Grant Agreement: 644047

INtegrated TOol chain for model-based design of CPSs

INTO-CPS

D1.2- Case Studies 2

Deliverable Number: D1.2

Version: 1.0

Date: 11 2016

Public Document

http://into-cps.au.dk

Contributors:

Julien Ouy, CLE
Thierry Lecomte, CLE
Martin Peter Christiansen, AI
Andres Vill Henriksen, AI
Ole Green, AI
Stefan Hallerstede, AU
Peter Gorm Larsen, AU
Claes Jger, AI
Stylianos Basagiannis, UTRC
Luis Diogo Couto, UTRC
Alie El-din Mady, UTRC
Hassan Ridouanne, UTRC
Hector Moner Poy, UTRC
Juan Valverde Alcala, UTRC
Christian König, TWT
Natalia Balcu, TWT

Editors:

Julien Ouy, CLE

Reviewers:

Simon Foster, UY
Christian Kleijn, CLP
Stefan Hallerstede, AU

Consortium:

Aarhus University	AU	Newcastle University	UNEW
University of York	UY	Linköping University	LIU
Verified Systems International GmbH	VSI	Controllab Products	CLP
ClearSy	CLE	TWT GmbH	TWT
Agro Intelligence	AI	United Technologies	UTRC
Softeam	ST		

Document History

Ver	Date	Author	Description
0.1	06-15-2016	JO	Initial document version
0.2	07-06-2016	JO	CPS aspects of the use cases
0.3	08-25-2016	JO	first version of the CLE use case
0.4	08-30-2016	MPC	addition of the AI use case
0.5	10-25-2016	JO	addition of the complementarity table
0.6	10-28-2016	MPC	update of the AI use case
0.7	10-31-2016	CK	addition of the TWT use case
0.8	11-04-2016	SB	addition of the UTRC use case
0.9	11-08-2016	JO	internal review version
1.0	11-30-2016	JO	corrections following internal review
1.0	12-16-2016	JO	final version

This document shows the four industrial partners' case studies. A case study contains the description of the industrial case, the modelling and simulation of it using INTO-CPS baseline tools and the assessment of the baseline tools according to their requirements. The goal of this deliverable is to bring insights about industrial benefits and weaknesses of using INTO-CPS baseline tools.

Contents

1 Introduction

This document shows the four industrial partners case studies: the Railway case study from CLE, the Agriculture case study from AI, the Building case study from UTRC and the the Automative case study from TWT. Each case study contains the public description of the industrial case, the modelling and simulation of it using INTO-CPS baseline tools and the assessment of the baseline tools according to their requirements. Prior to the descriptions of the case studies, a section is dedicated to specific points addressed during the first year of the project, the cyber-physical aspects of the case-studies.

2 Cyber-Physical aspects of the Industrial Case Studies

Each of the case-studies presents in a different angle the aspects of a Cyber-physical system. Individually they only cover a fraction of the different features that characterize a CPS but altogether the case studies of INTO-CPS demonstrate all that can be modeled, checked, tested, simulated or generated using the tools provided by INTO-CPS.

2.1 Cyber-Physical aspects of the Railway case study

The Interlocking case-study is intrinsically a safety-critical Cyber-Physical System (CPS). Conceptually speaking the core component is a logical system. The traditional railway approach for this kind of system is to have a redundancy in the logical component between software running on a computer, or in most cases an industrial automaton, and wired electronic boards composed on large shelves. The logic is essentially represented in software as a system of equations and registers, while the hardware is made of circuits and electromagnetic relays. The fault-tolerant design with those two redundant parts of the core monitor each other so that a failure on one side is immediately detected and put the system in a safe mode. The actuators are also monitored by the safety core so that the green light for a tram is given not only when all the switches are set to the right position, but also when sensors on the switches confirm this position. The direct consequence of this self-awareness is the complexity and the cyber-physical aspect of the equations treated in the logical component.

The Interlocking case study focus on an innovation which add yet another cyber-physical element: the distribution of the system over several simple modules, which is considered a challenge for the traditional railway industry. Whereas it can be simple to prove that each module has the same local safety level as a centralized interlocking the fact that a module allows a route only when it is safe to the knowledge of its own sensors is challenging to argue for. It is significantly more difficult to demonstrate safety at the global level because protocols and networks intervene in the challenge. Yet an innovative distributed Interlocking would have benefits for the industry with simpler hardware and the possibility to reach new markets where the distance or the complexity of the geography would prevent the use of a classical system.

The use of heterogeneous models with co-simulation and/or model checking would also lower the time to market of such a product. In railways safety, systems are rarely Commercial of the shelf, since a large part of adaptation and parameterization are needed before it is possible to deliver the final product. Moreover, during the installation, the system or the equipment interacting with it are not necessarily available for testing purposes or not truely representative of the final system. For those reasons, reasoning and testing with a model-based representation of the system as carried out in the INTO-CPS project can be seen as a major cost saver in being able to test in a virtual setting before one have to go into an expensive series of physical tests.

2.2 Cyber-Physical aspects of the Agriculture case study

The agricultural robot platform is comprised of a network of interacting computer units that operate with inputs from and provide outputs to its physical environment. The physical environment the agricultural robot platform is operating in is influenced by geography, weather and terrain conditions. The robot platform is a mobile CPS that needs to incorporate concepts from multiple engineering disciplines such as agricultural, control, electrical, mechanical, signal processing and software design. The robot needs to be able to operate autonomously with minor levels of human surveillance, so it needs to have its own "intelligence". Thus, this system has a high degree of autonomy. Dramatic changes have been made to the overall system engineering design of the Robotti platform, and thus from a commercial perspective it is definitely advantageous to be able to carry out initial investigations in a virtual setting instead of wasting too much time and money on producing full-scale expensive physical prototypes. These aspects collectively make the agricultural case study a good example of a CPS in the INTO-CPS project.

2.3 Cyber-Physical aspects of the Building case study

The building case study is considered to be a full CPS, composed by several cyber and physical elements. The cyber part of the year 2 case study include components such as the FCU controllers that will be embedded in the hardware devices, the communication interfaces in order to handle communication locally between the FCUs and globally between the FCUs and the supervisor. Communication modules will also coexist within the same FCU hardware device and will handle distributed communications exchanged between the FCUs and the supervisor. Commands for both the controllers and communications modules will handle switching mechanisms for certain FCU elements that include the device itself (e.g. on), regulating the fan speed (e.g. medium speed), regulating the air-dumper position (e.g. 10% open), regulating the power of the coil (e.g. 20V) and regulating the water valves position (e.g. 50% open). The aforementioned elements will be targeted for code generation in order to proceed with Software-in-the-Loop and Hardware-in-the-Loop simulations for the aforementioned case study. Finally the communication medium used in the HVAC CPS scenario is also based on protocol message exchange between the supervisor and the FCU controllers (e.g. through UART). The physical part of the year 2 case study will include components that will be co-modeled with the cyber part in the multi-models evaluated in the INTO-CPS platform. Those components include the thermal modeling of the physical rooms and zones in order to study thermal affects of the areas including wall insulation parameters and air-mass heat capacity. Air pipes connection and air flow exchanged between the Air Handling Unit (AHU) and the FCUs is also continuously described by air mass flow inside the pipes used in the building to connect the AHU and the FCUs. Water pipes connection and water flow pressure exchanged between the Heat Pump Unit the FCUs and the AHU, is also governed by continuous equations describung fluid dynamics in the water pipes, pressurized in the Heat Pump and heated or colling at the AHU towards the FCUs. Current continuous models includes 21.000 variables with a total of 11.000 parameters, while the number of differential equations used to describe the system dynamics is close to 7000.

2.4 Cyber-Physical aspects of the Automobile case study

The automotive case study can be considered a Cyber-Physical-System (CPS) because it contains local intelligence and autonomy in the vehicle. This is assisted by information about its environment typically derived from a cloud context (here, information on weather and traffic / route) and the logic depends upon the physical dynamics of the electric vehicle. In the most innovative scenario that is described in section 6.2, a real driver is able to drive a simulated vehicle (using a virtual reality environment). Here the characteristic curve of the accelerator (gas pedal) is adapted based on the remaining range of the vehicle enabled by its battery. If the range is too low for reaching the designated destination, the curve will be ”softened”, resulting in less acceleration at a given position of the accelerator. The controller that changes this characteristic curve is inherently a *cyber* element (a controller, running for example on a Raspberry Pi), which interacts with the physical world (the human driver and the vehicle’s dynamics, which is modeled). This controller is developed using the INTO-CPS tool chain and its methodology, from a systems engineering perspective. Thus, the different constituent elements of the system can be analysed using the HiL capability, as well as features such as testing and code generation that are covered by the INTO-CPS tool chain. Since the Co-Simulation that is used to determine the remaining range of the electric vehicle, also uses information from its environment (route and weather information), it can be considered a smart connected system. At the time of writing, the HiL scenario is being designed, and its implementation is planned for year 3.

2.5 Complementarity of the Industrial Case Studies

The table 1 summarises the different aspects of CPS and the related major industrial needs covered by the case studies of INTO-CPS. Parenthesized yes means that the feature is present in the case study but not completely implemented yet. We see that except for size efficiency, all aspects appear in one or more case studies. Some of them, mainly formal verification, design test exploration and test automation are to be explored in Y3.

Table 1: Complementarities of the industrial case studies at Y2

Application area	Railways	Agriculture	Buildings	Automotive
Lead partner	CLE	AI	UTRC	TWT
Time Critical	Yes	(Yes)	Yes	No
Safety Critical	Yes	(Yes)	No	No
Autonomy	Yes	Yes	Yes	Yes
Power efficiency	No	No	Yes	
High Performance	Yes	(Yes)	Yes	Yes
Size Efficiency	No	No	No	No
Cost efficiency	Yes	Yes	Yes	No
Actuators modeling	Yes	Yes	(Yes)	Yes
Sensors modeling	Yes	Yes	Yes	Yes
Network modeling	Yes	(Yes)	(Yes)	(Yes)
Formal verification	(Yes)	No	Yes	No
SIL Co-simulation	Yes	Yes	(Yes)	(Yes)
HIL Co-Simulation	Yes	Yes	No	(Yes)
Design Space Exploration	No	Yes	(Yes)	(Yes)
Test-Automation	(Yes)	No	No	(Yes)

2.6 Key Performance Indicators

Key performance concerning WP1 relies on three indicators.

KPI-A, utility assessment We can affirm that this indicators has been reached, all case studies in Y2 are developed following the methods and tools from Into-CPS: development of SYS-ML models, CT and DE models of the different components, generation of FMU from Overture, OpenModelica or 20-sim, execution of co-simulation with the COE. This will be detailed in this document.

KPI-B, Reduction of development time for CPS This indicator is evaluated indirectly and has to be detailed for each case study. For CLE we can cite the early evaluation of the safety properties of the system that saved a lot of time: several propositions of distribution have been created and tested for safety properties as models, this phase could append before the choice of the hardware and the development of drivers or communication layers (middleware). This saved the time and money of developing several module prototypes. In the second phase, when the prototype has been tested, no reason came for questioning the choice of the distribution and requiring a second iteration. Automatic code generation of the prototype also saved a lot of time for two reasons, first the manual development of the code from the specification would have require roughly fifteen days whereas generating code from the formal model was almost instantaneous and the code for the middleware alone costed around five days. The time for the development of the model itself is shared with the system specification phase. For the agriculture case study, AI was able to work in parallel in the development of the SiL (tablet interface), which gave AI the opportunity to present results even before constructing a physical prototype. This saved AI an estimated time of 5-6 months in the development time, which would not have been possible without INTO-CPS.

KPI-C, Tool chain platform TRL This indicator can already be set above 4, the tools permit to build operational prototypes, with directly exploitable simulation and testing results. The integration of the different tools together is complete.

3 Railway case study

3.1 The Case study

In railway signalling, an interlocking is an arrangement of signal apparatus that prevents conflicting movements through an arrangement of tracks such as junctions or crossings. Usually interlocking is in charge of a complete line, computing the status of actuators (switches, signals) based on signalling safety rules that are encoded as so-called ”binary equations”. A typical interlocking is in charge of managing $\sim 180,000$ equations (see figure 2, for instance) that have to be computed several times per second. These equations compute the commands to be issued to track-side devices: they encode the safety behaviour that enable trains to move from one position to another through routes that are allocated then released.

Currently, there are attempts to find the right trade-off between efficiency of an interlocking system (availability of routes, trains' delays and cost of interlocking system) and safety (collision avoidance, derailment prevention, availability and efficiency of emergency system).

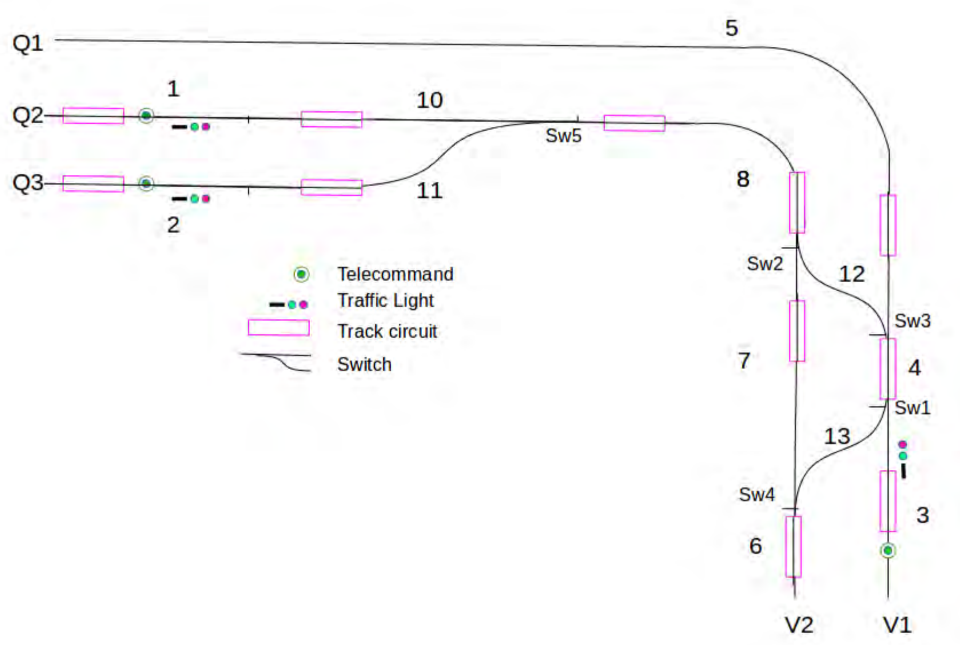


Figure 1: Partial scheme plan of a train line as seen from the sky, including track circuits, switches and Traffic lights

3.1.1 Interlocking challenges

A central interlocking is able to deal with a complete line, all decisions are made globally. However the distance between devices distributed along the tracks and the interlocking system may lead to significant delays to update the status of the devices. Moreover this architecture, well dimensioned for metro lines, is often overkill for simpler infrastructures like tramway lines.
So there is room for an alternate solution: distributed interlocking. A line is then divided into overlapping interlocked zones, each zone being controlled by an interlocking. Such interlockings would be smaller as fewer local devices have to

be taken into account and a local decision could be taken in a shorter amount of time and would result in potentially quicker train transfers. However, overlapping zones have to be carefully designed (a train cannot appear by magic in a zone without prior notice) and some variable states have to be exchanged between interlocking systems as the Boolean equations have to be distributed accordingly over the interlocking systems.

This distribution implies several engineering problems. An “optimal distribution” i.e. the decomposition of the line into overlapping areas such as to minimise delays, availability and costs, requires a smart exploration of the design space (decomposition is directly linked with railways signalling rules). It also implies that one has to define what information has to be exchanged between interlocking computers and how many equations have to run on any of them (20,000 equations maximum for example).

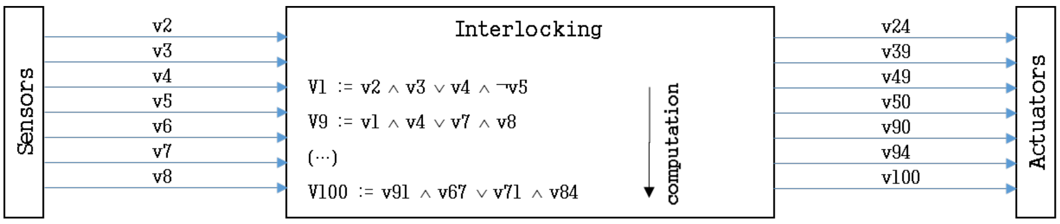


Figure 2: boolean equations that lead the signalling system

3.1.2 Accurate Train movement simulations and challenges

In order to have a realistic overview of the traffic of the trains and deal with safety concerns, the train movements along the track map have to be simulated in a realistic way. The finer the movement is simulated, the more one can ensure an efficient but safe interlocking system. Usually, a train receives/considers a Movement Authority Limit (MAL) : a stop point that the train must never overshoot. Such a MAL is updated in real-time by interlocking mechanisms and communication facilities. For an automated train, Automatic Train Operation (ATO) computes the best movement for reaching and stop at the MAL. In parallel, an Automatic Train Protection (ATP) guards against a failure of the ”normal” service mode (e.g. service brakes failure, ATO software/sensor loss of train position). ATP checks that in the worst case, the MAL will never be overshot. Exploring the behaviour of a “manual mode” train (with possible rollback movement) and ensuring a safe automatic protection is far from being clarified in the Railway community. Even

ClearSy - which has used the ProB animator for years (a high level discrete modelling language, similar to VDM-RT) for railway use cases animations- cannot achieve, in a discrete way, a smart handling of continuous movement, of the maximal assessments of physical parameters or of the continuous time problems such as differential equations, Zeno paradox or controlling precision results.

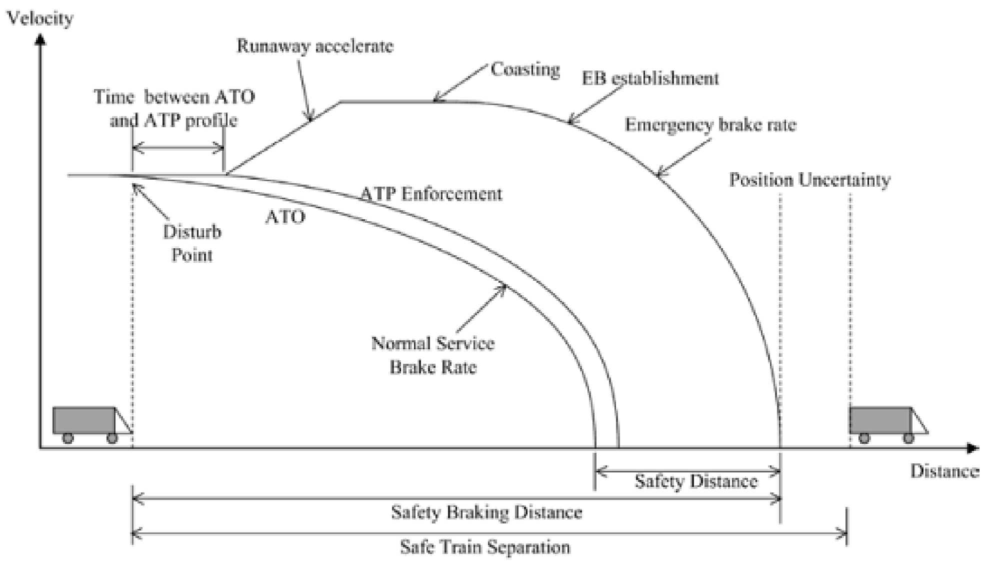


Figure 3: Usual Safe Braking Model

Simulating together and in real time an efficient interlocking system with train movement would enable to enhance train performance with respect to delay or availability whilst keeping it safe. In order to get a higher level of confidence about the safety of the case study the co-simulation conducted must be accompanied with other techniques developed for the INTO-CPS tool chain such as the model checking feature.

3.1.3 Design of a Distributed Interlocking

There can be several alternative solutions for the distribution challenge of this case study, depending on the properties we want to protect or features we want to provide. The distribution of the interlocking system has been driven by geographical distribution. The trackmap has been cut into five automatons/modules communicating via a ring network of UART. This solution is adapted to a large system because the compactness of the modules offers short distances between captors or actuators and the automaton they belong to while the long distance between modules can be covered by serial connections. The distribution of the equipment can be seen in figure 4.

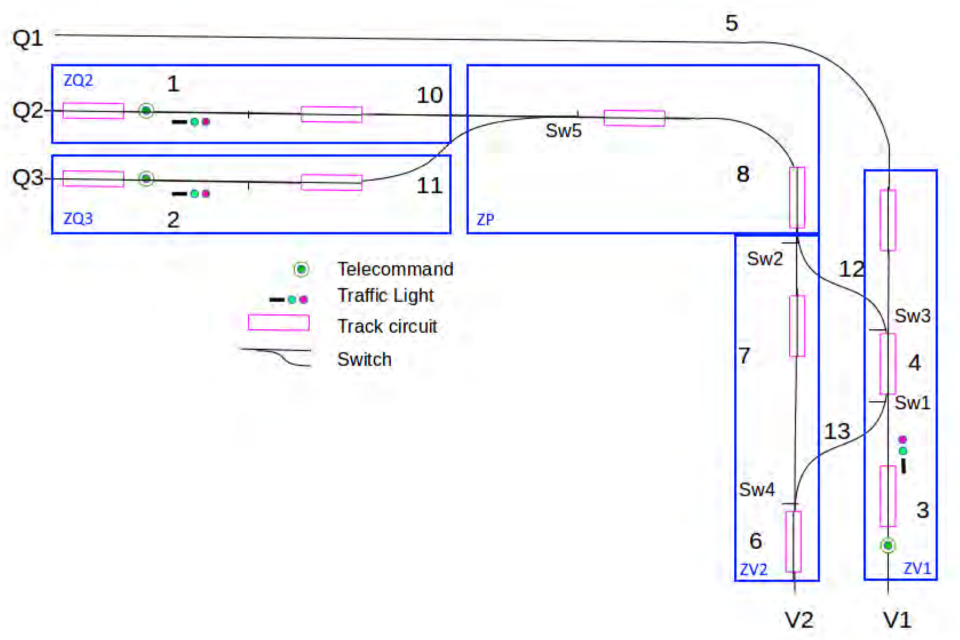


Figure 4: Distribution of the Interlocking system over the trackmap

3.2 Contribution during Year 2

3.2.1 Discrete Event Modeling (DE)

The DE model of the Interlocking is described using VDM-RT. The first centralised VDM-RT version of the interlocking system developed Year 1 was semi-automatically derived from Ladder code for a PLC. Since we did not need a Ladder version for Automaton implementation, modelling directly in VDM-RT was more efficient than reusing the version from Year 1. However, this distributed model is conceptually derived from the centralised version. Most of the equations are kept or duplicated between the different modules, a few have been created specifically for the distribution. So while the safety kernel of the centralised interlocking model contained 180 equations, the distributed one has 250 equations shared between the five modules (85 for ZV1, 49 for ZV2, 37 for ZQ2 and ZQ3 and 39 for ZP). The distribution of the model introduced a new and important feature to the system: message communication. The five interlocking modules need to communicate over long distance and with different signals. For this purpose we designed a ring network protocol and each module possesses a networking layer able to transmit and treat messages. This network function is also a part of the VDM-RT model and this gives benefits in the simulation.

