map), AstaZero
Description of contents	Develop the C-ITS test capabilities at AstaZero by setting-up cloud based 5G cellular communication system and a distributed cloud network to support the C-ITS use cases. Included in this C-ITS test system is also to connect AstaZero infrastructure to the cloud system, *e.g.*, traffic lights. Develop demonstrator(s) of use cases and associated logic, to be supported by the C-ITS test system. The system will involve a cloud sensor-fusion based dynamic map from which some C-ITS applications will be developed and tested, *e.g.* - Emergency Brake Warning - Intersection control (cloud-based traffic lights) - Vulnerable Road User Warning
Method/approach (when relevant)	1. Analyses of C-ITS use case development done in other external forums (5GAA, 5GCAR, *etc.*) and standardization bodies (ETSI TC ITS, SAE, IEEE, *etc.*) 2. Analyze the needs for the C-ITS use case and a. set-up the communication system to handle the requirements using, *e.g.*, network slicing and edge computing b. set-up the onboard system for the C-ITS use case
Delivery	7.1 Specify C-ITS Use Cases 7.2 Specify C-ITS test system architecture 7.3 Established an end-to-end C-ITS test system at AstaZero using existing 5G PoC (Proof of Concept) cellular network and using C-ITS infrastructure components such as cloud application(s), connected vehicles, and connected transport system(s). 7.4 Develop demonstrators C-ITS use cases 7.5 Perform tests and demonstrations of relevant C-ITS use cases in combination with other communication services supported in today's vehicles, *e.g.*, internet access, OEM services, *etc.*

WP7.1 Specify C-ITS Use Cases

Two Use Cases were specified and demonstrated. The specified Use Cases were Emergency Brake Warning, Figure 40, and Intersection Control, Figure 41.

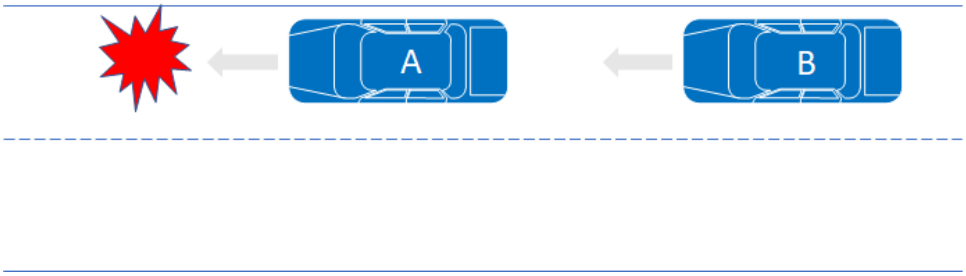


Figure 40. Emergency Brake Warning Use Case.

1. Vehicle A and Vehicle B continuously send their position to the C-ITS Server to make the C-ITS system aware of the traffic situation.
2. Vehicle A detects an obstacle on the road and activates its brakes which also triggers and sends an Emergency Brake Warning to the C-ITS Server.
3. The C-ITS Server receives the Emergency Brake Warning from Vehicle A and evaluates the traffic situation and sends an Emergency Brake Warning to relevant vehicles behind Vehicle A, i.e. Vehicle B in this case.
4. Vehicle B receives the Emergency Brake Warning from the C-ITS Server and activates its brakes.

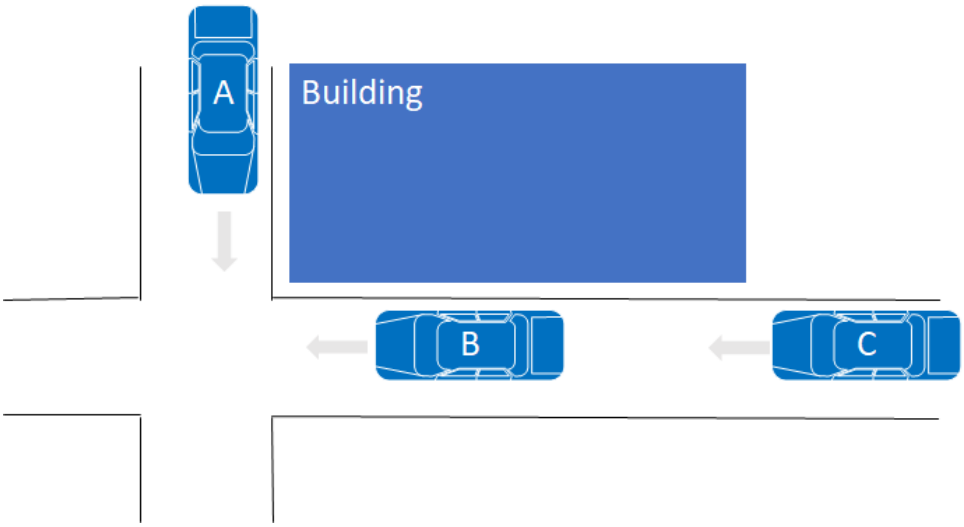


Figure 41. Intersection Control Use Case.

1. Vehicle A, Vehicle B and Vehicle C continually send their positions to the C-ITS Server to make the C-ITS system aware of the traffic situation.
2. The C-ITS Server detects a potential critical situation based on vehicle A and Vehicle Bs positions and starts sending Vehicle As position to Vehicle B.
3. Vehicle B analyzes the information about Vehicle A from the C-ITS server and activates its brakes due to an imminent crash with vehicle A. Vehicle B in turn also sends an Emergency Brake Warning to the C-ITS Server.
4. C-ITS Server receives the Emergency Brake Warning from Vehicle B and evaluates the traffic situation and sends an Emergency Brake Warnings to relevant vehicles i.e. Vehicle C.
5. Vehicle C receives the Emergency Brake Warning from the C-ITS Server and activates its brakes.

