Rail to Digital Automated up to Autonomous Train Operation

D36.1 – Demonstrator Specification

Architecture

Due date of deliverable: 30/11/2023
Actual submission date: 30/11/2023

Leader/Responsible of this Deliverable: DB

Reviewed: Y

Document status		
Revision	Date	Description
	From June 2023	Collaborative work and consolidation on concept boards
01	20/10/2023	Initial document version
02	23/10/2023	Internal review 1
03	30/10/2023	Internal review 2
03	01/11/2023	Submission for TMT Review
04	30/11/2023	Updated version after TMT Review

Project funded from the European Union's Horizon Europe research and innovation programme		
Dissemination Level		
PU	Public	x
SEN	Sensitive – limited under the conditions of the Grant Agreement	

Start date: 01/12/2022 (WP36 Kick-off 25+26/01/2023) Duration: 42 months

ACKNOWLEDGEMENT

This project has received funding from the Europe's Rail Joint Undertaking (ERJU) under the Grant Agreement no. 101102001. The JU receives support from the European Union's Horizon Europe research and innovation programme and the Europe's Rail JU members other than the Union.

Name	Company	Details of Contribution
Oliver Mayer-Buschmann	DB NETZ	Main author
Maik Fox	DB NETZ	Contributor Modular Platform
Benjamin Labonté	DB NETZ	Reviewer
Martin Kochinke	DB Systemtechnik	Reviewer
Zahoor Ahmed	DB NETZ	Contributor Onboard Communication
Friederike Maier	DB NETZ	Reviewer
Kai Schories	DB NETZ	Contributor Diagnostics
Thomas Martin	SBB	Contributor Physical Architecture
Rolf Mühlemann	SBB	Reviewer
Thomas Buchmueller	SBB	Reviewer
Wolfgang Wernhart	GTS	Reviewer
Reinhard Hametner	GTS	Reviewer
Ulrich Geier	Kontron	Reviewer
Manfred Taferner	Kontron	Contributor FRMCS-Onboard
Henrik Larsson	Trafikverket	Reviewer
Harald Roelle	SMO	Reviewer
Thomas Bernburg	SMO	Reviewer

ABBREVIATIONS AND ACRONYMS

Abbreviation	Definition
3GPP	3rd Generation Partnership Project
ATO	Automatic Train Operation
CCS	Control-Command and Signalling
CGMS	Configuration and Group Management
COTS	Commercial Off the Shelf
ERJU	Europe's Rail Joint Undertaking
ERTMS	European Rail Traffic Management System
ETCS	European Train Control System
EuLynx	European Initiative Linking Interlocking Subsystems
FRMCS	Future Railway Mobile Communication System
FVA	Functional Vehicle Adapter
GoA	Grade of Automation
IAM	Identity and Access Management
IM	Infrastructure Manager
ISO	International Organization for Standardization
IVV	Integration, Verification and Validation
MCx	Mission-Critical Services
MDCM	Diagnostics, Monitoring, Configuration and Maintenance
NeuPro	Neue Produktionsverfahren (Project)
OCN	Onboard Communication Network
OCORA	Open CCS Onboard Reference Architecture
OPC UA	Open Platform Communications Unified Architecture

Abbreviation	Definition
OS	Operating System
OSI	Open Systems Interconnection
PKI	Public Key Infrastructure
POSIX	Portable Operating System Interface
QoS	Quality of Service
R2DATO	Rail to Digital and Automated Train Operation
RAMS	Reliability, Availability, Maintainability and Safety
RBC	Radio Block Centre
RTE	Runtime Environment
RU	Railway Undertaking
SIEM	Security Information and Event Management
SNMP	Simple Network Management Protocol
SIL	Safety Integrity Level
SuC	System under Consideration
SOVD	Service Oriented Vehicle Diagnostics
SoW	Statement of Work
SRAC	Safety Related Application Conditions
TOBA	Telecom On-Board Architecture
TLS	Transport Layer Security
TRL	Technology Readiness Level
TCMS	Train Control and Management System
TSI	Technical Specifications for Interoperability
VIA	Vehicle Interface Adapter

Abbreviation	Definition
WP	Work Package

TABLE OF CONTENTS

TABLE OF FIGURES

1 INTRODUCTION

1.1 PURPOSE OF THIS DOCUMENT

This document aims to provide an initial architecture of the demonstrator setup, depicting and explaining the involved functional clusters, their main components, and interfaces.

It does not apply a sophisticated Model Based System Engineering (MBSE) approach and should be considered as an entry point for further architectural work, composing first architectural representations of the demonstrator, and aiding the comprehension for those who utilise the provided information.

This architecture intentionally stays quite high-level. It tries to incorporate aspects of existing pre-work known to the partners from initiatives as OCORA [15] and X2Rail4 [16] and ongoing specification work in the System Pillar.

Due to the iterative integration of functions over the implementation tasks, the architecture will need continuous refinement. This does not imply that further versions of this document will be published in future deliverables of WP36.

It is rather foreseen to use appropriate methods and tools, as far as they will be provided by R2DATO or internally by the participating partners. Therefore, this document won't be maintained and can't be considered as a working document.

1.2 BACKGROUND

It must be noted that there currently does not yet exist an aligned and standardised Onboard Architecture in the Railway Sector and first consolidation with existing and work in progress specifications as UNISIG SUBSET-26, -150, -147, -149, -119, -139, etc. is not expected before mid or end of the year 2024. Resulting specifications may still rather follow a bottom-up approach as currently started in the System Pillar Train-CS domain. The in parallel started top-town architecture following the MBSE ARCADIA methodology [17] may even take years.

In the Work Package 36 (WP36) "Onboard Platform Demonstrator" the partners Deutsche Bahn (DB), Ground Transportation Systems (GTS), Kontron (KONTRON), Schweizerische Bundesbahnen (SBB), Siemens Mobility (SMO) and Trafikverket (TRV) cooperate to validate the feasibility of a future-proof onboard platform suitable to host safety critical applications.

In the context of R2DATO this work package demonstrates a concrete implementation of a modular computing platform as envisaged in WP26 to a Technology Readiness Level (TRL) of 5/(up to 6)[1], enhanced by onboard connectivity to a train adapter, FRMCS communication modules and shared services (e.g., diagnostics and maintenance).

[1] *"TRL5 Pre-prototype tested in lab: Integration of components with reasonable and realistic supporting elements for testing in a simulated environment. High fidelity is achieved in laboratory."*

"TRL6 Prototype tested in relevant environment: The technology is tested in a relevant environment. It starts to be considered as a representative prototype to be tested in a high-fidelity laboratory environment or in a simulated operational environment".

The project timeline started in December 2022 (M1) and ends in May 2026 (M42). This document is provided within the deliverable D36.1. For a better understanding of the project scope, objectives, partner setup and timeline, the introduction document “Demonstrator Specification” and the “Statement of Work” (SoW) may be studied first.

1.3 PURPOSE AND SCOPE

Next to this document, the deliverable D36.1 contains two additional technical documents: the “System Definition” and the “User Stories & Test Cases”.

As depicted in chapter 2 of the “Demonstrator Specification”, the content of those two documents serves as the basis and main input to this document. While the System Definition describes the system as a black box, deriving system capabilities and describing relevant actors and external interfaces, the User Stories provide a more detailed view on the demonstrator objectives.

The Architecture combines this input and extends it to some more detailed representations, revealing a certain degree of internal structure, currently known and committed between the project's partners.

The main objective of this document is to serve as an initial reference, enabling the future realisation of the demonstrator, aligning on a set of applicable architectural and structural content and to describe relations between functional clusters and contained components. To obtain these goals, the architecture document contains a set of views, accompanied with respective descriptions based on the current knowledge.

The intention is to provide a fundament for future design work on the contained sub-system with the aim to propose a structured decomposition on both, logical (functional) and physical layers.

On some focus areas it provides more detailed views. It shows how to integrate newly to be developed and designated hardware and software components of existing solutions provided by the participating partners.

The definition of detailed interface descriptions, system behaviour as well as a structured identification of design constraints can’t be achieved at this stage. Such activities must be postponed to the later implementation phases. Further analysis and investigation will be necessary. Nevertheless, some degree of guidance for a future choice of technology and appropriate solutions shall be achieved.

For additional insight into steps and processes usually applied for the treatment of architectures, please refer to Annex A:. However, not all items listed in the annex are in the scope of this document.

2 STARTING POINT FROM THE SYSTEM DEFINITION

Starting point for the architecture is the Figure 5 in the System Definition's chapter "System Boundary", that identifies the main external actors and Interfaces of the System under Consideration (SuC). Omitting the interfaces and laboratory specific actors here, Figure 1 reduces to the external actors, expressing the system boundary and functional clusters for the onboard part in focus of the demonstrator.