Composition of a module
Each of the modules contains inputs, local and output variables. Inputs come from sensors (track circuits, states of relays, and states of switches), commands from the train and messages from other modules. Outputs are commands to relays (construction and destruction), commands to switches (left and right) and commands to light signals. Locals have two purposes: to compute intermediate variables and to reflect the state of a distant module. Those variables form the system of boolean equations.

3.2.2 Continuous Time Modeling (CT)

The CT model from Y1 has been reused and modified for use with the new INTO-CPS Co-simulation Orchestration Engine (COE). It consists of a train in motion over a changing track-map. The train starts or stops at a signal light, and advances on the track-map given the track chaining. The length of the track, the chaining of track and switches and the positions of the sensors and actuators are parameters of the CT mode. They are stored in text files and read during the execution of the simulation. The signal authorisation and the switche positions are granted by the interlocking system. The maximum speed of the train at a certain position is computed by the CT model, in accordance with the safe breaking methods. During Y2, we modified the model by better separating the parts: parameters, trains, Traffic_Light_Manager, RelayBox, MIC_Manager and CommandPcc as can be seen in figure 5.

The blocks help to simulate hardware parts of the interlocking. Mainly, RelayBox and Mic_Manager simulate the behaviour of the safety relays and the switches. They are triggered by the commands of the DE model and provides delayed control variables for proofreading. CommandesPcc was a set of signals used for command or parametrisation of the DE system. It was later removed and its contents were changed into parameters.

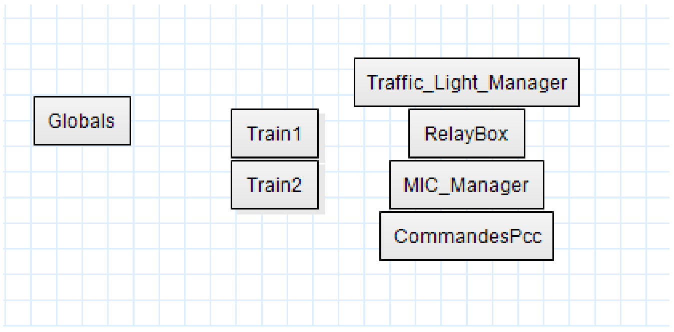


Figure 5: Highest block level of the CT model

Then the model was reshaped for modularity. A higher level of abstraction was added: figure 5 shows the simulation level which add the possibility to use different refinements and thus different simulations. SIL simulation can be run in two different tools: Crescendo or the new INTO-CPS COE. This abstraction model is the starting point of both kind of simulations. The Crescendo_Interface block contains the arrays of variables for the Crescendo co-simulation and both TrainFMU and Relay Box can generate an FMU for COE co-simulation.

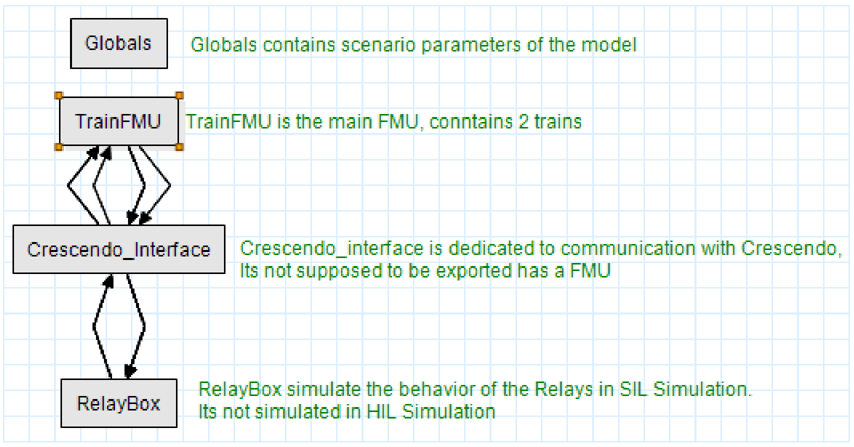

Globals
Globals contains scenario parameters of the model
TrainFMU
TrainFMU is the main FMU, contains 2 trains
Crescendo_Interface
Crescendo_interface is dedicated to communication with Crescendo,
Its not supposed to be exported has a FMU
RelayBox
RelayBox simulate the behavior of the Relays in SIL Simulation.
Its not simulated in HIL Simulation

Figure 6: Architecture of SIL setting

For Hardware in the loop (HIL) co-simulation, we fused the blocks Crescendo_-Interface and RelayBox into one abstract interface block. The interface model is basically a wrapper that communicates with the hardware outside of the computer, which will be detailed in section 3.2.5. The interface between both boxes is the same for SIL and HIL co-simulation: RemoteCommand and TrackCircuit from train to Interlocking and Switch positions and Traffic light from Interlocking to train.

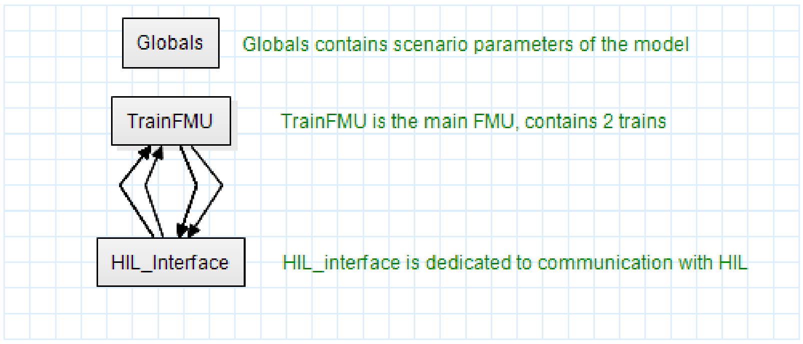

Globals
Globals contains scenario parameters of the model
TrainFMU
TrainFMU is the main FMU, contains 2 trains
HIL_Interface
HIL_interface is dedicated to communication with HIL

Figure 7: Architecture of HIL setting

Inside of the TrainFMU sub-model we added a way to manage a variable number of trains. Inputs of the sub-model are distributed to the instances of trains and outputs are computed from their outputs (boolean disjunctions or conjunctions depending on the signal).

2D Interface
For the purpose of demonstration, simulation graphs are not enough. We need to better represent the execution of the simulation. Therefore, we used the mechatronic module in 20sim to build an animated representation of the trains and their dashboards. This work is planed for Y2 but still in progress.

3.2.3 FMU Generation

CT model: Trains: Generating an FMU for the 20sim CT model of trains was not straightforward. Unfortunately, three aspects of the model were not supported at the moment of the test: tables, events by the 20sim FMU generator and arrays by the COE. Arrays have been flattened to scalars, tables were replaced by statical arrays put directly in the model. This lowered the possibility of evolution of the model but does not change its meaning. Events were a larger problem: the calculation of most equations of the model were based on events and it has not been possible to remove the concept of event without breaking the model. Thus instead we build a lighter version of the model. It modelled movement of the train and activation of the sensors but neither safe-breaking, derailment or slope management.

CT model: Relays: The FMU for Relays was generated without any problems.

DE model: VDM Interlocking: Generating an FMU from VDM required only to adapt the interface for scalars like the CT model and use the dedicated FMI library.

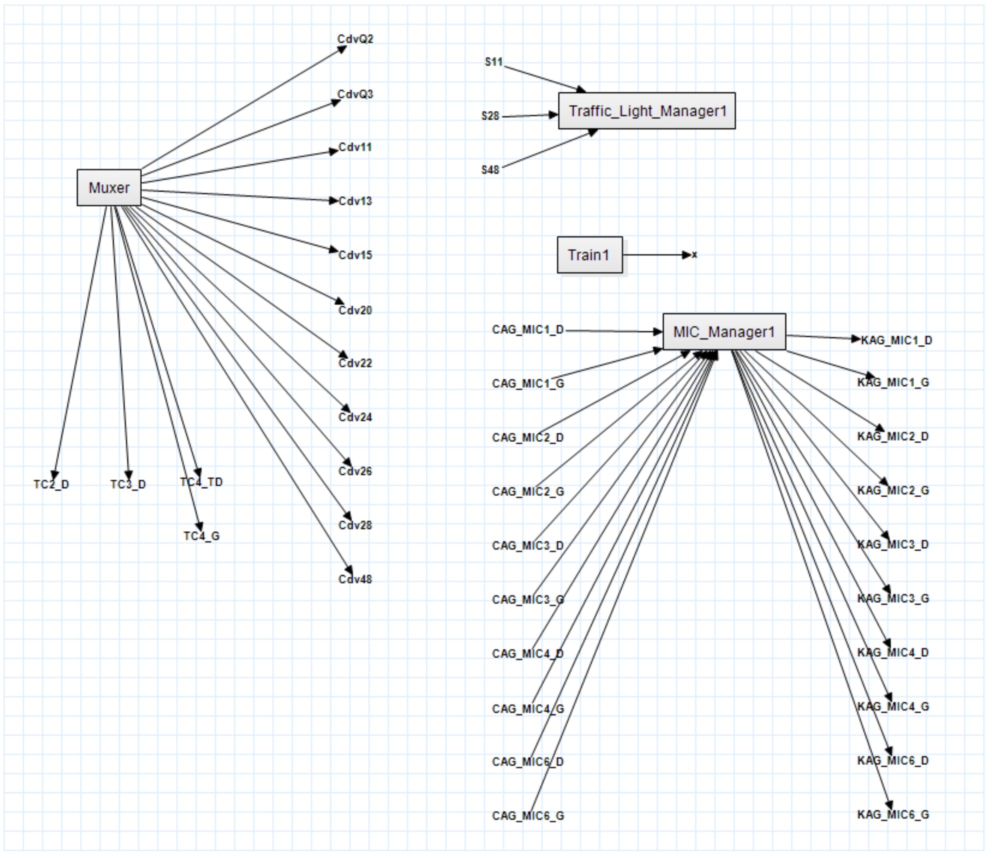


Figure 8: The TrainFMU Submodels

DE model: HIL wrapper: The HIL FMU wrapper is still in progress. The plan is to generate the interface with 20sim or Modelio and manually write the content of the FMU (message packing and sending through the network). This will be done until the end of Year 2.

3.2.4 Co-simulation

Co-simulation using Crescendo was a great success during Year 1 and is still succesfull with the models of Y2. Achieving the same goal using FMU faced some constraints on models (see before) and exploitation of the results but was also a success.

Crescendo Co-simulation
Crescendo was for us a reference for co-simulation, therefore the new CT and DE model needed to pass the test of Crescendo before going further in testing the

Simulation Name	Simulation duration	Crescendo duration	COE duration
Q3V2_init	15	24	18
Q3V2_full	40	55	20
Q3V1-V1Q3	70	92	40

Table 2: Duration of simulations

new INTO-CPS tool chain. The co-simulation was progressing well and without adjusting the model, mostly because those models were derived from a version that was already able to co-simulate in Crescendo. The most noticeable problem was the speed of the simulation. The full simulation of a single train passing (40 sec of simulated time) was taking 55 seconds on an average laptop (table 2).

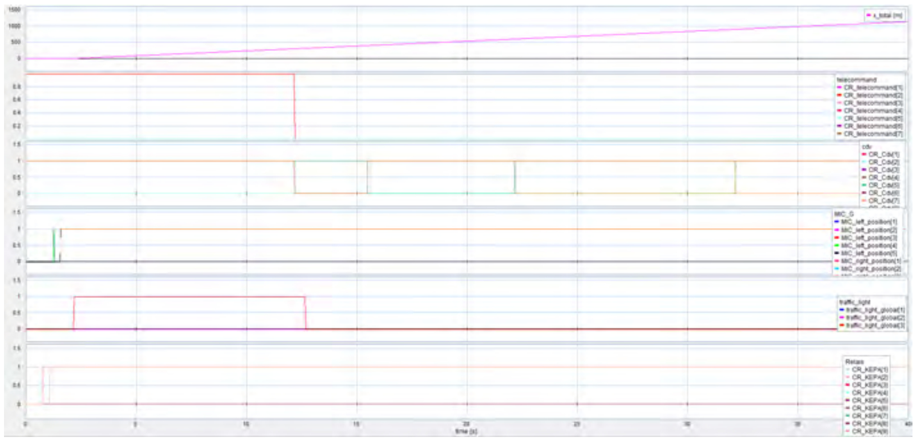


Figure 9: Co-simulation of the route Q3-V2 with Crescendo

COE Co-simulation

After correct generation of the FMUs, co-simulation with the INTO-CPS COE was a very simple task. Setting the configuration and connecting the interfaces in the application is very easy and there is very little room for user mistyping. The simulation is fast and gives immediate results. Graph results may not be as easy to read as in Crescendo but the raw Data can be read with external software.

FMU generation from 20sim and VDM and COE co-simulation is fast and reliable. The process of generating a new FMU for a modified model is very fast so it’s possible to immediately test variants in co-simulation.

3.2.5 Hardware In the Loop

Centralized Version

The centralised version of the interlocking has lead to the realisation of a dedicated Hardware in the Loop prototype. It consists of a micro-controller development board running the interlocking automaton software. On this board, we added LEDs and relays to reproduce the real interlocking system running on site. This board has been connected to the simulator running on the PC and was used to validate the accuracy of the simulation. The software running on the board has been translated by hand, being too early in the project for code generation. The co-simulation with centralised HIL was running only 20-sim on the PC: the 20-sim safe-breaking CT model and a 20-sim wrapper. This validated the continuous CT model but lacked the use of the COE. This should be improved with the distributed HIL prototype.

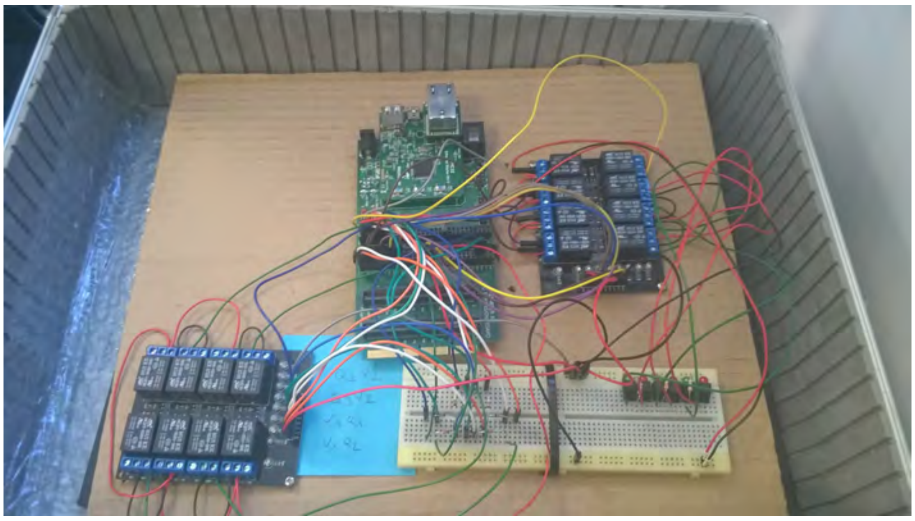


Figure 10: First Hardware in the Loop prototype

Distributed Version

The second version of the HIL prototype was for the distributed Interlocking. For this prototype we needed six independent modules, one for Ethernet communication with the simulator and five for the parts of the interlocking. We choose to design dedicated boards for the integration of the micro-controllers and their equipments. Each board is able to lock three relays for route reservation, manipulate two switches, three track circuits and one pair of signal lights. The central module plays only the role of an interface, it sends and receives messages from the computer running the simulation and converts them to TTL signals (Transistor-Transistor Logic) for the Interlocking modules, simulating the actual sensors and actuators of the system.

The communication between the modules is handled by a ring network of RS-232 connections. Each message is sent to the left neighbour and forwarded until it reaches its destination.

On each module, LEDs represent the state of the equipments, yellow for switches, blue for routes and red/green for the authorisation light.

This work still needs to be completed, the demonstration will be satisfying when we will be able to use the C code genereted from VDM-RT, embed it on the board and run the distributed system along with the CT simulation on the PC.

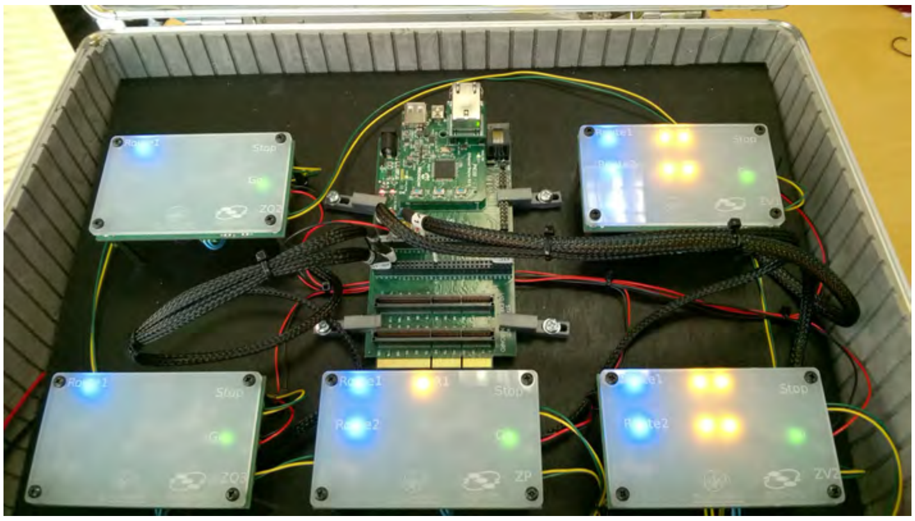


Figure 11: The distributed Hardware in the Loop prototype

3.2.6 Improvement for Industrial Development

Year 2 was the occasion to produce a new product, the Distributed Interlocking System, comparable to the one produced during Year 1, the Centralized Interlocking System but with the tools provided by INTO-CPS. The main benefits from using the INTO-CPS tools are in the verification and validation phases of the V-cycle. The immediacy of the simulator tests reduces greatly the time between successive versions. Moreover, most of the testing can be executed at a higher level of abstraction using models which is sufficient for co-simulation. Of course the implementation of the model via code generation will also be submitted to a serie of tests later in the process, but the corrections during these tests will concern only implementation problems (scheduling, I/Os, . . .) that could be found and solved with simple unitary tests.

3.3 Industrial needs and assessments

Table 3 presents an overview of ClearSy's need for the INTO-CPS tool chain. In the following part, we detail each ClearSy's (CLE) needs. Compared to Year 1, fulfilment of the needs by the INTO-CPS tools has been added to compare with the baseline tools.

3.3.1 Cle_1: ClearSy_time_Modelling

- **Description**
  The tool **must** enable to express delays of system's actions/steps. The tool **must** enable the description of delay and the duration of physical components such as the commutation of relays and the state changes of the motorised switches. Also, it must be possible to set the time cycle of a micro controller.
- **Related Baseline Tools Requirements**
  [LPO+16] Requirements 0003-0005 are related because the modeling of time step could be done at the co-simulation level.
- **Method of verification**
  While modeling the Railway signalling system using 20-Sim, the duration of state changes of the relays and switches were modelled using the 20-Sim operator "`tdelay`" (Indicators: succeed: yes/no, rate of success over cases). Cycle-time of interlocking software were also modelled using a Clock operation that consumes a duration (using the VDM-RT operator "duration").
- **Assessment: achieved**
  In all our case studies, modeling delays and cycles times were achieved in VDM-RT and 20-Sim. Indicator: 100%

3.3.2 Cle_2: ClearSy_time_Simulation

- **Description**
  The tool **should** simulate delays/cycle time of system's actions/steps. The tool **should** enable the description of computational delays and duration of commutation of relay in a signalling system.

- **Related Baseline Tools Requirements**
  [LPO$^+$16] Requirements 0003-0005 is related because the simulation of time could be done at the co-simulation level. Requirement 0024 - is important to be as precise as possible.

- **Verification Method**
  ClearSy attempted to simulate the Railway signalling system using 20-Sim and VDM-RT. The switching of the relays should be simulated. Cycle-time should be simulated (Indicators: succeed: yes/no, rate of success over cases).

- **Assessment: Partially achieved**
  The PLC time cycle and the delays of state changes for relays and motorised switches have been successfully simulated and also co-simulated with Crescendo. Thus, the `tdelay` operator from 20-sim is not implemented in FMU generation. The COE co-simulation required a modification of the model that removed this aspect. Indicator: 75%

3.3.3 Cle_3: ClearSy_time_Trigger_Modelling

- **Description**
  The tools **must** contain, at least in one of their languages, a trigger artifact based on time or delay.

- **Related Baseline Tools Requirements**
  [LPO$^+$16] Requirements 0003-0005 is related because the simulation of time could be done at the co-simulation level.

- **Verification Method**
  VDM-RT was tested for the ability to code time based trigger.