WP7.2 Specify C-ITS test system architecture

The C-ITS test system architecture is shown in Figure 42 and an interfaces summary is given in Table 6.

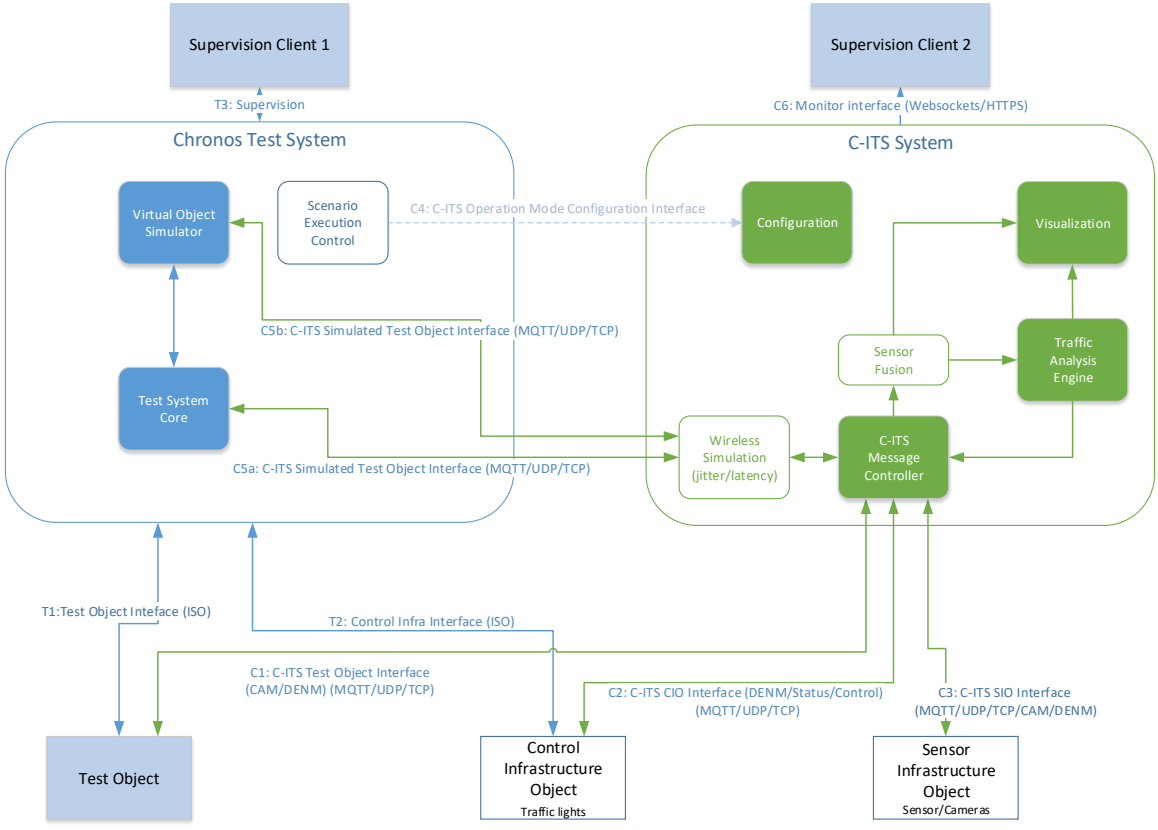


Figure 42. C-ITS Architecture.

Table 6. Interface summary.

C-ITS System	Protocol	Description
C1: C-ITS Test Object Interface	UDP+TCP, CAM + DENM, Specifications: ETSI EN 302 637-2 V1.3.1 ETSI EN 302 637-3 V1.2.1 ETSI TS 102 894-2 V1.2.1	C-ITS system reads the CAM information and sends CAM and DENM information to the Test Object.
C2: C-ITS Control Infrastructure Object Interface	UDP+TCP, DENM, Specifications: ETSI EN 302 637-2 V1.3.1 ETSI EN 302 637-3 V1.2.1 ETSI TS 102 894-2 V1.2.1	C-ITS system reads the DENM/status information and controls the infrastructure object. Control is only possible when the Test System is not controlling it. (Not implemented)
C3: C-ITS Sensor Infrastructure Object Interface	UDP+TCP, DENM, Specifications: ETSI EN 302 637-2 V1.3.1 ETSI EN 302 637-3 V1.2.1 ETSI TS 102 894-2 V1.2.1 Others TBD	C-ITS system reads the CAM/DENM information (Not implemented)
C4: Operation Mode Configuration Interface		C-ITS system receives a configuration specifying operation mode. (Not implemented. Only manual configuration possible)
C5: C-ITS Simulated Test Object Interface	See C1	See C1. CAM and DENM messages from the simulator and from test objects without direct C-ITS communication capabilities use this interface.
C6: C-ITS Monitor Interface	HTTPS/Websockets	Provides an overview of the C-ITS objects and traffic warnings.

WP7.3 Establish an end-to-end C-ITS test system at AstaZero

A C-ITS test system was established at AstaZero as specified in WP7.2. In the final deployment some parts were left out, these parts were the Wireless Simulator logic and the Configuration interface. The Wireless Simulator logic and Configuration Interface are not part of the basic C-ITS test system and focus was to deploy the parts needed to perform tests and demonstrations. Neither were the Control and Sensor Infrastructure Object parts implemented. The Control and Sensor Infrastructure Object parts were not needed for the specified use cases and by that not implemented.
The two C-ITS use cases (Emergency Brake Warning and Intersection Control) as specified in WP7.1 was deployed as cloud applications in the C-ITS edge server. One Volvo Car (same as used in WP8 Virtual Injection) and a Volvo Truck were equipped with C-ITS clients and used for tests and demonstrations. The C-ITS clients send and receive CAM and DENM to/from the C-ITS Server.