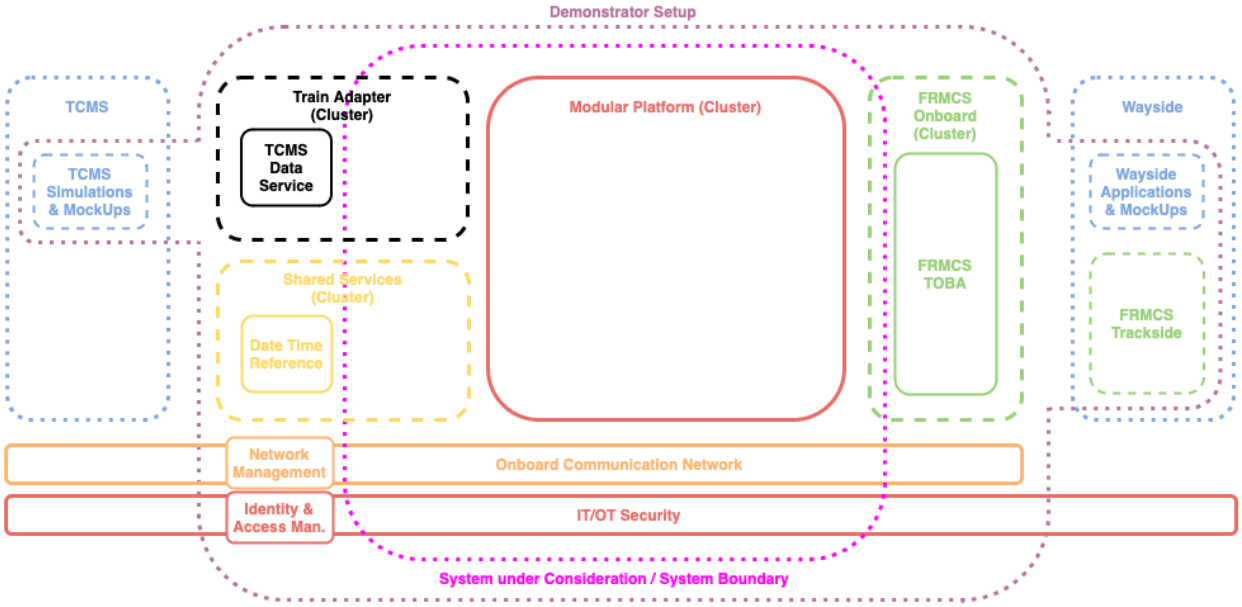

Figure 1: External Actors and Functional Cluster

The external Actors (and external Interfaces) get explained in detail in the System Definition document. In the next chapter a first look inside the System under Consideration (SuC) will be taken. There will be introduced (logical) views and logical components, to work out their relationships to each other and to the external actors they interact with.

3 LOGICAL ARCHITECTURE

3.1 PURPOSE

The logical architecture is the decomposition of the system into functional components. This can be done on system and sub-system level but from a technology-independent perspective. It defines logical components and their relation and interaction to each other. Furthermore, borders and constraints shall be considered, and interfaces shall be defined. As already explained in the introduction chapter the level of detail remains limited in this document. At least some views will be provided, working out explicit areas of interest, that have been discussed in task 36.1 between the participating partners.

3.2 INPUTS

Main inputs for the logical architecture are:

- System requirements and constraints
- System actors and their interaction with the system
- System capabilities and functions to be transferred to the logical architecture

3.3 STEPS TO BE CONDUCTED

- Define logical components and describe their functionalities
- Transition of all the actors to keep the function allocations
- Transition of external interfaces into the logical architecture
- Transition of all system capabilities to transfer the involved functions
- Definition of (internal) interfaces. Detailed definitions won't be realised for now, this must happen in the further steps along other activities relevant for the logical architecture.

3.4 MAIN LOGICAL COMPONENTS

The following Figure 2 introduces the main logical components of the SuC. This high-level overview will be further detailed in separate views on explicit areas of interest later.

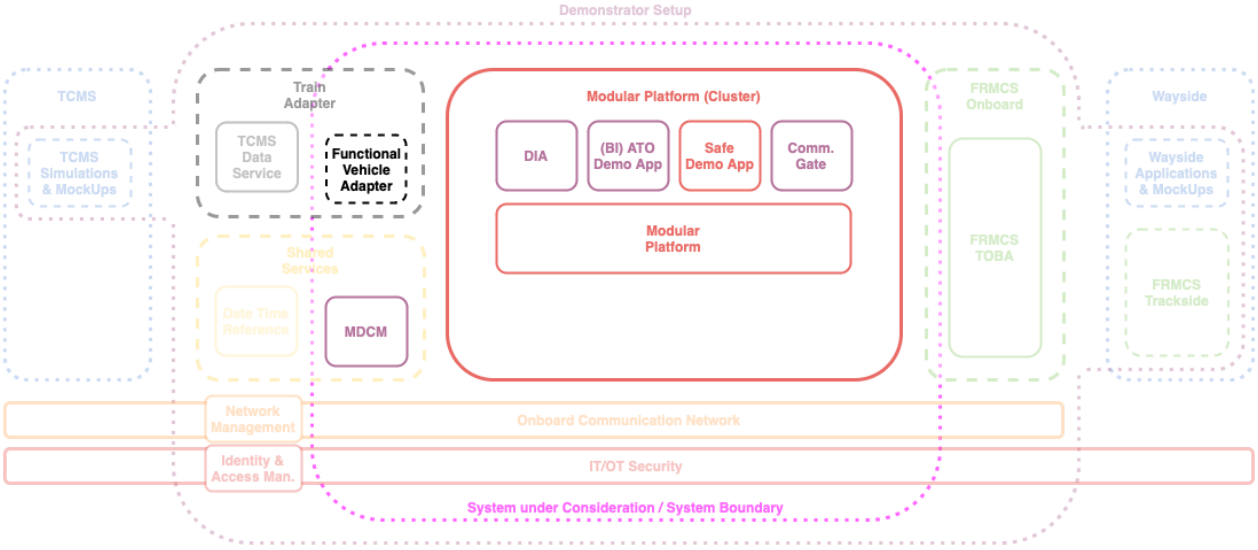

Figure 2: Logical Components of the SuC

The main motivation for realising a Modular Computing Platform is the ability to host mixed-SIL Onboard Applications in a modular and extendable way. Applications dedicated to a Safety Integrity Level up to SIL4 (e.g., ETCS) and basic integrity (e.g., ATO) will make use of services and abilities of the Modular Platform to showcase safe and secure hosting, communication, and the utilisation of diagnostics functions.

Access to the Onboard Communication Network (OCN), the ability to exchange data with the TCMS systems and transparent communication to trackside over FRMCS shall be realised based on the following logical components:

3.4.1.1 Modular Platform

In the context of WP36 the term Modular Platform refers to a functional cluster, comprised of hardware and software, on which safety related and non-safety related functional applications are hosted. The realised Modular Platform implementation of WP36 will be based on existing products and solutions provided by participating partners and shall follow and implement architectures, specifications and requirements for development, testing, and certification of modular and safe onboard computing platforms, considering available specifications of WP26, incl. standardised approaches for monitoring, diagnostics and configuration management.

3.4.1.2 Communication Gate

Component to enable communication between applications hosted on the Safe RTE and outside entities. Those may be applications running on the Basic Integrity Platform (please refer to chapter 3.4.1.7), applications deployed on separate hardware (communication over the OCN) or even on the wayside. A Communication Gate may be realised for specific use cases, granting accesses to OCN, FRMCS, potentially realising the integration of needed protocols.

A concrete example for FRMCS is explained in chapter 3.6.4.1.

3.4.1.3 Diagnostics Function

Platform services for diagnostics are services provided to applications e.g., for health and performance monitoring, logging, configuration, and maintenance purpose. A generic approach is foreseen in WP26 specification work in cooperation with System Pillar domains (Transversal CCS, potentially Computing Environment domain).

Existing platform specific solutions might be used for the implementation as a starting point to gain input for the specification work and requirements.

Diagnostics Modular Platform Services are responsible for the acquisition and processing of health and performance information provided by applications and system functions of the Modular Platform.

The interaction between the diagnostics services, the Modular Platform and hosted applications shall be based on existing and planned specification work of WP26 (in particular on deliverables D26.2 [1] and D26.3 [2]), adapting existing diagnostics interfaces and functions of the used platform implementation.

3.4.1.4 Monitoring, Diagnostics, Configuration and Maintenance (MDCM)

MDCM (Shared Services, [13], [14]) are diagnostics, monitoring, configuration and maintenance services, potentially provided to multiple onboard domains, e.g., for remote diagnostics purpose and software update. Logically the MDCM is located in the Shared Services cluster, which shall not imply, that it can't be realised/deployed on top of the Modular Platform.

3.4.1.5 Functional Vehicle Adapter

The Functional Vehicle Adapter (FVA) [12] is a software function of the Train Adapter cluster. Its purpose is to realise standardised interfaces between the CCS and TCMS domain for vehicle functions and vehicle information. Existing vehicles implement proprietary TCMS interfaces (not fully compliant to TSI-CCS Subsets SS-034, SS-119, SS-139 and SS-143, that do exist in different versions or are not fully specified yet). The FVA shall map, on a functional level, the commands sent, and the information received from specific TCMS systems as a configurable software function that shall be adaptable through parametrisation, in order to enable migration and simplified integration of future CCS/ETCS systems into vehicles. Logically the FVA is located in the Train Adapter cluster, which shall not imply, that it can't be realised, deployed on top of the Modular Platform.

3.4.1.6 Safe Demo Application

Safe Demo Application will be developed and integrated according to specified programming models and SRACs to showcase the ability of the modular platform to host safe onboard functions up to SIL4. The demo application will only show limited functionality. The goal is to embed and show the interactions with other parts of the demonstrator.

The ETCS application (as the Safe Demo Application) shall connect to a mock-up wayside application (simulated RBC over FRMCS) to obtain the authorization to move. With emulated location data the demonstrator could virtually travel and could test the proper behaviour with several test cases, involving the TCMS and the emergency brake.

3.4.1.7 Basic Integrity Demo Application

The ATO application (as the Basic Integrity Demo Application) could get a mission profile to run, supervised by the ETCS application. Running the mission profile would make use of the TCMS (throttle and brake, starting only when all doors are closed).

3.5 FUNCTIONAL CLUSTER MODULAR PLATFORM

3.5.1 Modular Platform View

The following view provides some first insights to the Modular Platform cluster, introduces some additional logical components and the internal interfaces as specified in the System Pillar Computation Environment domain. Further information can be found in chapter 3.5.4 and the specifications provided by the R2DATO WP26 [1], [2].