- **Assessment: achieved**
  It has been possible to set the duration of a computation unit using the VDM-RT CPU class. Cycle-time of interlocking software has also been modelled using a Clock operation that consumes a duration (using the VDM-RT operator ”duration”). The ”TON” and ”TOF” functions - from the LADDER code that enable to handle delay before triggering a signal or to hold a signal constant during a period of time- have been handled in VDM-RT by computing explicitly at each micro-step duration, the total elapsed time. IN function of the elapsed time the TON/TOF operators decide the triggering. Indicator: 100%

3.3.4 Cle_4: ClearSy_time_Trigger_Simulation

- **Description**
  The tool **should** simulate the trigger based on time or delay. In order to be able to synchronise with relays, and to express cycle time, it should be possible to simulate in the software logic the clock or delay enabling to postpone the triggering of operations.
- **Related Baseline Tools Requirements**
  [LPO+16] Requirements 0003-0005 are related because the simulation of time could be done at the co-simulation level. Requirement 0024 is important to be as precise as possible.
- **Verification Method**
  VDM-RT will be tested for simulating the code of time based trigger. (Indicators: succeed: yes/no, rate of yes over cases).
- **Assessment: achieved**
  It has been possible to simulate the duration of a computation unit using the VDM-RT CPU class. Cycle-time of interlocking software has also been simulated using a Clock operation that consumes a duration (using the VDM-RT operator ”duration”). The ”TON” and ”TOF” functions - from the LADDER code which enable to handle delay before triggering a signal or to hold a signal constant during a period of time- have been simulated. Indicator: 100%

3.3.5 Cle_5: ClearSy_checking

- **Description**
  The tool **should** enable to check logical consistency at the level of co-simulation.
- **Related Baseline Tools Requirements**
  [LPO+16] Requirements 00032 to 00035 are Model-checking based requirements(discrete Model-checking, continuous Model-checking and global(co-simulation) Model-checking). Their achievements are important for Cle_5 requirement industrial achievement.
- **Verification Method**
  Non collision invariant or Overall delay should be checked. For instance the duration of switching of relays which was missing in ClearSy’s first prototype and which caused an error at the testing phase on the industrial site,

could be earlier found out with the help of the model checker.
Indicator: succeed yes/no, rate of yes over cases
This should be done by model-checking by VDM-RT/RT-Tester/20-Sim. (Indicator: number logical of consistencies checked/ all logical consistencies)

- **Assessment: not achieved**
  At Y2, global Model-checking has not been tried yet. It has been possible to check invariant at the level of VDM-RT, and to inject error warning in the 20-Sim model when continuous invariant is falsified (detection of derailment). Indicator: 25%

3.3.6 Cle_6: ClearSy_Simulation_Scalability_1

- **Description**
  The tool **could** simulate real size Railway map evolution and trains movements.
- **Related Baseline Tools Requirements**
  [LPO+16] Requirement 0024, since it is important to control Simulator in order to avoid side effects from computation latency.
- **Verification Method**
  Indicator: Number of simulated tracks (and track circuits sensors), cross/joins (and join sensors) and related equations of simulation of train movement (20-Sim).
- **Assessment: Partially achieved**
  The railway case study provided several csv files that store track map data (joins, traffic light, track circuit...) . Modeling the data for FMU generation is more complicated than for baseline tools. Data in csv files could not be read by the CT FMU. CT model has been adapted for FMU generation and csv import has been removed. Indicator: 75%

3.3.7 Cle_7: ClearSy_Simulation_Scalability_2

- **Description**
  The tool **could** simulate real size railway signalling variable evolutions.
- **Related Baseline Tools Requirements**
  [LPO+16] Requirement 0024, since it is important to control Simulator in order to avoid side effects from computation latency.

- **Verification Method**
  Indicator: number of signalling variables and logic that are simulated into VDM-RT.

- **Assessment: achieved**
  During Y2, the number of variables of the VDM-RT model of interlocking has increased due to the distribution. The simulation and co-simulation has been successfully achieved. Indicator: 100%

3.3.8 Cle_8: ClearSy_Simulation_Exploration_Scalability

- **Description**
  The tools **should** integrate several heterogenous models seamlessly.

- **Related Baseline Tools Requirements**
  [LPO+16] Requirements 0018-0020 could be necessary in order to assess the maximal/minimal value from a range of parameters. The guidance requirements 0073, 0076 would be welcome.

- **Verification Method**
  Indicators: possibility to assess the maximal/minimal/optimal value of a parameter from a range of test. Use case : rollback case maximal value assessment, availability maximal value assessment, maximum duration assessment of train movement.

- **Assessment: not achieved**
  The INTO-CPS tools now allow to sweep a range of parameters at the level of co-simulation, only the train speed parameter has been tested so far. Indicator: 25%

3.3.9 Cle_9: ClearSy_Simulation_Accuracy_Confidence

- **Description**
  The tool **could** make clear the mechanisms, the accuracy and confidence of the simulation. It **could** be possible to handle and make clear the simulation of ordinary differential equations, with discontinue acceleration. It **could** be possible to model, explain and simulate multi-masses movement.

- **Related Baseline Tools Requirements**
  [LPO+16] Requirements 0045, 0047, 0055, 0058, 0061, 0065 : Semantics are necessary in order to keep accuracy and confidence. Quantifiable

simulation tolerance at the INTO-CPS co simulation are necessary to keep accuracy.

- **Verification Method**
  -20-Sim discontinuity handling (yes/no, accuracy/explanations)
  -20-Sim-Crescendo/COE maximal/minimal value assessment for a range of test case at the co-simulation level, margin error (rollback, is there margin error)

- **Assessment: not achieved**
  The 20-Sim modeling successfully handles the discontinuity of the acceleration because of the change of the track map (and so because the slope may change), or because of an emergency braking. However, discontinuity is not supported for FMU generation. Indicator: 25%

3.3.10 Cle_10: ClearSy_Seamless_integration

- **Description**
  The tool **should** easily enable integrating several heterogeneous models. The co-modeling level **should** enable modeling a continuous train movement, model the track map (with discrete information), model the interlocking signalling software and model the electrical relays/switches.

- **Related Baseline Tools Requirements**
  The [LPO+16] guidance requirements 0067, 0071, 0076 are concerned. Help for modeling at the co-simulation level is concerned: Requirements 0049 and 0050, 0051, 0052.

- **Verification Method**
  The Crescendo gluing/orchestration engine tool has been tested.
  FMI based co-simulation has been tested.
  Indicator: co-simulation: yes/no/partially achieved, rate duration to ”develop co-simulation”
  Is it FMI compliant ?

- **Assessment: partially achieved**

  The FMI co-simulation of VDM-RT and 20-Sim has been successfully achieved but with the use of a degraded CT model(see above part of co-simulation).
  Indicator: 75%

3.3.11 Cle_11: ClearSy_Distributed_Modelling

- **Description**
  The tool **should** enable the modeling of distributed hardware, with communication delay.
- **Related Baseline Tools Requirements**
  In [LPO+16] the requirement 0024, related to model communication delays at the co-simulation level is concerned.
- **Verification Method**
  The co simulation with 20-Sim, VDM-RT and the COE has be assessed against the use case of distributed interlocking (distributed communicating Hardware)
- **Assessment: achieved**
  During Y2 a distributed model of Interlocking has been produced and used for co-simulation, using simulated communications. Indicator: 100%

3.3.12 Cle_12: ClearSy_Code_generation

- **Description**
  The tool **should** easily enable generating code (C) or binary (HEX) with compatible facilities, complying with safety critical standards without too much need for manual patch.
- **Related Baseline Tools Requirements**
  [LPO+16] requirements 0037, 0042, 0044
- **Verification Method**
  VDM-RT Interlocking software generation (for code execution) on Pic 32 micro controllers for simulating interlocking.
  Indicator: achieved or not, duration to set the generation
- **Assessment: not achieved**
  At M22 code generation is still in progress. Early prototypes have been tested and look promising. Indicator: 50%

3.3.13 Cle_13: ClearSy_certification_Safety

- **Description**
  The INTO-CPS tool chain **should** provide quality arguments for a possible

certification kit (or any means to ease safety case) or redundant validation chain.
- what is the global level of confidence ? what elements are available to be used for safety case ? For each formalised modeling language (such as OpenModelica and VDM-RT) the language provider should also provide evidence that the corresponding simulator adhere to the formal semantic of their language.

- **Related Baseline Tools Requirements**
  The requirements 091 and 092 from [LPO+16] are critical for justification of well-foundedness and safety handling. [LPO+16] Requirements 0045, 0047, 0055, 0058, 0061, 0065 : Semantics are necessary in order to keep accuracy and confidence. Quantifiable simulation tolerance at the INTO-CPS co simulation are necessary to keep accuracy.

- **Verification Method**
  Indicator: Safety certification kit/method: yes/no

- **Assessment: not achieved**
  At the first year, there is no available clear (formal) semantic of VDM-RT, 20-Sim or OpenModelica and Crescendo. Moreover, there is no certification that the baseline tools behave as their specified semantic. Neither the co-simulation engine, or simulation engine, provide tolerance margin of the resulted simulations. There is no redundant validation chain for safety purpose. There is no certification kit. Indicator: 0%

3.3.14 Cle_14: CLearSy_Failure_Modelling

- **Description**
  The tool could enable modeling degraded mode at the co-simulation level.

- **Related Baseline Tools Requirements**
  [LPO+16] guidance requirement 0082 is critical. The interrupts mechanisms are also important at requirement 0056.

- **Verification Method**
  VDM-RT/20-Sim. Model Emergency braking phase. Indicator: yes achieved /no/ partially

- **Assessment: not achieved**
  It hat not yet been tested to model faulty behaviour in our CT model at the co-simulation level. Indicator: 0%

3.3.15 Cle_15: ClearSy_Traceability

- **Description**
  The tool should enable to coherently organise requirements and systematically to warn the user about missing checking of requirements against simulation or automatic checking tools.
- **Related Baseline Tools Requirements**
  From the deliverable [LPO$^+$16], the requirements 0089 and 0090 are critical for traceability and impact analysis. [LPO$^+$16] guidance requirement 0074 is welcome. Requirements 0012-0017 are important.
- **Verification Method**
  Modelio testing of requirements handling (Indicator: duration to set a requirement),
  traceability (indicator : yes/no)
  easy checking (indicator duration to Set/launch checking)
  dealing with versions , indicator : yes /no
- **Assessment: Partially achieved**
  Duration to set a requirement w.r.t. internal ClearSy tools: writing a few Excel requirements: 10 min, writing the same Modelio requirements: 12 min.
  There is traceability facility, but not between a requirement and a piece of code (some area in the code), or a document (system, Hardware) and not documented in the INTO-CPS project yet.
  An easy checking is possible in theory but not documented in the INTO-CPS project yet.
  Indicator: (50%) .

3.3.16 Cle_16: ClearSy_3D animation

- **Description**
  In the baseline tool 20-Sim it is possible to have a 3D animation of the progress while being simulated. It would be essential to keep this kind of functionality in the multi-model FMI based co-simulation as well. A 3d animation would be useful for better co-simulation understanding, with an increasing number of variables, it becomes difficult to follow the progression of a simulation.
- **Related Baseline Tools Requirements**
  From the deliverable [LPO$^+$16], the requirements 0093 is critical.

- **Verification Method**
  Existence of such available 3D-animator using FMU
- **Assessment: Partially achieved**
  The 3D animator embedded in 20-Sim has been made compatible with FMI. A model is being designed for the Railway Use-case but has not been tested in cosimulation yet.
  Indicator: (60%) .

Table 3: ClearSy's Needs

Needs.	Insight	Priority	Objective	Y1 indic.	Y2 indic.	Diff.	Accept.
Cle_1	Express delays	M	Year 1	100%	100%	+0%	Achieved
Cle_2	Simulate delays	S	Year 1	100%	75%	-25%	Partially achieved
Cle_3	Model trigger time	M	Year 1	100%	100%	+0%	Achieved
Cle_4	Simulate trigger time	S	Year 1	100%	100%	+0%	Achieved
Cle_5	Checking logical reqs.	S	Year 2	25%	25%	+0%	Not Achieved
Cle_6	Simulate track map	C	Year 1	100%	75%	-25%	Partially achieved
Cle_7	Simulate signal logic	C	Year 1	100%	100%	+0%	Achieved
Cle_8	Easy Space exploration	C	Year 2	0%	25%	+25%	Not achieved
Cle_9	Ssimulator Accuracy	C	Year 3	25%	25%	+0%	Not Achieved
Cle_10	Integrability of Models	S	Year 1	65%	75%	+10%	Part. Achieved
Cle_11	Distributed Systems	S	Year 1	0%	100%	+100%	Achieved
Cle_12	Generate (C or Hex)	S	Year 3	25%	50%	+25%	Partially Achieved
Cle_13	Certification kit	S	Year 3	0%	0%	+0%	Not Achieved
Cle_14	Degraded mode	C	Year 2	0%	0%	+0%	Not Achieved
Cle_15	Organize reqs.	S	Year 2	50%	50%	+0%	Part. achieved
Cle_16	3D Visualisation	S	Year 1	50%	60%	+10%	Part. achieved

4 Agriculture Case Study

This is the public version of the agricultural case study which provides a general overview. A more detailed version of the same document can be found in the confidential version.

4.1 Introduction

This case study is provided by the Danish company Agro Intelligence (AI) and it is focused on the evaluation and development of an agricultural robotic platform. This document also builds on the work submitted in the D1.1c deliverable after the first year of the INTO-CPS project [EGH15].

Figure 12 illustrates the three different generations of the robot over time, until the writing of this deliverable. The work presented in this deliverable will primarily be focused on the current version under development and secondly provides an update on the modelling of version 2 of the robot. The reason why the main focus is on the new version of the robotic platform is to evaluate how the INTO-CPS tool chain can be used to support rapid development of such an agricultural system.

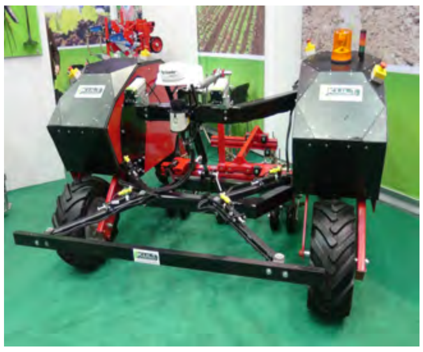
4.2 Agriculture case study

This case study is focused on the agricultural robotic platform Robotti. An early version of this robot can be seen in Figure 12a. Robotti is designed as a low cost, semi-autonomous machine to apply different kinds of soil and crop treatments through agricultural tools called implements. Examples of implements could be sowing, weeding, spraying or row cleaning tools.

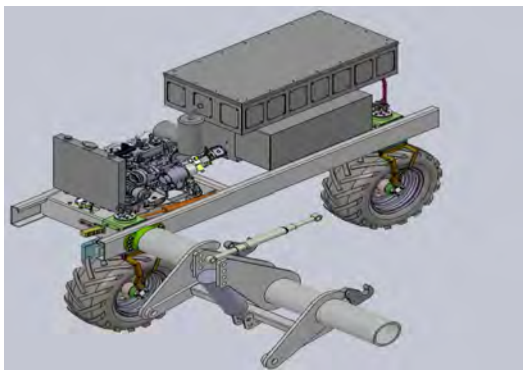
4.2.1 Description

The robot was initially conceived to be deployed and operated in fields structured in rows. An overview of the working environment is shown in Figure 13. This figure shows the start position where the robot is normally deployed and it represents with arrows the trajectory it follows. The robot transits from one row to the adjacent one in the turning areas. The field in which the robot is deployed is not necessarily flat and it could present different kinds of slope changes. In addition to the plants and the soil that have to be treated, there are other external elements

(a) Front view of the first generation Robotti.



(b) Presentation of the second generation Robotti



(c) CAD drawing of the third generation Robotti

Figure 12: Pictures of three generations of Robotti as of first of August 2016.

in the environment that can hamper the normal operation of the robot. These are obstacles of different sizes, such as animals or humans in the way, that require different kinds of actions. Small obstacles can be dealt with autonomously by the robot by handling the height of the implement, while big obstacles demand a machine full stop.

Figure 14 shows a specific kind of implement operated by the robot is in the field. In the agricultural domain there are many different kinds of implements for different purposes. The ones considered in this case study are row cleaning implements.

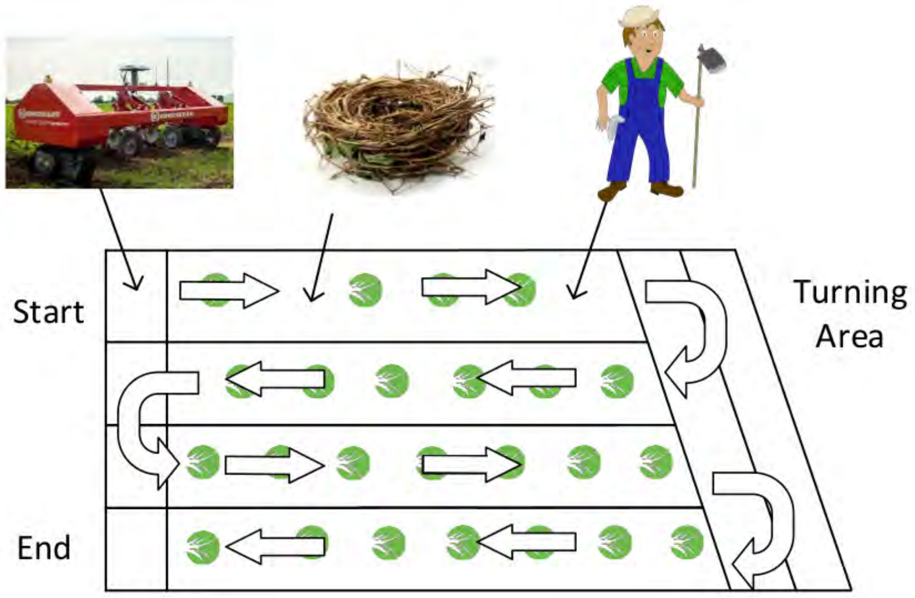


Figure 13: Example of a field where Robotti could be deployed.

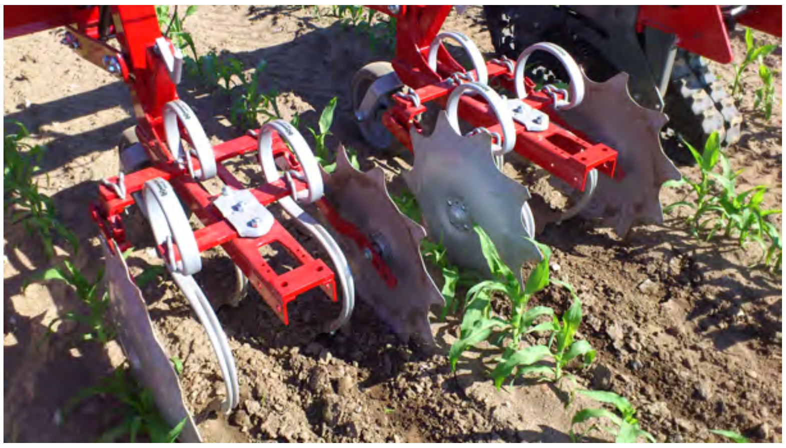


Figure 14: Two different row cleaning implements mounted on Robotti and tilling the soil.

4.2.2 Requirement and Specification

The industrial project Robotti is composed of a number of requirements and scenarios. In order to present an approachable case for the INTO-CPS research project a subset of eight functional requirements has been formulated as Use Cases (UC). An overview of these use cases is provided in Figure 15.

The overall use case is R1 and it is decomposed into additional seven use cases.

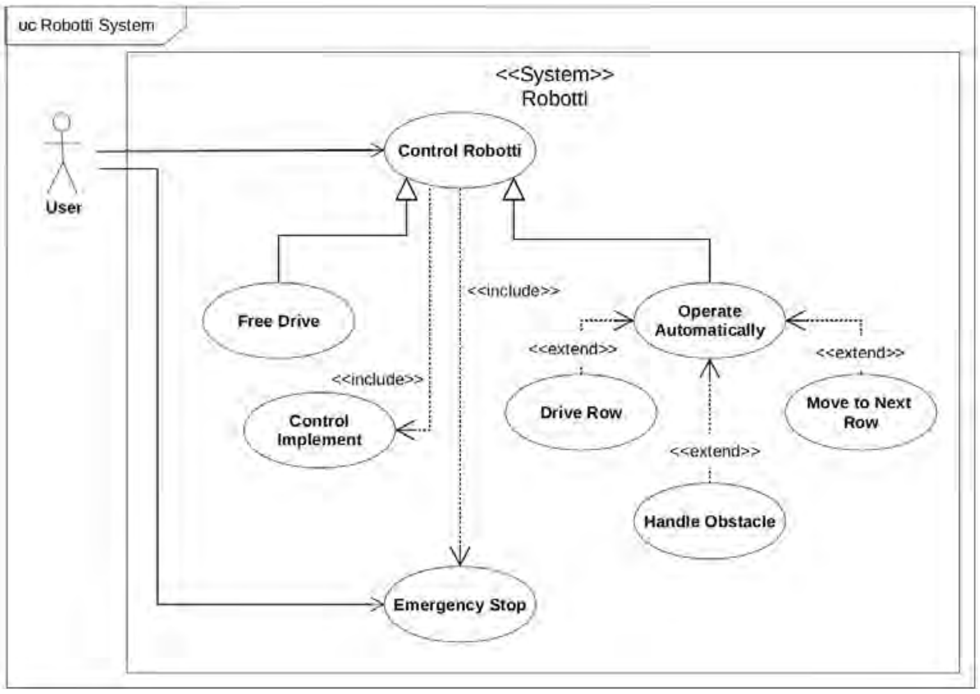


Figure 15: SysML use case diagram giving an overview of the services the robot controller has to provide.

The requirements specified in this section are related to robot navigation and implements safety control. Most of the requirements presented in Agro Intelligence (AI) are functional requirements and are specified as UC. More detailed descriptions of the use cases are provided in the sub-sections below.

Requirement 1: UC Control Robotti (Top Level UC)

- **Description**: The user should be able to control Robotti so it is possible to work the crop-fields. This is a top level use case that is refined in more concrete services offered by the system in the following UC descriptions.

- **Method of Verification**: Model simulation and system testing.

Requirement 2: UC Operate Manually (Free Drive)

- **Description**: The human operator should be able to operate manually Robotti in the crop-fields, being able to steer it and control its speed. The manual mode is denoted as Free Drive, since it allows the human operator to control the robot externally.
- **Method of Verification**: Model simulation and HiL system testing.

Requirement 3: UC Operate Automatically

- **Description**: Robotti should be able to navigate a field independently without requiring the constant input of a user.
- **Method of Verification**: Model simulation and system testing.

Requirement 4: UC Control Implement

- **Description**: It should be possible to activate the implement and control its height if needed. When Robotti operates automatically the implement height can be changed depending on the robot's position in the field.
- **Method of Verification**: Model simulation and system testing.

Requirement 5: UC Drive Row

- **Description**: Robotti should drive along the crop-rows of the field in which it is deployed and stay within them.
- **Method of Verification**: The model of the robot will be simulated taking the effect of disturbances and irregularities of the crop and soil into consideration.

Requirement 6: UC Move to Next Row

- **Description**: Robotti should be able to transit between rows through a turning area. Implements should be lifted/deactivated before starting the turn and lowered/activated when the turn is completed. Dependent on the implement, the robot might need to be paused in its steering and driving, until the operation is completed.

- **Method of Verification**: The model of the robot will be simulated having the robot deployed close to the end of a row.

