

OCORA

Open CCS On-board Reference Architecture

MBSE Modelling Guideline

This OCORA work is licensed under the dual licensing Terms EUPL 1.2 (Commission Implementing Decision (EU) 2017/863 of 18 May 2017) and the terms and condition of the Attributions- ShareAlike 3.0 Unported license or its national version (in particular CC-BY-SA 3.0 DE).



Document ID: OCORA-TWS01-041

Version: 2.1

Date: 30.06.2023

Management Summary

OCORA gathers contributors from the various European entities which participate as partners in this initiative. Therefore, the technical work is performed by experts with different backgrounds.

One of the OCORA goals consists on specifying the architecture of the CCS-OB unit, identifying its building blocks and external interfaces. The strategy selected for this purpose is following a MBSE approach using Capella as modelling tool, and a own OCORA modelling methodology inspired in the ARCADIA method.

In this context, assuming that there might be several experts working on the CCS-OB architecture, it is important to clearly describe a consistent and homogeneous modelling methodology. This will allow to keep coherence within the model despite the challenge associated to the heterogeneous working environment and the complexity of the CCS-OB unit.

The purpose of the MBSE Modelling Guideline is to document such a methodology. The methodology is detailed by describing a modelling process, a workflow indicating the sequence of steps and modelling rules to be applied on the diagrams or the elements contained in this diagrams.

Moreover, the development of the MBSE Modelling Guideline is supported by a Proof of Concept model which confirms the usefulness of the guideline.

Revision History

Version	Change Description	Initials	Date of change
1.0	Official version for OCORA R2 Release	SGG	22.06.2022
2.0	Official version for OCORA R3 Release	SGG	18.11.2022
2.1	Official version for OCORA R4 Release	ML	30.06.2023

Table of Contents

1	Introduction	7
1.1	Purpose of the document	7
1.2	Applicability of the document	7
1.3	Context of the document	7
2	Modelling process	8
3	Overall modelling rules	13
4	Modelling rules for System Analysis	14
4.1	Diagram types	14
4.1.1	Overall	14
4.1.2	Mission Capabilities Blank	15
4.1.3	System Data Flow Blank	17
4.1.4	System Architecture Blank	18
4.1.5	Exchange Scenario	21
4.1.6	Mode State Machine	23
4.2	Elements	25
4.2.1	System Mission	25
4.2.2	System Capability	25
4.2.3	System Actor	28
4.2.4	System Function	28
4.2.4.1	Overall	28
4.2.4.2	Functions allocated to the System	29
4.2.4.3	Functions allocated to Actors	30
4.2.5	Functional Exchanges	30
4.2.6	System Lifecycle Phases	32
4.2.7	ERTMS Modes	32
4.2.8	State Transitions	32
5	Modelling rules for Transition from System Analysis to Logical Architecture	32
6	Modelling rules for Logical Architecture	34
6.1	Diagram types	34
6.1.1	Overall	34
6.1.2	Logical Architecture Blank	34
6.1.3	Logical Dataflow Blank	35
6.1.4	Logical Exchange Scenario	35
6.2	Elements	37
6.2.1	Logical Function	37
6.2.1.1	Overall	37
6.2.1.2	Logical Function allocated to Logical Component	38
6.2.1.3	Logical Function allocated to Logical Actor	38
6.2.2	Functional Exchange	39
6.2.3	Logical Actor	39
6.2.4	Logical Component	39
6.2.5	Logical Component Exchange	39
6.3	Views	40

6.3.1 Overall	40
6.3.2 Capability Logical Architecture	41
6.3.3 Capability Logical Dataflow	41
6.3.4 Capability Exchange Scenario	42
6.3.5 Building block external interfaces view	43
6.3.6 Building block internal view	44
6.3.7 Detailed Scenario	45
6.3.8 Overall CCS-OB	51
7 Modelling rules for Physical Architecture	51
8 Model review	51

References

Reader's note: please be aware that the document ids in square brackets, e.g. [OCORA-BWS01-010], as per the list of referenced documents below, are used throughout this document to indicate the references to external documents. Wherever a reference to a TSI-CCS SUBSET is used, the SUBSET is referenced directly (e.g. SUBSET-026). OCORA always reference to the latest available official version of the SUBSET, unless indicated differently.

[\[OCORA-BWS01-010\] – Release Notes](#)

1 Introduction

1.1 Purpose of the document

The purpose of this document is to provide a framework in order to guarantee that all the users participating in the OCORA specification activities through the MBSE approach follow a common methodology and have a common understanding of the content to be developed and its format.

This document applies to the modelling activity in Capella.