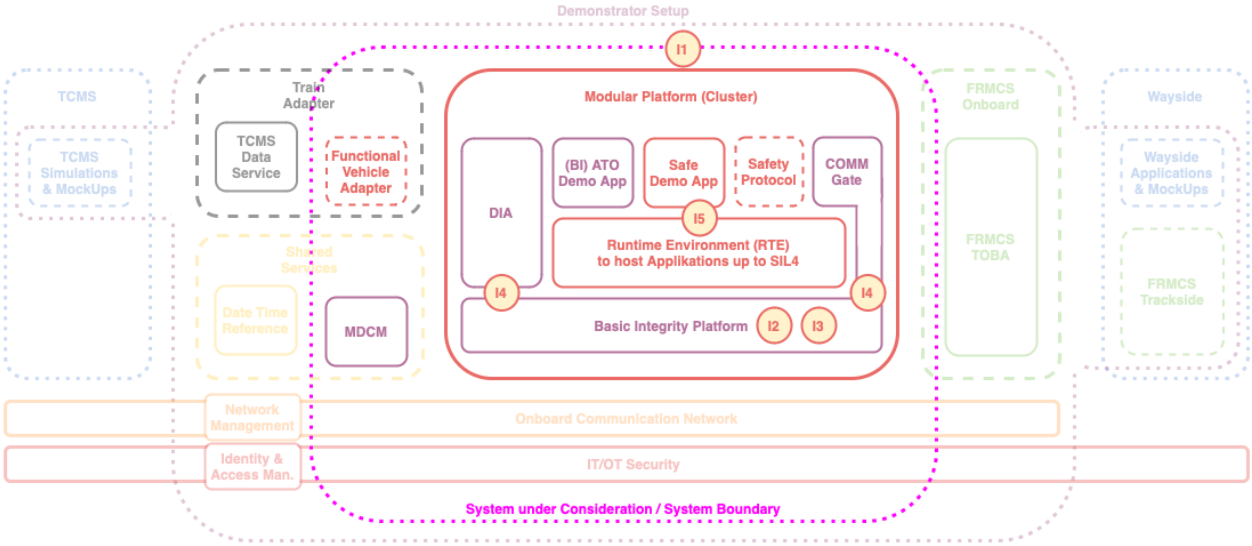

Figure 3: Modular Platform View

As already mentioned, this view shall not imply, that the FVA and MDCM can't be hosted on top of the modular platform. The above view shall not present any physical deployment. Later, a physical architecture option may depict one or both components on top of the above introduced RTE components, if feasible.

Nevertheless, Figure 3 provides a first interface mapping to logical components and a high-level idea on the internal architecture. This little inconsistency (not a pure logical architecture) is foreseen for the sake of understanding and exceeds the view by arranging the components. Red colouring is an indication for safe components while purple implies rather basic integrity. The FVA is here coloured in red due to its safety functions but stays logically inside the Train Adapter cluster, while the MDCM components has been coloured purple to indicate its basic integrity.

3.5.2 Context of the Modular Platform

The Modular Platform cluster is representing the core of the SuC. Functions realised within the Modular Platform cluster interact with fairly all described external actors, using all external interfaces as defined in the System Definition in chapter "System Boundary".

3.5.3 Logical Components of the Modular Platform

3.5.3.1 Safety Protocol

An implementation of a Safety Protocol is needed for safe end-to-end communication between safe applications. Due to the lack of a supplier for onboard applicable Safety Protocols as EuroRadio (refer to SS-037-2), another appropriate and existing safety protocol might be utilised. Safety protocols may as well be omitted or implemented in a simplified way, potentially not realising a full safety link between applications.

3.5.3.2 Runtime Environment

Runtime Environment (RTE up to SIL4) adds a safety layer to enable the hosting of applications with a Safety Integrity Level up to SIL4, voting, and safe and secure communication between mixed-SIL applications running on top.

3.5.3.3 Basic Integrity Platform

The Basic Integrity Platform builds up the basis for the Modular Platform. It consists of hardware (preferrable redundant COTS) and basic software, providing the ability to integrate basic integrity applications. It contains all needed Operating System (OS) services, realises hardware access, real time scheduling, (and potentially containerisation or virtualisation technologies, etc.) and is serving as a base system for guests such as the Safe Runtime Environment (RTE).

3.5.3.4 Communication Gate

This component has already been introduced in chapter 3.4.1.2. It enables communication between applications hosted on the Safe RTE, the Basic Integrity Platform and outside entities. Communication partners may be applications running on the Basic Integrity Platform of the same or of different hardware instances or even on wayside assets.

3.5.4 Interface Definitions

The relevant internal interfaces of the Modular Platform are adopted from current System Pillar Computing Environment output and described in the deliverable D26.2 [1].

The interfaces shown in Figure 3 are explained in the following subchapters.

3.5.4.1 I1: External Diagnostic, Logging, Orchestration & IT Security Interface

This interface is actually a group of interfaces needed for integration of the Modular Platform into its environment. I1 is covering aspects such as providing diagnostics data of the Modular Platform, e.g., health data, access to logging information, orchestration control and IT security, e.g., identity and access management.

3.5.4.2 I2: Hardware Abstraction Interface

This interface will provide means to decouple generic software implementation from properties of the actual hardware implementations available.

3.5.4.3 I3: Virtualisation Interface

Here, together with a virtualisation layer, a common interface for the RTE running on top of this layer is defined. This way, different RTEs and virtualisation layers can potentially be used in a Modular Platform.

3.5.4.4 I4: Basic Integrity Platform Independence Interface

A defined programming interface and a set of assumptions build the basis for potentially portable basic integrity applications, when they adhere to the specification of this interface. This interface provides basic and advanced functions, such as accessing platform functionality, communication between applications, thread management, file system access and network access, to name a few. To allow certification of applications, the usage of this functionality needs to adhere to a given programming model and other assumptions.

3.5.4.5 I5: Safety Platform Independence Interface

Like I4, but more restricted in scope and functionality to allow safe applications up to SIL4 to benefit from a common interface.

As shown in Figure 3, the location of the interfaces I2 and I3 is considered to be part of the “Basic Integrity Platform”, I4 and I5 are the actual interfaces between Basic Integrity Platform respectively the RTE and their hosted applications, and I1 is an external interface, e.g., towards the MDCM.

Note: Looking at the sources cited at the beginning of this subchapter, the definition of the interfaces is still at an early stage representing concepts rather than specifications ready for implementation. As such, this project will map these interface concepts into the architecture to show where they are applicable and decide within the appropriate tasks how the implementation is handled.

3.6 Functional Cluster FRMCS Onboard

The majority of FRMCS components can be considered rather as FRMCS infrastructure or as part of the external actor FRMCS TOBA. FRMCS is a quite sophisticated topic, and the reader shall be motivated to study related specifications for a better understanding.

No major FRMCS functions will be implemented inside the SuC. There are components that intend to use FRMCS as “FRMCS Clients” via the standardised OBapp interface. Please refer to chapter “FRMCS Interfaces” below for further details.

3.6.1 View on FRMCS Onboard

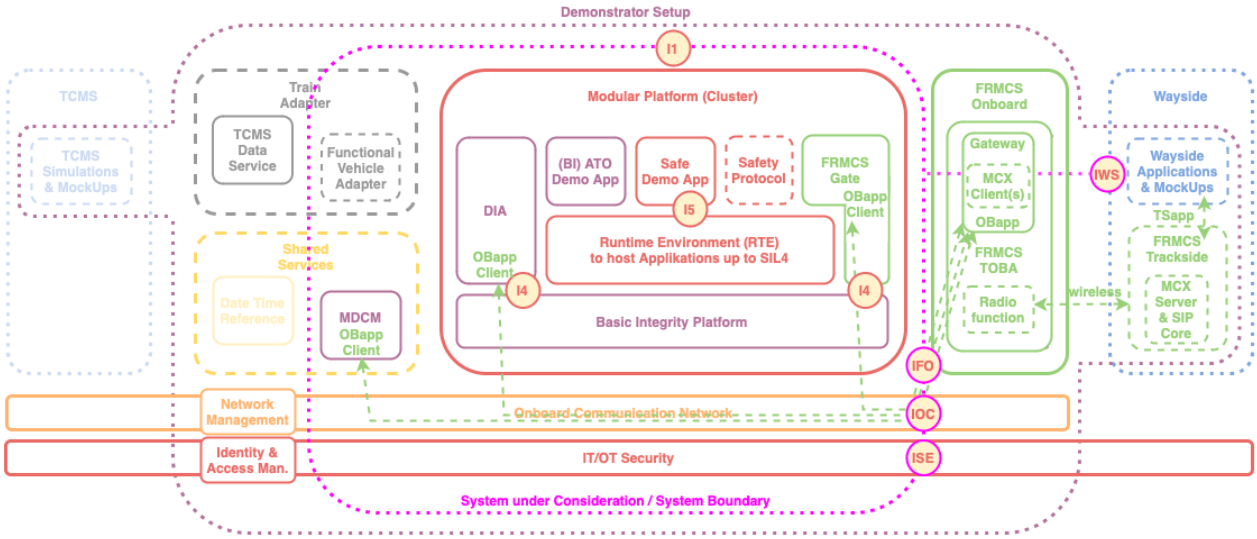

Figure 4: FRMCS View with relevant Interfaces

3.6.2 Context of FRMCS Onboard

The FRMCS Onboard is a system enabling FRMCS communication to onboard applications. The FRMCS Onboard achieves a decoupling between onboard application(s) and FRMCS communication service as well as transport service.

For completeness and better understanding we describe in the following as well the logical components of the FRMCS TOBA actor and some corresponding wayside FRMCS components.

3.6.3 Logical Components of FRMCS Onboard outside the SuC

3.6.3.1 FRMCS Gateway Function(s)

This software entity manages the data flows stemming from the applications connected to the FRMCS Onboard.

