D3.4 – Conclusions on Integration of Subsystems into FDF and SF V1.1

Project number:	826073
Project acronym:	Safe4RAIL-2
Project title:	SAFE architecture for Robust distributed Application Integration in roLling stock 2
Start date of the project:	1st October 2018
Duration:	34 months
Programme:	H2020-S2RJU-OC-2018

Deliverable type:	Report
Deliverable reference number:	ICT-826073 / D3.4 / 1.1
Work package	WP3
Due date:	July 2021 – M34
Actual submission date:	29th July 2021

Responsible organisation:	IKERLAN
Editor:	Iñigo Odriozola
Dissemination level:	Public
Revision:	1.1

Abstract:	This report includes conclusions on the integration of a train application into FDF and SF and provides a methodology for development of train applications on FDF and for integration into SF
Keywords:	Train applications integration, Methodology, Functional Distribution Framework, Simulation Framework

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 826073. The information and views set out in this document are those of the author(s) and do not necessarily reflect the official opinion of Shift2Rail Joint Undertaking. The JU does not guarantee the accuracy of the data included in this article. Neither the JU nor any person acting on the JU's behalf may be held responsible for the use which may be made of the information contained therein.

Editor

Iñigo Odriozola (IKL)

Contributors (ordered according to beneficiary numbers)

Xabier Mendialdua (IKL)

Núria Mata (ETAS)

Martin Zauner (LIEB)

Executive Summary

This delivery reports on the conclusions of the integration of train applications into the Functional Distribution Framework (FDF) and the Simulation Framework (SF).

The FDF provides an abstraction for the development of railway functions so that software applications are portable between different FDF implementations.

The SF consists of a set of tools for train subsystem virtualization, communication emulation and simulation, which allows integrating software applications to a mixed validation environment to conduct the validation.

The integration of a train application is based on the Application Profile (AP). The Application Profile is a key element for the standardization of the train function, as it defines the use cases the train application must implement and the data interfaces for the communication between the train subsystem and the TCMS.

A case study carried out for the Heating, Ventilation and Air-Conditioning (HVAC) subsystem is described in the first part of this document. The case study shows how an HVAC Subsystem has been implemented and integrated into two different implementations of the FDF. The aim of these integrations is to demonstrate the portability of the HVAC Subsystem (and in general of any train function) for different FDF implementations. The integration of HVAC Subsystem and validation was reported in the confidential D3.3 document. The current report summarizes the most relevant aspects of the integration and validation tasks, trying to give an overview while preserving confidential information.

The second part of the document describes the methodology that should be followed to carry out the integration of train applications into the FDF and the SF. Some recommendations and best practices for the integration of train applications, from the experience gained in the implementation of the HVAC case study, are also provided.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

The FDF provides an abstraction for the development of railway functions so that software applications are portable between different FDF implementations.

The SF consists of a set of tools for train subsystem virtualization, communication emulation and simulation, which allows integrating software applications to a mixed validation environment to conduct the validation.

The integration of a train application is based on the Application Profile. The Application Profile is a key element for the standardization of the train function, as it defines the use cases the train application must implement and the data interfaces for the communication between the train subsystem and the TCMS.

The integration of the Heating, Ventilation and Air-Conditioning (HVAC) subsystem was already reported in the confidential deliverable *D3.3 - Report on Integration of HVAC Subsystem and Validation*. Some contents extracted from D3.3 have been included in chapter 2, which describes the HVAC case study and the integration activities.

The aim of the case study is to demonstrate the portability of the HVAC Subsystem (and in general of any train function) for different FDF implementations. Thus, the case study describes how the HVAC Subsystem has been implemented and integrated into two FDF implementations.

These integrations have been validated using different validation scenarios defined by CONNECTA-2 and developed jointly with Safe4RAIL-2 partners. The HVAC Subsystem monitors and controls de HVAC device. Both HVAC Subsystem and HVAC device have been validated in simulated and real hardware, depending on the setup of each validation scenario. The simulated HVAC device provided by Safe4RAIL-2 has been implemented as a Functional Mock-up Unit (FMU) and provides the standardized Functional Mock-up Interface (FMI). This simulated HVAC enables to conduct the validation using the Simulation Framework. The SF is developed and runs at the train manufacturer.

Chapter 3 describes the methodology and the workflow to be used for the integration of train applications into the FDF and the SF. Some recommendations and best practices are derived from the experience gained in the implementation of the HVAC case study.

Chapter 2 HVAC Case Study and Demonstrator

Heating, Ventilation, and Air Conditioning (HVAC) is the technology of indoor and vehicular environmental comfort. An HVAC system's goal is to provide thermal comfort and acceptable indoor air quality. For this purpose, the HVAC comprises a set of subsystems used for moving air between indoor and outdoor areas, along with heating and cooling.

The document "Application Profile Definition Guideline and Example" [1] defined by CONNECTA describes the use cases to ensure "proper climate" functionality in a railway vehicle.

Based on the application profile of the HVAC Subsystem, the Technical Application Profile for HVAC [2] by CONNECTA-2 defines a functional grouping and describes the interfaces to be provided by Consist Level Control and HVAC Subsystem Control applications.

Based on the Technical Application Profile for the HVAC the following subset of uses cases have been implemented, after the agreement with CONNECTA-2 project.

CTA-T4.3-UC-Hvac-1	Startup and manage HVAC system
CTA-T4.3-UC-Hvac-3	Manage HVAC operational mode
CTA-T4.3-UC-Hvac-10	Monitor vehicle outside temperature
CTA-T4.3-UC-Hvac-11	Monitor comfort zone inside temperature
CTA-T4.3-UC-Hvac-12	Monitor comfort zone HVAC functional state
CTA-T4.3-UC-Hvac-14	Define comfort zone setpoint temperature
CTA-T4.3-UC-Hvac-15	Adjust comfort zone temperature offset
CTA-T4.3-UC-Hvac-17	Monitor HVAC failures

Table 1: Implemented use cases

The HVAC Function, as depicted in Figure 1, is decomposed in three components (yellow boxes): TCMS, HVAC Control and IO Control. The IO Control can be either a physical HVAC device or a plant software model to simulate the HVAC hardware.

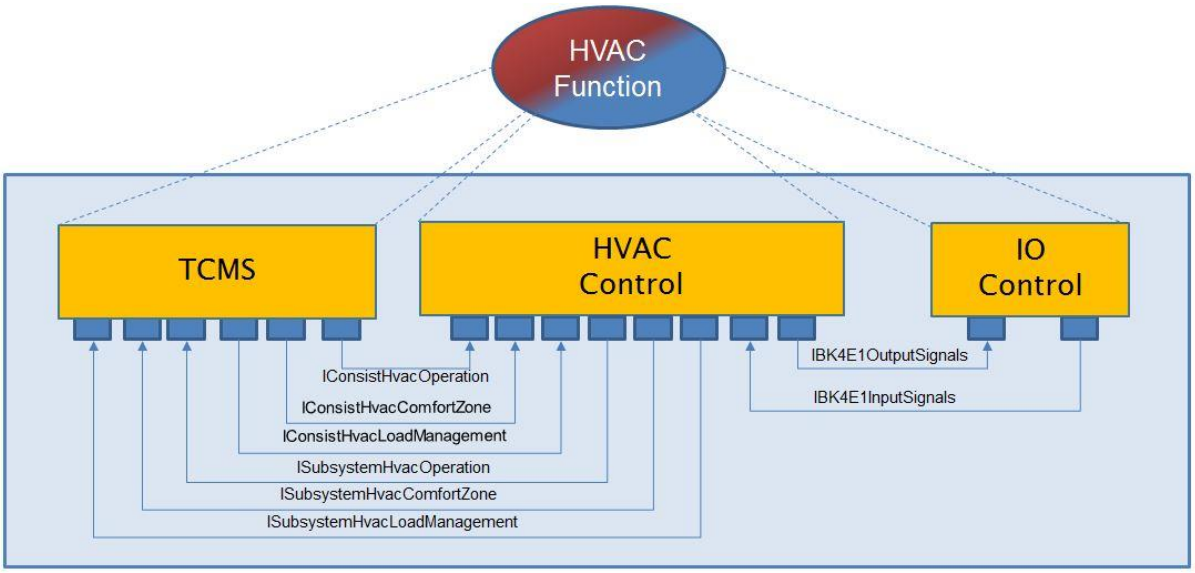


Figure 1: Components implementing HVAC Function and communication between ports using service interfaces.

The blue boxes attached to the components are ports to either offer services to other components or subscribe to services. The service interfaces used for the communication between the HVAC Control and TCMS have been defined in the CTA-2 HVAC Application Profile in [1] and [2].

The service interfaces between the HVAC Control and the IO Control are not defined by the application profile, since those interfaces are typically HVAC supplier specific because they depend on the hardware of the HVAC device. Therefore, interfaces IBK4E1OutputSignals and IBK4E1InputSignals in Figure 1 are Liebherr's hardware platform (BK4E1) specific and thus they are defined and provided by Liebherr.

The integration of Liebherr's HVAC described in this HVAC case study has comprised the activities explained below. The HVAC Control application has been first implemented. This Control application has been then integrated on top of the FDF using its API. Furthermore, a TCMS application that communicates via the FDF-API with the HVAC Control application has been implemented and integrated as well. Finally, the validation activities have been carried out in the different validation scenarios defined by CONNECTA-2 partners.

2.1 HVAC Subsystem Implementation

The HVAC Control is the software that monitors and controls the HVAC System. It is together with the TCMS application, integrated in a single CCU on top of the FDF (see Figure 1). The same HVAC Control software interacts with the physical HVAC device or a simulated HVAC Model through IO signals like IBK4E1OutputSignals and IBK4E1InputSignals (see Figure 1), monitoring the status of its sensors and devices and sending commands to the actuators that operate the HVAC.

On system start-up, HVAC Subsystem implements a start-up procedure, in which TCMS and HVAC Control negotiate the power needed by the physical HVAC to start in operation and the available power to perform such operation.

The HVAC Control notifies the power required by the HVAC device and waits for the available power release from the TCMS.

The HVAC waits then for the authorization token from the TCMS and does not perform any operation until the authorization is set. When the authorization token is received by the HVAC the startup process finalizes. From this moment, while the available power does not go below the power needed, the HVAC Control will continuously monitor the power the physical HVAC is consuming and will inform about it to the TCMS. The HVAC Control will also be ready to receive from the TCMS the global operational mode for the HVAC. The global operational mode allows to switch the HVAC device from the normal operation mode to other particular modes (e.g., test, washing or fast off, as defined inside the Application Profile) to carry out specific operations in the HVAC device.