Modelling is a collaborative activity which involve specialist with their own work methodology. It is therefore necessary to define common rules in order to have a harmonisation of practices so that the model is homogeneous.

Capella already has a methodology applied by default, called Arcadia. Therefore, compared with other modelling tools it is more restrictive and the user is more guided in his modelling activities. In this document, a set of rules in addition to this Arcadia method is provided. The objective is narrow more the different modelling choices that the user could make in order to ensure the homogeneity of the model. This guideline is derived from RCAs' modelling methods and uses the definitions as close as possible, to enable future integrations between the different models for CCS trackside and CCS On-Board.

These rules cover both principles for the modelling process (the order of tasks) and representation rules. Indeed, Capella is based on the definition of elements and graphical representation.

The document is addressed to readers with some knowledge in MBSE principles and in the Capella tool. However, for readers with a more basic level in this topics, the guideline is complemented by a [MBSE User Guide](#).

1.2 Applicability of the document

This document applies to modelling activities in OCORA in order to obtain an architectural view and integrate all the principles in this centralised view. The document is meant for OCORA internal application.

1.3 Context of the document

This document is published as part of the OCORA Release, together with the documents listed in the release notes [\[OCORA-BWS01-010\]](#).

Capella is a MBSE tool based on the Arcadia method. Capella is used in the OCORA project in order to enhance communication among stakeholders. It supports the architecture and design activities. It also ensures the coherence and completeness of the architecture.

The guideline development has been supported by the development of a proof of concept model. The figures showing Capella diagrams have not been extracted from the official OCORA model.

Furthermore the document has been reviewed and approved following the process indicated in [Review Instructions](#) (which was originally written for requirements review).

2 Modelling process

The modelling process is based on the ARCADIA method. Arcadia promotes four distinct perspectives:

- **Customer Operational Need Analysis** - definition of the Problem

Focuses on analysing the customer needs and goals, expected missions and activities. It structures the need in terms of actors/users, their operational capabilities, and activities.

Note: This perspective is not built in the OCORA modelling activities

- **System Need Analysis** - formalization of system requirements

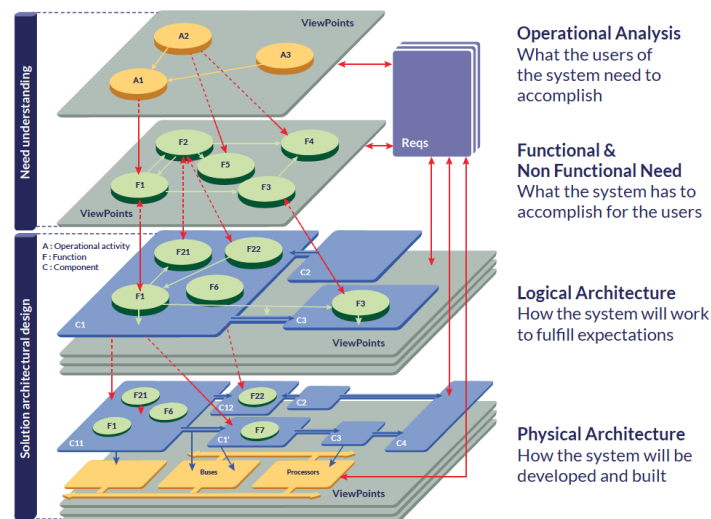
Focuses on the System itself, to define how it will satisfy the compiled operational need - zeroing in on functions and its related exchanges, non-functional constraints (e.g. safety, security, etc.) as well as role sharing between system and actors.

- **Logical Architecture (Notional Solution)** - definition of solution architecture

Aims at building a coarse-grained component breakdown of the system. This involves taking important engineering decisions which are unlikely to be challenged at a later stage. The system is decomposed into logical components, functions are allocated to components. This building process is where the majority of the OCORA design objectives and design rules will be considered.