Requirement 7: UC Handle Obstacle

- **Description**: Robotti should be able to detect obstacles in the row in which it is operating. Examples of obstacles can be a bird nest, a person or a deer.
- **Method of Verification**: The model of the robot will be simulated in different scenarios with obstacles of different sizes placed in different parts of the field.

Requirement 8: UC Emergency Stop

- **Description**: Robotti should be able to perform an emergency stop and execute a predictable stop behaviour. Supply to the motor (current/diesel) will be cut via hardware and a human operator notified. The controllers will stop their current operation and move back to idle mode.
- **Method of Verification**: The model will be simulated through different safety/critical scenarios.

Requirement 9: NF Platform usage

- **Description**: The solution modelled and considered in this case study should target the controller platforms intended to be used on Robotti. These are Linux platform running with the Robot Operating System (ROS) [QCG+09] and B&R Programmable Logic Controllers (PLC)[1].
- **Method of Verification**: Does not apply.

4.3 Modelling

This section describes the modelling that has been done using the INTO-CPS tools during this second year of the project. The modelling tools that have been used for this chapter are, Overture, 20-sim, OpenModelica and Gazebo/ROS. The main focus is the third generation of the Robotti vehicle platform and the different aspect

[1] B&R is a company producing industrial automation controllers`http://www.br-automation.com/en/`

of the on-board sensors and actuators. The reason why Agro Intelligence has chosen to extend the chain with Gazebo/ROS [QCG+09] is explained below.

4.3.1 Gazebo and ROS

Simulation and 3D-visualisation make it possible to rapidly test algorithms, design robots, and perform experimentation and testing using realistically modelled scenarios. Figure 16 illustrates one of the first versions of the next generation of Robotti before the CAD models were made.

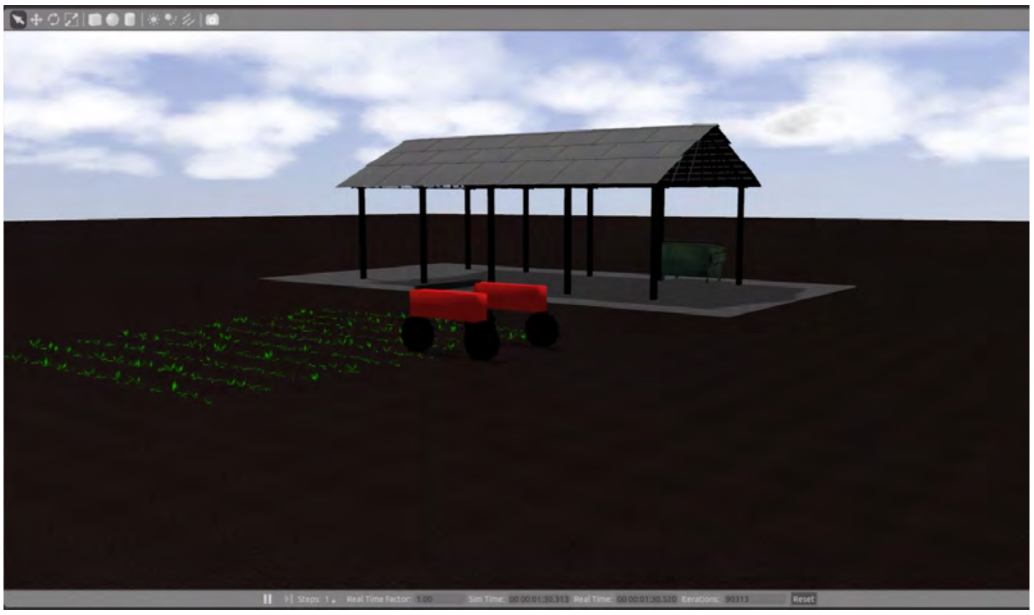


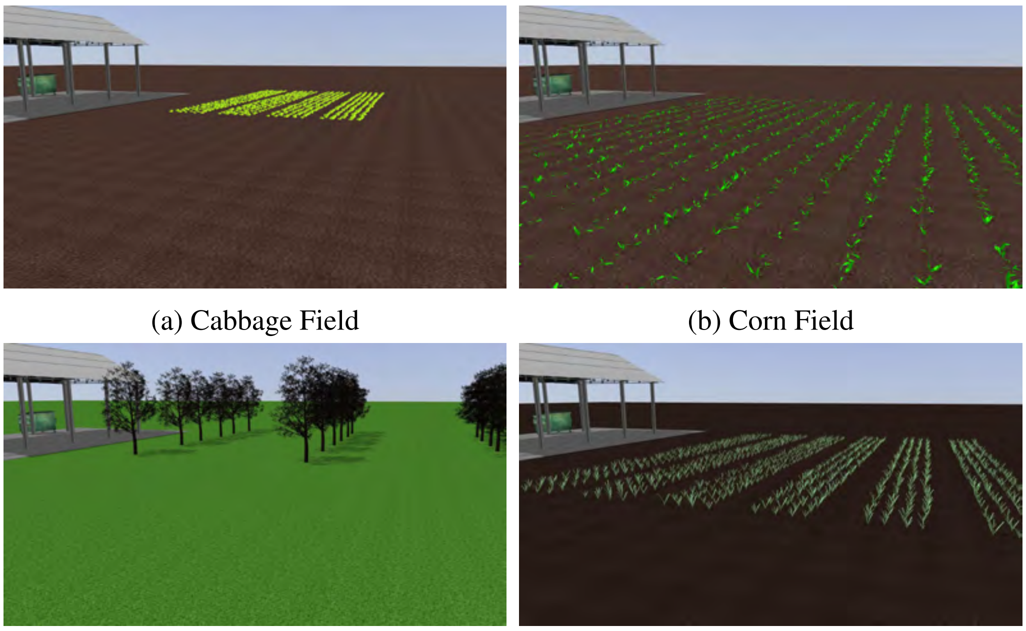
Figure 16: Visualisation of the upcoming third generation of the Robotti platform in Gazebo.

Gazebo offers the ability to accurately and efficiently simulate and visualise robots in indoor and outdoor environments. In this case study, we mainly use Gazebo for visualisation of the simulation result from COE, made with the models from 20-sim, Overture and OpenModelica. This kind of Gazebo visualisation of approach allows our development team to get early feedback from end users with a lesser understanding of all the utilised CPS technologies. Feedback from end users like an onion or cabbage farmer can provide valuable product feedback early in the development process, to ensure the development meets their demands.

Gazebo is mainly used for visualisation as stated in the description above, with the exception of visual sensing and the field environment. Visual sensing can be a camera or a laser-range scanner sensor, which is intended to be used on Robotti. To model a camera system, it needs an actual input of a crop field and obstacles to provide the correct output. By using the already created visualisation from Gazebo, a simulated image can be generated. To connect the INTO-CPS tools with Gazebo, ROS is used to delegate communication back and forth.

In some of AI's other research projects, the Gazebo/ROS combination is also being used by the academic and industrial partners. We have chosen to also follow this path in this case-study, to allow for better inter-collaboration with the other projects.

Creating crop-fields for Robotti The different field scenarios Robotti must operate in is an important part of the case study. The crop type and field structure have a significant impact on how Robotti should perform its task in the field. In Figure 17, examples are found of such crop-fields generated for Robotti to operate in. The simulated fields allow us to evaluate the results from the simulations, and are used as input for the visual sensor as objects in the environment.



(a) Cabbage Field (b) Corn Field

(c) Apple Orchard (d) Onion Field

Figure 17: Generated crop field examples

4.3.2 Robotti Third Generation

The third generation of the robotti platform comes in two different versions, a version with four-wheel steering and one with two-wheel steering as illustrated in Figure 18. Regarding industrial production, the first version currently intended to be put into production is the two-wheel steered version. In both versions each steered wheel can be operated individually to provide a high degree of freedom in driving.

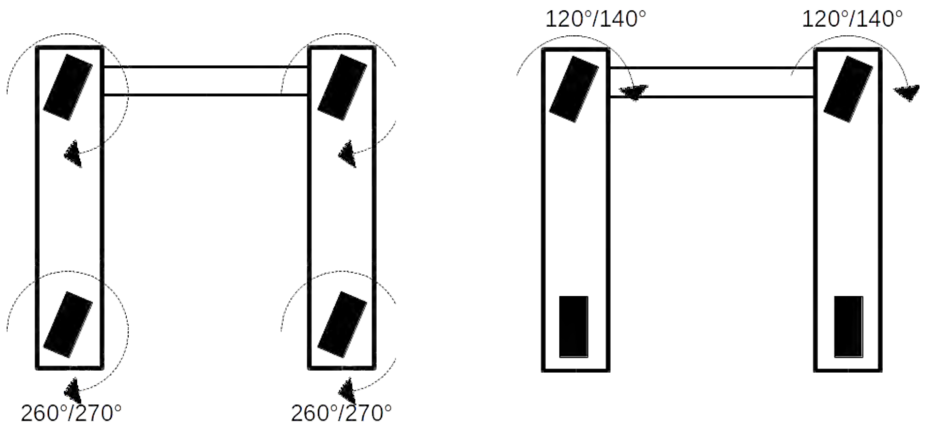


Figure 18: Planned steering freedom in the third generation of Robotti.

The two-wheel steered version is designed to be similar to a tractor and is intended to provide similar functionality with and without a human operator driving the robot. The main difference is that the implement (examples could be cultivator or sprayer) will be mounted inside the frame and not behind.

A vehicle CT model 4.3.2 4.3.2 have been made for the third generation of Robotti. The main reason for the different models is to account for the two types of robots and the progress of exporting capabilities from 20-sim this year. To export the more complex vehicle models, it requires variable step-size solver for FMU to work.

Actuators To steer the wheels on Robotti a hydraulic actuator is used for each wheel as illustrated in Figure 20. The actuator drives the wheel using a rack and pinion gearing system that translates linear motion into rotation.

This steering system has been modelled as a first order transfer function in 20-sim. The limiting block ensures that the response is within the boundaries of the actual actuator. Each wheel is steered individually with a setup as illustrated in figure 19

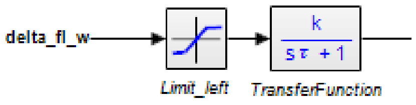


Figure 19: Modelling of the left front steering actuator.

for the front left wheel. This allows us to model the intended time-behavior of the actuator response.

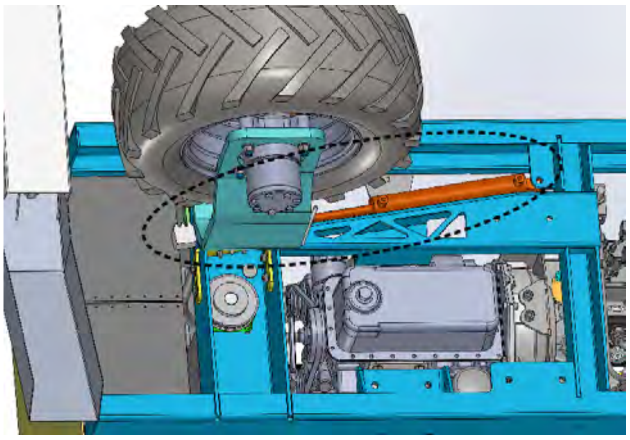


Figure 20: Actuator system used to on the Third generation of Robotti.

Second or higher order systems will be used to model the actuator response in the future in order to provide more realistic responses. Sample data from the actual Robotti will be needed to improve the modelling.

Wheels - Kinematic model The kinematic model is used as a simple first model to describe the transfer from wheel rotation speed (angular velocity) to linear velocity. This model takes the estimated radius of the wheel R_{ee} and converts the rotational speed (ω_{xx}) of the wheel xx into linear velocity V_{xx}. Conversion is performed as illustrated in figure 21 and calculated as:

$$V_{xx} = \omega_{xx} R_{ee} \tag{1}$$

Vehicle Body Dynamics - Bicycle model The CT-model defining the vehicle is a non-linear model with three Degrees of freedom (DOF), i.e., the longitudinal, lateral, and yaw directions, irrespective of the suspension and described in [CLJ15]. The model of the vehicle utilises the bicycle approach, meaning that

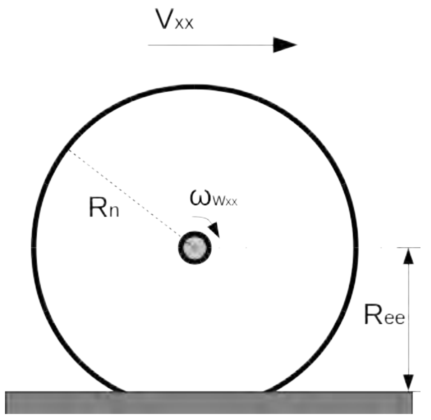


Figure 21: Concepts in the kinematic tyre model

the lateral forces on the left and right wheels are assumed to be equal and summed together. This assumption holds for typical agricultural vehicle operation velocities (<7.5 m/s) [KS10]. The bicycle structure is also known as a half-vehicle (Figure 22). The model allows for yaw and lateral motion through adjustment of the front wheel angle δ_f.

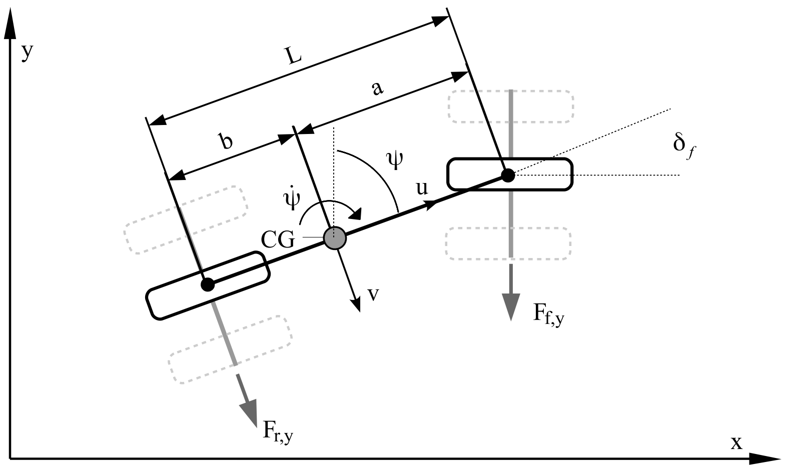


Figure 22: Dynamic bicycle model of the body of the Third Generation of Robotti.

The velocities u, v are at the Center of gravity (CG) of the vehicle. L is the wheelbase, where a is the longitudinal distance to the front wheel, and b is the longitudinal distance to the rear wheel. For a constant forward velocity, the vehicle motion is given by

$$m(\dot{v} + u\dot{\psi}) = F_{f,y}cos(\delta_f) + F_{r,y} \tag{2}$$

where r is the angular rate about the yaw axis. Similarly, the vehicle yaw motion is expressed by

$$I_{zz}\ddot{\psi} = aF_{f,y} - bF_{r,y} \tag{3}$$

where I_{zz} is the moment of inertia along the yaw axis.

Implements In agriculture, implements are the tools that are mounted onto a tractor, e.g. a cultivator or a sprayer. On the current version of Robotti the implement is attached to the vehicle by a three-point linkage, which is the standard way utilised by tractors. How this three-point linkage operates differ from vehicle to vehicle, but the general mechanical design is the same. A first three-point linkage model was created, as illustrated in Figure 23a, to mount different types of implements.

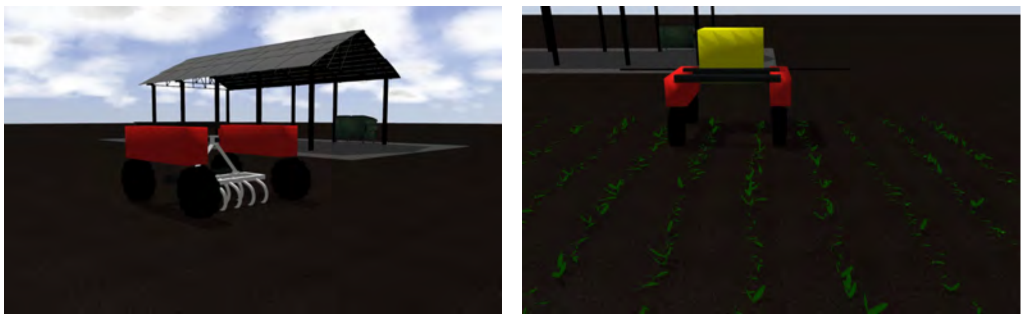

(a) Cultivator modelling - first attempt (b) Sprayer modelling - first attempt

Figure 23: First attempts in adding implements to the Robotti tool career.

A revision of this first CT-model of the connector will be needed when the final version of Robotti has been made. In this second year of INTO-CPS, a sprayer system has been the main focus of Robotti's implements, illustrated in Figure 23b, since it can be used for multiple applications. In its current version, the model of the sprayer assumes all nozzles are driven using the same valve. In later versions, the intention is to make a model where each nozzle can be activated individually, in order to allow spot spraying.

4.4 Robotti Second Generation

The movement of the robot has been studied from the CT side by creating a kinematics model of the vehicle. The 20-sim representation of the system is shown in Figure 24. This model takes the radius of the wheels R_{ee} and the separation between them T_c into consideration, to describe the robot movement. Based on these parameters and on the distances travelled by the wheels, it is possible to determine how the robot moves and what orientation it has.

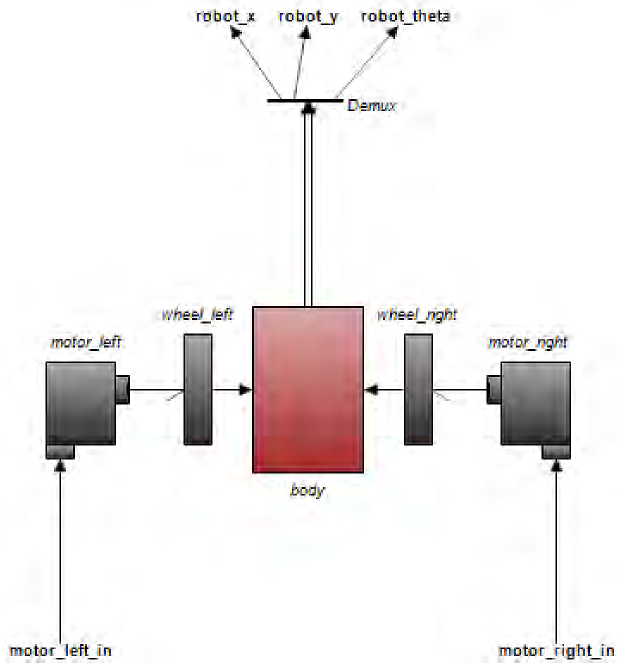


Figure 24: 20-sim model of the Robotti second generation.

The model is symmetric since the construction of the robot is as well, meaning that the mechanics of the left side of the robot are identical to the ones used on the right side.

The robot movement is described through a number of equations. The speed of the robot V_c is considered from the center of the robot, and described in terms of the rotational speeds of the right (ω_r) and left wheels (ω_l), as shown in equation 4 and based on 4.3.2.

$$V_c = \frac{(\omega_r R_{e_{ee}} + \omega_l R_{l_{ee}})}{2} \tag{4}$$

Considering the separation between the wheels T_c, it is also possible to determine the yaw rate of the robot ($\dot{\psi}$), as shown in equation 5. ψ is an angle defined with respect to the XY frame. X and Y are defined with respect to the field.

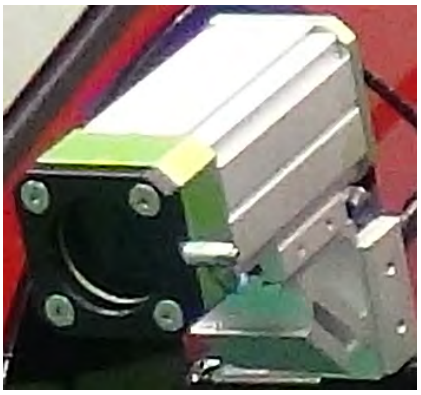
$$\dot{\psi} = \frac{(\omega_r - \omega_l)}{T_c} \tag{5}$$

Finally, based on these calculations it is possible to determine the location of a robot using the equation 6 for the X and Y coordinate axes respectively.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = V_c \begin{bmatrix} \cos(\psi) \\ \sin(\psi) \end{bmatrix} \tag{6}$$

4.4.1 Sensors

Camera row-tracking For the camera sensor, we have two model versions for the crop-row detection, one for using Gazebo and a simpler version using 20-sim.


(a) Basler camera intended to be used for row-tracking on Robotti.

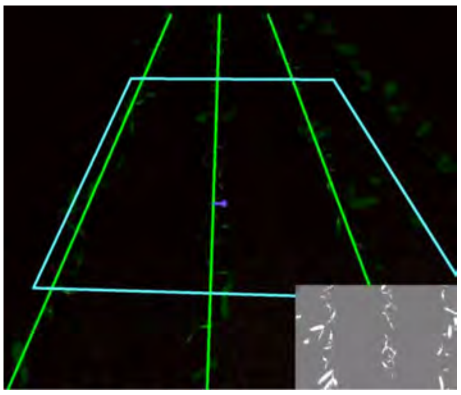

(b) Output from the tracking algorithm with the Gazebo image.

Figure 25: Camera and simulated output.

The detection of the crop-row is done to ensure that Robotti can move safely in the field without harming the individual plants. The Gazebo version utilises a camera image for the crop-row tracking algorithm, which provides an offset and an angle of the rows, as it is illustrated in Figure 25. The 20-sim version provides an output similar to the tracking algorithm, but without the use of an image, as it is based on a map aware of the current robot and crop positions. The intention behind the 20-sim version is to imitate what the actual tracking algorithm would provide, and allow it to run with configurations that are not possible with the algorithm. An example of these configuration parameters could be higher real-time update rates, or improved or degraded detection results.

4.4.2 Encoders

The rotary encoders for the Third Generation Robotti are based on the NovoTechnic RFC-4800 with a CANopen interface. The encoders are used for measuring both speed and wheel angle, and they provide regular updates at each CANopen sync message. Figure 26 illustrates the current version of the model of the rotary encoders.

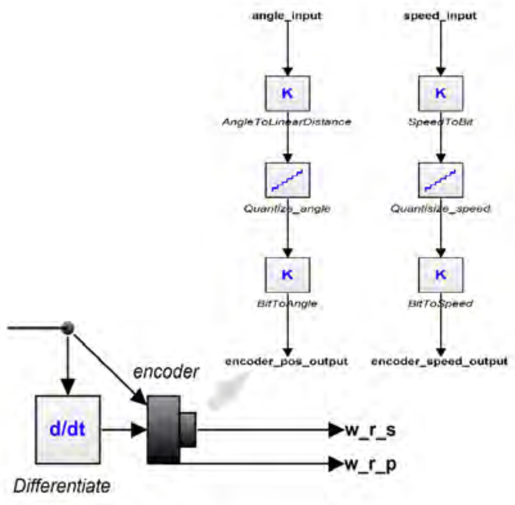


Figure 26: 20-sim encoder model.