Setting from global operational mode to the normal operation mode allows the user to select one of the available operation modes for a control zone (e.g., Standby, Automatic, Ventilation Only, …)

The HVAC Subsystem measures the temperature outside the vehicle using two sensors and calculates an average of these sensors' values. The measured outside temperatures are sent to the TCMS to be used for the overall mean outside temperature calculation. That mean outside temperature received from the TCMS is used for operating the HVAC.

Similarly, the HVAC Subsystem measures inside temperature of a comfort zone using two sensors and calculates an average of these sensors' values. The calculated temperature is sent to the TCMS to be used for the calculation. The temperature value received from the TCMS is used for operating the HVAC.

HVAC Subsystem's functional state is continuously monitored.

Temperature setpoint for a comfort zone can be defined within a certain range. This range depends on the chosen normative temperature curve and the maximum offset value.

The HVAC Control application, implementing the behaviour described above, has been implemented by Liebherr as a Matlab Simulink model. After an internal validation based on Matlab Simulink toolset in laboratory, ANSI C code has automatically been generated using Matlab Simulink code generation toolset. This automatically generated code has been tested on different hardware architectures, like X86_64, ARM and PowerPC.

It has first been integrated and tested into Liebherr's hardware platform (BK4R1). Then, Liebherr have tested it using the RTA-VRTE EAP environment from ETAS. Additionally, a separate version running natively on Windows has been introduced, in order to give the CONNECTA-2 partner CAF the ability do to some front up simulation tests before integrating it into their embedded architecture. Finally, the HVAC Control software has been natively validated on CAF's embedded architecture, running INTEGRITY RTOS. For this last activity, a library for PowerPC architecture has been created. This library has been compiled together with a test application and deployed into the CCU.

In this way, the compatibility and integrability of the code generated by Simulink from HVAC Subsystem model has been validated on the different platforms where it is later integrated in the different demonstrators.

Once, validated the HVAC model and the compatibility and integrability of the generated code in the different target hardware architectures, the integration of the HVAC Subsystem into the FDF implementations has been carried out, as described in the next section.

2.2 Functional Distribution Framework

The FDF provides an abstraction for the development of railway functions so that software applications are portable between different FDF implementations. For this purpose, the FDF defines an API that enables the subsystem providers to develop one software solution that may be integrated by different rolling stock system integrators.

Figure 2 depicts a conceptual view of the subsystem integration in a machine running a FDF instance. The lower green box represents the FDF implementation. The yellow box on top of it represents the FDF API, i.e., the set of C++ interfaces the subsystem developer may use to integrate the subsystem implementation into the FDF. The top side orange boxes represent the subsystem application. On the one hand, the box at the right side, Subsystem Control, represents the application that will control the physical system to be integrated. On the other hand, the box at the left side represents the part of the TCMS that will communicate with the subsystem control application to command and monitor the subsystem. The data interfaces for this communication are specified in the technical interfaces of the Application Profiles (see Figure 11 in section 3.1.1). Finally, the external light blue box represents a machine running the FDF and the integrated subsystem application.

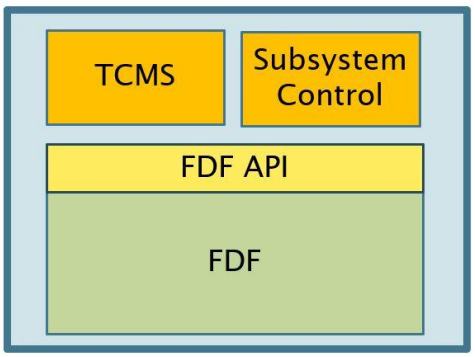


Figure 2: Subsystem integration in FDF

2.3 HVAC Subsystem Integration

Once the implementation of the HVAC Control application has been finalized and validated, the integration of the subsystem into the FDF has been carried out.

The aim of the HVAC Subsystem integration has been to demonstrate the portability of a single HVAC Subsystem implementation to different hardware/software platforms. This HVAC Subsystem integration has been developed as a proof of concept with the aim that other train functions besides HVAC could also be integrated by different rolling stock manufacturers into their solutions following the methodology used for HVAC.

To validate the portability of the HVAC Subsystem it has been integrated in two different implementations of the FDF. On the one hand, the AUTOSAR Adaptive Platform based FDF implementation from ETAS and, on the other hand, the CAF's proprietary FDF based on INTEGRITY RTOS. Therefore, a single implementation of the HVAC Subsystem from Liebherr, has been integrated into both solutions and tested in different validation scenarios.

The integration of the HVAC Subsystem has consisted in the integration of the HVAC Control Application on each FDF implementation, in the development of the TCMS that interacts with the HVAC Control Application and the deployment of both applications in the CCUs for the different validation scenarios.

In order to carry out this integration, in addition to the FDF, the Technical Application Profile [2] for the HVAC was required. The Technical Application Profile defines the data interfaces for the communication between the TCMS application and the HVAC Control application (see Figure 1)

- IConsistHvacOperation
- ISubsystemHvacOperation
- IConsistHvacConfortZone
- ISubsystemHvacConfortZone
- IConsistHVACLoadManagement
- ISubsystemHvacLoadManagement

These data interfaces define the inputs the TCMS application must provide to the HVAC Control Application before its execution and the outputs the HVAC Control Application will provide to the TCMS after its execution.

Two different communication patterns have been used for the integration of the HVAC Control in the FDF implementations. The FDF based on INTEGRITY RTOS uses a signal-based communication to read the inputs and write the outputs. The AUTOSAR Adaptive Platform based FDF instead uses a service-based based communication. The HVAC Control will subscribe to a service offering the input signals and provide a service containing the output signals.

Therefore, the integration of the HVAC Control application into each FDF implementation has been carried out following a different communication approach. However, a single HVAC Control application code has been integrated in both solutions. It is important to mention that the need of following two different integration strategies does not invalidate the major goal of demonstrating the portability of the HVAC Function between the two FDF implementations.

As depicted in Figure 2, the HVAC Control application is integrated with the FDF implementation through a standardized FDF-API. Once integrated the application, it is compiled together with the FDF implementation to generate the executable file that is deployed in the CCU. The same applies to the TCMS application that is deployed as another executable (in the same CCU or in a different one, depending on the target scenario).

Once validated the integration of the HVAC Subsystem into the two FDF implementations, the validation of the HVAC System has been carried out in different scenarios. A summary of the validation activities is explained in the next section.

2.4 HVAC System Validation

The last activity carried out in the HVAC case study was the integration of the HVAC System consisting of:

- the HVAC Subsystem
- the physical HVAC or a HVAC simulation model
- other HVAC relevant train subsystems

with the purpose of validating the HVAC Function in a simulated environment before the HVAC will be built in the real train.

The physical HVAC is ideally the real HVAC equipment providing heating, ventilation, and air conditioning in a closed chamber. This chamber reproduces the conditions of a train HVAC zone. Such a test bench is costly and not always available when new HVAC control software must be tested and validated. A model of the physical HVAC is alternatively used in Hardware-in-the Loop (HIL) and Software-in-the-Loop (SIL) systems.

Similarly, the train environment is typically modelled so that the HVAC can interact with the other relevant train subsystems.

Both the train environment model and the HVAC model (if needed) provide the plant model for the Simulation Host. How the Simulation Host is implemented depends very much on the railway supplier's validation strategy.

Each CONNECTA-2 partner has its own solution and therefore, the validation of the HVAC Function has been driven by the demonstrators defined by CONNECTA-2 project. Integration of the technology developed in Safe4RAIL-2 have been done in collaboration with CONNECTA-2 partners. These integrations have enabled to validate the developments and test their integrability in the final demonstrators to be built by CONNECTA-2, which must incorporate technologies developed by Safe4RAIL-2 project.

The CONNECTA-2 Urban Demonstrator defines a remote Hardware-in-the-Loop scenario where the CCU running the HVAC Subsystem will monitor and control a remotely located real HVAC device. The CONNECTA-2 Regional Demonstrator instead, defines several virtualized and simulated environments, where the HVAC device is substituted by an HVAC Simulation model.

Liebherr HVAC unit MACS 8.0 has been used for the validation with real HVAC device. Further information on the Liebherr HVAC unit MACS 8.0 can be found in [6] and [7].

e

Figure 3: MACS 8.0 module used for remote control

For the validation scenarios where an HVAC Simulation model has been used, Liebherr has provided CONNECTA-2 partners an FMU HVAC Simulation model, which each CONNECTA-2 partner has integrated in their Simulation Framework. In addition to the HVAC model, these validation scenarios have required a simulation of the thermal vehicle and environment simulation model. These simulation models have been provided by the vehicle manufacturer.

2.4.1 Data protocol for System Validation

The validation of the HVAC Function requires a communication between HVAC Control Application running in the CCU and the physical HVAC or the simulated HVAC.

This communication was carried out using Train Real Time Data Protocol (TRDP) and the following process data packets, where the exchanged signals are specified, were defined.

- PD_HvacCtrl_HvacIO: Process data packet sent from the HVAC control application to the HVAC (physical or simulated),
- PD_HvacIO_HvacCtrl: Process data packet sent from the HVAC (physical or simulated) to the HVAC Control application

Figure 4 shows, on the left side, the process data packets for the communication with the physical HVAC device. The same process data packets are required for the communication with a simulated model, as shown in Figure 5.

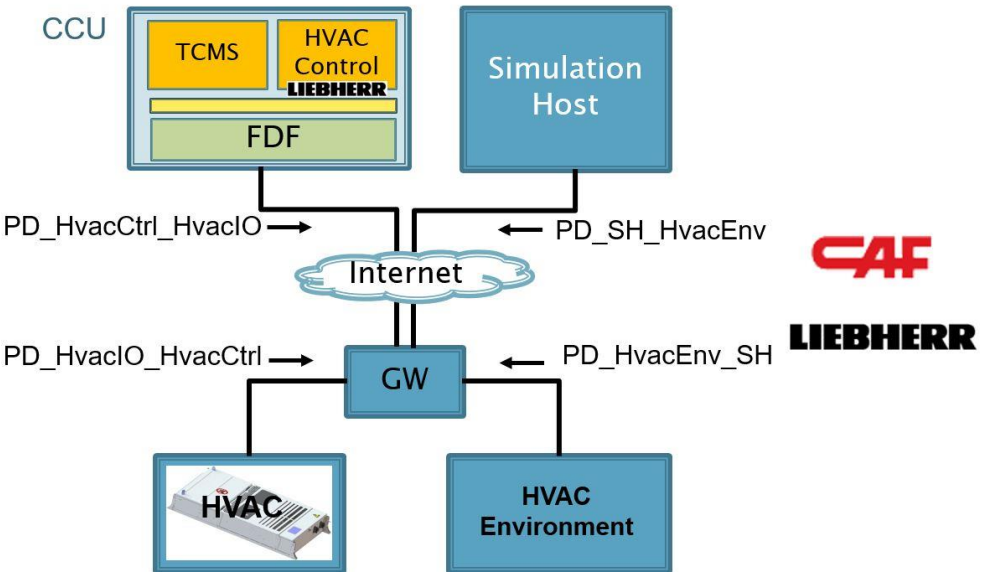


Figure 4: Communication between the HVAC Control and the Physical HVAC in the remote Hardware in the Loop scenario.