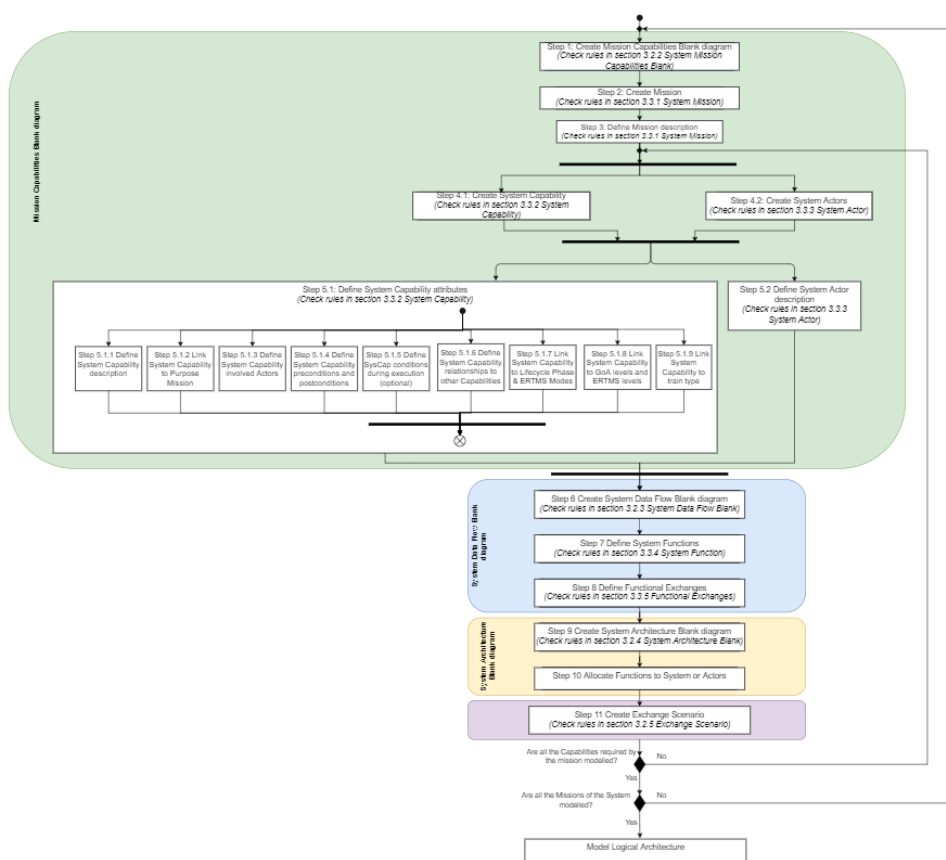
- **Physical Architecture** - definition of solution architecture

Makes the logical architecture vision evolve according to implementation, technical and technological constraints, and choices.