The encoder model provides measurements in both speed and angle in absolute terms. When generated, the FMU is currently unable to address the CANopen part of the system, and this part is therefore added when the HIL testing is needed. In the HIL CANopen cases, the back conversion from bit value will not be performed, and the bit values are considered as outputs directly. Since the the encoder model contains both CT and DE elements, it can be categorised as a hybrid model.

4.4.3 Emulated sensor fusion

In most cases where mobile robots are navigating in an environment, information from different sensor sources are combined (fused). This is known as sensor fusion. This approach of localisation by a robot is used to ensure the robot can navigate in an outdoor environment since no single sensor source can provide the necessary measurements.

Making a complete sensor-fusion setup can seem a bit extreme in a modelling context, when it is wanted, for example, to test a navigation algorithm. To simplify the process, a model has been made that emulates the expected behaviour of sensor fusion algorithm using 20-sim. The model can currently be in three different states:

1. Provides no data about the robot's localisation
2. Provides current position but not the direction of the robot
3. Provides current position and the direction of the robot (pose)

For both case 2 and 3, the localisation data comes with an uncertainty estimate.

4.4.4 Controllers

The controllers documented here are designed for the third generation of Robotti, but should in theory also be applicable to the second generation of Robotti, with some minor modifications.

4.4.5 High-level control structure

The state diagram in Figure 27 is used to represent how the robot reacts to different external events, toggling between different operational modes. The manual mode is used for the operational setup, like mounting the implement, testing actuator functionality and driving the robot by wire. For high-level control, the robot always starts in idle mode with all actuators in passive, but all sensors are booted up. Currently, it is only possible to switch to FreeDrive directly, since to robot needs a heading in order to go into AutomaticOperation. The heading determines which way the robot is turned globally, and is used to guide the robot correctly into the field, since both position and heading need to be known. The Emergency Stop occurs when a bumper action, safety button press or an unhandled situation occurs.

4.5 Industrial needs and assessment

The industrial needs have been updated to match the current needs of Agro Intelligence. The description has been focused, and no new industrial needs have been added. An overview of AI industrial needs after year two is presented in Table 4.

Table 4: Summary of Agro Intelligence industrial needs.

Req.	Category	Sub Cat.	Description	Priority	Status	Ver.
AI_1	Functional	Tools	C++ code generation from VDM-RT	M	Partially achieved	2.1
AI_2	Functional	Tools	C-code generation from 20-sim	M	Partially achieved	2.0
AI_3	Functional	Tools	HiL and SiL simulation	M	Partially achieved	2.1
AI_8	Non-Functional	Process	Development Methodology	S	Partially achieved	2.1
AI_9	Functional	Tools	Model snapshot and version control	C	Achieved	2.1
AI_10	Functional	Tools	Simulation results snapshot and version control	C	Partially achieved	2.1
AI_11	Functional	Tools	Gazebo integration	S	Partially achieved	2.1
AI_12	Functional	Tools	3D animation and visualisation	S	Partially achieved	2.1

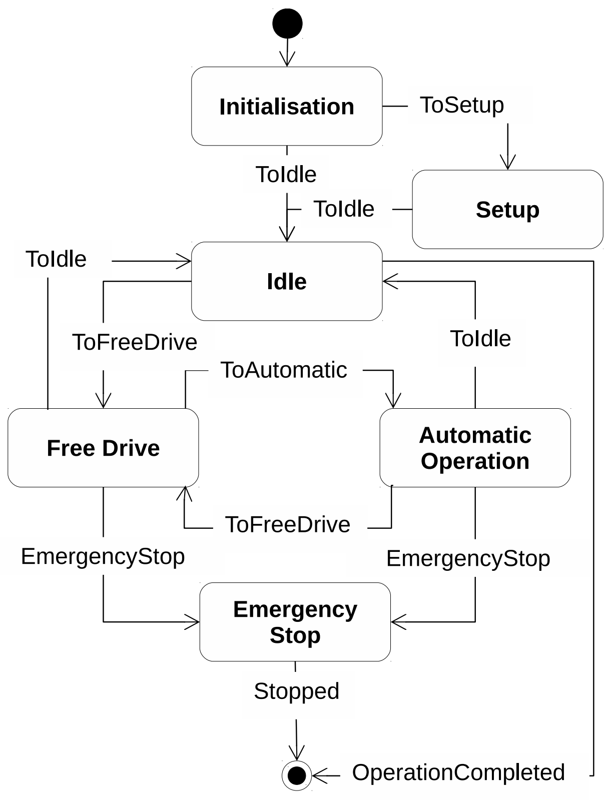


Figure 27: Overal Control statemachine structure in SysML form

The following sections provide a description of the industrial needs for the agricultural case study. These industrial needs are formulated as requirements.

4.5.1 AI_1: Code generation from VDM-RT

- **Description:** The tool must facilitate code generation from VDM-RT models to C++. The target platform for this is embedded Linux and it must run either as a standalone executable or within a ROS node.
- **Method of Verification:** Code generation from models of different control logic of varying complexity. Testing of the generated code in a controlled setup and comparison with expected results based on models. Measurement: percentage of VDM-RT modules translated; suitability as control software.
- **Assessment:** Partially achieved

The code generation support from VDM-RT is close to fully developed for C-code, according to the baseline tool requirements. However, due to prioritisation of the developers of the VDM code generator to support C-code generation in the first place, its functionality is not fully suitable for Agro Intelligence since we need code generation to C++, and C++ code cannot run in C-code environment. As AI has a different development time schedule than in the INTO-CPS requirements, i.e. before the code generator was required to be ready, we had to code in both C++ and VDM manually. AI has provided an example VDM-RT project for our target platform (embedded Linux with ROS) that can be used as a first step to validate when a C++ code generator is ready for testing.

- **Related baseline tools requirements:** 0038
- **Degree of achievement:** 75%

4.5.2 AI_2: Code generation from 20-sim

- **Description:** The tool must facilitate code generation from 20-sim to C/C++ software. The target platform for this is embedded Linux, and it must run either as a standalone executable or within a ROS node.
- **Method of Verification:** Code generation from models of different control logic of varying complexity. Testing of the generated code in a controlled setup and comparison with expected results based on models. Measurement: percentage of 20-sim moduls translated; suitability as plant and control software.
- **Assessment:** Partially achieved.
  20-sim models can be exported into C-code and FMU units for fixed step-size ODE solvers (Euler, Runge-Kutta 2, Runge-Kutta 4), that can be executed on Windows- and Linux-based computer units. The variable step-size solve has been implemented to allow the C-code generation to match the response in 20-sim. The Backward differentiation formula (BDF) solver is currently the only tool able to support some of the more vehicle models. This functionality is still needed to be implemented for code generation.
- **Related baseline tools requirements:** 0038, 0039, 0040
- **Degree of achievement:** 65%

4.5.3 AI_3: HiL and SiL simulation

- **Description:** The tool must facilitate Hardware-in-the-Loop and Software-in-the-Loop simulation of 20-Sim and VDM models.
- **Method of Verification:** Co-execution of different models of increasing complexity with Hardware and Software realisations. Measurement: Yes/No for HiL and SiL; Percentage of simulated cases (from total of "pure" model simulations).
- **Assessment:** Partially achieved.
  At the writing of this chapter, only 20-sim and OpenModelica can provide export of code that can be used for HiL and SiL testing. HiL is done via the FMUs and implemented manually on the particular Operating system and controller.
- **Related baseline tools requirements:** 0042, 0043, 0084, 0086, 0087, 0088
- **Degree of achievement:** 30%

4.5.4 AI_8: Methodology for the development of embedded real-time systems.

- **Description:** The project must provide methodological guidelines to apply a model-based engineering approach using the tools from INTO-CPS to the development of embedded real-time systems that constitute Cyber-Physical Systems.
- **Method of Verification:** Provided/Not provided. Applicability assessed by using the tools in the case study. Measurement: Weighed percentage of activities not applicable and activities missing in proposed methodology.
- **Assessment:** Partially achieved.

  The methodological guidelines for the development of embedded real-time systems are in general terms achieved, but are lacking some specific examples, which are relevant for mobile robots, such as the Robotti case study. We think these should be included in requirement 0084 with deadline in Y3. It has been challenging to adapt them to our setup on Linux/ROS platform, and specific guidelines for mobile robots and their interaction with the environment. For example, we would have liked to have more specific guidelines for data exchange for GIS, and for sensor data transmission of GNSS, Camera, IMU or Laser-range scanner, which are important for a CPS technology like a mobile robot. A different development time schedule between AI and the INTO-CPS requirements, caused the mismatch between our needs and the available guidelines.

- **Related baseline tools requirements:** 0070 - 0076, 0084
- **Degree of achievement:** 60%

4.5.5 AI_9: Model snapshot and version control

- **Description:** The tool-chain should provide a way to keep track of the different versions of the models that compose a co-model.
- **Method of Verification:** Provided/Not provided. Applicability assessed by using the tools in the case study. Measurement: Yes/No. Degree of support in multi-simulation setup.
- **Assessment:** Achieved.
  The way the INTO-CPS COE operates allows the FMU models to be distributed in sub-folders, that allow for tools like git and svn the ability to version control multi-models for the case studies. The same goes for the models made within the tools 20-sim, Overture and OpenModelica. A challenge still is to do version control directly on 20-sim *.emx files, since they can represent visual blocks.
- **Related baseline tools requirements:** none.
- **Degree of achievement:** 99%

4.5.6 AI_10: Simulation results snapshot and version control

- **Description:** The tool should provide a way to keep track of the different simulation results of a co-model. This could enable that at any point of time a certain set of simulation results can be linked to a concrete version of the models as well as the parameters used for the simulation.
- **Method of Verification:** Provided/Not provided. Applicability assessed by using the tools in the case study. Measurement: Yes/No. Degree of support in multi-simulation setup.
- **Assessment:** Partially achieved.
  When a simulation is executed via the COE, the exchange variables are

stored in a time-stamp log file. A challenge that is still present is the ability to match git/svn version with the simulation result.

- **Related baseline tools requirements:** none.
- **Degree of achievement:** 50%

4.5.7 AI_11: Gazebo integration

- **Description:** The project could enable the co-simulation of high-level control models with Robot models running in the Gazebo environment.
- **Method of Verification:** Co-simulation of models of different complexity. Reading of sensors and control of actuators that are deployed in Gazebo. Interaction with the Gazebo simulated world. Measurement: Yes/No. Degree of support in simulation/multi-simulation setup.
- **Assessment:** Partially achieved.
  It is possible to interface with Gazebo using the interface Agro Intelligence has made 4.3.1. A direct Gazebo FMU interface to the COE is still missing for full completion.
- **Related baseline tools requirements:** 0001 - 0006
- **Degree of achievement:** 50%

4.5.8 AI_12: 3D animation/visualisation

- **Description:** The project should enable the visualisation of Robotti models. This will help to understand complex behaviour expressed in the model and facilitate communication in multi-disciplinary engineering teams.
- **Method of Verification:** Provided/Not provided. Visualisation of co-simulation models. Ease of use. Quality of visualisation. Measurement: Yes/No. Weighed: Degree of support in simulation/multi-simulation setup; Time to create visualisation compared to "conventional" tool; Acceptance by engineers.
- **Assessment:** Partially achieved.
- **Related baseline tools requirements:** 0093
- **Degree of achievement:** 75%

5 Building Case Study

In this section we provide the public version of the building case which provides a generic overview of the Year 2 case study of an HVAC Cyber-Physical System. A more detailed description can be found in the confidential version.

5.1 Introduction

The Year 2 case study contains a description of the public the building scenario developed by UTRC, the design of the HVAC infrastructure the modeling of the involved Cyber-physical components and the results generated after the co-simulation using the INTO-CPS platform. The HVAC scenarios studied, will include a Cyber-physical system surrounding a building area of a floor, consisting of two rooms and one zone composed by two areas. Each room temperature will be controller by a Fan Coil Unit (FCU). The zone temperature will be controlled by two collaborated FCUs, each one for one of the two areas of the zone. The FCU will consist of several vital elements that will control the fresh air temperature provided by the Air Handling Unit (AHU) to the room using air dumper, fan, coil and valves. The water that will heat or cool the air will be pressurized and provided to each FCU by the Heat Pump Unit (HPU), currently not included in the CT model. The described scenario will be modeled using the INTO-CPS baseline tools as well as state-of-the-practice commercial tools and evaluated entirely in the INTO-CPS platform. After descriptions of SysML modeling, CT and DE models developed, we discuss the co-simulation results taken from the current version of the INTO-CPS engine for a series of controller scenarios. Finally we conclude with FMI co-simulation experiences and next steps towards year 3 use case.

5.2 Building Case Study

The building case study will focus on modeling and analysis of energy and comfort systems that control the temperature of connected rooms or areas inside a building premise as shown in figure 28. This Year 2 case study models encapsulate several modules including a) Physical rooms and air flow modeling, b) Fan Coil Units, c) Supervision of the FCUs d) Communication interfaces between master-slave FCUs, e) Communication between FCUs and Supervisor, f) Air-pipe connections between FCUs and AHU, g) Water-pipe connections between FCUs and HPU, h) Air-Handling Unit controller and i) Heat Pump Unit load. Results have been generated from the newly developed Continuous and Discrete models for Year 2 that are based on commercial HVAC products and control requirements.

The current control strategy will compose a multi-model that will regulate FCU operation, Supervision of FCUs, AHU operation and overall HVAC functionality. User inputs will be taken into account from rooms or zone thermostats and will compared with current Room Air Temperature (RAT) sensed by the FCUs, triggering certain actions to the FCU(s). The FCU(s) having a direct connection with a supervisor device, will take a series of actions to reach the desired temperature by a) regulating the air flow using its fan and the air-dumper, b) power up the coil to heat the air if necessary, c) regulate the water pipe valves to control the cooled or heated air originated by the AHU, d) regulate the water pressure inside the water pipes originated by the HPU, e) synchronize with the supervisor to coordinate with the rest of the FCUs and respectively -if needed- with the AHU and the HPU.

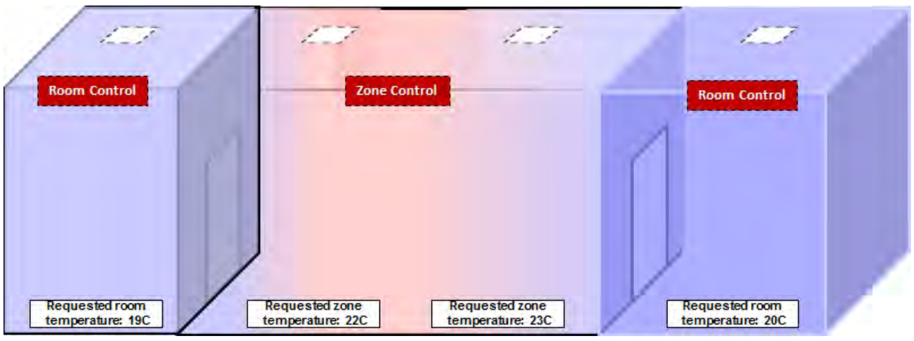

Room Control
Zone Control
Room Control
Requested room temperature: 19C
Requested zone temperature: 22C
Requested zone temperature: 23C
Requested room temperature: 20C

Figure 28: Rooms and Zone level schematic for the Year 2 building case study

The main objective for the Year 2 case study is to respect high level requirements that span from temperature control of rooms and the zone , energy consumption and safe operation of all of the devices that are involved in the HVAC system. One of the main reasons why the aforementioned fact remains a challenge, is the product line engineering approach currently followed by building automation industries. In the current work-flow, different types of engineers contributing to the creation of the same device are involved and affecting respectively the system design. Verification of the generated models or code is enabled in stages of the product life-cycle that leave the system open to delays due to late error discovery. To this end, INTO-CPS and co-simulation solutions will not only bridge the identified gap between engineers but also, enable verification of the output models early in the design phase, thus, rapidly increasing the product life-cycle while respecting system requirements.

5.3 Modeling

Bridging between logical and physical models is performed through the control model, where a control strategy should be developed to regulate HVAC equipment according to the indoor temperature response, the user selected settings and the current operation of the devices. The control model optimizes the building performance through maximizing user comfort, while minimizing energy consumption. User comfort maintains the comfort characteristics (e.g. indoor temperature, CO_2, humidity) in the standardized comfort band.

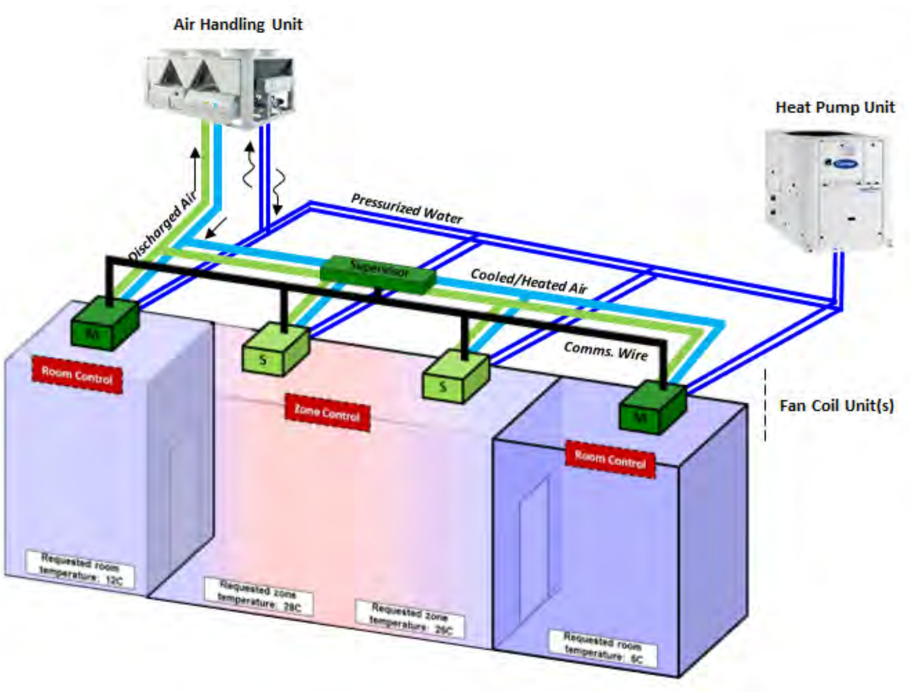


Figure 29: Cyber-Physical schematic of Year 2 building case study including FCUs, communications with supervisor, air exchange between FCUs and AHU, and water exchange between HPU, AHU and FCUs

For example, if a user selects a specific room temperature that differs from the sensed RAT that an FCU reads, FCU will try to meet the selected temperature by switching on its coil. But if the neighbor the FCU had requested a temperature point that required additional water pressure from the Heat Pump Unit to heat the air, then the initial FCU could avoid switching the coil on as it can regulate water valves to allow more (already heated) water to be driven inside it while closing valves in the second FCU through the supervisor. Such a decision will increase energy savings, as the the coil will not be switched on and the Heat Pump will not require to be activated through a water rerouting solution.

Design Modeling

In this section initial requirements are described according to core functionality described for the FCUs unit, the supervisor and communication network, the AHU unit and the system as a whole. At the current level of the building use case, we focus on the assessment of a subset of those requirements, as our main interest for the Year 2 is to evaluate INTO-CPS platform usage.

The SysML design model defines the architecture of the CPS at a high level. From this high level description, we derive the specifications for the FMUs that compose the multi-model of the system and the connections between them. The FMU specifications are satisfied by the constituent models elaborated using the Overture VDM and Dymola Modelica technologies.

The structural view of the building case study system is shown in Figure 30. The system model is decomposed into a CT plant and a DE controller. These two entities form the primary FMUs of the system. In our work, we have discovered that centralising all DE control in a single FMU is beneficial since it enables the use of powerful VDM-RT communication abstractions. Otherwise, communication would have to be done through manual encoding and decoding of strings sent across FMU ports. On the CT side, we have also realised that usage of a single Dymola model improves performance and makes the simulation less vulnerable to step size issues. We could have partitioned the Dymola model into some arbitrary set of FMUs but that would not provide any benefit at this time.

The plant and controller are further decomposed into their most relevant components. While this decomposition is not necessary to construct the constituent models and run co-simulations, it is important to properly document all the relevant entities in the system to ensure consistent use of terminology and to facilitate team coordination and communication. Of these components, we draw attention to the FCU, which is shared between the Plant and the Controller. Indeed, the FCU lies at the boundary between the CT and DE worlds – the system is inherently a hybrid system. As such, part of the FCU is modeled in Dymola and part is modeled in Overture – the two models will be connected via FMI. Finally, since the system must interact with human users, we also include an FMU to abstractly represent the behaviour of a system user, although it is primarily used for the

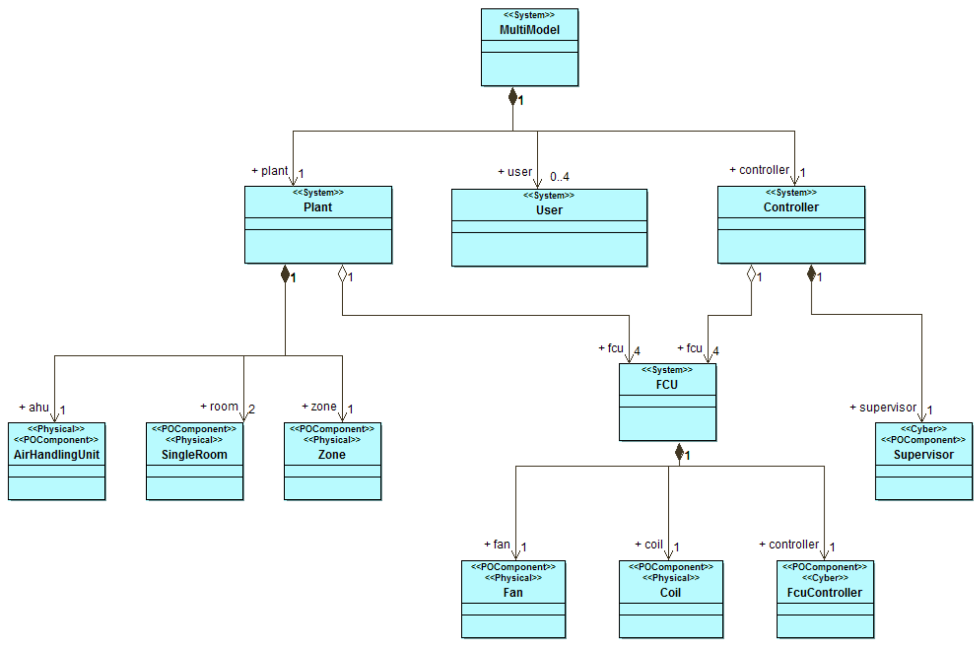

<<System>> MultiModel
+ plant 1
+ user 0..4
+ controller 1
<<System>> Plant
<<System>> User
<<System>> Controller
+ fcu 4
+ fcu 4
<<System>> FCU
+ ahu 1
+ room 2
+ zone 1
+ supervisor 1
<<Physical>> <<POComponent>> AirHandlingUnit
<<POComponent>> <<Physical>> SingleRoom
<<POComponent>> <<Physical>> Zone
<<Cyber>> <<POComponent>> Supervisor
+ fan 1
+ coil 1
+ controller 1
<<POComponent>> <<Physical>> Fan
<<POComponent>> <<Physical>> Coil
<<POComponent>> <<Cyber>> FcuController

Figure 30: Structural view of the building case study model, realised as a SysML INTO-CPS Architectural Structure diagram.

purpose of simulating different scenarios.