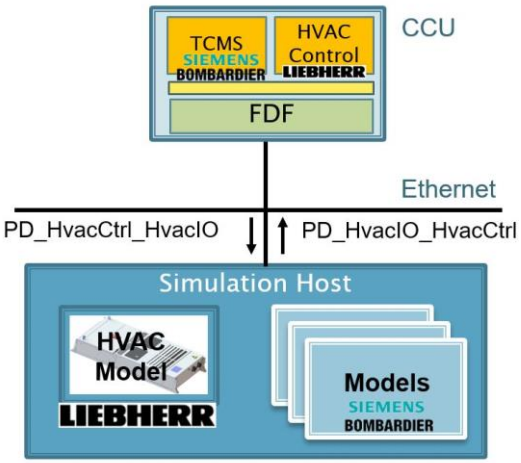


Figure 5: Communication between the HVAC Control and the Simulated HVAC Model.

In addition to these process data packets for the communication between the HVAC Control application and the HVAC, two additional TRDP process data packets have been defined to remotely control the HVAC environment via the Simulation Host. These process data packets are also shown in the left side of Figure 4.

- PD_SH_HvacEnv: Process data packet sent from the Simulation Host to the remote HVAC environment
- PD_HvacEnv_SH: Process data packet sent from the HVAC environment to the Simulation Host application to the HVAC (physical or simulated)

A set of different scenarios defined by CONNECTA-2 for the validation of the HVAC System where fully or partially implemented in Safe4RAIL-2 and are summarised below:

- Remote Hardware in the Loop validation

The HVAC Control application running in the CCU monitors and commands a remotely located physical HVAC unit thought the Internet.

- Hardware-in-the-Loop Validation
  The HVAC Control application and the TCMS run in a CCU and the HVAC Control application interacts with a HVAC Simulation Model running in a Simulation Host.
  Another alternative for Hardware-in-the-Loop Validation have also been implemented where the TCMS and the HVAC Control application are running in two separate CCUs.
- Virtual train validation
  All the components of the HVAC Subsystem run together with the HVAC model and additional environment components in a Windows PC. The HVAC Control application together with the TCM runs in a virtual CCU, while the HVAC model and other environmental models run in a Windows Simulation Host.

Chapter 3 Train Subsystem Integration Methodology

The methodology for the integration of a train application is based on the Application Profile concept defined by CONNECTA project. The Application Profile is the key element for the standardization of the train function. It defines an interface between the TCMS and a subsystem and for this purpose, it describes the signals that must be exchanged between TCMS and the subsystem.

The Application Profile defines the uses cases the train application must implement as well as the data structure that standardizes the information exchange for the communication between the Consist Level Control and the train subsystem. Therefore, a specific Application Profile must be defined for each train subsystem to be integrated.

To carry out the integration, the FDF provides an abstraction for the development of railway functions so that train applications are portable between different FDF implementations. The FDF enables the integration of the application that implements the Application Profile. The FDF provides an API, defining a set of C++ interfaces [2], which enables that subsystem providers develop a single software solution that may be integrated by different rolling stock system integrators.

The SF consists of a set of tools that enable to run simulation models of the train subsystem and the train environment allowing the communication of these simulation models with the train application to carry out the validations in virtualized environments.

3.1 Subsystem Integration into FDF

3.1.1 Subsystem Application Profile

An Application Profile is defined in CONNECTA project using a modelling approach based on SysML. The Application Profile defines the data interface between the TCMS and a subsystem. Therefore, for each subsystem that needs to be integrated, an Application Profile is required.

The Application Profile is defined as a set of a SysML interface blocks. An Interface block is a special kind of block for typing proxy ports. It contains a set of flow properties. A Flow Property signifies a single flow element to or from a Block, and thus the set of flow properties define the inputs and outputs for the subsystem.

In Figure 6 an example of an Interface block from [2] is depicted. This example shows the interface between the TCMS and the HVAC system, one of the two interfaces that make up the Functional HVAC Application Profile.

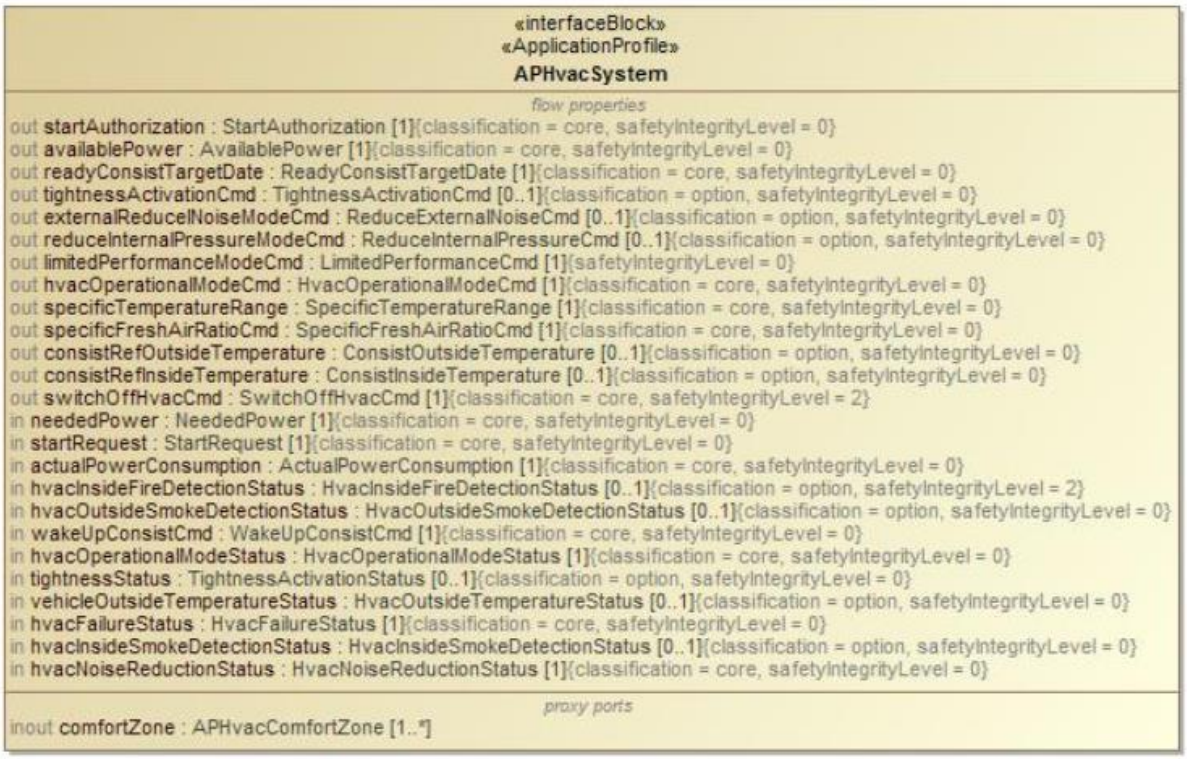


Figure 6: APHvacSystem, one of the Interface Blocks of the Functional HVAC AP (Example from [2])

Each Flow Property in an Interface Block must be specified as in, out or in-out, depending on it is an input, an output or is both an input and output for the subsystem. In this regard, an in typed Flow Property indicates that subsystem provides this information to the TCMS, while and out typed one indicates that the subsystem requires this information from the TCMS. Finally, an in-out typed Flow Property indicates that data flow is bidirectional, TCMS needs to provide it to the subsystem but the subsystem may update its value to send it back to the TCMS.

Each Flow Property is typed by a Signal. A Signal defines a unit of information that must be interpreted and thus transmitted as a whole. Thus a signal may contain one or many attributes.

◄|►	**Flow property** *Name : Type [Multiplicity]*	**Attributes** *Name : Type [Multiplicity]*
out	startAuthorization : StartAuthorization [1] safetyIntegrityLevel = 0	devId : String [] Device identifier that identifies the target device for this start authorization. tokenAuthorization : Boolean [] Represent the TCMS authorization given to Hvac system in order to start one of its heating/cooling equipment. • true • false
out	availablePower : AvailablePower [1] safetyIntegrityLevel = 0	powerAvailable : power[kilowatt] [] Indication on the power value available for the HVAC system.
out	readyConsistTargetDate : ReadyConsistTargetDate [1] safetyIntegrityLevel = 0	targetDateTime : DateTime [] Target date when the vehicle must have a proper environment to accommodate passengers. L year : Integer [1] The year (Gregorian calendar). L month : Integer [1] The month (1 through 12). L day : Integer [1] The day (1 through the number of days in month). Unit of time, equal to 24 hours [ISO 8601, 2.2.5]. L hour : Integer [1] The hour (0 through 23). Unit of time, equal to 60 minutes [ISO 8601, 2.2.4]. L minute : Integer [1] The minute (0 through 59). Unit of time, equal to 60 seconds [ISO 8601, 2.2.3]. L second : Integer [1] The second (0 through 59).
out	tightnessActivationCmd : TightnessActivationCmd [0..1] safetyIntegrityLevel = 0	activeTightness : Boolean [] Tightness activation request. • true • false

Figure 7: Flow Properties for HVAC System InterfaceBlock (Example from [2])

Each attribute of a Signal will be typed and this type could be basic data type provided by SysML (e.g. Boolean or Integer) or a user defined data type (e.g. celsiusTemperature, powerInKilowatt or activationStatus), that will provide a particular meaning to the attribute that compounds the signal.

The Application profile for one subsystem can define one or several Interface Blocks. When more than one Interface Block is defined for a subsystem, the flow properties included in each Interface Block will depend on the functions of the subsystem that will be managed through each Interface Block.

In addition to the SysML Interface Blocks, the Application Profile also includes the scenarios for exchanging signals between the TCMS and the subsystem. These scenarios are described using UML/SysML use cases. Thus, by means of the use cases an AP describes what a consist, that owns the TCMS and the subsystem can do.