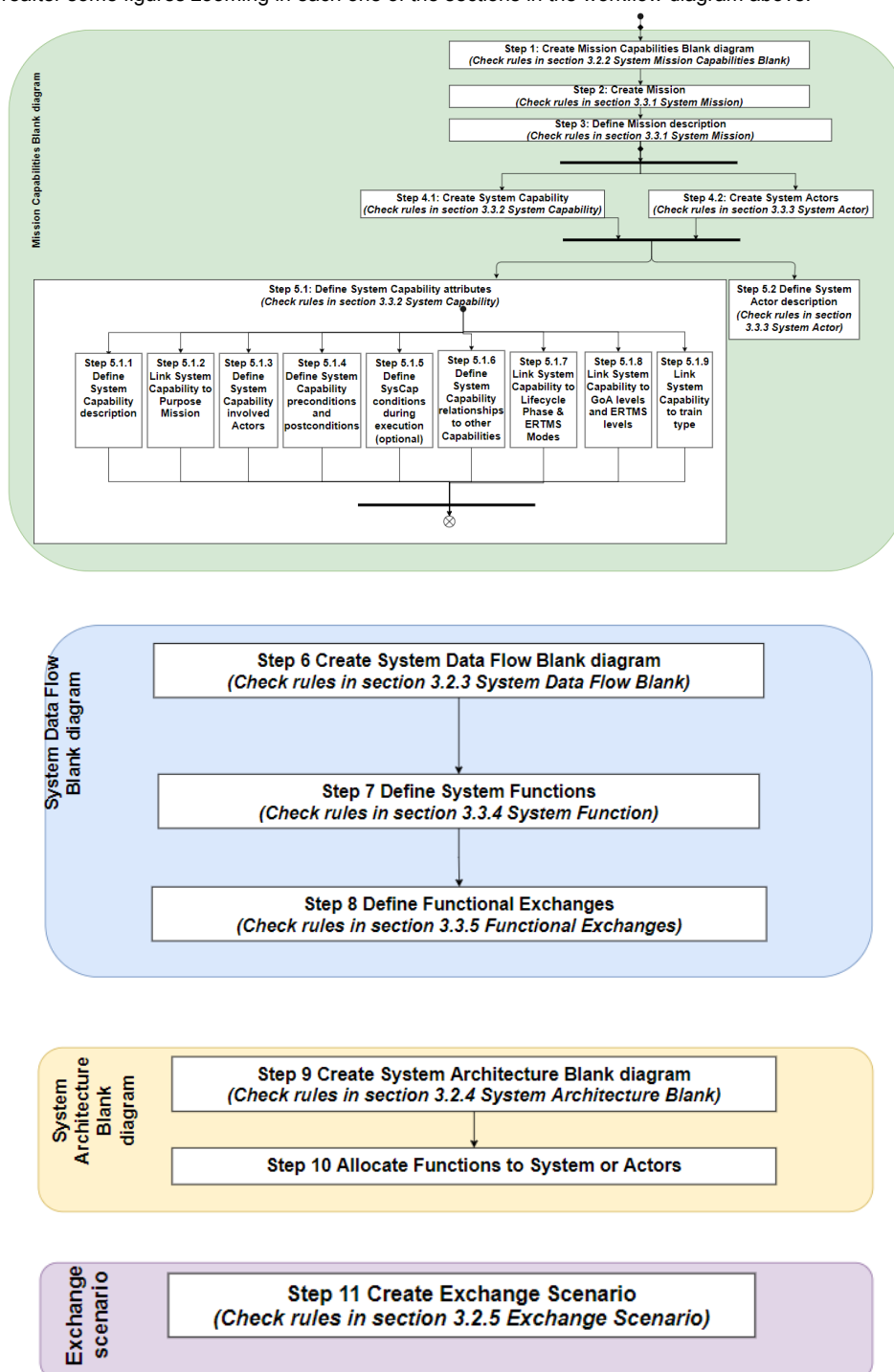


The following diagram shows the detailed workflow to be followed when modelling the System Need Analysis.

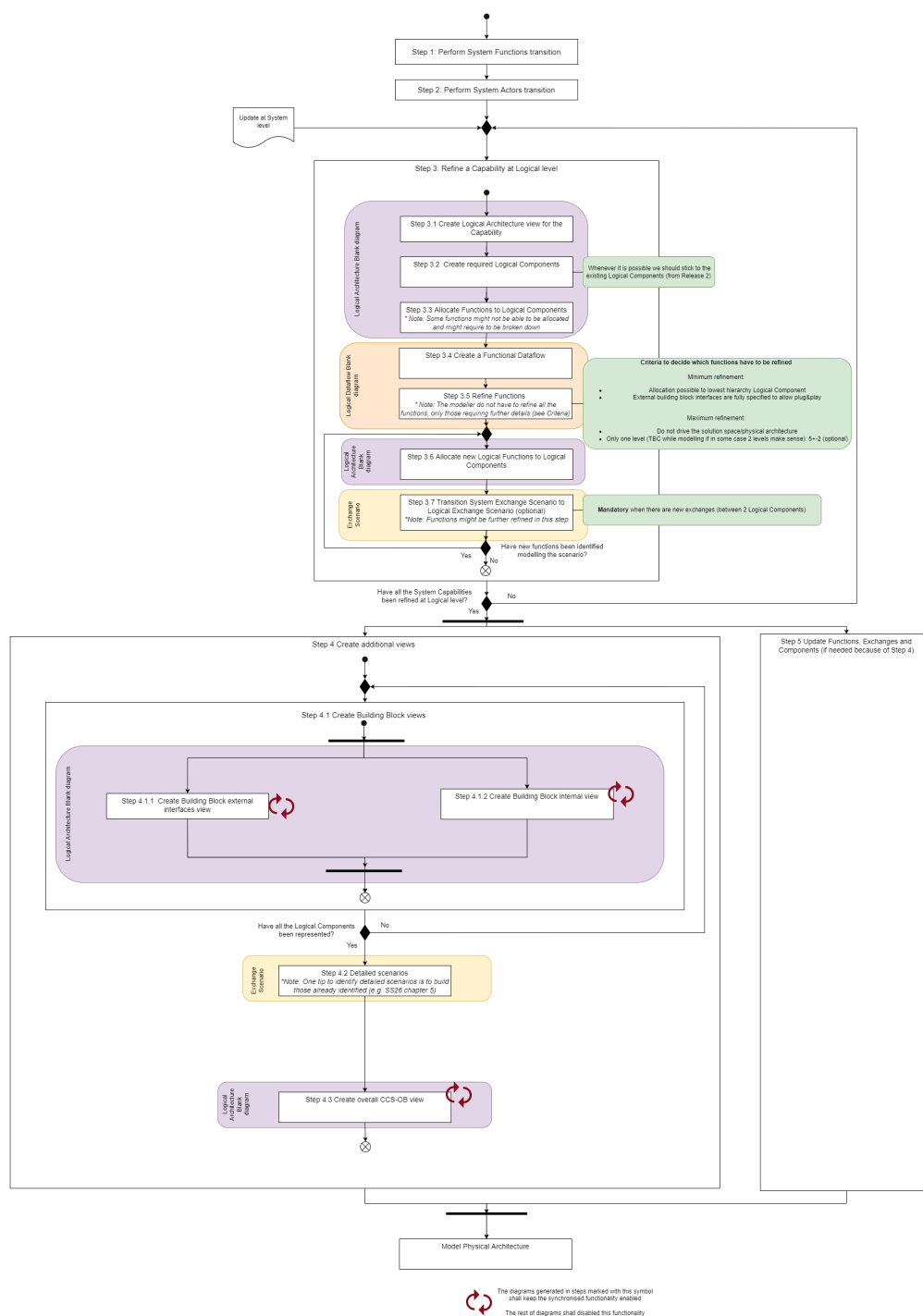
Note: The missions and capabilities identification and description is an input from TWS01-WP11 - System Capabilities