The connections between the FMUs in the multi-model are displayed in Figure 31. As the diagram implies, the topology of the system is static. Indeed, this is already hinted at in Figure 30, where the cardinality of all associations is a fixed number. This is because the INTO-CPS tool chain does not support dynamic specification of FMU connections – all connections between all FMU instances have to be explicitly specified a priori.

The building case study model contains 4 FCUs, therefore the connections diagram contains 4 instances of the same set of connections. The connections are as follows:

ValveOpening enables the controller to set the opening of the FCU valve in the plant model.

FanSpeed enables the controller to set the operating speed of the FCU fan in the plant model. Together with the valve opening, this connection enables the controller to affect the physical world.

RoomTemp enables the plant model to communicate current temperature in the room to the controller. This signal provides an abstract model of a tempera-

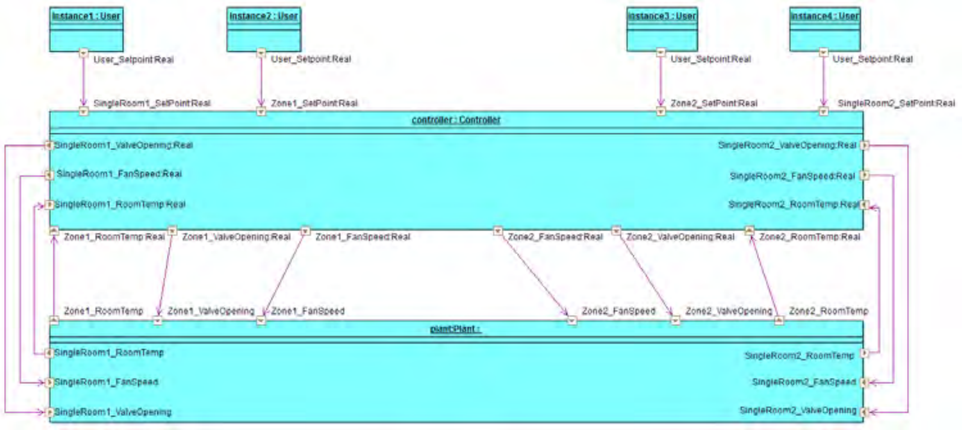


Figure 31: FMU Connections in the building case study , realised as a SysML INTO-CPS Connections diagram.

ture sensor – it can be further refined inside one or both constituent models.

SetPoint enables an outside entity to define the set point to be targeted by the PI controller of the FCU. This allows a building occupant to control the desired temperature of the system.

Discrete Event Modelling with VDM

The VDM model represents the discrete behaviour of the system. It represents the individual FCU Controllers, the overall Supervisor and the communication between them.

The primary task of the FCU controller is to run a PI loop to enable the FCU to control the room temperature. The PI controller was originally designed and tuned in Dymola and was then discretised and inserted in the VDM model. PI controller implementation with imperative languages is well established so our approach here was straightforward. In addition to running the PI loop, the FCU controllers support a Master/Slave behaviour. Broadly speaking, the Master overrides the set point and measured temperature of its slaves to ensure more consistent control over a zone.

We model the Master overrides by using indirection on the data acquisition steps of the PI loop. The Master itself has an additional loop over its slaves that it uses to override the aforementioned values. The Master override loop occurs less frequently than the main PI loop (which a Master FCU also carries out) so some time tracking is necessary. The relevant operations for Master and Slave, shown in Listing 1, are:

```
public acquireMeasuredTemp :() ==> real
acquireMeasuredTemp() == (
  if role=<MASTER> then
    return getTempValue()
  else
    return masterMV;
);

public acquireSetPoint : () ==> real
acquireSetPoint() == (
  if role=<SLAVE> then
    return masterSP;
  if superSP <> nil then
    return superSP;
  return setpoint.getValue();
);

public updateSlaves : () ==> ()
updateSlaves() == (
    for all s in set slaves do(
      s.setMasterMV(temp.getValue());
      s.setMasterSP(setpoint.getValue());
    )

);
```

Listing 1: FCU Controller operations that encode Master and Slave behavior.

To enable the Supervisor activity, the FCU Controllers expose an API for the Supervisor so that it may control them using a higher level set of instructions. Part of this behaviour is also encoded directly in the FCU Controllers as can be seen in the `acquireSetPoint` operation. The API exposed to the supervisor is shown in Listing 2

```
public setSuperSetPoint : real ==> ();

public promoteToMaster : set of Controller ==> ();

public demoteToSlave : () ==> ();

public turnOff : () ==> ();

public getRole : () ==> Role

public getSlaves : () ==> set of Controller

public getSPValue : () ==> real
```

Listing 2: FCU Controller operations that form the API exposed to the Supervisor.

CT Model of Building and HVAC System The building and HVAC model implemented for the year 2 case study describes a $140m^2$ floor in an office building supplied by a hydronic system in heating mode. The HVAC system is made of a HP providing hot water to the FCUs in the occupied space and to the AHU responsible for maintaining fresh air requirements in the zones. The floor is supplied by four FCUs; one for each individual room and two complementing each other to serve a large zone. The thermal performance of the building is reasonable leading to a thermal demand of about 50 kWh/m^2/year. This performance is based on typical northern Europe climate conditions.

Modeling the complexity of the integrated system was not possible in OpenModelica (OM). Fluid flow and heat transfer phenomena induced by the system components are not currently supported by OM. Instead the project team used Dymola 2016, which supports FMI 2.0. Dymola is commercial tool, based on Modelica, for modeling and simulation of integrated and complex systems. It is a multi-engineering tool widely used in automotive, energy systems, and aerospace applications. A schematic overview of the CT model is illustrated in Figure **??**. These are complex models that take into account solar radiation and occupancy levels in the building and maintain acceptable levels of water and air pressures as per manufacturer specifications. The model has 8 inputs being the fan speed and valve opening position of each of the four FCUs and 4 outputs being the actual room temperature of each room and area measured at the FCU level.

The indoor temperature in each room is controlled using an FCU device, where the FCU uses a water coil and fans to heat or cool the circulated air inside each room. The air Handling Unit (AHU) supplies fresh air to each FCU, while the Heat Pump (HP) supplies hot or cold water to the water coils in the FCU. The

Table 5: Statistics of CT FMU generated from Dymola

Item	**Statistics**
Constant	7373 scalars
Free Parameters	6101 scalars
Parameters depending	5738 scalars
Inputs	8 scalars
Outputs	4 scalars
Continuous time-states	300 scalars
Time-varying variables	4635 scalars

water that flows into the coil is pressurized by the HP, whereas a Fan blows the circulated room air through the coil to regulate the indoor air temperature. An FCU PI-controller regulates the fan speed and the rate of the water flow from the heat pump to the coil in order to maintain a set temperature in the room in which the FCU is located.

The model is based on mass and energy balances in a given room/zone. Two major assumptions were used to simplify the model for a given zone:

- The zone air is well mixed at all times
- Long wave radiation exchange between surfaces is ignored

The model is compiled in Dymola 2016 using FMI export and selecting the following options: model exchange including source code and 64-bit binaries. This is the only export option that is currently compatible with the INTO-CPS COE. The complexity of the generated FMU is summarized in Table 5.

Code Generation Proceeding to code generation, after incorporating the necessary tools required for the targeted hardware platform (e.g. the distributed FCU units and the supervisor), the code will be generated based upon the developed models as described in the previous sections. Those will describe the overall behavior of the FCUs, communications and supervision of them. For the targeted 3 FCUs mainly, for the 2 areas in the zone and room 2, the embedded device will be based on a 16-bit PIC24FJ embedded micro-controller. In order to excite advanced controller and prognostics capabilities for several FCU operations (e.g. controlling the Fan motor), we incorporate the Zynq Platform motor controller for room 1. Zynq 7000 consists of 2 A9 ARM processors and a XilinX FPGA. Finally the supervisor will be flashed to the Zybo platform (similar to Zynq) which for now is handling FCU set points provided to the rooms. Connections from the 3 PICs and the Zynq platform with the Zybo Supervisor are being handled through UART cable. The year 2 current hardware setup is shown in Figure 32.

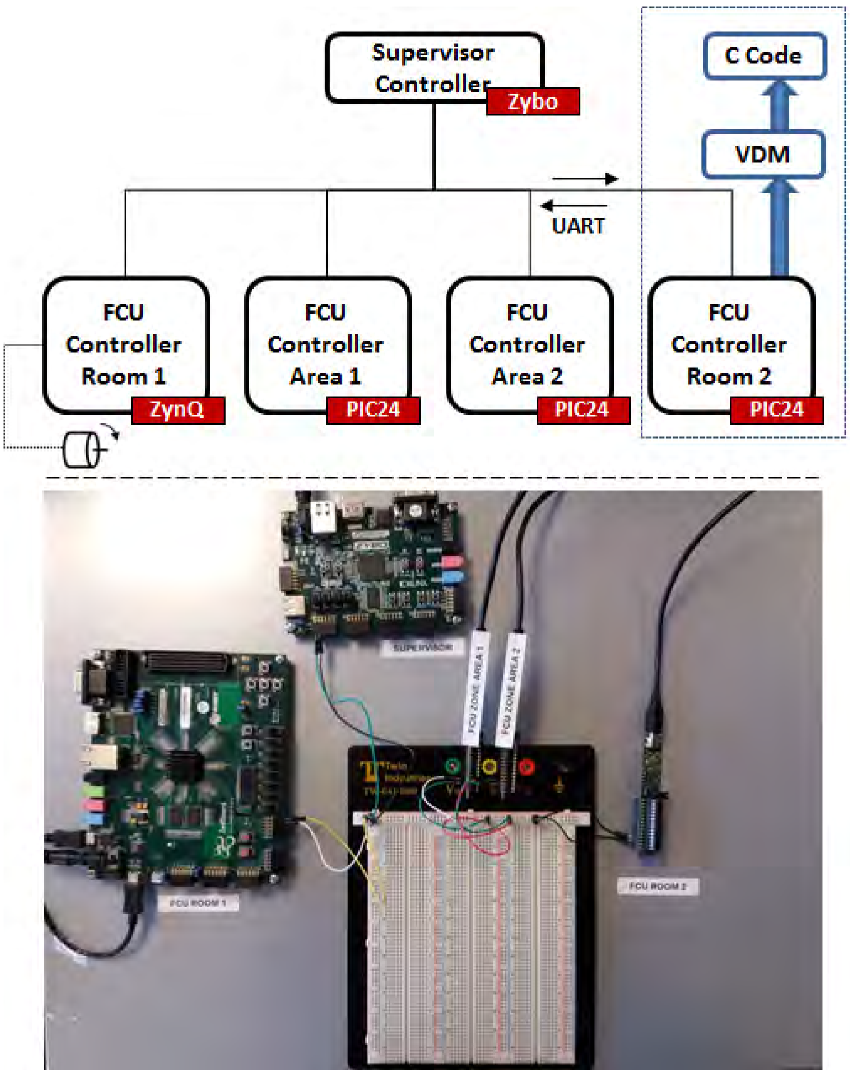


Figure 32: Hardware Setup for Year 2 case study towards code generation

We have extracted a proof of concept version of the PI controller regulating the fan and valve modules of the FCU, based on the sensed room air temperature. For the room 2 instance, as shown in Figure 32 a proof of concept PI controller modeled in Overture using VDM (example taken from year 1 use case) has generated the C code for the 16-bit PIC embedded micro-controller. Code has been refined in order to meet hardware platform characteristics of the PIC architecture. Compilation and flashing of the C code to the micro-controller was successful, executing basic controller commands. Currently automated code generation for the distributed case is not supported. Results from the code compilation and flashing to the PIC micro-controller have accumulated an 8.6% (706 bytes) usage of data memory utilized by the PID controller, while the program data used to realize the controller operations is accumulated at 38.1% (50049 bytes).

5.4 HVAC Co-simulation Results

In this section we describe co-simulation results for several HVAC scenarios for the building case study, based on the INTO-CPS platform. Our focus will be on the design and evaluation of the control algorithms deployed both on the Fan Coil Units (FCUs), the Supervisor and the Air Handling Unit (AHU) for the Year 2 case study. Co-simulation will based on FMUs generated from the models described in the previous sections; we define different scenarios for which we configure our co-simulation parameters in order to evaluate control functionality of the HVAC system as a whole. Table 6 outlines the co-simulation experimental setup for the different scenarios evaluated within our Year 2 models.

Table 6: Scenario Experimental Setup: Control Operation in Variable Temperature conditions

FMUs	2 FMUs with 4 connections (1 encrypted)	INTO-CPS
Connections	inputs, outputs	JSON file created by Modelio
DE	1 FMU describing functionality of 4 FCUs and Supervisor	Overture
CT	1 FMU describing functionality of rooms and AHU	DYMOLA
Total Experiment	Variable Step Size [0.5 - 60]: 4000 sec	INTO-CPS
Scenario 1	Environmental conditions (OAT) remain unchanged	
Scenario 2	Outside Air Temperature (OAT) Decreases	
Scenario 3	Outside Air Temperature (OAT) Increases	

Scenario 1: Evaluate Controls Operation in normal conditions

For scenario 1 the CT and DE FMUs are co-simulated in normal temperature conditions. In this case, the thermal effects of room materials, air-mass flow characteristics and external environmental conditions are mapped by the CT and DE representations as described in previous sections.

Figure 33 shows the behavior of different controlled variables of the model, such as room air temperature (RAT), of all the 4 rooms defined in our use case. The room air temperature set point is considered constant at 294.5K (21.5 ^{o}C) and communicated by the CT model to each FCU controller. Each FCU controller regulates the valve position and fan speed to manipulate the Entering Water Temperature (EWT) and Supplied Air Temperature (SAT), respectively for each room. We observe the controls operation leading to the steady temperature increase from

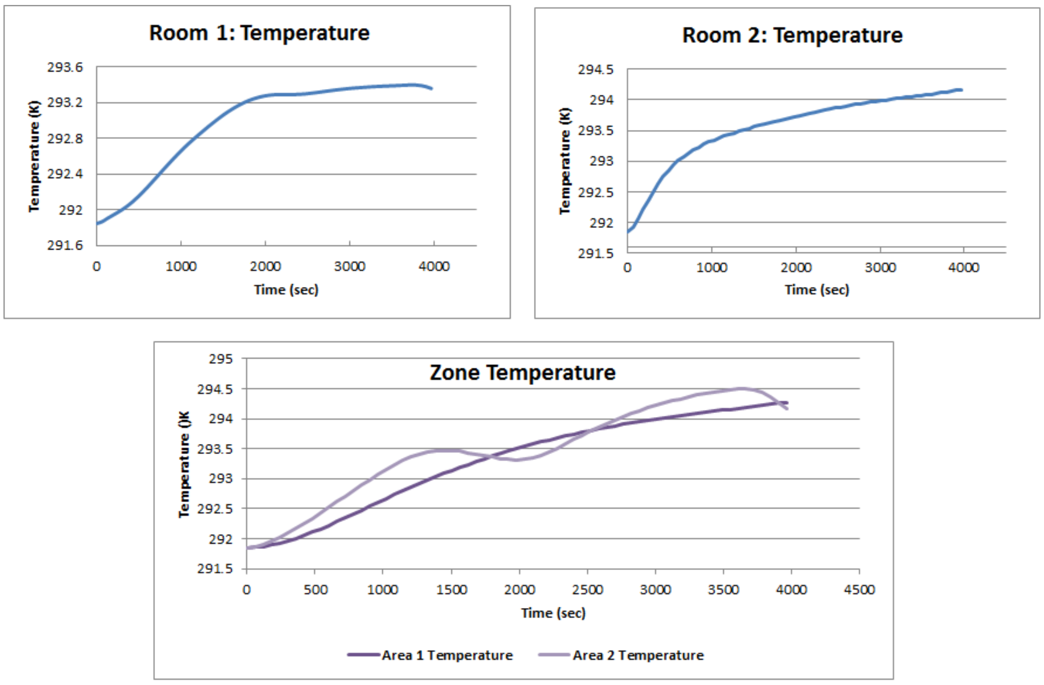


Figure 33: Scenario 1 INTO-CPS Co-simulation Results for Room 1, Room 2 and Zone (Area 1 and Area 2): Normal Operation

the initial value. Both rooms and the zone temperatures set-points are not reached justifying the continuous increase of the temperature for the whole duration of the co-simulation, shown in Figure 33.

Scenario 2: Evaluate Control Robustness when temperature values radically decrease due to environmental changes

The co-simulation results allow for visualization and post processing of various parameters of interest in building applications such as room air temperature, wall temperature, FCU valve opening, FCU fan speed, and controller output. These parameters can be used to calculate a number of indexing quantities to evaluate building performance that for this normal scenario, are comfort and energy consumption. That is the reason why the PI controller – modeled in VDM-RT – tries to reduce the distance between the initial room temperature set point and the real temperature (RAT) by regulating valves (for heated/cooled water flow) and Fan (warm/cold air), until the distance between them equals zero.

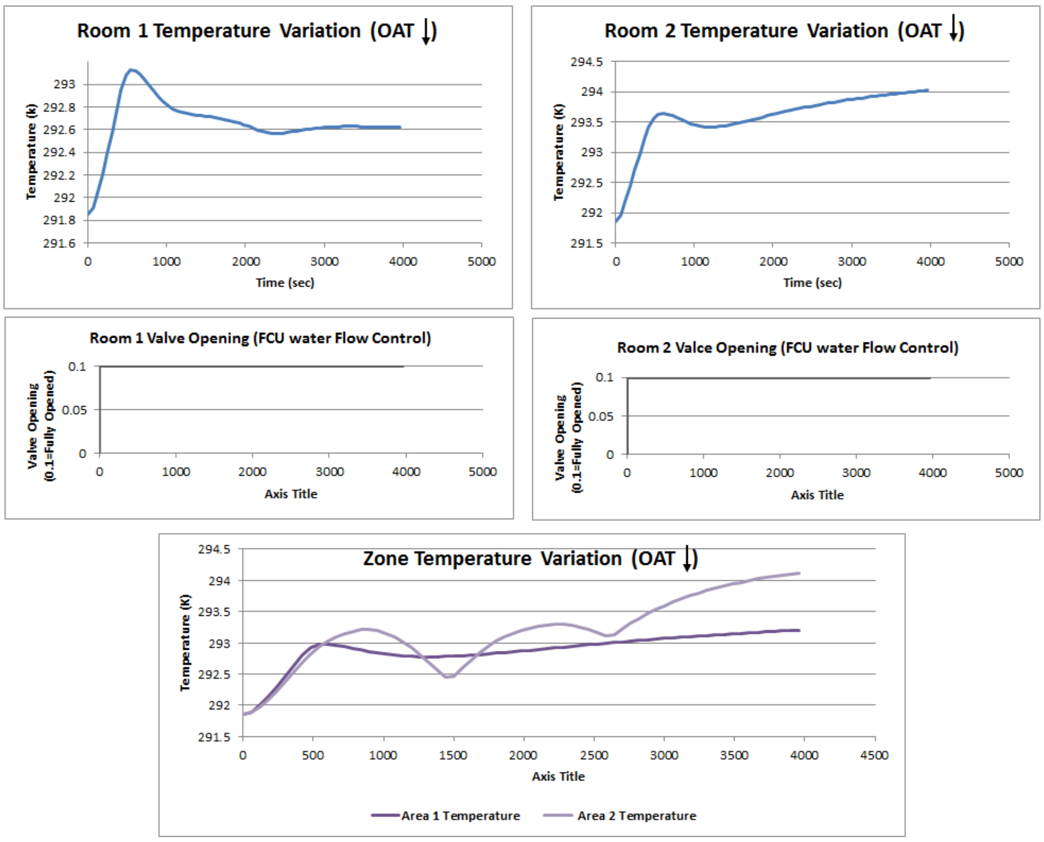


Figure 34: Scenario 2 INTO-CPS Co-simulation Results for Room 1, Room 2 and Zone (Area 1 and Area 2): OAT decreases

Figure 34 shows the temperature when outside air temperature is radically changed (OAT ↓) for Scenario 2. We evaluate the controller robustness when the building is exposed to sudden changes in the outside air temperature (OAT). In particular the OAT was subject to a step function decrease of 20 ^oC from its normal value of 5 ^oC. Figure 34 depicts the building response to this sudden change. Presented variables are the room air temperatures and valve openings.

5.5 Industrial Needs and Assessment

In this section we proceed with the second assessment of the requirements as shown in Table 7 evaluating the INTO-CPS platform as described above. Initial requirements are defined according to core functionality for the Year 2 use case.