Interfaces to the C-ITS edge cloud server from the Simulator (see WP2) and to the CHRONOS server (see WP3) were also established. The simulator sends CAM for the simulated objects to the C-ITS Server and the CHRONOS server sends CAM for the physical objects not able to carry any own C-ITS client. These objects are RC Cars, Low Rider, *etc.*

The 4G/5G cellular network at AstaZero was used for the C-ITS related communication between the vehicles equipped with C-ITS clients and the C-ITS Server. For the demonstrations only 4G network was used due to the limited availability of 5G modems.

WP7.4 Develop demonstrators C-ITS use cases

C-ITS demonstrators were developed based on those use cases specified in WP7.1.

Totally three demonstrations were performed, two for the Emergency Brake Warning use case and one for the Intersection Control use case.

The Emergency Brake Warning use case was demonstrated with two different set-ups of Vehicle A (see WP7.1 the leading vehicle), in one demo vehicle A was a simulated virtual vehicle and in the second demo vehicle A was an RC Car. In both cases the vehicle B was a real car equipped with a standalone C-ITS Client not connected to any vehicle system. The standalone C-ITS Client was a Raspberry PI with a touch screen, interfaced to a GPS receiver and a cellular modem. The Raspberry PI was also controlling a flashing warning light which indicated when an Emergency Brake Warning DENM was received i.e. to make the DENM reception visible to the demo spectators.

The Intersection Control use case was demonstrated with the following set-up (see WP 7.1 for vehicle denominations):

- Vehicle A was a simulated virtual vehicle. The simulator sent the vehicle information to the C-ITS Server.
- Vehicle B was a Volvo Car with virtual injection, see WP8. For C-ITS client implementation, see Figure 42.
- Vehicle C was a Volvo Truck with a vehicle integrated Telematics Control Unit (TCU). The TCU was updated with cellular modem to handle the C-ITS communication.

The Intersection Control demo was except for Vehicle C based on the same principle as the Virtual Injection demo in WP8. The difference between this demo and the virtual injection demo was that in the Virtual Injection demo the simulated vehicle information was received from the Chronos Test Server and was handled as sensor information with the sensor view blocked by the building. In the C-ITS demo, the simulated vehicle information was received by Vehicle B via the C-ITS System and by that not blocked by any building.

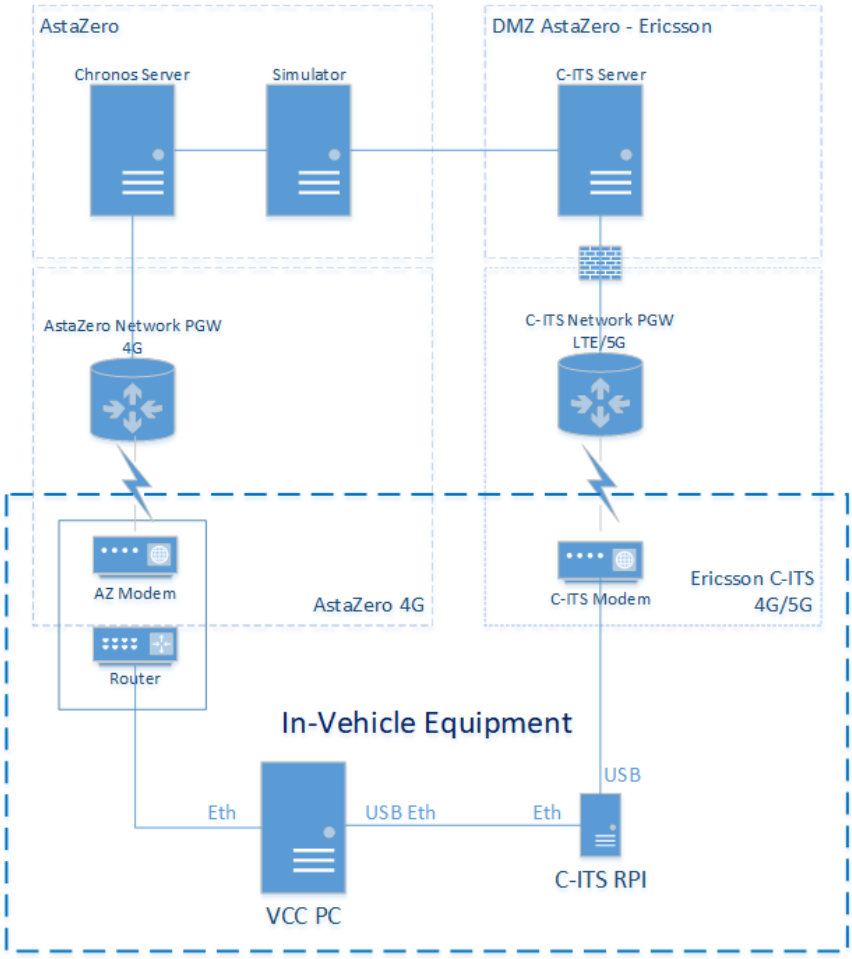


Figure 43. C-ITS Client implementation in the VCC Vehicle B.

In Figure 43 the communication paths related to the VCC vehicle are shown and also how the C-ITS client is implemented in the vehicle. For the C-ITS demo the C-ITS RPI and C-ITS Modem was added, compared to what was in the vehicle for the Virtual Injection demo, see WP8. The C-ITS RPI converted the information from/to the VCC PC to CAM and DENM messages which are used on the C-ITS system side.