The use cases describe the different functions to be performed by the subsystem. Firstly, the description of the use case explains how the interaction between the subsystem and the TCMS will take place. On the other hand, the Flow Properties involved are specified so that the subsystem can carry out the function. Finally, by means of an activity diagram, the behaviour associated with the use case is modelled, in which the interactions between the subsystem and the TCMS and the information exchanged in each interaction are explicitly described.

An activity in an activity diagram may send a signal, or receive a signal, from the TCMS to the subsystem or vice versa.

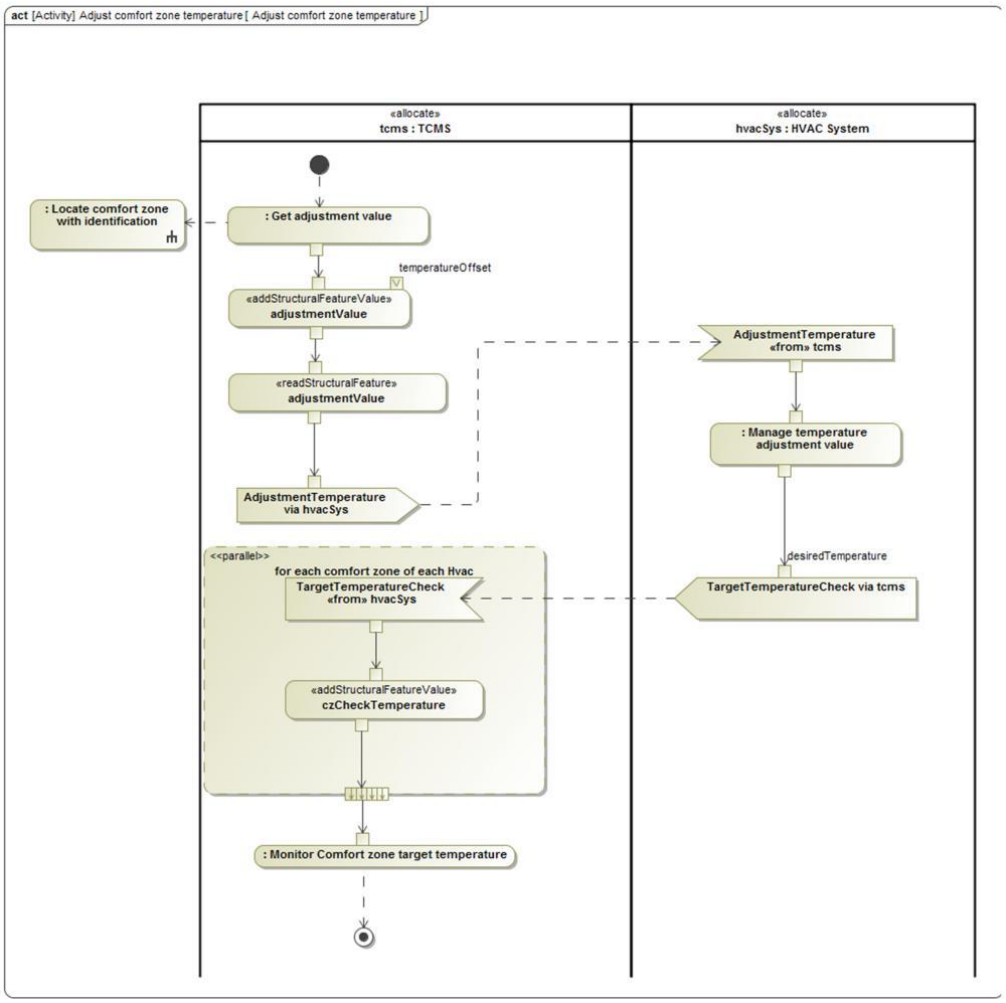


Figure 8: Activity diagram for Adjust comfort zone temperature (Example from [2])

Activity diagrams are complemented by sequence diagrams when it is necessary to provide more detailed information about the behaviour of the subsystem in a use case.

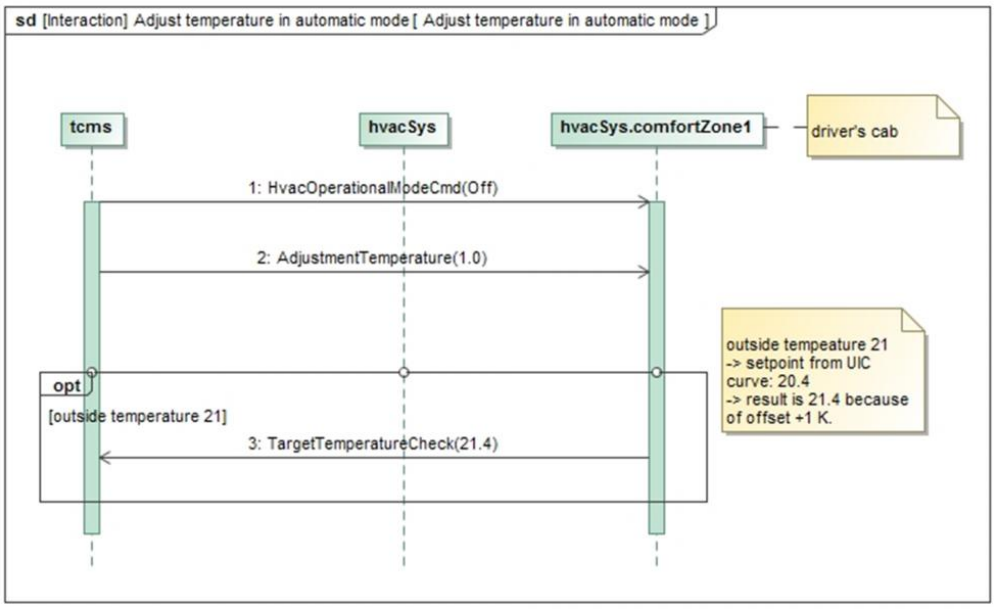


Figure 9: Sequence diagram for Adjust comfort zone temperature (Example from [2])

Thus, behaviour of a use case will be defined by means of an UML activity diagram which can be complemented with one or several sequence diagrams when additional detail of the behaviour of the subsystem is required.

Technical Interfaces

The SysML Interface Blocks in the Application Profile provide a general view of the signals to be exchanged between the TCMS and the subsystem. This information is refined in the technical interface, which decomposes the system in several interfaces.

Each interface groups a set of flow properties outgoing from the TCMS to the System or incoming from the system to the TCMS.

As a bidirectional communication is required between the TCMS and the subsystem, two different interfaces will be defined for each operation/function of the subsystem.

For the HVAC subsystem, integrated in the case study, the following technical interfaces were defined:

- ISubsystemHvacLoadManagement
- IConsistHvacLoadManagement
- ISubsystemHvacOperation
- IConsistHvacOperation
- ISubsystemHvacComfortZone
- IConsistHvacComfortZone
- ISubsystemHvacFireDetection

Interfaces starting with ISubsystem, define the interfaces that must be provided by the Subsystem and required by the TCMS, while those starting with IConsist, are the interfaces provided by the TCMS and required by the Subsystem. Therefore, IConsist interfaces define the outgoing flow properties from the TCMS to the Subsystem while the ISubsystem interfaces define the flow properties going from the Subsystem to the TCMS.

These interfaces can be grouped in pairs, since they define information to be exchanged between the TCMS and the subsystem for the implementation of the different functions by the subsystem. The suffix in the interface name indicates the function for which the interface will be used.

Figure 10 shows the interfaces required and provided by the applications for the HVAC function.

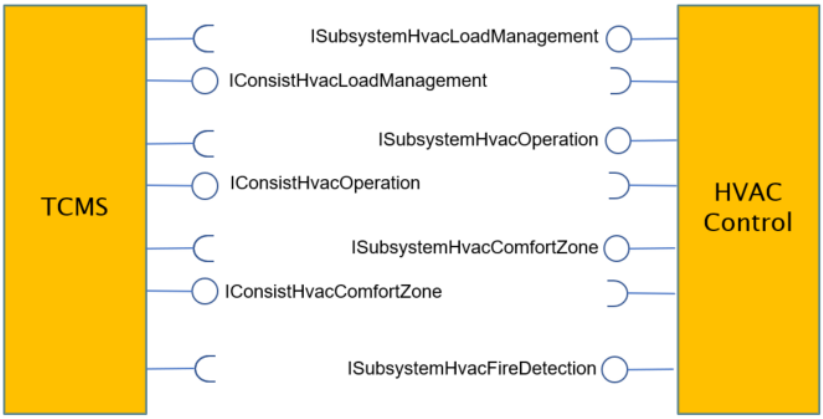


Figure 10: Interfaces for the HVAC Subsystem

In general, the application profile for a subsystem will define all the interfaces for such subsystem, those that must be provided by the Subsystem control application together with the interfaces required by the application, which will be provided by the TCMS.

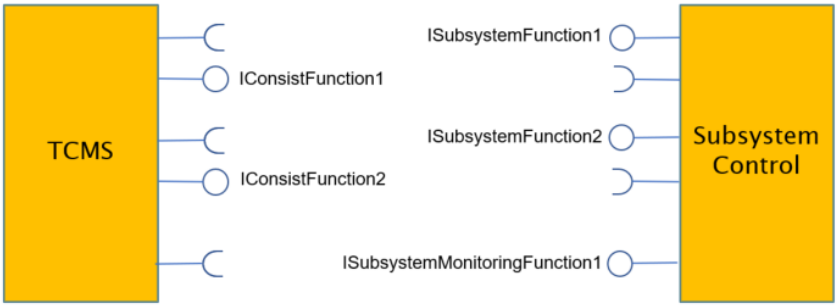


Figure 11: Interfaces for a Subsystem

Additionally, service identifiers for each interface are provided as well in the Application Profile.

These identifiers are required for the design of the services when a service-oriented architecture is used. Each service needs to be identified with a unique identifier that must be known by the components that will require that service. Thus, the service identifier will consist of a unique string and a unique integer value.

In the example below the service identifier for the IConsistHvacOperation interface defined in the application profile for the HVAC is shown.

Service identifier (string):	rail::subsystem::interface::hvac::IConsistHvacOperation
Service identifier (integer):	50656 (“A71800”H)

Figure 12: Example of service identifier (Example from [2])

Figure 13 shows a general view of the technical interfaces defined by CONNECTA-2 in the application profile for the HVAC.