It provides a control plane (as per definition in TOBA SRS document [9]) for the applications and oversees the so-called Onboard Application (OBapp) interface to distribute user plane data from applications over the various FRMCS radio modules depending on the application's Quality of Service (QoS) requirements and priorities between applications.

The functions include:

- The applications local binding
- The OBapp messages reception and transmission
- The processing of OBapp message content

The FRMCS Gateway is also in charge of allowing the end-to-end control plane communication to be able to open the corresponding data sessions based on the Mission-Critical Services (MCx) framework as defined in 3GPP TS 23.280.

This control plane establishment is based on the creation of a MCx Client for each loose coupled application. This function also manages the life cycle of those MCx Clients.

The FRMCS Gateway function relies on the modem information raised by the FRMCS Radio Module function to dispatch the data flow between modems.

3.6.3.2 FRMCS Radio Module

This module provides 3GPP and non-3GPP radio access technologies supported by the FRMCS system by integrating different modems which are embedded within the FRMCS Onboard.

The FRMCS Radio Module is monitoring the status of those modems (connected, not connected, etc.) to inform the FRMCS Gateway on the availability of the modems.

3.6.3.3 MCx Clients

MCx Clients are used to access the FRMCS services. MC or MCx stands for Mission-Critical Service as defined in corresponding 3GPP specifications (cf. 3GPP TS 23.280 and therein referenced specifications). The x refers either to data, video, or Push to Talk (PTT), depending on the data format manipulated.

For each loose coupled application connected to the FRMCS Onboard Gateway, an MCx client is created to manage the Mission-Critical Services, the connection to the MCx Server and the data sessions to the onboard applications.

3.6.3.4 MCx Server

The MCx Server is part of the trackside services and consists of different components which represent the MCx core system. The MCx Server provides the service layer and communication services to higher layer applications.

The MCx Server together with the SIP Core (explained in next chapter 3.6.3.5) implement communication services, that go beyond the usual range of functions of a conventional office switching system to satisfy the needs of a mission-critical railway communication solution.

It is responsible for all MCx signalling, including signalling for voice communication channels as private, group, emergency calls, etc.

For data-oriented functions, it relies on mission-critical common services, providing Configuration and Group Management (CGMS) and Identity and Access Management (IAM/IdMS) functions.

- The IAM system is the Identity and Access Management platform, that is managing digital identities to control user access to the system by applying a framework of policies, processes, and technologies. The IAM assigns users to specific user groups and ensures, that only identified users get access to resources and functions of the system.
- The Configuration and Group Management Server (CGMS) allows the configuration of service applications in groups, by defining functional roles and group profiles.
- A Media Resource Function is taking care of media streams, codecs, transcoding, and announcements.

3.6.3.5 SIP Core

IP Multimedia Subsystem (IMS) is part of the MCx solution on the trackside. The IMS service layer is represented by the IMS-based SIP Core implementing switching services.

The core switching system allows real-time mission-critical communication over IP and builds the basis for the provision of enhanced services by the application servers. The SIP Core implements and manages the SIP protocol which is defined in rfc3261 [22].

The network functions of the SIP core correspond to the 3GPP mission-critical architecture based on IMS architecture defined in 3GPP 23.228. The mission-critical application server, hosting the service logic, is connected towards the SIP core using the IMS Service Control (ISC) interface. All clients are provisioned in the Home Subscriber Server (HSS) database to allow authentication and registration in the system.

3.6.4 Logical Components of the FRMCS Onboard inside the SuC

3.6.4.1 FRMCS Gate

This logical component can be seen as a part of the Modular Platform. It includes the FRMCS Client, potentially in a redundant implementation to increased availability. Its main purpose is the translation from multiple platform internal safety message queues originating from the Safe Demo App (voting etc. is to be handled by the platform safety layer) towards the internal OBapp implementation and vice-versa. This gate is needed to establish communications between the external FRMCS TOBA Box and the Safe Demo App hosted on the modular platform.

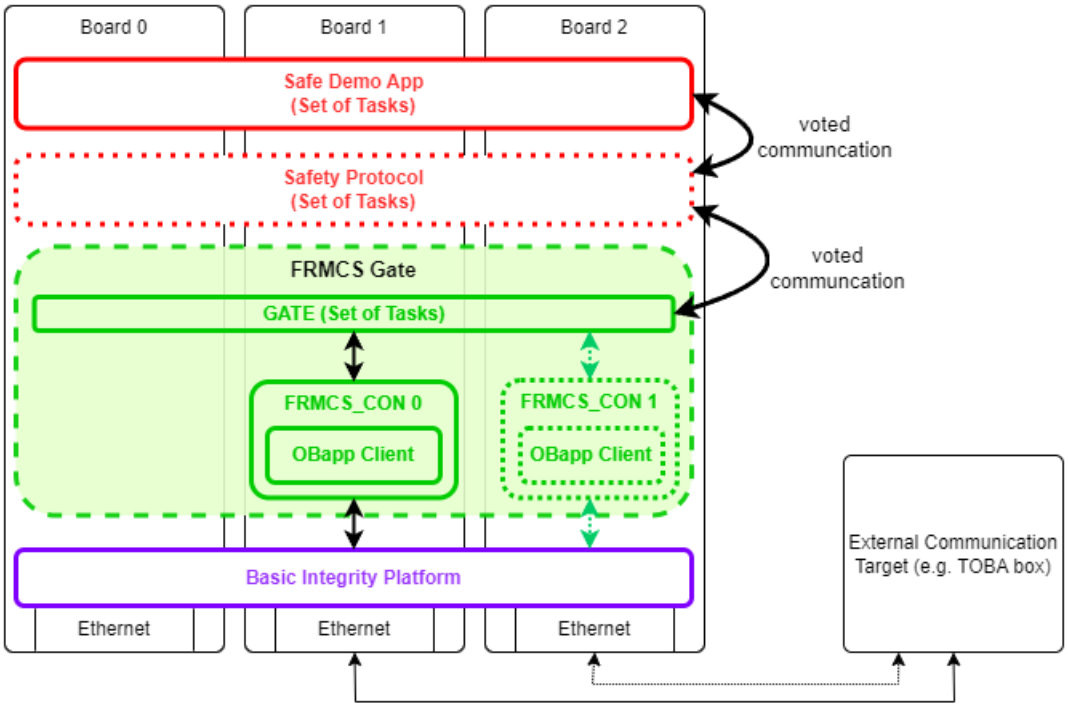

Figure 5: FRMCS Gate Communication Flows

As shown in Figure 5 above, the FRMCS Gate (in green) is in the middle of the communication flow from Safe Demo App to an external communication target, such as the FRMCS TOBA box. The figure also shows more detail coming from the Modular Platform architecture, especially the execution on multiple “boards” for redundancy. These “boards” can be dedicated computers or virtual machines, depending on the use case. Nevertheless, they are distinct hosts for tasks and host as well single tasks that are part of a so-called task set. Applications that need to fulfil certain safety goals are usually implemented as a set of tasks distributed over multiple computers, respectively “boards". The tasks of a single set are all executed independently from each other, but their communication (e.g., using message queues) is managed by the platform.

The distributed execution of task sets allows for certain measures, such as voting, to be applied to check the outputs of the individual tasks. Inputs towards the tasks also need special handling, such as synchronization and copying for the individual tasks.

The FRMCS Gate is located at the interfaces of the Safe Demo Application respectively an intermediate Safety Protocol implementation (both are safe applications, needing task sets), and the

Ethernet interface on a single board, as provided by the Basic Integrity Platform. The FRMCS Gate has two main responsibilities:

1) Implementation of a so-called Gate, translating between voted task set communication and single-task non-voted communication.
2) Implementation of the single-task, non-distributed functionality “OBapp Client” on a single board and Ethernet interface (called “FRMCS_CON” in the figure above).

The Gate is handling the generic tasks of taking the voted communication of the safe applications and translating them for a single-task application executed on a dedicated board with a corresponding Ethernet connection and vice-versa. The Gate is versatile in the way that it can also implement fail-over for two redundant single-task applications, potentially increasing availability when the external communication partner supports it (e.g., by having two dedicated Ethernet ports as well).

The redundant usage of the FRMCS_CON tasks (0 and dashed 1) is only shown here as an example. Whether a redundant implementation is possible or not will be decided in the corresponding implementation tasks later in the project.

The FRMCS_CON task itself accepts the local communication by the Gate, as discussed above, and implements an OBapp client to connect to the external TOBA box. This connection is established using the generic network capabilities as offered by the Basic Integrity Platform.

3.6.5 FRMCS Interfaces

The interfaces of the FRMCS system are described specifically in [10] and [11]. For the general understanding of the FRMCS requirements and the system architecture please see also [7], [8] and [9]. Since OBapp interface is essential here and shown in the view a brief introduction gets included here.

3.6.5.1 OBapp Interface

The OBapp corresponds to the interface between the Onboard Application(s) and the Onboard FRMCS. This interface ensures management of and access to the communication services allowing the authentication, authorisation and quality of service profile management requested by those applications.

3.7 FUNCTIONAL CLUSTER DIAGNOSTICS

The functional cluster Diagnostics comprises and implements diagnostic functions of the CCS Onboard, including acquisition and processing of Health and Performance information, provided by Applications and Functions of the Modular Platform and the provision of diagnosis data to the wayside, enabling services for remote diagnostics. Therefore, it comprises services for monitoring, diagnostic, configuration and maintenance tasks to onboard and trackside clients.

3.7.1 View on the Diagnostics

The following figure provides an overview of external actors, proposed diagnostics components as well as relevant internal and external interfaces that are involved in different diagnostic use cases. In this view it gets obvious, that basically the complete demonstrator setup may be involved in diagnostics scenarios.