WP7.5 Perform tests and demonstrations of C-ITS Use Cases

Demonstrations were performed as developed in WP7.4, e.g., the Emergency Brake Warning and Intersection Section Control. The demonstrations worked out as planned which can be seen from videos:

- Emergency Brake Warning with RC-car, https://www.youtube.com/watch?v=0Q-F_bpAHqE
- Intersection Control, https://www.youtube.com/watch?v=KKHZYGi1DRA

WP8 Virtual injection - VUT hardware

WP8	Virtual injection - VUT hardware
Leader (role and responsibility)	VCC (Development of interface to inject virtual objects into VCC vehicle ECU)
Other participants (roles and responsibilities)	AB Volvo (Interface to inject virtual objects into truck's ECU), Autoliv (Support on how to test Virtual Injection using Autoliv's Sensors and/or ECUs as well as providing input to the definition of interfaces)
Description of contents	This work package has developed a common hardware interface to inject virtual targets in VUTs from different OEMs and set requirements for the V2V simulation in WP2.
Method/approach (when relevant)	The virtual targets will be injected as object tracks in the vehicle ECU. As a first step we will investigate which inputs from the virtual target simulator are required to inject the object tracks in the ECU of each OEM and will define a common interface architecture. Different hardware interfaces to the ECU will then be developed by each OEM, based on the common interface definition. Volvo Cars already has prototyped injection of virtual targets in its HIL environment. An analysis of the current V2V hardware solutions will be carried out in order to set requirements for the injection of virtual V2V signals from the simulator (WP2).
Delivery	8.1 Document describing the common interface for object injection 8.2 Hardware and software interfaces to ECUs 8.3 Demonstration of virtual target injection in VUT from different OEMs (Volvo Cars and AB Volvo) 8.4 Report on requirements for V2V simulation.

WP8.1 Document describing the common interface for object injection

The Chronos 2 architecture for virtual object injection is described in Figure 44. The traffic simulator runs the test scenario and generates virtual objects. Virtual object information is passed through the Chronos 2 server to the virtual object injection interface in the vehicle via a 4G link.

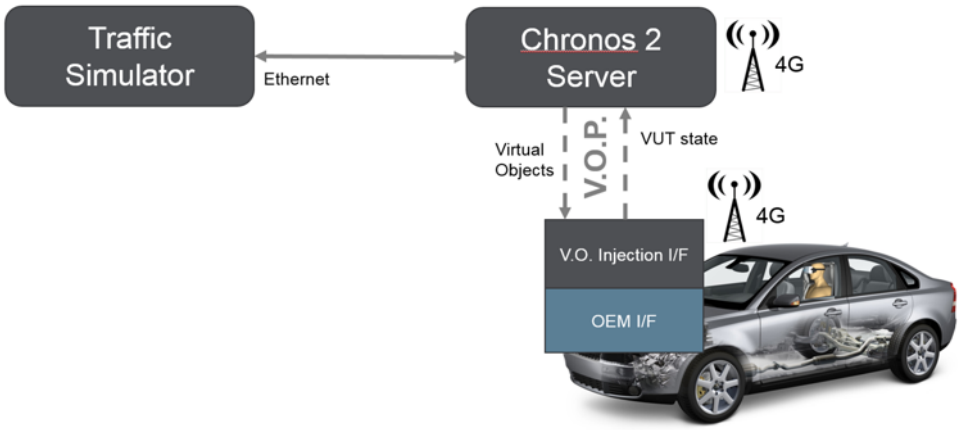


Figure 44. Virtual injection in the Chronos 2 test system. The virtual object protocol (VOP) describes the communication between the Virtual object injection interface and the Chronos 2 server.

The virtual object injection interfaces are composed of two parts: one OEM-specific interface which manages the injection of virtual objects to the vehicle's active safety ECU, and one test system interface which manages the communication with the Chronos 2 server.

This setup enables virtual injection to different OEM's vehicles without disclosing sensitive information, *e.g.*, about the specific signals needed to activate a certain function in the active safety ECU. All sensitive communication with the active safety ECU is managed by the OEM-specific injection interface, while the test system interface can be standardized and shared.

In Chronos 2 the test system interface consists of a 4G modem, connected to the AstaZero private network, and a virtual injection protocol (VOP) which describes the communication between the Chronos 2 server and OEM-specific interface.

The VOP is detailed in Appendix C. The OEM-specific interface is described in the next section.

WP8.2 Hardware and software interfaces to ECU

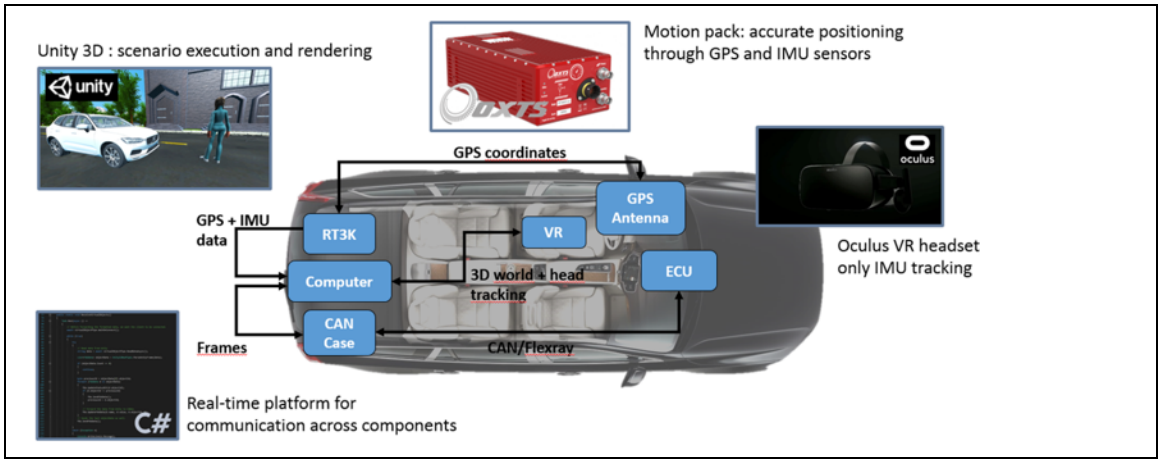


Figure 45. Architecture of Volvo Cars platform for virtual target injection.