5.6 UTRC-Req-001: Simulation Completeness

- **Description**
  INTO-CPS baseline tools must offer simulation completeness (for continuous, discrete and co-models) based on the allowable modeling parameters selected by the end-user. Completeness will be assessed with respect to co-simulation platform functionality (crashes, logging and run-time errors) during execution of co-simulations.
- **Related to Baseline Tools Requirements**
  Requirements 0003-0005 are related because the modeling of the time step could be done at discrete, continuous and co-simulation level
- **Method of Verification**
  For all co-simulations executed by UTRC stuff, we successfully co-simulated thermal, air and physical components through FMUs extracted by Dymola. Discrete FCU controllers were modeled in Overture from which FMUs also have been extracted. Co-simulation results are considered to be complete.
- **Assessment:** Achieved
  In all of our models in the Year 2 case study, the simulation completeness was shown using the INTO-CPS platform. Indicator: 100%

5.7 UTRC-Req-002: Simulation Delays

- **Description**
  INTO-CPS platform should execute simulations by providing extra time delays at minimum. With extra time delays we describe simulation executions where simulation solvers might exhibit various delays into acquiring results.
- **Related to Baseline Tools Requirements**
  Requirements 0003-0005 are related because the modeling of time step could be done at discrete, continuous and co-simulation level
- **Method of Verification**
  For the Year 2 case study and the CT and DE models, simulation delays were negligible for all the co-simulations performed. Some delays occurred when selecting a fixed point co-simulation step size which has been reported to the project consortium
- **Assessment:** Achieved
  In all of our models in the building case study, no simulation delays were

encountered when executing co-simulations using FMUs from both Dymola and Overture. Indicator:95%

5.8 UTRC-Req-003: Design Space Exploration

- **Description**
  Design space exploration techniques should be applied to both continuous and discrete models when executing a series of simulations
- **Related Baseline Tools Requirements**
  Requirements 0018-0020 could be necessary in order to assess the maximal/minimal value from a range of parameters.
- **Method of Verification**
  This requirement should be verified using simulations and functional testing. Currently there is not support for design space exploration within the baseline tools. Early design state exploration experiments have been applied offline in the building case study independently from the tools evaluated in this report.
- **Assessment:** not achieved Currently the design space exploration (DSE) is not supported within the INTO-CPS toochain. CT and DE model evaluation started but not completed. Indicator:25%

5.9 UTRC-Req-004: Simulation Accuracy and Precision

- **Description**
  INTO-CPS should be able to provide a varied level of simulation accuracy with a user-selected variable precision. This will increase confidence in results from co-simulation executions
- **Related Baseline Tools Requirements**
  Requirements 0045, 0047, 0055, 0058, 0061, 0065 : Semantics are necessary in order to keep accuracy and confidence. Quantifiable simulation tolerance of the INTO-CPS co simulation are necessary to keep accuracy.
- **Method of Verification**
  Having in place formal semantics for INTO-CPS baseline tools is of paramount importance in order to enable simulation accuracy and precision. Currently INTO-CPS tools have a limited support with respect to simulation and co-

simulation manipulation (e.g. rollback or test case evaluation) with accurate results.

- **Assessment:** partially achieved
  In the current evaluation study, tools have depicted certain levels of precision in the simulation experiments. This includes variable step co-simulation artifacts (e.g. raw cosimulation data, graphs) generated by the orchestration engine and the INTO-CPS application. We expect to increase simulation results accuracy (using time-tags, debugging log files, memory consumption) in co-simulation in the next year of the INTO-CPS project. Indicator:40%

5.10 UTRC-Req-005: Scalable Simulation

- **Description**
  INTO-CPS platform should adequately produce simulation results in large continuous, discrete or co-models where complexity is significantly high.
- **Method of Verification**
  Although this requirement is often bound to the available resources, from the co-simulation engine point of view, the INTO-CPS toolchain should be able to produce simulation and co-simulation results for a certain level of model complexity. As this verification will depend on the model complexity we argue that the specific requirement will be re-evaluated in year 3 of INTO-CPS project.
- **Assessment:** partially achieved
  In the current Year 2 evaluation, INTO-CPS platform has exhibited certain levels of simulation efficiency when tackle a certain level of model complexity, particularly for our CT model. This may include solver evaluation and assessment during fixed or variable step co-simulations, from the INTO-CPS toolchain. Indicator:60% achieved

5.11 UTRC-Req-006: Code generation

- **Description**
  INTO-CPS platform should allow code generation with respect to selected hardware in certain programming languages (e.g. C) automatically from the models.
- **Related Baseline Tools Requirements**
  Related to requirements 0037, 0042, 0044.

- **Method of Verification**
  Code generation from the VDM-RT model on PIC micro-controllers (UTRC FCUs) for executing FCU PI commands

- **Assessment:**partially achieved
  INTO-CPS code generation support is not yet mature particularly for the Overture tool where we model our supervisor and the PI controllers. Proof of concept results have been shown for basic PI FCU controller in the selected hardware platform but not configured for the distributed case. Indicator:30%

5.12 UTRC-Req-007: Platform Independence and FMU support

- **Description**
  INTO-CPS platform should be independent from external tool dependencies and support FMI-based cosimulation (version 2.0) for both encrypted and plain FMUs.

- **Related Baseline Tools Requirements**
  New requirements.

- **Method of Verification**
  Evaluate both encrypted and non-encrypted FMUs' cosimulations;

- **Assessment:**partially achieved
  Major contribution of the INTO-CPS platform being used from the web-app. Still there is a certain level of matureness that can be achieved in year 3. Indicator:70%

5.13 Conclusion - Reporting Experiences

The goal of this deliverable was to bring insights about industry benefits and weaknesses of using INTO-CPS platform, and to plan for the next technological advances of INTO-CPS features according to the project planning for Year 3. By evaluating our HVAC controllers through different co-simulation scenarios, we are in the position to adjust and refine INTO-CPS tools requirements, providing valuable feedback to the consortium. We discuss below and reflect our experiences of working with FMI-based tool chains for the design and development of CPSs.

Table 7: UTRC’s requirements

Req.	**Category**	**Sub Cat.**	**Insight**	**Priority**	**Status**	**Version**	**Assess**
UTRC-Req-001	Non-Functional	System	Simulation Completeness	M	Year 1	1	achieved , 100%
UTRC-Req-002	Non-Functional	System	Simulation Delays	S	Year 1	1	achieved , 95%
UTRC-Req-003	Non-Functional	Efficiency	Design Space Exploration	C	Year 1	1	not achieved, 25%
UTRC-Req-004	Non-Functional	Tools	Simulation Accuracy and Precision	C	Year 1	1	Partially, 40%
UTRC-Req-005	Non-Functional	Tools	Scalable Simulation	S	Year 1	1	Partially, 60%
UTRC-Req-006	Tools	Scalability	Code Generation	S	Year 1	1	Partially, 30%
UTRC-Req-007	Functional	Software	Platform Independence and FMU support	S	Year 1	1	partially achieved, 70%

- It is important for a project to have a common CPS design description that is central and shared by all project members. This design should evolve over time but these evolutions should be visible and agreed upon by all members. Ideally, this design should be enforced through distribution and adherence to constituent model specifications through FMI model descriptions. Where the tools do not mandate this, it should be project policy to do so nonetheless. Tools that do not support importing of FMI model descriptions should not be avoided if at all possible.
- To combat undocumented assumptions slipping through the FMI tool chain, all restrictions, units and other kinds of assumptions must be properly documented. Ideally, this should be done in the main design document or SysML model. If this is not possible, an independent document containing the assumption should be created. This document must be shared and accessible by all partners. From a tooling perspective, such assumptions should be encoded in the constituent models and the FMI interface.

An important aspect was as well the need for specific INTO-CPS features such as design space exploration and code generation which will elevate the platform efficiency towards scalability and SiL/HiL simulations respectively. As next steps in Year 3 of the INTO-CPS we anticipate to apply INTO-CPS technologies in order to analyze more complex HVAC cases in order to evaluate scalability of the orchestration engine. Having in mind current state-of-the-practice commercial co-simulation platforms we aim to evaluate new HVAC industrial needs through INTO-CPS features including mature code generation for SiL and HiL simulations, design space exploration and model checking.

6 Automotive Case Study

6.1 Specifications

The main goal for the automotive case study in INTO-CPS is to create a tool for range prediction of electric vehicles. This shall be done by simulating the drive train of the electric vehicle and other relevant electric auxiliary loads such as the air conditioning, while integrating information from weather, topography or traffic. The range optimization assistant shall enable the driver of an electric vehicle to select a route which offers the maximum range of the vehicle with the given electric battery, in order to alleviate the drawbacks of the limited range.

For the initial case study (see also deliverable D1.1a [HLB+15]), we distinguish between two scenarios :

- *Offline:* A user gives as input a desired start and end destination. A specific route is simulated and the energy consumption is calculated.
- *Online:* The online mode is activated as soon as a route was chosen and the car starts moving in a certain direction. In this situation, real-time data coming from the vehicle like torque requirements, current speed, available battery power (state of charge) etc can be accessed.

  Here the following factors play an important role in the work flow:

  1. The state of the car (state of charge of the battery) is updated periodically . If the energy level deviates from the prediction (done in the off-line mode) and the energy level isn't enough to reach the destination another route is simulated. If no route is found under the energy constrains the user is alerted. If a new route was found the user is alerted and has to choose whether the route should be updated.
  2. The environment (weather or traffic situation has changed) is seen as a trigger that causes a new simulation to start and generate another route. The user has to actively agree upon the old route being overwritten by the newly generated route.
  3. The driver deviates to another route and thus triggers a new route that needs to be generated. Automatically, the old route is overwritten by the new route .

  An illustration of the triggers inside the system is shown in figure 35 below.

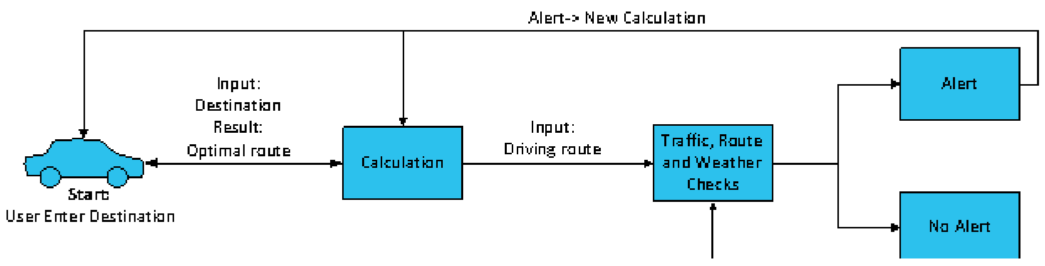


Figure 35: Alert system of the case study.

To demonstrate the alert system the following situation will be shown in the case study. The driver starts the application and enters a destination. The system returns the optimal route. Now there are three possible scenarios:

1. The weather situation changed and a new Route A is the better one.

2. Nothing happens and the driver arrives at the destination.
3. The traffic situation has changed and a new Route C is the better one.

These scenarios are depicted in the following figure 36.

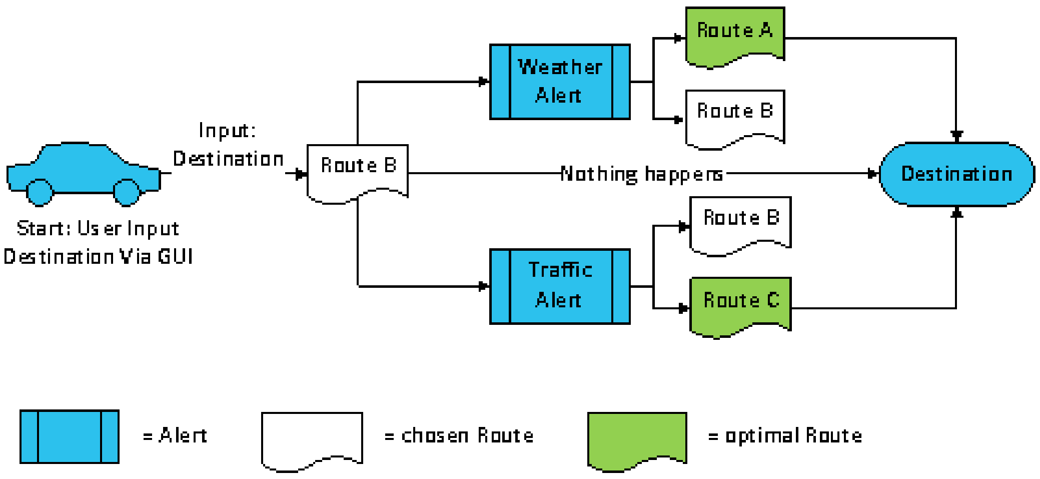


Figure 36: Scenarios for selection of different routes depending on the different alerts.

The alarm system is described in more detail below, and was implemented as a prototype by the end of year 2.

6.2 Specification of the HiL scenario

In order to increase the level of complexity of the automotive case study, a HiL scenario is proposed that goes beyond the originally planned scenario. The goal of this is to better evaluate the INTO-CPS tools, by developing and optimizing the gas pedal controller (see below) with the help of the INTO-CPS tools.

This setup contains the following components that will be described below. It partially builds on top of a setup for a driving simulator that was used in the HoliDes research project (see for example `http://www.holides.eu/content/adapted-automatic`

Driver: A person interacting with the driving simulator by means of pedals (brake and accelerator) and steering wheel. As we only consider the longitudinal dynamics in the case study, only the gas and brake pedal are relevant for this scenario.

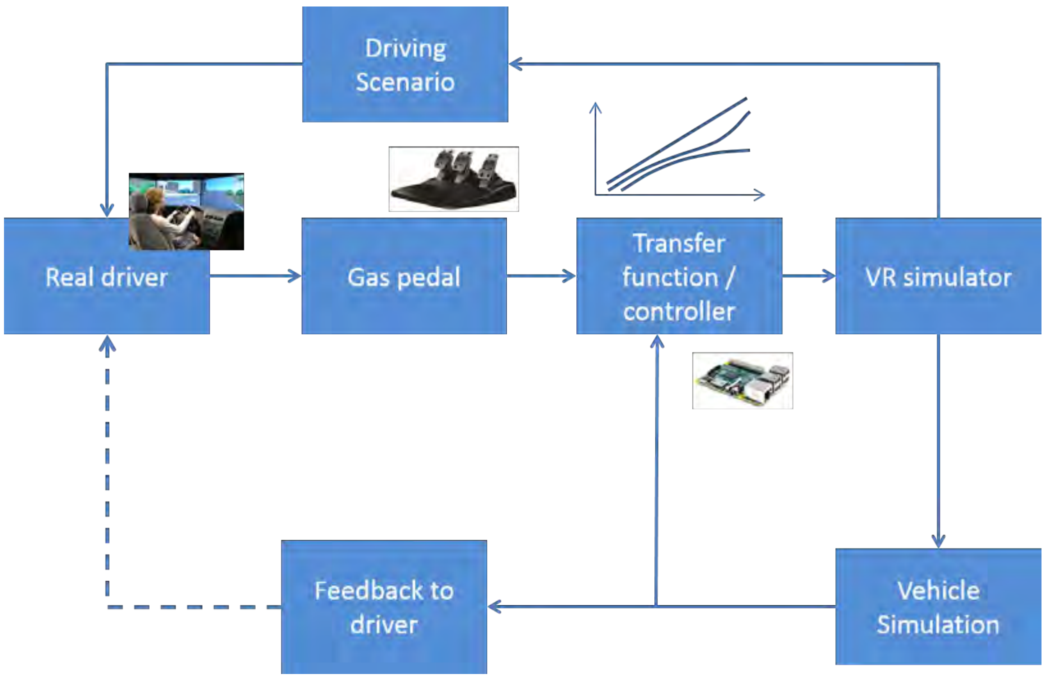


Figure 37: Schematic scenario for the automotive case study with a HiL setting.

Steering Wheel and pedals: The Human-Machine-Interface consists of a physical steering wheel and a gas and braking pedal (Logitec G27 Racing Wheel).

Controller: A controller (here to be implemented on a Raspberry Pi microcontroller) that uses as input the gas pedal signal and modifies this signal for the output, based on parameters provided by vehicle simulation. The controller implements an algorithm that calculates the parameters for modification of the gas pedal signal. Based on the remaining range or the SoC level, the output level of the gas pedal is reduced. This controller, and in particular its function for adapting the gas pedal curve, has to be developed from scratch. This will be done using the INTO-CPS tools. An initial proposal for this function can be seen below in Figure 38. There, the output of the controller is initially equal to the input. Depending on the state of the vehicle, this curve can be modified by the controller, such that the output is lower than the input and the gas pedal is "softened". When the input is at its maximum (i.e. the driver requests full acceleration), the output must also reach the maximum, since it can be crucial for the driver's safety.

Virtual reality simulator: A complete setup for a driving simulator is based on the OpenDS simulation (see `www.opends.eu`).

Vehicle simulation: The bulk of the simulation that has been developed so far in INTO-CPS (see [HLB+15]). The input signals for the vehicle simulation are generated by the VR simulator in form of velocity, position of the car, or acceleration. Input signals such as temperature or atmospheric pressure are generated by the weather module based on the current car position.

Feedback: A visual signal, in form of a simple LED that changes its color is given to the driver in the attempt to influence his driving behavior to improve range and power efficiency. The LED turns green every time the driver accelerates slowly and in case of a kick-down or a sudden push of the gas pedal the LED will turn red.

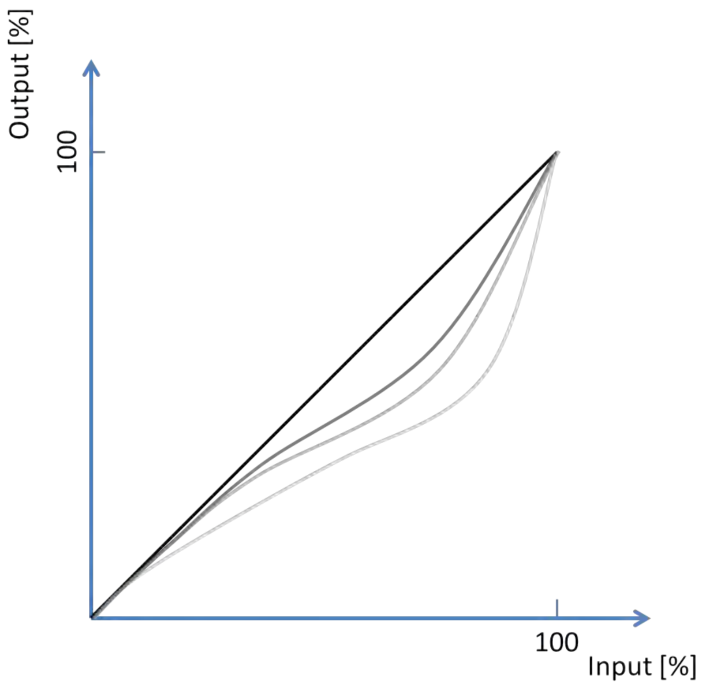


Figure 38: Output of the gas pedal controller, as a function of the input. Initially the output is equal to the input (black diagonal line), but depending on the remaining range of the vehicle, the curve can be ”softened” (gray lines).

Of these components, mainly the feedback and the controller need to be developed, using the INTO-CPS tools. The other components already exist, but need to be adapted.

6.3 Model

The system model consist of the following parts:

- Longitudinal dynamics of an electric vehicle, including the battery and air conditioning
- Route planning including traffic information
- Ambient conditions
- Alarm System

These models are coupled with the INTO-CPS Co-Simulation Orchestration Engine (COE). Most of these models are described in detail in the previous deliverable [HLB+15], and their description will not be repeated here, for the sake of clarity. Since the CT models of the automotive case study are written in Matlab, a Matlab-FMU wrapper had to be developed by TWT, that allows coupling of the models to the COE (see also Deliverable D6.2 [KFP+16]).

The SysML connections diagram for the multi-model, consisting of the route module, the weather module and the longitudinal dynamics module (internally called ”ArtSim”) is shown in Figure 39 below.

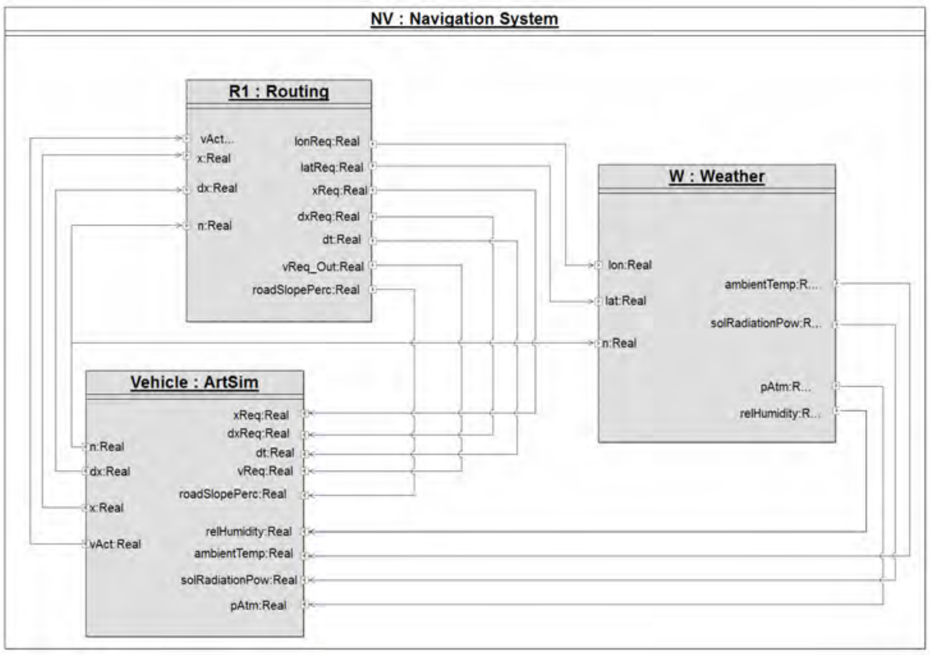


Figure 39: SysML Connections diagram including the longitudinal dynamics, the weather module and route module.

6.4 Route module (CT)

During year 2, the route module that was previously described in Deliverable D1.2e [KB15] was improved. Now, it takes the current position and velocity (calculated from the longitudinal dynamics) as input for the next section of the route (this is also shown in the connections diagram in Figure 39). This in turn makes the whole simulation more realistic, as it smoothes the acceleration curve.

6.5 Alarm System (DE)

We consider a controller, which we call an alarm system, with the function to monitor the state of the vehicle, in terms that are relevant for the route assistant that is developed in this case study. It consists of four functions which monitor the state of the battery (i.e. the SoC), the traffic, the weather and the route. If any of these changes, the simulation is reset, to take the changed conditions into account. It is foreseen to use real data in the long run, but for developing this alarm system, simulated data can be used. The state diagram of this alarm system is displayed in figure 40.

The default state is *SignalMonitoring*, in which the alarm system continuously checks the vehicle parameters. If any of these is out of its boundaries, the Co-Simulation is re-started. If the driver decides to take a different route than the proposed one, the route needs to be calculated before these values then are handed over to the weather module, for delivering the weather values to the Co-Simulation. A sudden change in weather conditions also triggers the restart of the weather module, and in consequence of the Co-Simulation. If the measured velocity of the vehicle is on average significantly lower than the planned velocity, a traffic jam is assumed, which requires a re-start of the Co-Simulation as well. Finally, if certain vehicle parameters (such as the battery SoC) suddenly change, the new values need to be taken into account.