In the context of this document, it might not be feasible to break down all relevant scenarios and provide appropriate level of detail. Therefore, the intention is to introduce the topic by describing some characteristics of the main components contained in the cluster.

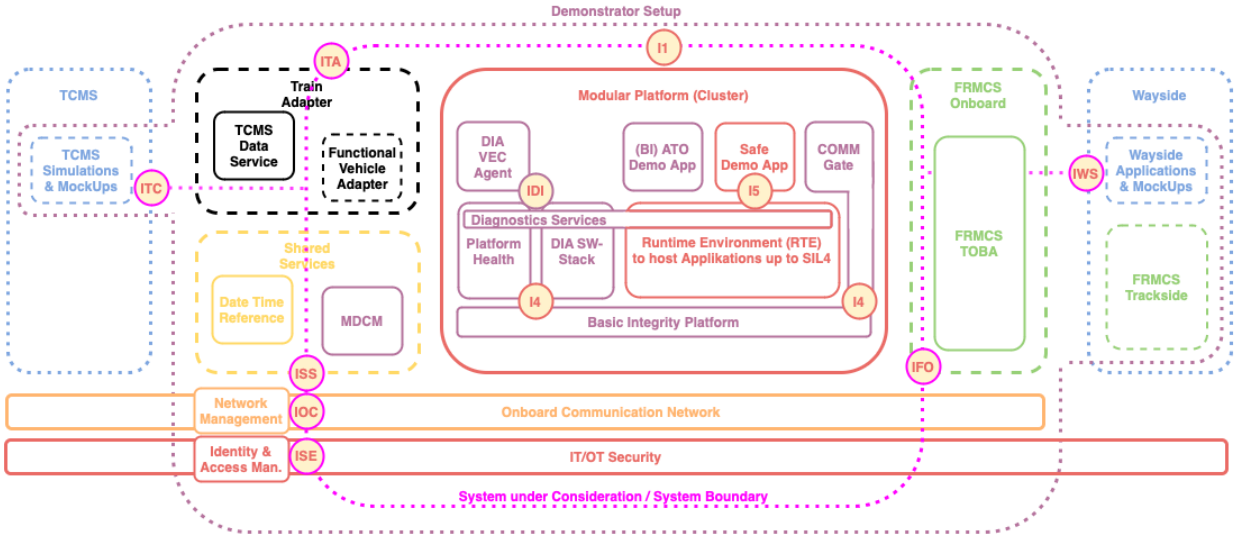

Figure 6: Modular Platform Diagnostics and MDCM

3.7.2 Context of Diagnostics

The diagnostic use cases that are foreseen in the project can be divided into four parts:

- Vehicle Diagnostics (DIA-VEC)
- Diagnostics Modular Platform Services
- MDCM (please refer to chapter 3.4.1.4)
- Diagnostics of the hosted Basic Integrity and SIL Applications

3.7.3 Logical Components of Diagnostics

This chapter provides description and characteristics of the components contained in the cluster.

3.7.3.1 DIA-VEC

DIA-VEC can be considered as a diagnostics application, receiving diagnostic related and unified TCMS data from the train systems via the TCMS-Data Service, that is based on the specification of EuroSpec [18].

The main function of DIA-VEC is assessing the health and performance of selected train systems by aggregating signals for further determination of potential impact and failures of the train operation. The aggregated data shall be provided to onboard and potentially as well to wayside ATO functions with the intention to realise use cases of automatic train operation up to GoA4 as a supportive basic integrity source of relevant TCMS data.

3.7.3.2 Modular Platform Diagnostics Services

Modular Platform Diagnostics Services are provided to applications e.g., for health and performance monitoring, logging, configuration and maintenance purpose. A generic approach is foreseen in

specification work, either in WP26 as part of I4 and I5 (interfaces between the platform and hosted applications) or in System Pillar. At this point in time, it is still expected but vague to which extent I5 will be specified for DIA Services in D26.3. In case there is the need, existing platform specific solutions might be used for the implementation as a fallback to gain feedback and evidence for the planned specification work.

3.7.3.3 Platform Health

The component Platform Health is responsible for the acquisition and processing of health and performance information provided by applications and system functions of the Modular Platform.

3.7.3.4 DIA SW-Stack

The interaction between the diagnostics services, the modular platform and hosted applications shall be based on specification work of WP26 (in particular deliverables D26.2 [1] and D26.3 [2]), adapting existing diagnostics interfaces and functions of the used platform implementation.

It is foreseen to evaluate and integrate standard software components and diagnostic protocols for the realisation of the "DIA software stack" component, potentially utilising existing COTS solutions from the automotive industry. As a first step the existing services of the Modular Platform supplier shall be evaluated as well.

3.7.3.5 MDCM

Monitoring, diagnostics, configuration and maintenance services have already been introduced in chapter 3.4.1.4. Those services are potentially provided to multiple onboard domains, e.g., for remote diagnostics purpose, configuration and software update. Please refer to specification work in OCORA [13], [14] and corresponding user stories in section 2.5 of the User Stories & Test Cases document [19] for further details.

3.7.4 Diagnostics Interfaces

3.7.4.1 Interface IDI

This Modular Platform internal interface between the Diagnostics Services and hosted applications represents the diagnostics related part of the Interfaces I4 and I5. WP36 will depend on WP26 specification activities defining such an interface. First requirements have been collected in prior work of OCORA, please refer to deliverables D26.2 [1] and D26.3 [2] for further details. The name "IDI" is just a working title for identification and may not exist in WP26 or System Pillar specifications.

For the actual realisation in WP36, existing concepts and implementations of the used Modular Platform solution may be utilised.

The interface may comprise the following functions:

- Regular logging of hosted applications, mainly for (remote or local) debugging purpose. A potential solution may be similar to the POSIX syslog mechanism. It is essential to have a minimum set of attributes like timestamp, application name and severity of the log message. Several configuration possibilities regarding buffer size and remote access may be provided and security topics need to be considered as well.
- Specific mechanisms to query internal states or variables of the hosted applications, both, remotely and locally. Applications are supposed to provide corresponding data in a standardised way, which would then be encapsulated with proper protocols from a platform

service. Here several standardised protocols may be utilised, such as SNMP, OPC-UA or solutions from the automotive sector. Further details are provided in chapter 3.12.3.

- Specific mechanism to send commands to an application. Same protocols as introduced above could be used.
- Provision of methods and tools for deep debugging and developing purpose, also a very light-weight console-like interface could make sense.

3.7.4.2 Interface I1 of the Modular Platform

The Interface I1 of the Modular Platform has already been introduced in chapter 3.5.4.1. It shall specify external accessibility of the Modular Platform's diagnostics data as health status, logging information, etc. to external entities.

At the current point in time the I1 interface is not yet specified in detail, and it is not clarified how diagnostics data related to the hosted applications shall be provided and distributed. The definition of further interfaces will be necessary and WP36 is planning to evaluate existing solutions from wayside systems and standards from the automotive sector as mentioned above in chapter 3.7.3.4.

Interface I1 is generally divided into several parts, all communicating to external entities:

- External Diagnostics Interface: An interface ideally following open and wide-spread standards to query the system for specific information. This can be information from an application running on it, but also information from the platform itself or from a potentially utilised virtualisation layer.
- External Command Interface: An interface to command selected activities to the system. Similar protocols as in the points above could be utilised. Examples for those commands could be "reboot", "reconnect" or application specific commands like "config reload", etc.
- External Update Interface: Used to realise software updates of the hosted applications, the used Operating System (OS), configuration or (maybe) also the virtualisation layer. This interface is of particular interest as it needs to be embedded into a specific process to ensure safety and availability. Appropriate processes may heavily depend on the user (IM/RU) and on operational rules. Unifying those rules is seen as one of the big challenges in the standardisation endeavour.
- External Logging Access: It shall be possible for externals, potentially by remote access and as well by local entities to obtain log-files for expert analysis and scenario replay. It is currently discussed, how much of legal aspects need to be considered for such an interface.

3.7.5 Diagnostics and FRMCS

As depicted in Figure 4 the Onboard Diagnostics cluster may implement as well a FRMCS Client(s) to realise a FRMCS link to wayside diagnostics services via the I1 interface. It is not yet fully clear where this FRMCS Client(s) will be situated, and if all diagnostics communication will be centrally routed via the MDCM, which might be the most straight forward approach. This topic is marked further study and is a result of further design work and evaluation.

3.8 FUNCTIONAL CLUSTER TRAIN INTEGRATION

The aim is to show how the modular platform can be integrated into a train. The functional cluster Train Integration includes all components and interfaces that are necessary to realize this.

Context of Train Integration

The Train Control and Management System (TCMS) is an on-board system that controls and manages the train equipment. For that purpose, it has interfaces to other train-borne systems. Depending on the type of vehicle, the vehicle generation and the supplier, the structure and interfaces of the TCMS can differ significantly. To be able to interact with diverse types of TCMS, an adaption is necessary. A concept for a Train Adaption and the associated Functional Vehicle Adapter has been developed within the OCORA initiative [12].

Logical Components of Train Integration

3.8.1.1 TCMS (Simulations)

In the context of WP36 real TCMS systems won't be integrated due to the absence of a supplier for such a system. Therefore, a simulation-based solution is foreseen as part of the demonstrator lab setup. This demonstration setup is intended to provide various functions of a TCMS system (e.g., door and brake status). A full TCMS simulation is not foreseen.

3.8.1.2 TCMS Data Service

The TCMS Data Service is an approach to realise TCMS data formalisation, the definition and demonstration of methods for a generic TCMS data model and a data service interface for particular use cases e.g., door and brake status. The basic idea in the context of WP36 is to show, that the TCMS Data service is capable to harmonise TCMS data from TCMS simulations, representing different vehicles that are providing differing vehicle data.