The OEM-specific injection interface was developed by Volvo Cars, the setup is described in Figure 45.

The virtual injection platform is composed of:

- Volvo XC90 test vehicle equipped with instrumented active safety ECU.
- Desktop PC installed in the trunk of the test vehicle, connected with a 4G modem to AstaZero private cellular network.
- Real-time application running on PC and managing all communication between the platform components and with the Chronos 2 server.
- Unity 3D application installed on PC, providing a virtual environment for visualization and injection.
- Vector CAN + Flexray interface connected to the PC through USB
- Oxford Technical Solutions RT3000 motion GNSS+IMU motion pack, connected to PC through Ethernet.
- Oculus Rift virtual reality (VR) headset.

Unity 3D environment

The core of the injection solution is a 3D environment created with the Unity 3D gaming engine. The environment contains the 3D model of the vehicle under test (VUT), static environment, 3D models for virtual objects, and a sensor model.

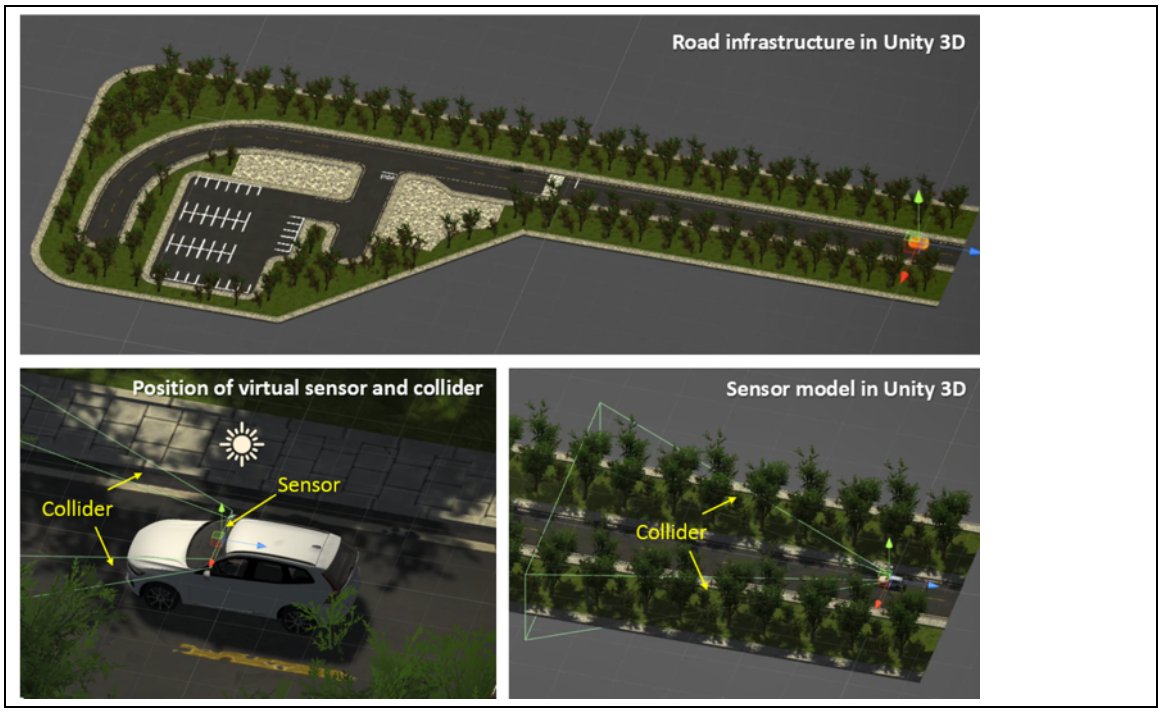


Figure 46. Components of the virtual environment in Unity 3D.

The static environment contains 3D models of the road infrastructure, buildings and other static content, such as trees, sky and background.

The 3D model of the VUT is a CAD model of a Volvo XC90, complete with exterior and interior details. Most components of the model, such as opening of the doors, visualization of information in the instrument cluster, and steering wheel movement, can be controlled via Unity 3D scripts. Virtual targets are represented by green XC90 CAD models, including only exterior meshes.

Both VUT and virtual target models can be controlled by providing x, y, z, and heading information through a Unity 3D plugin. In the Chronos 2 project, the virtual targets are controlled by the simulator through the VOP protocol over the Chronos 2 server.

The sensor model is based on the production sensor installed on the physical test vehicle, a combined radar and camera system positioned behind the windshield (Figure 46). The model is composed of a trapezoidal collider mimicking the field of view and range of the physical sensor, and a ray-casting function to model sensor occlusion. If the 3D model of a virtual target enters the field of view of the sensor (i.e. collides with the trapezoidal collider) a ray is cast from the sensor to the target to determine if the line of sight is occluded or not. In case of free line of sight, the target is considered as detected by the sensor, and the information about its position is sent to the real time platform to be injected into the active safety ECU.