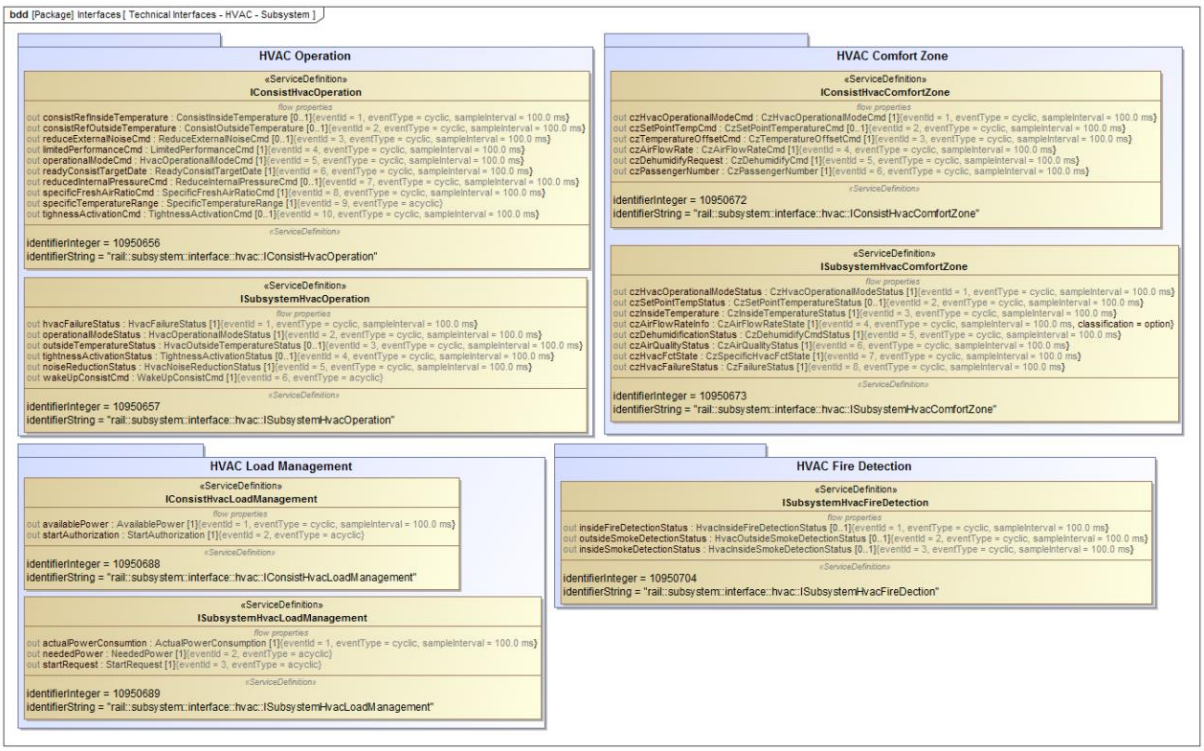


Figure 13: Technical interfaces for the HVAC function (Example from [2])

3.1.2 Train application components and interfaces

A train application, as depicted in Figure 14, is decomposed in three components (yellow boxes): TCMS, Subsystem Control and IO Device. The IO Device can either be a physical device or a plant software model to simulate the real hardware.

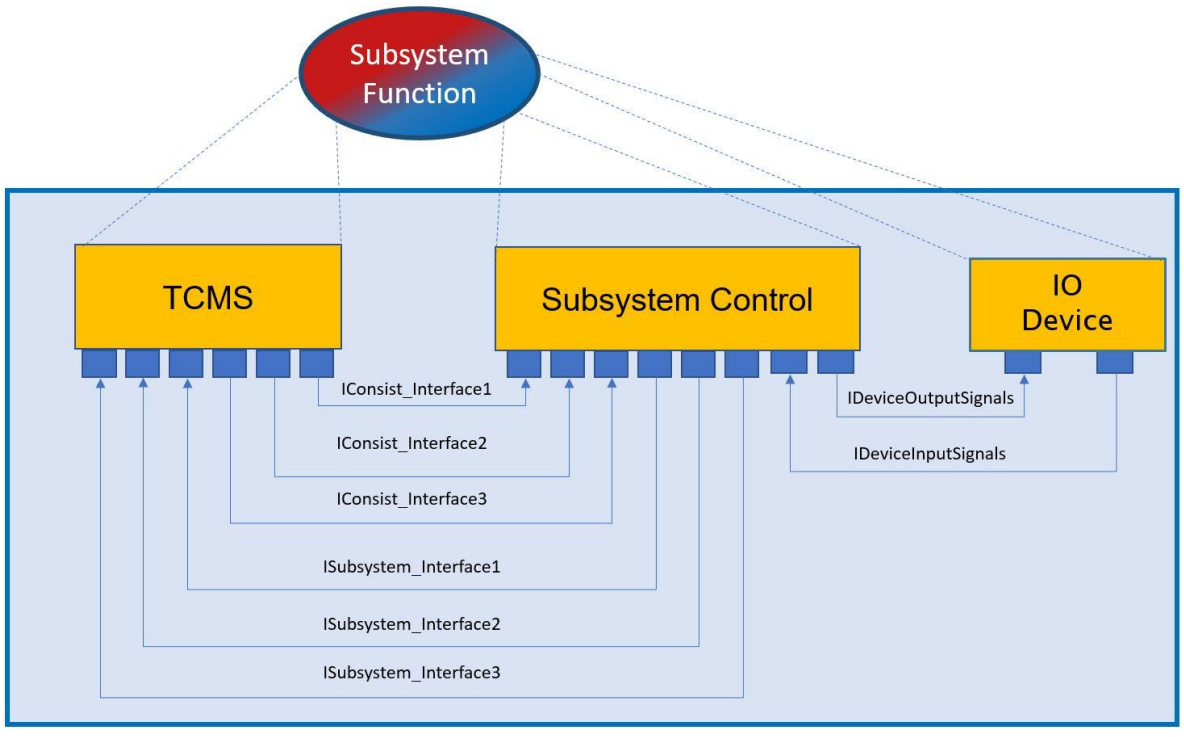


Figure 14: Components implementing a train application and communication interfaces.

The blue boxes attached to the components are ports to either offer services to other components or subscribe to services. The service interfaces used for the communication between the Subsystem Control and TCMS are defined in the Technical Application Profile.

The service interfaces between the Subsystem Control and the IODevice are not defined by the application profile, since those interfaces are typically train system supplier specific because they depend on the hardware of the device. Therefore, interfaces IDeviceOutputSignals and IDeviceInputSignals in Figure 14 are hardware platform specific and thus needs to be defined and provided by system supplier. The IDeviceInputSignals is used to retrieve the IO signals from the device hardware, whereas the IDeviceOutputSignals interface sends IO signals to the device hardware.

The IODevice is a plant model to simulate the behaviour of the subsystem hardware. This component will enable to speed up the testing during the implementation and integration phase, but when the real hardware is available is replaced by the physical hardware.

3.1.3 Subsystem integration

3.1.3.1 Signal based vs service-oriented communication

Signal-based communication architectures have long been used in communication protocols such as CAN and LIN, as well as in train control system communications. This communication is best suited for applications where hardware and software are closely coupled and communication between the ECUs are defined statically. In signal-based communication data is sent over the network whenever the data values are updated or modified. The sender is not concerned about whether the data is required by any node in the network.

Service-oriented communication architectures, enable a dynamical stablishment of communication between the communicating nodes. In this case data is sent by senders only if a receiver needs it. Therefore in these communication architectures senders need to know if there is any receiver waiting for the data.

The FDF API defines a set of functions to enable the integration of a control application on top of it. Not to restrict the FDF solutions to a single communication paradigm and to allow different FDF providers to offer the communication solution that better fits to an end user need, it does not determine how the Consist application and the Subsystem must communicate with each other. In this sense, a different communication approach has been followed in each of the FDF implementations used in the HVAC case study described in section 2.

CAF's proprietary FDF implements a signal-based communication approach while AUTOSAR Adaptive based FDF provided by ETAS implements a service-oriented communication.

Both communication approaches are based on the technical interfaces defined in the Application Profile, as described in section 3.1.1. These interfaces define the communication needs between the Consist application and the Subsystem application and indicate which data will be provided by each application to the other.

In the signal-based communication the interfaces defined in the application profile for each functional grouping are implemented as C/C++ data structures. A push-pull pattern gives the applications access to the input and output variables. Each application accesses to its inputs and output variables through those data structures that contain the signals defined in the application profile and the FDF is responsible for providing the inputs and extracting the outputs.

For the integration and deployment of the HVAC application into CAF's FDF, which follows a signal-based approach, CAF's proprietary tools have been used. Therefore, in order to preserve CAF's IP no further technical details on how it has been carried out is provided in this document.

On the other hand, for the service oriented communication, in addition to the data interfaces, service identifier information is also required, as mentioned in section 3.1.1.

Integration following service-oriented communication can be carried out using ETAS provided RTA-VRTE EAP toolset. Further detail on how service-oriented communication is implemented is provided in section 3.1.3.5, using an example.

3.1.3.2 Implementation and Validation of the Subsystem Control Application

The implementation of the Subsystem Control Application will consist in developing the software responsible of monitoring and controlling the subsystem to be integrated. This software is vendor specific and will depend on the behaviour of the subsystem.

The implementation of this code is independent of the FDF. It can be hand coded or it can be implemented using model driven development methodologies (as in the case of the HVAC), in which its behaviour is modelled and then code is automatically generated. In this sense, there is no restriction on which process must be followed for the implementation of subsystems control application. The only requirement is that final source code, either hand-coded or automatically generated from a model, must be implemented in C/C++ in order to be integrated into the FDF.

3.1.3.3 Integration of the Subsystem Control Application into the FDF

The integration of the Subsystem Control Application into the FDF will depend on the communication approach provided by the FDF implementation. Despite the communication approach followed by each FDF implementation can be different, a subsystem control code can be integrated in a FDF following a signal-based communication or in a FDF following a service-based communication approach. Depending on the communication approach, the steps that will be followed for the integration will vary slightly.

If signal-based communication approach is followed the InputSignals and OutputSignals data structures needs to be implemented. This data structures are based on the technical interfaces defined in the application profile. InputSignals data structure defines the set of input data required by the Subsystem Control application for its execution. Thus, this data needs to be provided by the FDF to the Subsystem Control application for its execution. OutputSignals data structure defines the set of output data that FDF will get provided by the Subsystem Control application after its execution.

The Subsystem Control Application is an application that will be cyclically executed, on a basis of a step by step execution. Thus integration of the Control Application on the FDF will consist in the following sequence of five steps that will be periodically executed by the FDF:

- fill the inputs data structure (InputSignals) with the current values of the variables managed by the FDF.
- execute *readInput* function of the subsystem providing as parameter the updated data structure filled in the previous step. The execution of this function will update the information in the subsystem and will get it ready for its execution.
- execute *step* function of the subsystem. Step function will execute the logic implemented for the subsystem and will update the values for the output variables that it internally uses.
- execute *writeOutputs* function of the subsystem passing as parameter an OutputSignals type data structure, which will be filled by the subsystem with the current values for the output variables.
- get the values provided by the subsystem on OutputSignals data structure and update the values for the FDF managed variables.