The alarm system is being implemented in VDM and coupled to the COE as an FMU. At the time of writing this deliverable, this is still work in progress.

6.6 Simulation results

To simulate the range-prediction, a route in the vicinity of Stuttgart (Germany) is selected. The route has a length of approximately 34km and is largely on country road. The height profile is depicted below in Figure 41. Initially, the road has a

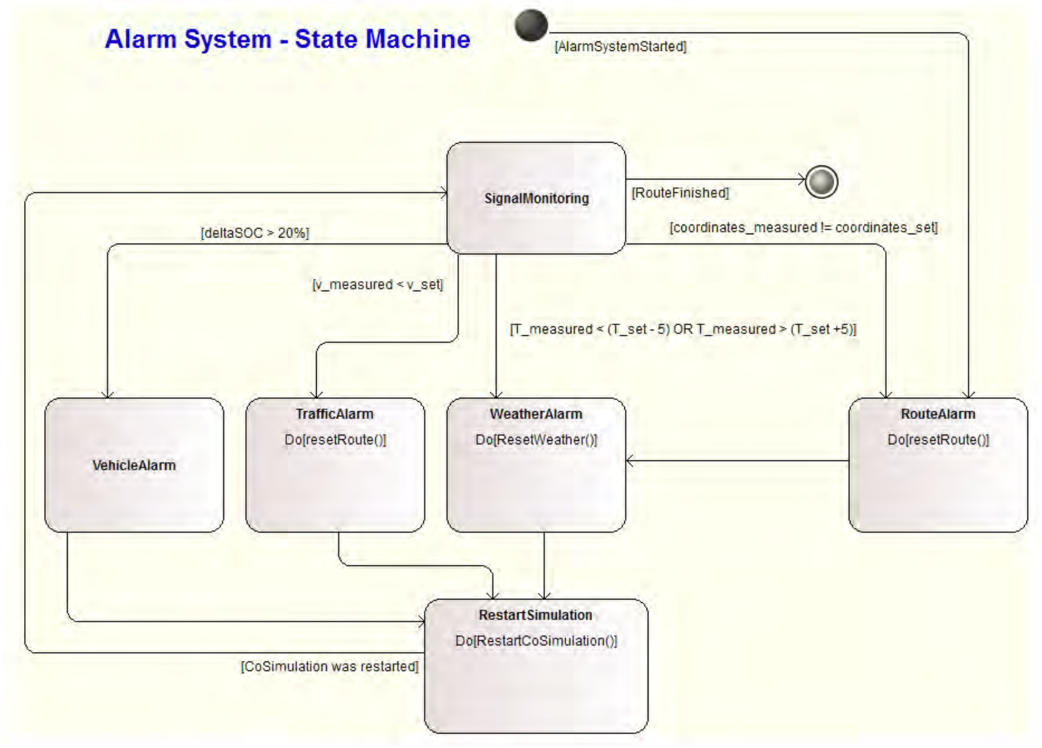


Figure 40: State diagram of the alarm system.

rather steep ascent, until it reaches an height of approximately 750m above sea level. After several ascents and descents, the route shows an overall downwards slope until it ends at 650m above sea level.

The simulation results for the vehicle state are plotted below in Figure 42. Initially, the vehicle speed is around 25 km per hour, since the route is on an inner city road. After the first kilometer, the route leads onto a country road for the remainder of the trip. The figure also shows that the vehicle is able to follow the set velocity closely, despite the slope of the road shown in the previous Figure 41. As there is no gearbox considered in this electric vehicle, the motor speed follows exactly the vehicle speed. The battery voltage starts at around 330 V and drops finally to around 310 V. Along the trip, the voltage oscillates within a range of about 10 V, due to the acceleration and recuperation that is required by the profile of the road. As expected, the battery SoC (State of Charge) drops from initially 100 % to approximately 70 % at the end of the trip.

Results for motor torque, battery current and the slope of the route are displayed in Figure 43. All three values show very similar behavior, which is closely coupled with the battery voltage. The motor generates a torque of up to 150 Nm in the

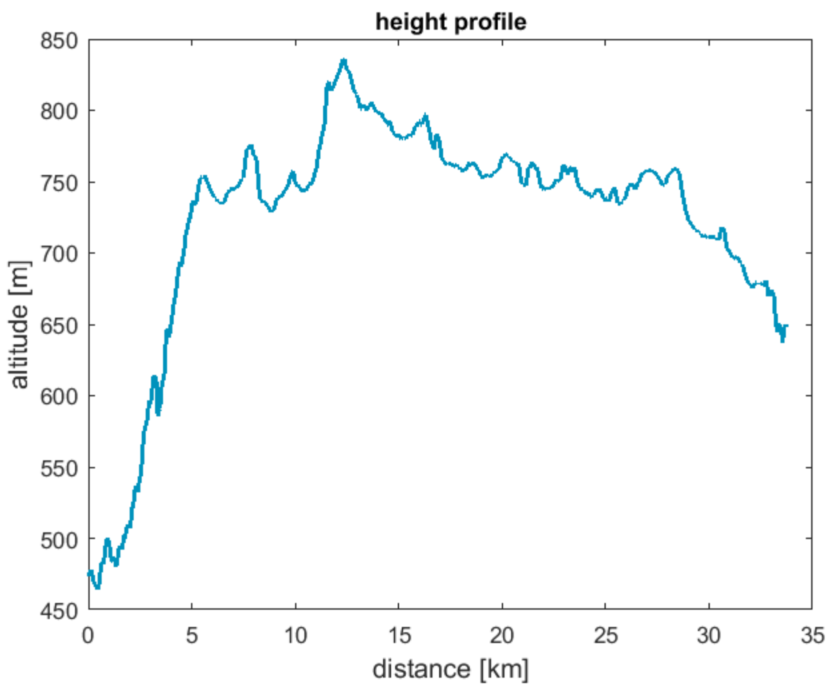


Figure 41: Height profile of the selected route.

steep sections of the road. Accordingly, a current of up to 200 A is drawn from the battery, in particular in those sections that have a slope of up to 20 %.

The temperatures that are calculated from the air conditioning module (for a more detailed description of this model, see Deliverable D1.1e [KB15]) are displayed in Figure 44. Initially, the temperature of the air inside the vehicle is 20 C, while the temperature outside is 5 C (e.g. when the vehicle was parked in a garage in winter). The setpoint for the temperature controller is 23 C. While the temperature of the air inside the vehicle quickly rises, the fixtures (e.g. the seats) only heat up slowly. At the same time, the temperature at the windows and at the vehicle case drops quickly and reaches a steady state at 7 C.

6.7 Evaluation of INTO-CPS tools

In this section, the evaluation of the emerging INTO-CPS tools from the autmotive's case study's perspective is given.

- Modelio

  The SysML diagrams for describing the system model that were generated

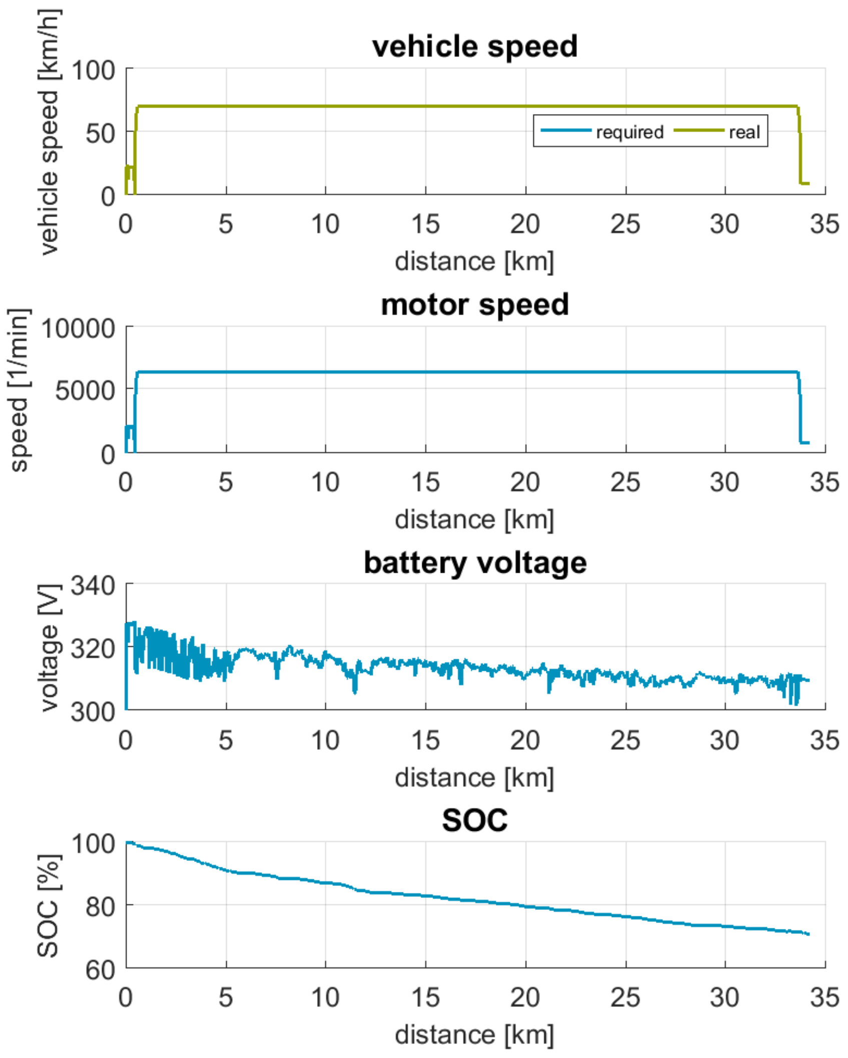


Figure 42: Simulation results for vehicle speed, motor speed, battery voltage and SoC.

in Modelio in year 1 were transferred to Architectural diagrams and Connection diagrams, to make use of the INTO-CPS extensions of Modelio. Exporting a COE configuration from this Connections diagram (see for example Figure 39) makes the process easier than manual configuration of the COE, and less error prone.

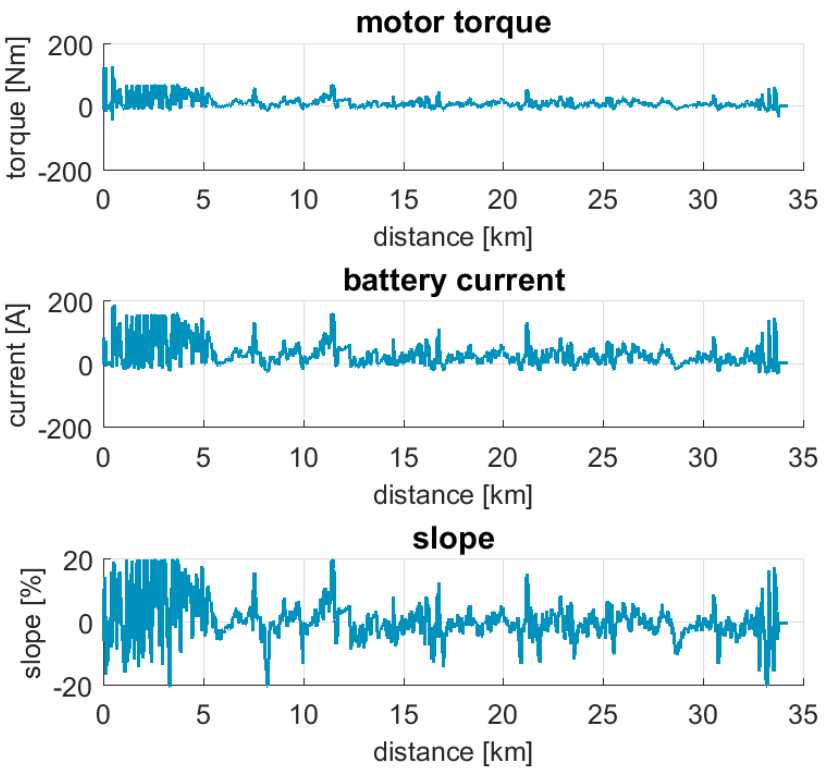


Figure 43: Simulation results for motor torque, battery current and slope of the route.

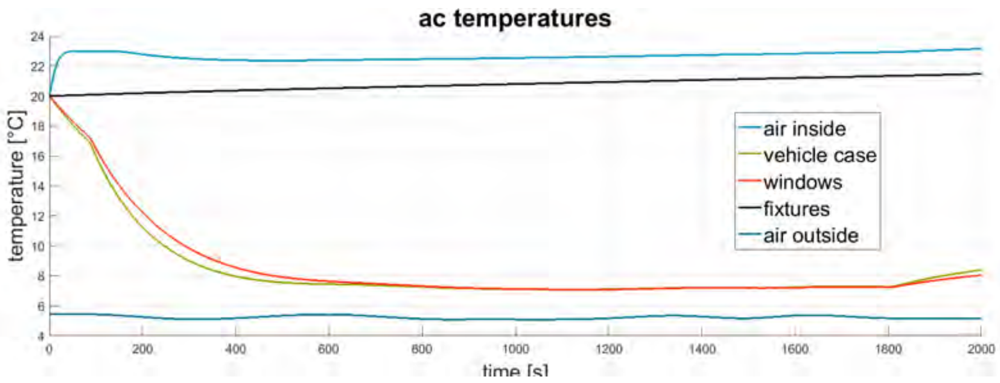


Figure 44: Simulation results for temperatures inside and outside the vehicle.

- VDM / Overture

  A simple version of the Alarm system (see section 6.5) was implemented in Overture. Using the new FMU export option, this model was exported to an FMU and connection to the COE (i.e. coupling the alarm system to the remainder of the models) is being performed at the time of writing of this deliverable. The new FMU export option in Overture significantly simplifies the process. In combination with the import of the MODELDESCRIPTION.XML from Modelio, the whole workflow is integrated further, and errors (e.g. due to different naming of signals) are reduced.

- Co-Simulation Orchestration Engine / INTO-CPS application

  The different simulation models were coupled with the COE, using the INTO-CPS application. The configuration of the multi-models was created from Modelio, and the Co-simulation was configured in the INTO-CPS application. The INTO-CPS application is user-friendly and rather simple to use, which makes the shift to this tool quite simple. Throughout year 2 of INTO-CPS, the COE has become more stable, which improved usability.

Im summary, the core parts of the emerging INTO-CPS tools and workflows were used and evaluated in the automotive case study. The integrated tool-chain offers many benefits for a smooth workflow. The fact that the tool-chain is open to other tools that support the FMI standard (such as Matlab) allows re-using existing know-how, and is therefore a great advantage for potential users. Feedback was given from the automotive case study to the tool developers regarding user-experience, requirements and potential bugs.

6.8 Requirements and Assessments

6.8.1 TWT_1: Validity checking

- **Description:** The validity of system models, i.e. the correctness of the connected input and output signals to and from the different models is checked by the tools.
- **Method of Verification:** Manual variation of input and output signals to check if the tools detect the changes. If the tools notify the user of a mismatch, this requirement is fulfilled.
- **Assessment:** Complete. The INTO-CPS application detects if the name of a input / output signal is false, and gives a warning to the user.
- **Related baseline tools requirements:** 0035 - 0036
- **Degree of achievement:** 100%

6.8.2 TWT_2: System structure

- **Description:** A system structure, describing the connections between the different models and their hierarchy, is created and saved by the tools, in order to re-import and modify later. This simplifies the generation, distribution and modification of the whole system architecture.
- **Method of Verification:** Manual generation of a system model to import into the tools. If the system structure is correctly imported, this requirement is fulfilled.
- **Assessment:** Complete. The INTO-CPS application allows saving of complete projects, including their hierarchy. SysML diagrams, describing the architecture of the system, can also be exported.
- **Related baseline tools requirements:** 0001, 0005, 0012
- **Degree of achievement:** 100%

6.8.3 TWT_3: Results tracing

- **Description:** Simulation results are traced to the requirements and the models where the results originated from. This makes testing of the requirements easier and thus increases trust in the simulation results.
- **Method of Verification:** Comparison of results and requirements in simple models. For a measurement, the number of requirements that are linked to simulation results can be compared to the total number of requirements.
- **Assessment:** Not started. The INTO-CPS tools do not yet support traceability functions.
- **Related baseline tools requirements:** 0015 - 0017
- **Degree of achievement:** 0%

6.8.4 TWT_4: Requirements assessment

- **Description:** An indicator is provided to determine how well a requirement has been met, not just yes/no.
- **Method of Verification:** Requirements can be varied systematically to assess if the resulting indicator / ranking function reacts accordingly.

- **Assessment:** Not started. The INTO-CPS tools do not yet provide the traceabilty between requirements and results.
- **Related baseline tools requirements:** 0021
- **Degree of achievement:** 0%

6.8.5 TWT_5: System creation from SysML

- **Description:** Starting from a SysML model of the system, a skeleton system model (in FMI) is created. This will simplify the workflow by connecting the different tools for system modelling and simulation.
- **Method of Verification:** Comparison of an automatically generated skeleton system model with its SysML model.
- **Assessment:** Complete. The INTO-CPS profile for Modelio offers export of FMI modelDescription.xml files.
- **Related baseline tools requirements:** 0049
- **Degree of achievement:** 100%

6.8.6 TWT_6: Version checking

- **Description:** The tools should be able to check if the interfaces between different models have changed due to different versions. This is relevant for the development process of modelling, especially when different parties are responsible for different models.
- **Method of Verification:** Selection of different model versions with different inputs and outputs. If different versions of the same model can be selected, or the user is notified of differences between model versions, this requirement is fulfilled.
- **Assessment:** Ongoing. The INTO-CPS application detects if input / output signals do not match the corresponding names. However, versioning support is not yet implemented.
- **Related baseline tools requirements:** 0090
- **Degree of achievement:** 80%

6.8.7 TWT_7: Version selection

- **Description:** The user selects different versions of the same model (either single model or system model).
- **Method of Verification:** Provided / not provided.
- **Assessment:** Not started.
- **Related baseline tools requirements:** It remains to be discussed if this requirement will be incorporated in the general requirements document, deliverable D7.5 [LPO$^+$16].
- **Degree of achievement:** 0%

6.8.8 TWT_8: Parameter variation

- **Description:** Model parameters are systematically and automatically varied within defined limits. This allows systematic optimization of the models.
- **Method of Verification:** Provided / not provided.
- **Assessment:** Not started. Design Space exploration functions are still under development and have not yet been tested with this case study.
- **Related baseline tools requirements:** 0020
- **Degree of achievement:** 0%

6.8.9 TWT_9: Parameter selection

- **Description:** If the user wants to parametrize a whole model at once, parameter sets for models are selected from a parameter file. This can be helpful if relatively complex models have large parameter sets, describing for instance different variants of a product.
- **Method of Verification:** Provided / not provided.
- **Assessment:** Not started.
- **Related baseline tools requirements:** It remains to be discussed if this requirement will be incorporated in the general requirements document, deliverable D7.5 [LPO$^+$16].
- **Degree of achievement:** 0%

6.9 Conclusion

In this part, the progress of the automotive case study that was made during year 2 is presented. Based on the know-how that was already available, TWT has created a co-simulation using CT models and have started with DE modeling, to create a cruising range prediction tool for electric vehicles. This tool takes into account a model of the physical behaviour of the vehicle (the longitudinal dynamics, the battery, electric engine and the air conditioning) as well as the topology of the chosen route, and weather conditions (relevant for the air conditioning) and a system for monitoring the state of the vehicle. It will therefore allow the user to predict the remaining battery charge, based on on realistic data (vehicle physics, route, weather). It can be used for example for realistic calculation of electric vehicle fleets or for integration in the vehicle's software, as an addition to the navigation system, or for optimizing vehicle settings for maximum range.

The focus of this year's work was the transfer of the case study to the INTO-CPS tool-chain. This primarily meant usage of the COE, and as a consequence, connecting the Matlab models in a FMI-compliant fashion. To achieve this, a Matlab FMU wrapper was developed during year 2. Furthermore, integration of the workflow from the abstract system models (in SysML) to creation of a Co-simulation configuration, was achieved.

To further develop the case study, a HiL scenario is specified in this deliverable. This will be largely based on existing technologies, and serve to evaluate more aspects of the INTO-CPS tool-chain, such as code generation or design space exploration.

The evolving INTO-CPS tools will be further evaluated in the third year. One focus here will be the evolving functionalities regarding traceability, which are very interesting for Systems Engineering applications. In terms of implementation, the HiL scenario that is described in this document will be the main effort in year 3, along with the assessment of the requirements.

References

[CLJ15] Martin Peter Christiansen, Peter Gorm Larsen, and Rasmus Nyholm Jørgensen. Agricultural Robotic Candidate Overview using Co-model Driven Development. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3 Submitted 4-3-2015.

[EGH15] Jose Esparza, Ole Green, and Stefan Hallerstede. Case Study 1, Agriculture, (Confidential). Technical report, INTO-CPS Confidential Deliverable, D1.1c, December 2015.

[HLB$^+$15] Francois Hantry, Thierry Lecomte, Stelios Basagiannis, Christian König, and Jose Esparza. Case Studies 1, Public Version. Technical report, INTO-CPS Public Deliverable, D1.1a, December 2015.

[KB15] Christian König and Natalie Balcu. Case Study 1, Automotive, (Confidential). Technical report, INTO-CPS Confidential Deliverable, D1.1e, December 2015.

[KFP$^+$16] Christian König, Peter Fritzson, Adrian Pop, Christian Kleijn, Peter Gorm Larsen, Mette Stig Hansen, Jörg Brauer, and Stylianos Basagiannis. Dissemination and Exploitation Report - Year 2. Technical report, INTO-CPS Deliverable, D6.2, December 2016.

[KS10] Manoj Karkee and Brian L. Steward. Study of the open and closed loop characteristics of a tractor and a single axle towed implement system. *Journal of Terramechanics*, 47(6):379–393, December 2010.

[LPO$^+$16] Peter Gorm Larsen, Ken Pierce, Julien Ouy, Kenneth Lausdahl, Marcel Groothuis, Adrian Pop, Miran Hasanagic, Jörg Brauer, Etienne Brosse, Carl Gamble, Simon Foster, and Jim Woodcock. Requirements Report Year 2. Technical report, INTO-CPS Deliverable, D7.5, December 2016.

[QCG$^+$09] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, 3, page 5, 2009.