The real-time platform

A real-time application written in C# was developed by Volvo Cars to manage all communication between the different hardware components of the virtual injection platform. This application also manages the communication between the Chronos 2 server and the injection platform.

The communication with the Chronos 2 server is based on the VOP protocol [Appendix C], which contains the states of the vehicle under test and other test objects, including virtual targets generated by the simulator.

The real time application also manages all coordinate conversions. All coordinates in the Chronos 2 VOP protocol are passed in simulation coordinates defined by the traffic simulator, the position of the VUT is recorded in GPS coordinates by the OTSx 3000, while the injection interface requires Unity 3D coordinates. The conversion is done by fixing an origin in GPS coordinates together with a North-East heading for each test scenario. Based on this fixed reference system, GPS coordinates are translated into local planar coordinates and rotated to match both the simulator and Unity 3D coordinates.

The motion of the VUT in the Unity 3D project is achieved by passing the converted GPS coordinates and heading to the 3D vehicle model via a C# script. Similarly, the position of virtual objects in is controlled by a C# plugin attached to the 3D asset in Unity 3D. This plugin which receives the x, y, z, heading coordinates from the traffic simulator via the Chronos 2 server.

Virtual object injection solution

The virtual injection solution is based on an instrumented version of the production active safety ECU installed in the Volvo Cars XC90 test vehicle. In its normal operation mode, the active safety ECU uses a camera and a radar to detect road infrastructure and traffic objects. The fused information from the two sensors is passed the active safety software, which is responsible for the activation of active safety functions (Figure 47).

In order to inject virtual targets, the instrumented ECU must enter “injection mode”, this is achieved by passing the right signal via the CAN interface. When in injection mode, both sensors are disabled, and virtual targets are injected as fused objects directly into the active safety software module.

The structure of the injected fused object depends on the specific interface between the sensor fusion and active safety software modules on the ECU. In our case, fused object properties are injected as CAN signals to the instrumented ECU interface. For each detected traffic virtual object, these signals include:

- Object ID
- Lateral and longitudinal position
- Lateral and longitudinal velocity
- Object type

- Other signals needed to assess threat severity and detection confidence
- Virtual road infrastructure, such as lane markings and road edges, can also be injected as fused objects in a similar manner.

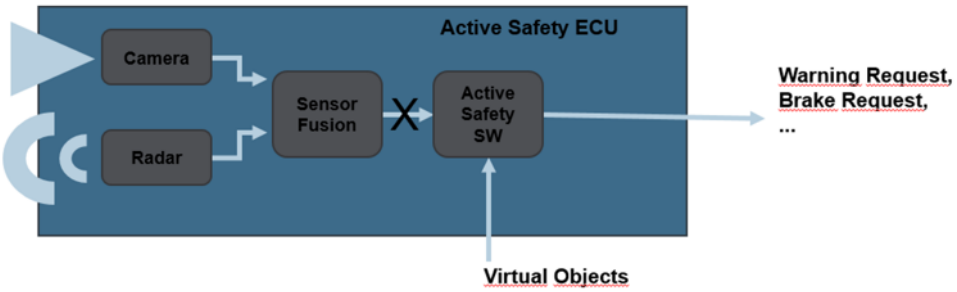


Figure 47. Injection of virtual targets into Volvo Cars active safety ECU. When the ECU is in "injection mode", signals from the sensor fusion module are bypassed and fused virtual objects are directly injected to the active safety software.

This setup allows to test real production software on the active safety ECU and the effect of real vehicle dynamics on the function performance. However, the performance of the perception system, including sensor fusion, cannot be tested as these are bypassed when in "injection mode". For the same reason, this setup does not allow scenarios with a mix of real and virtual targets.

Virtual reality visualization

In order to allow the test driver to visualize the virtual targets, and to test driver / function interaction in simulated scenarios, WP 8 also developed a virtual reality setup for the driver.

The setup is made of an Oculus Rift virtual reality headset connected the main PC, a Vector Flexray interface, and a C# application managing the headset tracking (see Figure 45).

The headset view is positioned in the driver seat of the vehicle under test in the Unity 3D scenario used for virtual target injection. This makes it possible for a test driver wearing the headset to drive the test vehicle in the real world and at the same time visualize the simulated Unity 3D scenario.

The tracking of the headset position and orientation in a moving vehicle is not trivial, as it is normally designed to operate in a stationary environment. The Oculus Rift tracking relies on fused information from IR cameras and inertial measurement units (IMU) to achieve a fast and stable performance. However, the algorithm fusing camera and IMU data assumes a stationary environment and is not suited for a moving vehicle subjected to lateral and longitudinal acceleration. To overcome this limitation, a solution was found by disabling the camera tracking and only using information from IMUs. The resulting system allows for a good tracking of the headset orientation, while limiting

translational movement in the x, y, z axis. This limitation was found to be acceptable for the test scenarios considered in this project.

In order to maximize immersion and provide feedback to the driver, the Flexray interface can be used to visualize information from the car internal network. This is done by connecting specific signals from the car network to Unity 3D assets via C# plugins. For the scenarios considered in this project, feedback information included:

Steering wheel angle: the steering wheel in Unity 3D moves according to the measured position of the real steering wheel

Speed and gear: the virtual instrument cluster reports real speed and gear information

Forward collision warning: when the forward collision warning function is activated, an icon flashes on the instrument cluster

The setup for an automatic emergency brake scenario is shown in Figure 48.