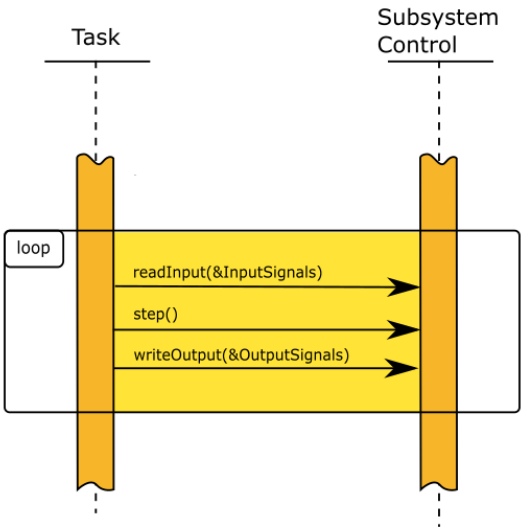


Figure 15: Integration of the Subsystem Control Application in the signal-based FDF

The FDF following the signal-based approach where the integration of the subsystem for the case study has been carried out is a proprietary solution which requires proprietary tooling for the integration and the configuration of the communications, thus no step by step detailed information about the integration process can be provided.

To integrate the Subsystem Control application following a service-based communication approach, as in the AUTOSAR Adaptive based FDF implementation, service interfaces compliant with the technical application profile needs to be implemented. Details on how these service interfaces are defined and deployed is given in section 3.1.3.5.

3.1.3.4 Test vectors for integration validation

The validation of the integration of a subsystem control application and the consist application that communicate with it is carried out using a set of test vectors.

Test vectors are based on the use cases described for the subsystem in the application profile. As a general rule, a test vector should be defined for each use case. However, for complex use cases, several test vectors can also be defined, in order to simplify the test cases.

Each test vector consists of a sequence of executions that must be carried out with the control application. For each execution the values that must be provided to the inputs (flow properties defined for required interfaces in the technical application profile) are specified first. Then the expected values for the outputs (flow properties for provided interfaces in the technical application profile) are specified. Each execution step, must be clearly defined in the test vector, including a step number counter and a description (e.g. No input changed, outside temperature changed, etc.).

This information must be provided by the subsystem provider, who knows how the subsystem must behave and it will be used by subsystem integrators to validate the integration. Therefore, these test vector specifications enable a subsystem integrator to carry out a step by step test execution to check that control application provides the expected output values when it is executed after providing some specific values to the inputs.

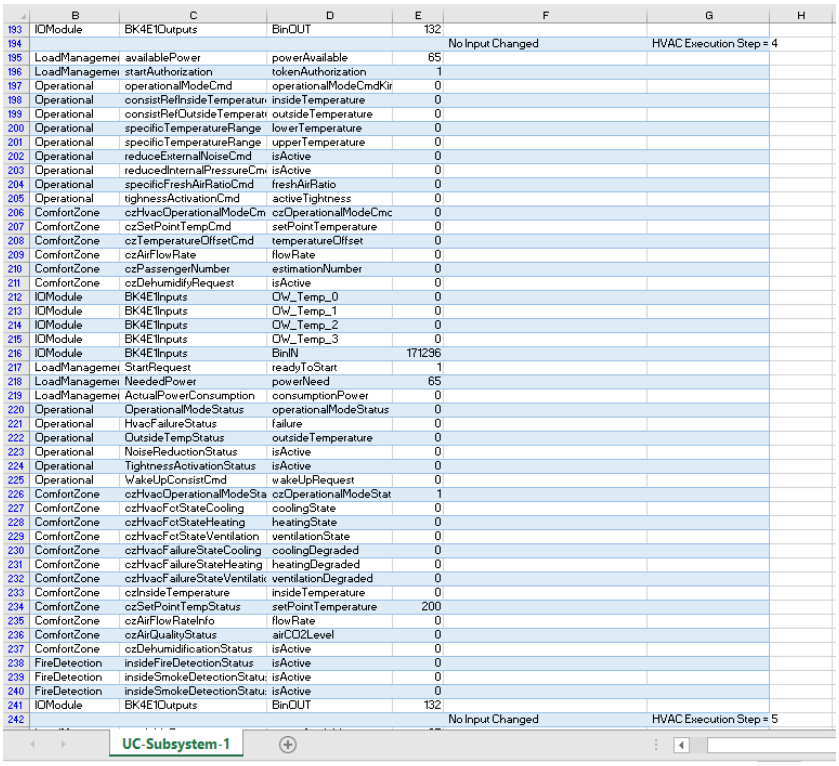
To unambiguously define a test vector, values for all the flow properties of each interface should be specified for each step.

The following information must be provided for the definition of a test vector:

Data	Description	Values
Input/Output	Defines if it is an input data that FDF must provide to the control application or it is the result that must be obtained after the execution of the control application	It can take these values: • IN: input data for the control application • OUT: output data provided by the control application
Interface	Interface from the Technical Application profile	The name of one of the interfaces from the application profile
Flow Property	Name of the flow property	The name of one of the signals from the interface selected
Attribute	Attribute of the Flow Property	One of the attributes of the signal selected in Flow Property
Value	Value of the attribute	Value that must be provided, when it is an input. Expected value, when it is an output.

Figure 16: Test vector definition data

All this information can be provided in an Excel file as depicted in Figure 17, which presents an example of a test vector for the case study presented above.



	B	C	D	E	F	G	H
193	IOModule	BK4E1Outputs	BinOUT	132			
194					No Input Changed	HVAC Execution Step = 4	
195	LoadManagemer	availablePower	powerAvailable	65			
196	LoadManagemer	startAuthorization	tokenAuthorization	1			
197	Operational	operationalModeCmd	operationalModeCmdKir	0			
198	Operational	consistRefInsideTemperatur	insideTemperature	0			
199	Operational	consistRefOutsideTemperat	outsideTemperature	0			
200	Operational	specificTemperatureRange	lowerTemperature	0			
201	Operational	specificTemperatureRange	upperTemperature	0			
202	Operational	reduceExternalNoiseCmd	isActive	0			
203	Operational	reducedInternalPressureCm	isActive	0			
204	Operational	specificFreshAirRatioCmd	freshAirRatio	0			
205	Operational	tightnessActivationCmd	activeTightness	0			
206	ComfortZone	czHvacOperationalModeCm	czOperationalModeCmc	0			
207	ComfortZone	czSetPointTempCmd	setPointTemperature	0			
208	ComfortZone	czTemperatureOffsetCmd	temperatureOffset	0			
209	ComfortZone	czAirFlowRate	flowRate	0			
210	ComfortZone	czPassengerNumber	estimationNumber	0			
211	ComfortZone	czDehumidifyRequest	isActive	0			
212	IOModule	BK4E1Inputs	OW_Temp_0	0			
213	IOModule	BK4E1Inputs	OW_Temp_1	0			
214	IOModule	BK4E1Inputs	OW_Temp_2	0			
215	IOModule	BK4E1Inputs	OW_Temp_3	0			
216	IOModule	BK4E1Inputs	BinIN	171296			
217	LoadManagemer	StartRequest	readyToStart	1			
218	LoadManagemer	NeededPower	powerNeed	65			
219	LoadManagemer	ActualPowerConsumption	consumptionPower	0			
220	Operational	OperationalModeStatus	operationalModeStatus	0			
221	Operational	HvacFailureStatus	failure	0			
222	Operational	OutsideTempStatus	outsideTemperature	0			
223	Operational	NoiseReductionStatus	isActive	0			
224	Operational	TightnessActivationStatus	isActive	0			
225	Operational	WakeUpConsistCmd	wakeUpRequest	0			
226	ComfortZone	czHvacOperationalModeSta	czOperationalModeStat	1			
227	ComfortZone	czHvacFotStateCooling	coolingState	0			
228	ComfortZone	czHvacFotStateHeating	heatingState	0			
229	ComfortZone	czHvacFotStateVentilation	ventilationState	0			
230	ComfortZone	czHvacFailureStateCooling	coolingDegraded	0			
231	ComfortZone	czHvacFailureStateHeating	heatingDegraded	0			
232	ComfortZone	czHvacFailureStateVentilati	ventilationDegraded	0			
233	ComfortZone	czInsideTemperature	insideTemperature	0			
234	ComfortZone	czSetPointTempStatus	setPointTemperature	200			
235	ComfortZone	czAirFlowRateInfo	flowRate	0			
236	ComfortZone	czAirQualityStatus	airCO2Level	0			
237	ComfortZone	czDehumidificationStatus	isActive	0			
238	FireDetection	insideFireDetectionStatus	isActive	0			
239	FireDetection	insideSmokeDetectionStatu	isActive	0			
240	FireDetection	insideSmokeDetectionStatu	isActive	0			
241	IOModule	BK4E1Outputs	BinOUT	132			
242					No Input Changed	HVAC Execution Step = 5	

UC-Subsystem-1

Figure 17: Test vector example

3.1.3.5 Implementation of the communication based on the technical application profile

The technical application profile defines the data interfaces provided and required by the application.

Therefore, based on the technical application profile the implementation of the communication will be carried out.

For the signal-based communication approach followed in the case study proprietary tooling and technology have been used, and no further detail will be provided in this regard.

On the contrary, the implementation for the communication for the service-based approach has been carried out by the ETAS provided RTA-VRTE EAP toolset and in the following section further details on how this implementation can be carried out is detailed.

Service Interface definition and deployment

The AUTOSAR Adaptive based FDF implementation uses a service-based communication approach. This implies that interfaces defined in the Subsystem Technical Application Profile (see section 3.1.1) must be implemented as Function Service interfaces.

The Standard AUTOSAR Adaptive Platform Representation (ARXML) is used for the definition of these Function Service interfaces. The ETAS proprietary Highlighted Application Design Language (HADL), helps to specify what kind of data structure is used and how the interfaces are put together in a readable way. The standardized AUTOSAR ARXML interface definition files are created and updated each time a HADL description is changed.

Both the AUTOSAR ARXML definition tool and the HADL are provided in RTA_VRTE EAP.