The received harmonised data shall be provided to a generic diagnostics application, the so-called DIA VEC agent as introduced in chapter 3.7.3.1.

3.8.1.3 TCMS Data Service Interface

A first published version of the TCMS Data Service is available at EuroSpec [18].

3.8.1.4 TCMS Data Service View

The following figure shows the data formalisation approach of the TCMS Data Service from the WP36 perspective, appropriate information will be available soon in the corresponding deliverable D31.2 TCMS Data Service Concept [3].

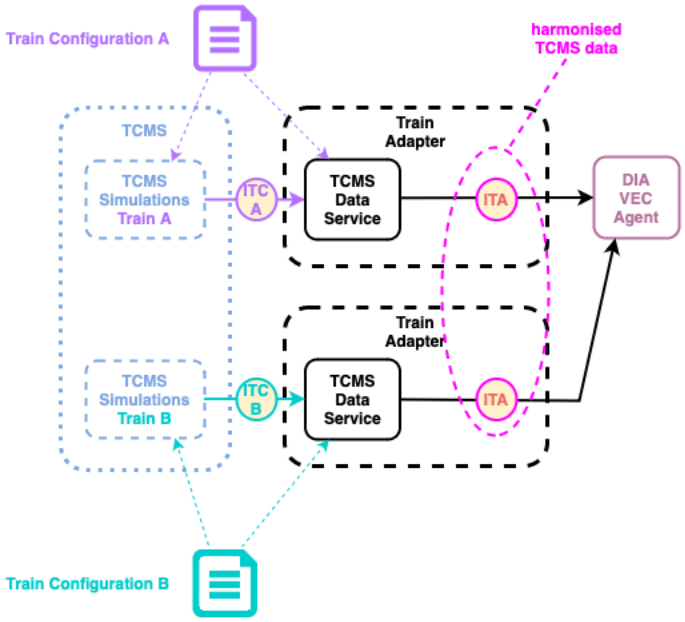

Figure 7: TCMS Data Service Harmonisation View

3.8.1.5 Functional Vehicle Adapter

The Functional Vehicle Adapter (FVA) is a computation function whose job is to provide a unified and standardized interface towards CCS onboard applications. The general concept for such an adapter was developed as part of the OCORA initiative [12]. The adapter allows to map data exchange between a specific TCMS and standardised CCS onboard application.

The interface to the onboard applications is defined in SUBSET-034, SUBSET-119 and SUBSET-139 among others. In the OCORA initiative, this interface is referred to as SCI-TCMS, for which further definitions have been made in [20]. The interface to the TCMS is named CI-TCMS and varies depending on the used TCMS system.

Various high-level requirements for a FVA were outlined in document [21]. This includes for example requirements for variable mapping and configuration. These requirements should be considered as part of the integration. Figure 1 from [12] provides an overview of the integration of the FVA and its interfaces.

3.9 Functional Cluster Onboard Communication Network

The Onboard Communication Network functional cluster provides deterministic and low-latency connectivity between the onboard functional clusters including Modular platform, FRMCS Onboard, Shared Services and TCMS. To achieve the use cases defined in section 2.7 of User Stories and Test Cases, the Onboard Communication Network needs to haven the following functional characteristics:

- Data Transfer: functions to transfer data between other functional clusters
- Redundancy: functions to provide uninterrupted connectivity
- Accessibility: functions to allow connections for configurations
- Reconfigurability: functions to detect changes in the configurations
- Manageability: functions to manage and monitor the communication
- Time Synchronization: functions to provide synchronization of referenced time

It is worth mentioning here that in addition to above mentioned functions, this functional cluster shall support the onboard communication requirements of other functional clusters i.e., data rates, determinism, safety and security.

A basic set of requirements for onboard communication networks is specified in the UNISIG SUBSET-147 (referencing additional relevant norms and standards). SUBSET-147 shall be extended towards higher ISO OSI network layers within the deliverables D23.2 [5] and D23.3 [6].

Proposals on corresponding network solutions (e.g., potentially foreseen protocols, etc.) are expected in later WP23 and WP24 deliverables and may be considered. Related work must be postponed and classified “for further study”, since those deliverables are not available yet.

3.10 Functional Cluster IT/OT Security

As the name suggests, this functional cluster provides services to secure the overall system. A threat and risk analysis is required to identify a thorough set of security functions. The following functions are proposed in section 2.8 of User Stories and Test Cases document:

- Network Separation
- Authentication and Authorization
- Identity and Access Management
- Encryption
- Public Key Infrastructure (PKI)
- Security Information and Event Management (SIEM)

Since Security topics are not in the main focus of the demonstrator, the security services of WP36 will for sure only realise a subset of those functions. For instance, the realisation of PKI and SIEM is assumed as very unlikely, since no contributing partners are available that could provide and integrate the needed technology. Even the conducting partner for the risk analysis is not yet defined but consulting from WP3 has been proposed.

The identification of concrete security components of the demonstrator setup, relevant interfaces and needed controls must be considered “for further study”.

PHYSICAL ARCHITECTURE

3.11 ACTUAL STATUS AND OBJECTIVES

It has to be noted that the development of a concrete physical architecture has just started. The partners of the demonstrator project have discussed planned option for realisation rather on a higher level as reflected in the SoW and its implementation plan. Constraints have been identified and further alignment is necessary during the implementation tasks of the project.

The main aim of a physical architecture is to gather different viewpoints to obtain an overall picture including the deployed logical components, their logical exchanges and exchanges to external actors, and their realized functionalities. Additionally, the physical architecture consolidates into an explicit architecture from all the possible alternatives.

Main inputs are the Logical architecture and identified constraints.

3.12 DEMONSTRATOR ARCHITECTURE

3.12.1 Modular Platform Physical Architecture

As described in detail in the Statement of Work's (SoW) Implementation Plan, the demonstrator development will evolve in a stepwise approach from pure virtual environments, mix of virtual and partly physical setups, potentially distributed laboratory environments towards an integrated laboratory setup, integrating hardware and software components up to TRL6.

The target setup will comprise multiple computing platforms (TRL6) that will host the Modular Platform cluster as specified in the logical architecture chapter 3 (safe und basic integrity applications, diagnostics services, etc.), implementing hardware and software solutions.

In Task 36.2 a purely "virtual" environment will be realised where first demo applications will be developed, integrated and executed within containers on a provided datacentre and server infrastructure.

The main intention here is to setup the infrastructure for development purpose and to concentrate mainly on the realisation of functional oriented aspects, without caring too much for resource and performance constraints.

The final physical architecture of the Modular Platform cluster to be realised starting from Task 36.3 and mainly being conducted within Task 36.4 will contain:

- Modular Computing Platform with at least three Computing Elements (CPU boards, see picture below). Expected safety realisation based on the supplier GTS solution on a concrete hardware, similar to onboard hardware including, the needed sub-rack, back panel and power supplies.
- One separate Safe Input/Output (IO) Hardware solution
- Software artefacts:
    - POSIX based Application Programming Interface (API)
    - Middleware and configurations based on GTS safety solution
    - A Linux based basic integrity platform and real-time Linux kernel
    - Needed tool- and supply-chains, etc.

Figure 8: CPU Board

Three or more CPU-board with a modern CPU architecture as shown in Figure 8 will be plugged into a sub-rack with a proper back panel and three independent power supplies as shown in Figure 9. This Modular Platform hardware will be connected to the rest of the system as drafted in chapter 3.12.5.

Figure 9: Sub-Rack with multiple CPU Boards

Another essential part for demonstration will be the "Safe IO hardware". Here a small, hat-rail mounted safe computing unit with safe Input/Output (IO) will be used to control a lamp or something similar, which can be switched on and off as a demonstrator showcase. It communicates with the Modular Platform hardware potentially using a GTS solution supported safety protocol.

3.12.2 FRMCS TOBA Box

The FRMCS TOBA Box is a software Virtual Machine (VM) embedded in a in TRACe hardware (SR-TRACe-RM404 [23] of Kontron Toulon) as shown in figure below:

Figure 10: SR-TRACe-RM404 TOBA Box

The VM is composed of different software entities implemented in an Ubuntu Linux guest-OS.

The principal goals of the software entities are:

- To manage the OBapp interface
- To manage the different integrated modems
- To implement MCx Clients on behalf of Onboard applications
- To manage the different applications connected and the different sessions opened
- To manage multi-connectivity

The different modems controlled by the VM are directly embedded into the TRACe hardware and configured in a passthrough mode by the KseOS (TRACe host OS).

3.12.2.1 Connectivity and HW Connectors

Regarding the different external connections present on the TRACe hardware, they can be described by the figure below:

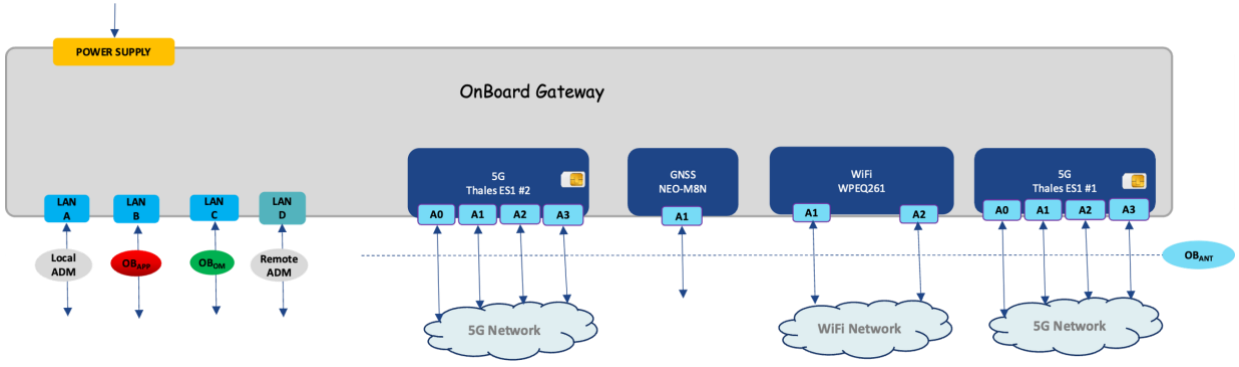

Figure 11: TRACe connectivity

The TOBA Box can be connected to the GTS TAS platform, through the Onboard Communication Network and network switches, using the LAN B (M12 Rail Cable) connector.