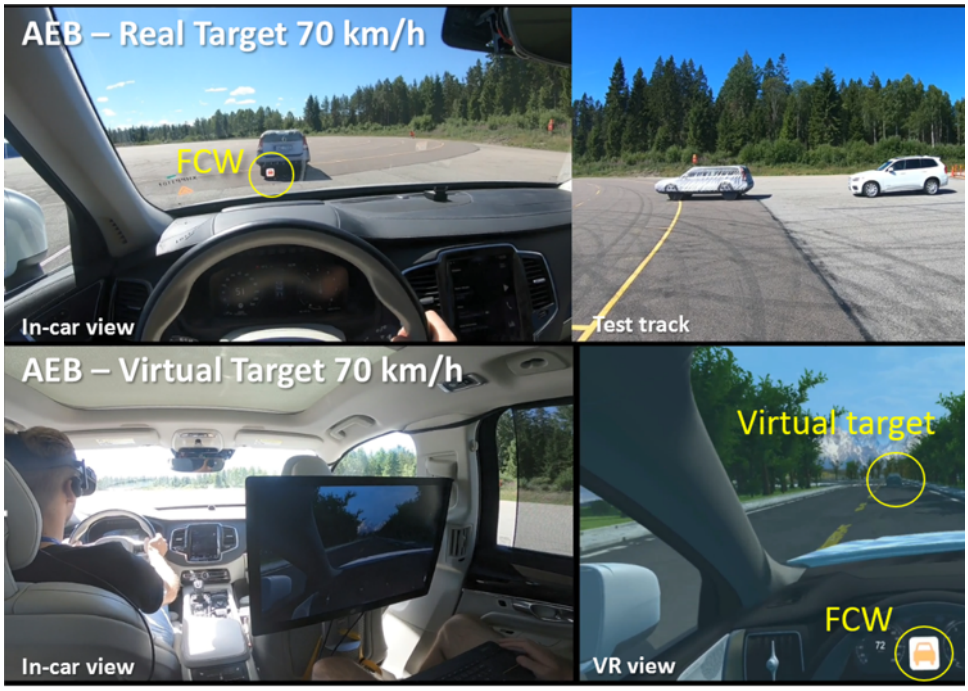


Figure 48. Comparison of real (top) and virtual (bottom) testing of the same automatic emergency braking (AEB) scenario. The virtual target correctly triggers the forward collision warning (FCW) and then activates AEB.

WP8.3 Demonstration of virtual target injection in VUT from different OEMs

During the project, the following active safety and driver assistance functions were successfully tested with the virtual injection + virtual reality headset setup:

- Forward collision warning (FCW)
- Automatic emergency brake (AEB)
- Oncoming collision mitigation
- Lane keeping assist (LKA)

- Pilot assist (PA)

An FCW + AEB scenario was selected for the project demo, which was performed at the final project meeting 12 March 2020. The demonstrated scenario is described in Figure 49.

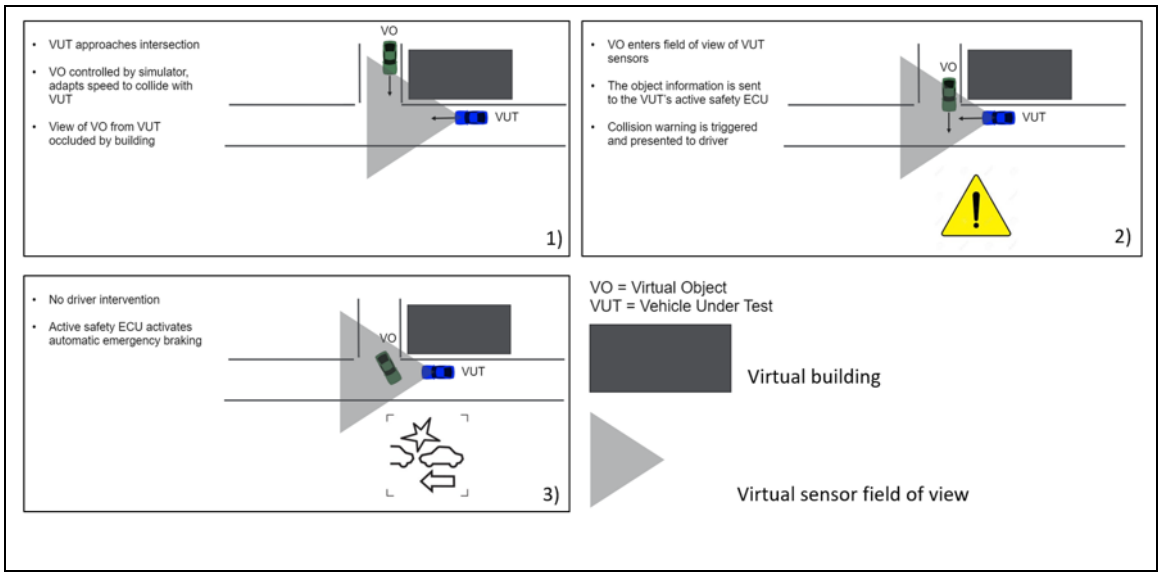


Figure 49. Storyboard of virtual injection FCW+AEB scenario demonstrated at Chronos 2 final meeting 12 March 2020.

Only the Volvo Cars test vehicle was used in the demonstration of virtual target injection. Due to lack of resources, AB Volvo had to withdraw from the WP8 work and focus remaining resources on C-ITS and scenario language work.

8.4 Report on requirements for V2V simulation

WP8 and WP7 investigated the possibility of simulating a vehicle-to-vehicle (V2V) communication service to test C-ITS applications. The details of this work are reported in the WP7 Section of this document.