In the listing below an example of an HADL definition for a component that implements a train function is provided

```
import cta.org.IConsistFunctionA
import cta.org.ISubsystemFunctionA
import cta.org.IConsistFunctionB
import cta.org.ISubsystemFunctionB
import cta.org.IDeviceInputSignals
import cta.org.IDeviceOutputSignals

package cta.org{
        component subsystem {
                require consistFunctionA  for IConsistFunctionA
                provide subsystemFunctionA  for ISubsystemFunctionA

                require ideviceInputSignals for IDeviceInputSignals
                provide ideviceOutputSignals for IDeviceOutputSignals
        }

        component tcms {
                require subsystemFunctionA  for ISubsystemFunctionA
                provide consistFunctionA  for IConsistFunctionA

        }

        component iodevice {
                provide ideviceInputSignals for IDeviceInputSignals
                require ideviceOutputSignals for IDeviceOutputSignals
        }
}
```

Figure 18: Components implementing a train function and communication between ports using service interfaces.

The process of deploying service interfaces into machines is done in two steps. Interfaces are deployed in a first step and service instances are deployed then.

1. Interface Deployment (see Figure 19):
    a. The service interface is given the identification number, the so-called Deployment ID, which must correspond to the service identifier value defined in the technical interface in the application profile (Figure 12)
    b. The elements of the service interface (events, methods, fields) are provided an identification number (ID). Each Flow Property in the Technical interface will be mapped as an Event and an identification number (Event ID) will be provided.

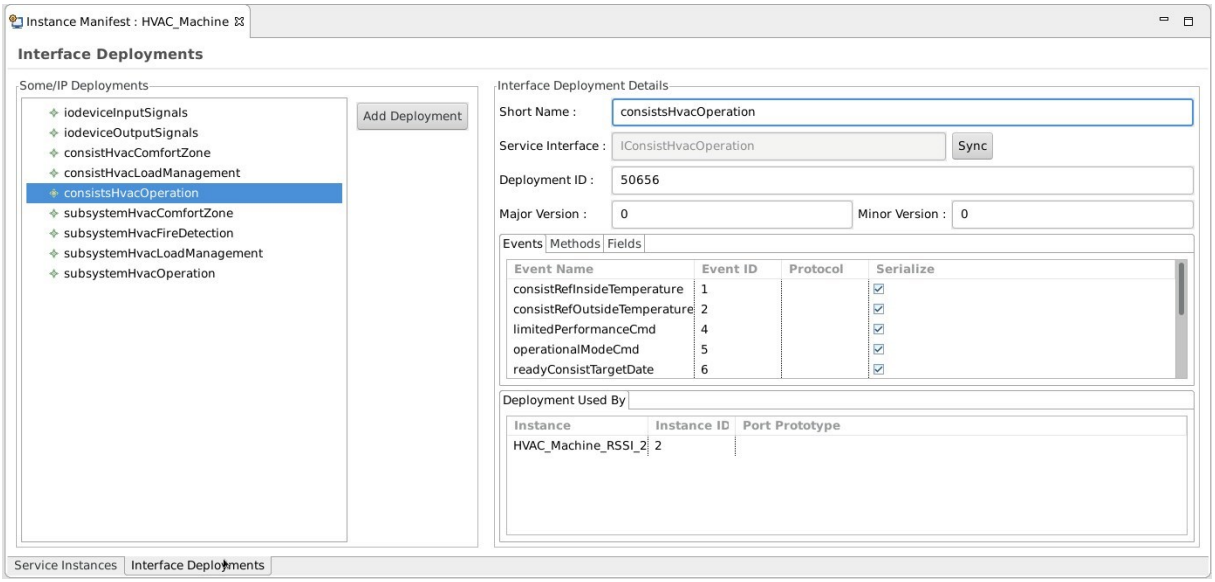


Figure 19: Deployment of an interface using RTA_VRTE EAP (HVAC example).

2. Service Instances deployment (see Figure 20):
    a. Interfaces are added to machines
    b. Instance IDs are provided to interfaces

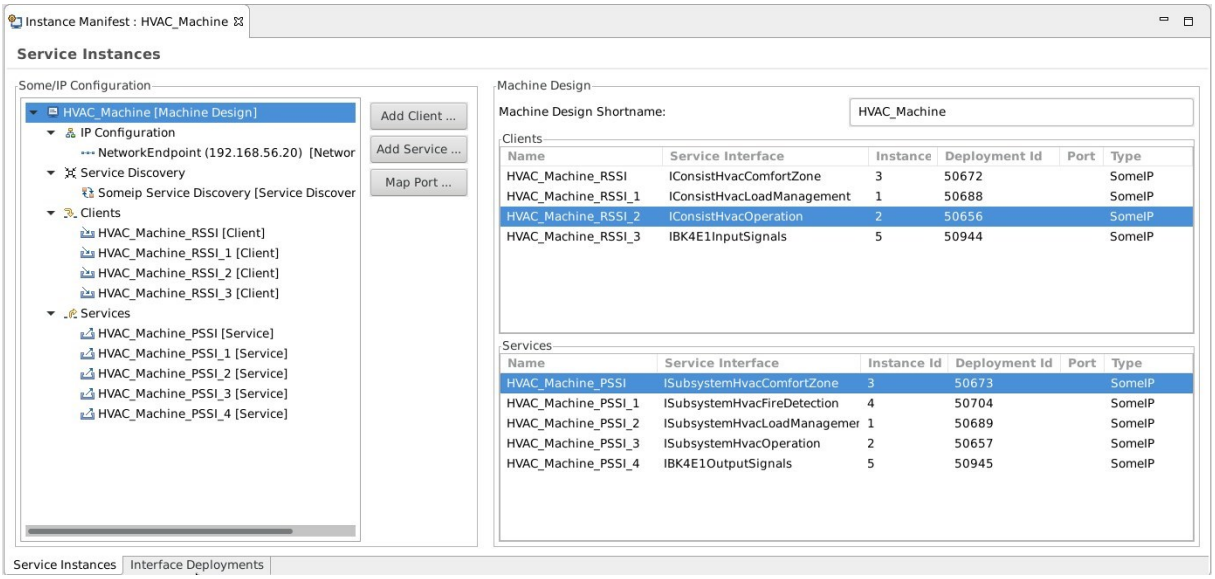


Figure 20: Service interfaces deployment using RTA_VRTE EAP (HVAC example).

The Technical Application Profile for a Subsystem must specify all the Deployment IDs and Instance IDs for the Subsystem. These values will be assigned to the service interface instances of the applications developed using RTA-VRTE EAP, the AUTOSAR Adaptive Platform FDF-based implementation provided by ETAS.

3.1.4 Deployment of the Applications to Machines

Once the service interfaces are deployed, the applications are deployed to machines. A machine is an instance of an FDF. The communication between applications running in different machines is done over Ethernet using a communication protocol, i.e. TRDP.

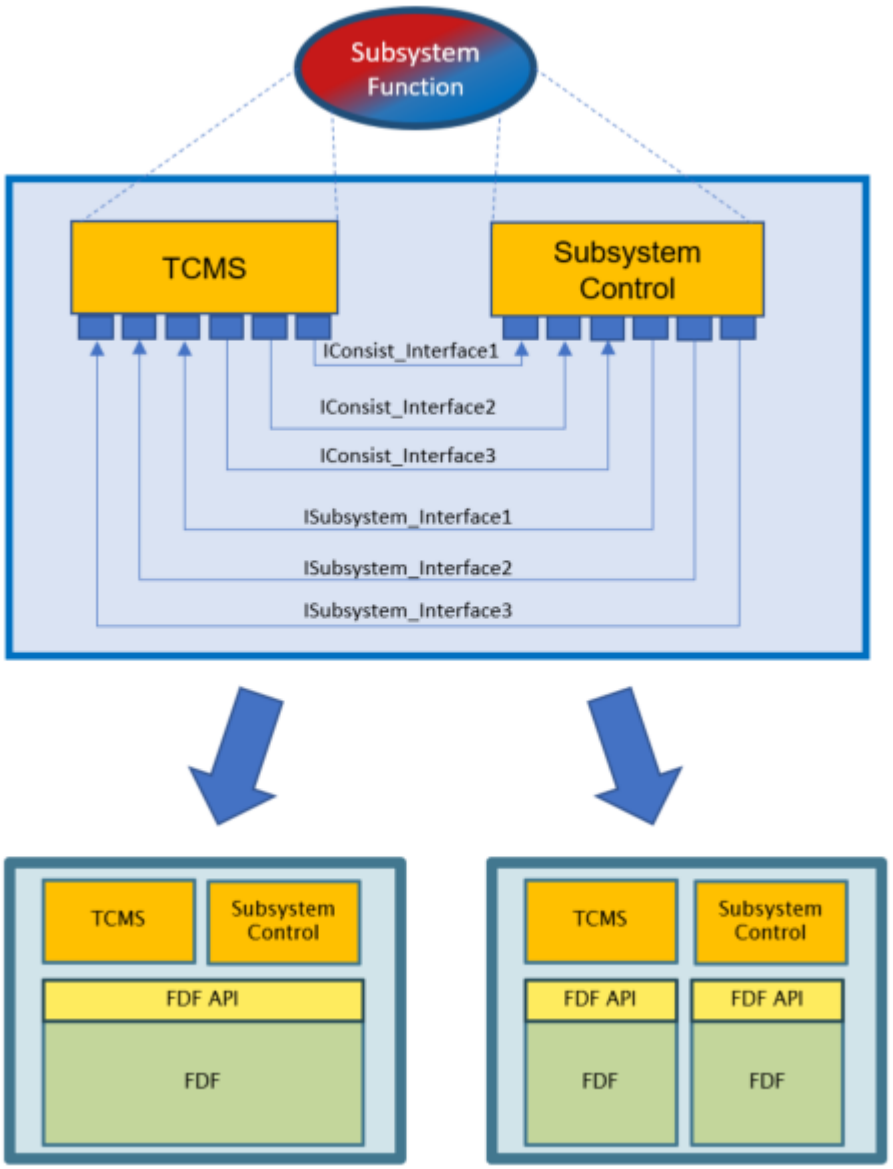


Figure 21: Subsystem Function deployed on one or two machines running on one CCU.

Applications deployed on the same machine will communicate using Inter-Process Communication (IPC) while communication between applications deployed on different machines is performed via communication protocols over Ethernet.

3.1.5 Toolset for the Integration

The integration process of train applications into the FDF using service-oriented communication, can be carried out using the RTA_VRTE EAP by ETAS.

RTA_VRTE EAP is the development environment for the integration of the subsystem function into the FDF as it provides the AUTOSAR Adaptive based FDF implementation and the FDF API to integrate applications in the FDF.