The LAN A and the LAN D ports are used for the administration of the FRMCS TOBA Box like maintenance and debugging purposes.

The different embedded modems that will be used during the tests should be connected to the corresponding antennas.

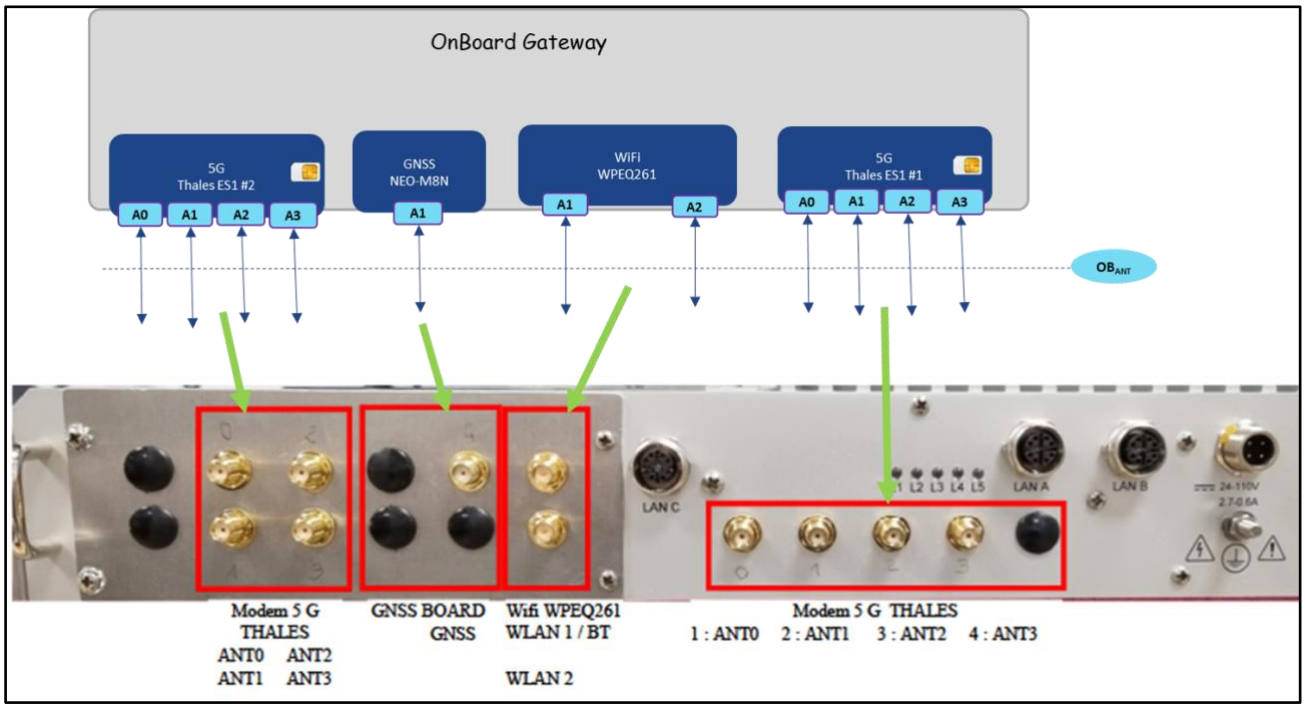

Figure 12: Modem connectors

3.12.2.2 Security protocols

Regarding the security aspects, as of today the release 17 of the MCx framework guarantees the authentication, authorization and integrity for the data and control flows for the Onboard-to-Wayside communication.

The communication between the applications and the TOBA Box are secured based on the http2 over TLS protocol for the control interface between application and the TOBA box (OBapp).

3.12.3 Diagnostics

As introduced in chapter 3.7.3, it is foreseen to utilise standardised protocols, existing solutions from the Modular Platform vendor and COTS software components from the automotive sector. Relevant software components shall be integrated on top of the Modular Platform Interface I4 (please refer to chapter 3.7.3.4 and Figure 6 in chapter 3.7.1).

Diagnostic communication to onboard functions (applications hosted on the Modular Platform) may be realised using automotive diagnostic standards, as ODX (Open Diagnostic eXchange format) and UDS (Unified Diagnostic Services).

For an onboard remote access link to the wayside, SOVD (Service Oriented Vehicle Diagnostics) could be utilised.

Existing solutions of the Modular Platform supplier shall be evaluated as well and might be used for specific uses cases like (remote) software-update.

For diagnostics applications there do exist different possibilities to choose from:

- Syslog-System: Possibility to log activity with different severity levels, and meta-data as timestamps and source information. The text-based description provides a high degree of flexibility and can be formatted according to specific needs. Remote access and configuration are foreseen, following well defined security rules.

  Note: Platform services themselves make extensive use of the Syslog-System.

- The utilised platform provides a SNMP protocol implementation. The application can use it and needs to define two things: The MIB (application-specific Management Information Base) and controlled and well-defined means to exchange data between the SNMP-daemon and the application.

  Note: Those mechanism can be utilised for Basic Integrity applications as well as for SIL applications.

- The utilised platform also provides an OPC-UA protocol implementation. Using it is likely comparable to SNMP, of course respecting the protocol- and system differences between the two protocols.

  Note: SNMP and OPC-UA are described within specifications of NeuPro and EuLynx [24]. There, are also the MIBs (SNMP) and System Models (OPC-UA) already defined for certain purposes. Since the focus in those specifications applies rather to wayside object controllers so far, it must be investigated, how useful they are for onboard systems.

- Advanced debug tools are available. Usually, they are used during development and not within a productive system in operation. They allow deep insights into the fault-tolerance and safety system and can be an essential debugging resource.

- Of course, each application can implement its own diagnostics method using standard communication means of the platform, which are basically message-queues and TCP/IP sockets. It shall be examined during the demonstrator endeavour if it is possible and/or necessary, to standardise or simply use specific diagnostics protocols.

3.12.4 Train Integration

A concrete solution to demonstrate train integration hasn’t been selected yet due to the lack of a participating solution supplier. Nevertheless, the concept of the OCORA Functional Vehicle Adapter (as introduced in chapter 3.4.1.5) seems to be carried out to real railway products like the Vehicle Interface Adapter (VIA) from Siemens, which has been introduced in a railway trade magazine [25] recently.

3.12.5 Onboard Communication Network

The diagram below illustrates a high-level architecture intended to showcase a variety of on-board and off-board communication requirements. It's important to note that the actual number of hardware components connected to the Onboard Communication Network (OCN) may vary depending on the available resources in the laboratory. For example, the goal is to utilize six switches proposed by WP23, but there might be less switches used.

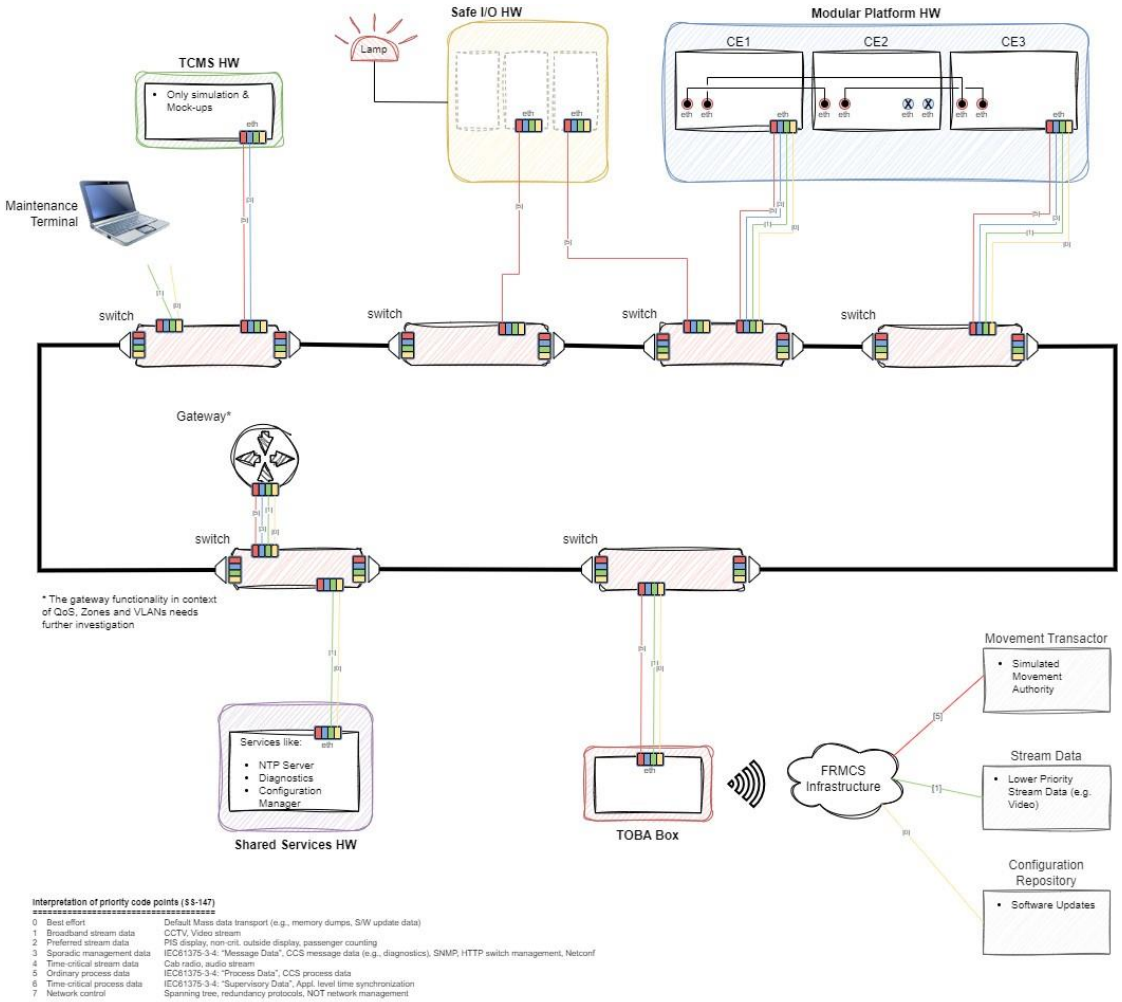

Figure 13: Network Architecture

In context of the OCN, our primary focus in this demonstration includes:

- Demonstrating communication links with distinct Quality of Service (QoS) requirements, such as different priorities for safe and non-safe data exchange, both between on-board systems and between on-board and off-board systems, with the latter achieved through communication via FRMCS.
- Implementing network segmentation through virtual networks.
- Investigating the impact of a gateway/firewall on QoS attributes (priorities).