The solution adopted is based on the virtual target injection platform described in Sections 8.1 and 8.2, with the only difference that the simulated targets are not sent directly via the Chronos 2 server but via a C-ITS Client connected by USB to the PC in the vehicle.

As a consequence, the requirements for V2V simulation are the same as those described for virtual injection in Appendix C.

7. Dissemination and publications

Dissemination

How are the project results planned to be used and disseminated?	Mark with X	Comment
Increase knowledge in the field	X	
Be passed on to other advanced technological development projects	X	
Be passed on to product development projects		
Introduced on the market		
Used in investigations / regulatory / licensing / political decisions	X	

The project has a webpage at https://www.astazero.com/chronos-part-2/ where results and movies from demonstrations are shared. Both the Coordinator and the partners have presented the project to internal and external audiences. For example, Euro NCAP has shown great interest and the project has been presented to some of its working groups.

Publications

[1] Björn Blissing and Fredrik Bruzelius. “Exploring the suitability of virtual reality for driving simulation”. In: Proceedings of the Driving Simulation Conference 2018. September. Antibes, France: Driving Simulation Association, 2018, pp. 163–166.

[2] Björn Blissing, Fredrik Bruzelius, and Olle Eriksson. “The effects on driving behavior when using a head-mounted display in a dynamic driving simulator”. 2020. Submitted for possible journal publication.

[3] Björn Blissing, Fredrik Bruzelius, Siddhant Gupta, and Francesco Costagliola. “Validation of driver behavior in the Driver and Vehicle in the Loop platform”. 2020. Submitted for possible conference proceeding publication.

[4] Björn Blissing “Driving in Virtual Reality -- Requirements for automotive research and development”. Department of Management and Engineering Linköping University. Linköping Studies in Science and Technology Dissertations No. 2085.

For details on the Volvo Cars student projects, see Appendix B.

8. Conclusions and future research

The Chronos 2 project has enabled technical progress which will change the way vehicles are tested in the future. What you see in the virtual environment during a test is taking place in the real world, with a few milliseconds delay. All inputs controlling the various robots originate in the virtual environment, weaving virtual and real world into one. This means that the testing is moving from the physical test track to the virtual, simulated track, and back again, with the virtual system being in charge of controlling the tests.

A cloud-based C-ITS test system using 4G/5G cellular network was set up, showing that different C-ITS applications can be tested and evaluated. Close integration between the AstaZero test system infrastructure and the C-ITS system was also shown making it possible to include physical, proxied (*e.g.,* radio-controlled cars) and virtual objects in the test and evaluation of C-ITS applications.

The concept of dynamic trajectories developed in Chronos 2 is now also being implemented in ISO-22133. The Chronos 2 project is hereby leading and giving important feedback to the development for this ISO standard.

Dynamic trajectories serve several purposes, for example:

6. The supervisor utilizes dynamic trajectories for calculating safe-way-out trajectories and can stop the running test scenario, in case of an emergency
7. It adds the capability for, and testing of, a subject vehicle with less constraints assuming that the vehicle acts more unpredictably compared to a predefined static path (*e.g.,* autonomous vehicles)
8. It is possible to avoid dangerous test scenarios where humans are at risk of injury
9. Complex test scenarios where a simulator generates the trajectories can be carried out
10. Both basic and more complex test scenarios can be made increasingly efficient with regard to time and cost

The project demonstrated how active safety functions can be tested with virtual target injection and developed a standard interface setup to connect vehicles from different manufacturers to the same injection platform. This system enables safe, repeatable, and efficient testing of active safety functions in dangerous scenarios that may be impossible to recreate in a traditional test-track setup. The project attracted the attention of Euro NCAP which is now considering to include virtual injection into its future testing protocol. The Chronos 2 consortium is planning to continue research in this area with close interaction with Euro NCAP.

Furthermore, the project has shown that it is feasible to use head-mounted displays to include the driver into the virtual injection testing. Hence, driver response can be tested along with the technical aspects. The dynamic trajectories are then a necessary

component to account for the non-deterministic behavior of a driver. The ecological validity of this setup's test environment was quantified in some known problematic situations such as speed perception and lane positioning using naïve test person in user experiments. The outcome of these experiments can be used as guidance when and how this technology should be used.

The work on simulation environment integration and scenarios in WP 2 gave many promising results, such as the testing capabilities enabled by virtual objects and V2X capable objects in the simulation. The same conclusion applies to the implementation of dynamic trajectories. For the work on scenarios, all involved partners achieved further insights in the scenario standard OpenSCENARIO and showed that other scenario standards from different tools can be executed in other environments (WP2.6.8).

Future work

For dynamic trajectories, at least two potential future work points were identified:

- Parametrizing the simulated vehicles to make them behave more similar to the physical objects that are to follow the generated trajectories
- Extend the concept of dynamic trajectories to implement a return-to-start functionality, see WP 2.4.5.

It may also be interesting to further explore the possibilities to centralize the different components (Chronos server, simulation environment, supervisor *etc.*) in a single hardware in order to reduce the time lag introduced by the flow of information between components. A centralized hardware could also be moved into the VUT and eliminate further lag from long-range communication over, *e.g.*, 4G. The benefits, disadvantages and challenges of this approach may be considered for future research.

9. Participating parties and contact persons

Organization	Contact person
AstaZero	Katarina Boustedt
AB Volvo	Lars Bjelkeflo
Ericsson	Roland Gustafsson
ESI	Jonas Fredriksson
Fengco	Martin Djup
Inceptive	Magnus Willner
RISE	Jonny Vinter
Veoneer	Rustem Elezovic
Volvo Cars	Francesco Costagliola
VTI	Fredrik Bruzelius