It also provides all the tooling required

a) to develop the control applications
b) to define the communications between the TCMS and the Subsystem Control application
c) to define how the applications must be deployed and make the deployments
d) to generate the virtual machines that will execute the applications and will enable to test them.

3.2 Subsystem Integration into SF

The integration of a subsystem into the SF enables to test/to run the subsystem into a virtual environment instead of in a real working environment.

This integration requires a simulation model of the system to be integrated so that the subsystem will monitor and control a simulated unit of the system.

In Safe4RAIL-2 this integration has been achieved by means of an FMU model. The HVAC unit simulation model was provided as a Functional Mock-up Unit.

A Functional Mock-up Unit is a software library in accordance to the specification of the Functional Mock-up Interface [3]. The Functional Mock-up Interface is a free and tool-independent standard that defines a container and an interface to exchange dynamic models using a combination of XML files, binaries and C code zipped into a single file (see [5] for details). The FMU can be loaded into several compatible tools to be integrated into a higher-level system simulation model. Many tools including Dymola, Matlab/Simulink National Instruments VeriStand, and Python (e.g. with PyFMI [5]) support the FMI standard [4]. FMI specifications are freely available under an open-source license so that there is no need to worry about additional license fees. FMU is well established in the industry and used by companies such as Siemens and Bosch.

FMI supports model exchange and co-simulation. For co-simulation, the FMU encapsulates the numerical solver for simulating the model. For model exchange the simulation relies on the numerical solver of the simulation tool. Within Safe4RAIL-2 project agreed with CONNECTA-2 project partners FMU version 2 for co-simulation has been used.

Specific licenses are not needed to simulate the model itself. Licenses may be required for the employed simulation tools.

In an overall simulation, in addition of the subsystem model, a train environment model is also required. The environment models are tightly dependent to the technology used in the implementation of the simulation host and will be provided by the vehicle manufacturer.

Figure 22 presents a simulation setup for the subsystem where the TCMS and the subsystem control application are deployed on the CCU to control the system model that is deployed in the simulation host, together with the train environmental model.

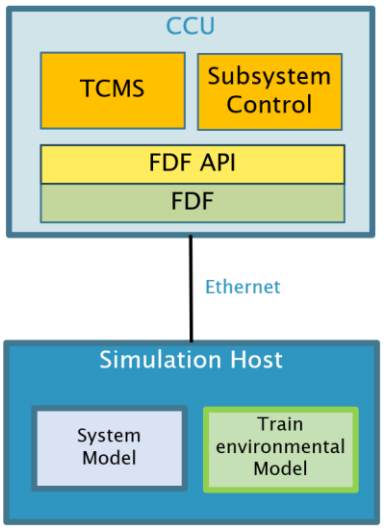


Figure 22: System model integration on the Simulation Host

3.2.1 Data protocol for simulation

Validating the train application using the simulation host requires a communication between the application and the model that simulates the physical device to be controlled by the train application.

For this purpose, the communication protocol (i.e. data packets) between the control application and the simulated device needs to be defined. This communication is carried out using Train Real Time Data Protocol (TRDP) over Ethernet and the following process data packets need to be defined:

- PD_AppCtrl_SubsystemIO: Process data packet sent from the control application running on the CCU to the device (physical or simulated)
- PD_SubsystemIO_AppCtrl: Process data packet sent from the device (physical or simulated) to the control application on the CCU

Additional process data packets needed to control other models (i.e., Train environmental model) need also be defined if they are required to interact with those models from the CCU.

3.2.2 Simulation model interfaces

The simulation model of the system must provide interfaces to communicate with the following entities via de Simulation Framework:

- Subsystem control application
- Vehicle/environmental models
- Simulation handler

The Simulation Framework must handle the communication between the system model and the control application running on the CCU. This includes packing/unpacking bus messages and reading/writing signals transferred via the bus from/to the system model. Furthermore, the Simulation Framework must also handle the communication between the system model and the environmental model.

The simulation model may provide an interface to indicate the status of the simulation, e.g. to indicate if the validity ranges of inputs and outputs are violated.

A simulation model could provide additional interfaces (inputs) to enable the injection of faults for the simulation.

3.3 Recommendation and Best Practices

This section summarises the main ideas described in the methodology with a special focus on the essential activities that need to be carried out to facilitate the integration of train applications.

The integration of a train application starts by defining in very deep detail in the application profile the use cases and the Interface Blocks.

The technical interface for the application must describe thoroughly all the interfaces required and provided by the application to be implemented.

The application profile in conjunction the technical interface defines the data that needs to be exchanged between the TCMS and the train application so this information is essential to define the communications properly.

In addition to the Application Profile, it is very convenient to define in an early stage of the development the test vectors associated to each use case defined in the Application Profile. This information becomes essential to validate the implementation of the train application itself as well as during the integration into the FDF and in a simulated environment through the SF.

If a service oriented approach is going to be used for the integration of the application into the FDF, the RTA_VRTE EAP provides in addition to the implementation of the FDF that fits this approach, all the tooling required for the definition of the communications, the integration of the application, capabilities for testing it in a running environment and for the deployment of the application following different deployment strategies.

Chapter 4 Summary and Conclusion

This deliverable has presented the methodology for developing train applications to be integrated into the Functional Distribution Framework (FDF) and Simulation Framework (SF).

The document first briefly describes the HVAC case study on how the HVAC function is integrated into the FDF and SF. The steps followed in the integration of this particular train function are generalized to describe the methodology that must be followed to integrate other train applications.

The methodology relies on the Application Profile concept, which defines the interfaces between the TCMS and the train application to be integrated, i.e. the data that TCMS and the application must exchange.

The communication between TCMS and the train application can be realized using two different communication paradigms: signal-based or service-oriented. Using one or the other will depend on the implementation of the FDF. The FDF APIs are generic and communication approach independent, which enables to integrate train subsystems using both communication paradigms. In the integration carried out in the HVAC case study both communication approaches have been used, over the same subsystem, validating the portability of the HVAC in two different FDF implementations.

The train application allows not only to control a physical device, but also a simulated one. For a simulated scenario, a simulation model of the device implemented as a Functional Mock-up Unit (FMU) and providing the standardized Functional Mock-up Interface (FMI) is required. This simulated scenario enables to conduct a train application validation using the SF instead of a real physical device.

The integrations carried out for the HVAC subsystem have enabled to validate both the signal-based and the service-oriented communication approaches, validating the portability of the subsystem in two different FDF implementations. In addition to being able to validate the subsystem integration using a real HVAC device, the use of FMU models has made it possible to validate the control application in various simulation environments thanks to the SF.

Chapter 5 Definitions and Abbreviations

5.1 Definitions

Application Profile	An Application Profile, according to CONNECTA project goal, describes a functional interface between the TCMS and a subsystem.
Functional Distribution Framework	Framework that aims to offer an execution environment for distributed TCMS applications up to SIL4 that ensures a strict spatial and temporal partitioning, location transparency and abstraction from the underlying network protocols and hardware, allowing at the same time the interoperability between different vehicle manufacturers
Functional Mock-up Interface	Open standard that defines a standardized interface to be used in computer simulations and for exchanging dynamical simulation models between different tools in a standardized format.
Functional Mock-up Unit	Model that conforms to the FMI standard.
INTEGRITY RTOS	Real-time operating system from Green Hills, used by CAF as operating system in its CCUs.
HVAC	Heating, ventilation and air conditioning is the technology of indoor and vehicular environmental comfort. Its goal is to provide thermal comfort and acceptable indoor air quality.
HVAC Simulation Model	Functional Mock-up Unit that simulates the behaviour of a physical HVAC.
HVAC Subsystem	Software that monitors and controls a HVAC (physical or simulated)
HVAC System	System consisting of a HVAC Subsystem, a HVAC (physical or simulated) and other HVAC relevant train subsystems.
MACS 8.0	Modular air conditioning system by Liebherr integrated as the remote physical HVAC equipment in the Urban Demonstrator.
Physical HVAC	HVAC equipment that provides heating, ventilation and air conditioning in a closed chamber that reproduces the conditions of a train HVAC zone.
Simulation Framework	Set of tools for train subsystem virtualization, communication emulation and simulation, which allows integrating the HVAC Subsystem in an early stage to a mixed validation environment.

Table 2: List of definitions.

5.2 Abbreviations

Abbreviation	Translation
AP	Application Profile
API	Application Programming Interface
AUTOSAR	AUTomotive Open System Architecture

Abbreviation	Translation
CAF	Construcciones y Auxiliar de Ferrocarriles, S.A.
CAN	Controller Area Network
CCU	Central Control Unit
CFB	Communication Framework Bridge
FDF	Functional Distribution Framework
FMI	Functional Mock-up Interface
FMU	Functional Mock-up Unit
HIL	Hardware in the Loop
HVAC	Heating, Ventilation and Air-Conditioning
IPC	Inter Process Communication
LIN	Local Interconnected Network
MACS	Modular Air Conditioning System
PD	Process Data
RTA-VRTE	Real Time Applications - Vehicle Run Time Environment, the AUTOSAR Adaptive Platform development environment from ETAS
RTA-VRTE EAP	Real Time Applications - Vehicle Run Time Environment Early Access Program
RTOS	Real Time Operating System
SF	Simulation Framework
SH	Simulation Host
SIL	Software-in-the-Loop
TCMS	Train Control and Monitoring System
TRDP	Train Real Time Data Protocol

Table 3: List of Abbreviations

Chapter 6 Bibliography

[1] CONNECTA, D4.3 - Application Profile Definition Guideline and Example, 2018. https://projects.shift2rail.org/s2r_ip1_n.aspx?p=CONNECTA

[2] CONNECTA-2, D1.2 - Definition of new FDF requirements and new Application Profiles
https://projects.shift2rail.org/s2r_ip1_n.aspx?p=CONNECTA-2

[3] Functional Mock-up Interface
https://fmi-standard.org, last accessed 17.09.2020

[4] FMI Tools
https://fmi-standard.org/tools/

[5] PyFMI
https://pypi.org/project/PyFMI

[6] Liebherr Modular Air Conditioning System
https://www.liebherr.com/shared/media/aerospace-and-transportation/transportation/downloads/products/liebherr-modular-air-conditioning-system.pdf

[7] MACS 8.0 brochure
https://www.liebherr.com/shared/media/aerospace-and-transportation/transportation/downloads/products/liebherr-macs-8.0-brochure.pdf

[8] RTA-VRTE EAP User's Guide
Safe4RAIL-2/CONNECTA-2 shared repository:
https://s4r2-cta2.technikon.com/02-FDF-SF/02-Demonstrators/Regional/Documentation/RTA-VRTE EAP/RTA-VRTE-EAP User Guide.pdf