The Modular Platform Hardware represents the Thales GTS TAS Platform, hosting on-board demonstration applications of varying criticalities. To visually depict safety-critical communication over the OCN, a demonstration application on the Modular Platform will control a light connected to

remote Safe I/O Hardware, which is external to the Modular Platform Hardware and linked to the OCN.

To assess the high availability aspects of a OCN ring network, both the Modular Platform Hardware and the Safe I/O Hardware will be connected in parallel to two different switches. This setup will demonstrate that a single switch failure, in combination with a single cable/connector failure, will not impact application availability or functionality.

The TOBA Box represents the FRMCS communication between on-board and off-board systems. In the context of the OCN, the demonstrator will illustrate how off-board services with varying QoS requirements communicate with on-board demonstration applications running on the Modular Platform. The primary focus will be on investigating and demonstrating how FRMCS service priorities can be effectively routed through the OCN to the corresponding demonstration applications.

The Movement Transactor, Stream Data, and Configuration Repository are examples of off-board services, used to illustrate communication with different QoS requirements. The Shared Services Hardware and the TCMS Hardware represent on-board systems offering services that are not hosted on the Modular Platform. The Maintenance Terminal serves as an indicative system used by the system administrator in the demonstration.

A breakdown in concrete software and hardware components will be conducted in later implementation tasks and can be foreseen as an open point.

4 NEXT STEPS AND CONCLUSION

Subsequential further work on the architecture will be necessary as proposed in Annex A:. This applies in principle to all areas of the architecture (only towards the degree of feasibility within a demonstrator project). From system, sub-system level, seamlessly continuing in design and implementation activities (in Tasks 36.2, 36.3 and 36.4), to integration and validation, the architecture needs to stay refinable in a consistent, comprehensible, and traceable way.

Such continuous refinement raises the need to use appropriate methods and tools, that enable traceability between requirements, architecture, implementation, and validation, realising configuration and release management. Those needs can only be fulfilled by proper modelling tools (as Capella, SysML), seamlessly integrated into an Application Lifecycle Management (ALM) tool as Polarion, and as it is supposed to be utilised in the ER JU System Pillar.

Finally, the statement shall be repeated, that this document won’t be maintained and can’t be considered as a working document, since it doesn’t fulfil the above-mentioned needs.

5 REFERENCES

[1] R2DATO D26.2 Intermediate Modular Platform requirements, architecture and specification
(not yet published)
https://projects.rail-research.europa.eu/eurail-fp2/

[2] R2DATO D26.3 Final Modular Platform requirements, architecture and specification
(not yet published)
https://projects.rail-research.europa.eu/eurail-fp2/

[3] R2DATO D31.2 Concept Document: TCMS formalisation
(not yet published)
https://projects.rail-research.europa.eu/eurail-fp2/

[4] R2DATO D31.3 Demonstrator: TCMS Formalisation
(not yet published)
https://projects.rail-research.europa.eu/eurail-fp2/

[5] R2DATO D23.2 Definitive and aligned requirements set for Onboard Communication Network
(not yet published)
https://projects.rail-research.europa.eu/eurail-fp2/

[6] R2DATO D23.3 List of solution candidates
(not yet published)
https://projects.rail-research.europa.eu/eurail-fp2/

[7] FRMCS Functional Requirements Specification FU-7120-1.0.0
https://uic.org/rail-system/telecoms-signalling/frmcs

[8] TOBA Functional Requirements Specification FU-7510-1.0.0
https://uic.org/rail-system/telecoms-signalling/frmcs

[9] FRMCS System Requirements Specification AT-7800-1.0.0
https://uic.org/rail-system/telecoms-signalling/frmcs

[10] FRMCS FFFIS AT-7950-1.0.0
https://uic.org/rail-system/telecoms-signalling/frmcs

[11] FRMCS FIS-7970-1.0.0
https://uic.org/rail-system/telecoms-signalling/frmcs

[12] OCORA-TWS04-010 CCS On-Board Functional Vehicle Adapter Introduction
https://github.com/OCORA-Public/Publications/blob/master/09_OCORA%20Release%20R4/OCORA-TWS04-010_FVA-Introduction.pdf

[13] OCORA-TWS08-010 MDCM Introduction
https://github.com/OCORA-Public/Publications/blob/master/09_OCORA%20Release%20R4/OCORA-TWS08-010_MDCM-Introduction.pdf

[14] OCORA-TWS08-030 MDCM SRS
https://github.com/OCORA-Public/Publications/blob/master/09_OCORA%20Release%20R4/OCORA-TWS08-030_MDCM-SRS.pdf

[15] OCORA
https://github.com/OCORA-Public/Publications

[16] X2Rail-4
https://projects.shift2rail.org/s2r_ip2_n.aspx?p=X2RAIL-4

[17] ARCADIA (MBSE Method)
https://en.wikipedia.org/wiki/Arcadia_(engineering)

[18] EuroSpec
https://eurospec.eu/tcms-data-service/

[19] R2DATO Onboard Platform Demonstrator User Stories & Test Cases

[20] OCORA-TWS04-012 TCMS (Functional Vehicle Adapter) Standard Communication Interface Specification Interface: SCI-TCMS
https://github.com/OCORA-Public/Publications/blob/master/00_OCORA%20Latest%20Publications/Latest%20Release/OCORA-TWS04-012_TCMS-Standard-Communication-Interface-Specification.pdf

[21] OCORA-TWS04-011 Functional Vehicle Adaptor (FVA) High-Level Requirements
https://github.com/OCORA-Public/Publications/blob/master/00_OCORA%20Latest%20Publications/Latest%20Release/OCORA-TWS04-011_FVA-Requirements.pdf

[22] Rfc 3261
RFC 3261: SIP: Session Initiation Protocol (rfc-editor.org)

[23] Kontron Introduces New TRACe-RM404 Railway 19-Inch Platform for Train Control

[24] EULYNX Baseline Set 4 Release 2
https://rail-research.europa.eu/wp-content/uploads/2023/06/20230629-EULYNX-BL4R2-Model-Export-Note.pdf

[25] Signalling + Data Communication (115) edition 11/2023.

ANNEX A: ARCHITECTURE ACTIVITIES OVERVIEW

The following list shows methodical process steps that may be covered in subsequential architectural work and (sub-)system design as needed and applicable. Nevertheless, essential activities needed in an implementation project were applied in WP36 Task 1.

Please note that the following list is by far not exhaustive and used to illustrate the complexity of the topic. The order of the items is not meant to show any chronological or logical order.

- Allocation of functional and non-functional requirements for the identified components, derived from the system requirements.
- Specification of performance, RAMS and security requirements for the control of interfaces where safe and reliable interaction can be compromised (may be done in a later phase for security based on a risk assessment).
- Analyse life cycle constraints (for applicable demonstrator life cycle only)
- Respect project constraints on the sub-systems, including cost, schedule, and technical constraints.
- Realise definitions of viewpoints to document the procedures for creating, interpreting, analysing, and evaluating architectural data models.
- Work out fundamental conception and explanation of rationales for design decisions and qualities (such as feasibility, performance, safety, and interoperability), architecture selection, technological/technical system element selection, and allocation and traceability between (system) requirements and architectural entities.
- Conduct and realise detailed modelling activities following architecture processes and methodologies.
- Identify conflicts and constraints.
- Define the logical component exchanges of the logical components between subsystems or between subsystems and actors.
- Create logical functions, exchange scenarios, functional exchanges and logical functional chains.
- Identify functions needed for robustness to failures.
- Identify functions needed for mitigating hazards.
- Apportion non-functional requirements to logical functions.
- Model data flows between logical functions.
- Conduct interfaces refinement (e.g., between subsystem and actor/subsystems).
- Identify subsystems and actors which share an interface.
- Display the subsystems (and/or actors depending on the interface), logical components, functions and component exchanges needed to represent the exchanges for interface with an actor or for inter-subsystem interface.
- Describe the physical layer of a specific functionality, create specific views to describe details of the physical realisations.
- Implement constraints affecting the physical layer on the several views.
- Consolidate the physical architecture, by allocating all the deployed logical components to the subsystems, relating all actors which will be linked to a subsystem.

- Evaluate the available physical architecture to identify conflicts.
- Identify the physical constraints affecting each of the alternatives, from the subsystem constraints.
- Perform trade-off assessment to provide further evidence to selected alternatives and create records to keep track of the reasoning and relevant information of the decision.