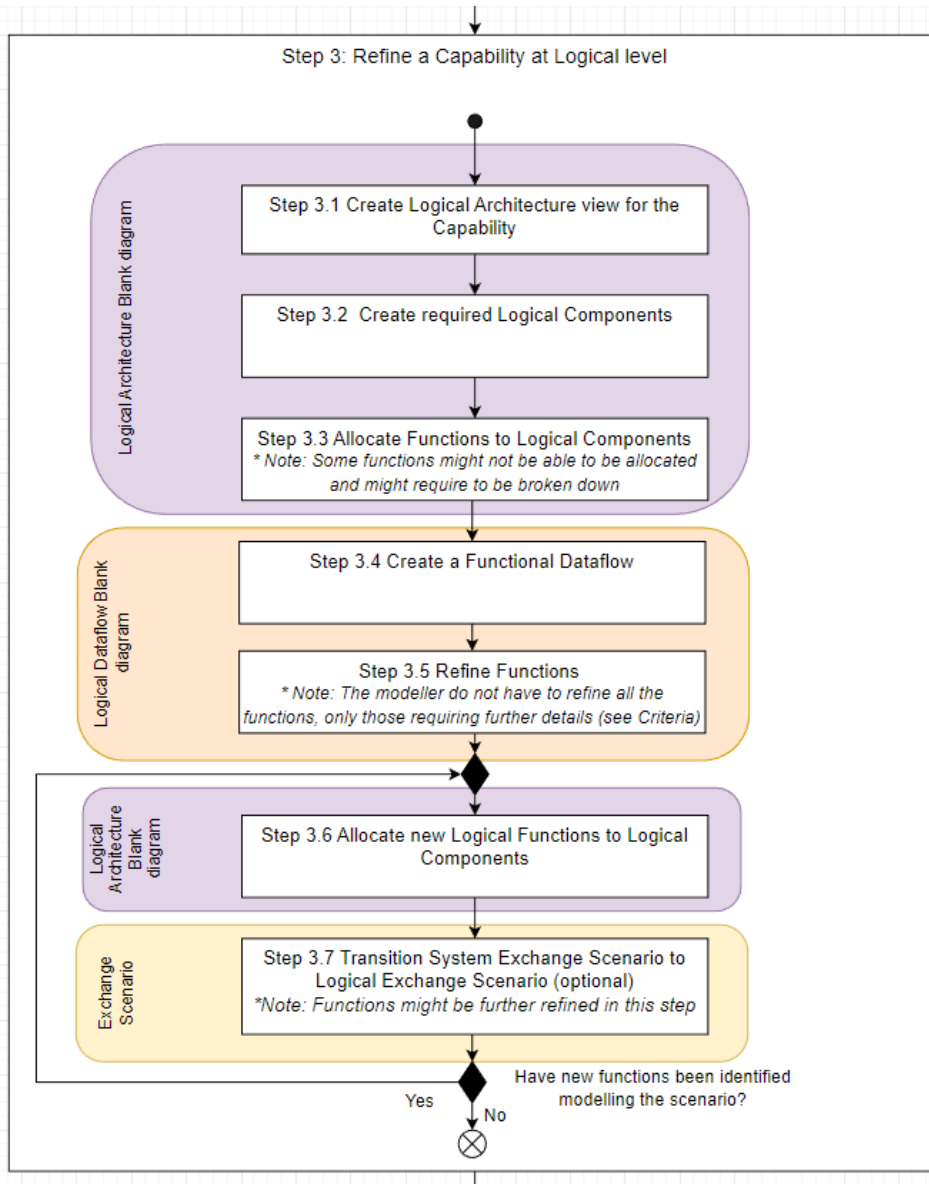
For readability, hereafter some figures zooming in each one of the sections in the workflow diagram above:

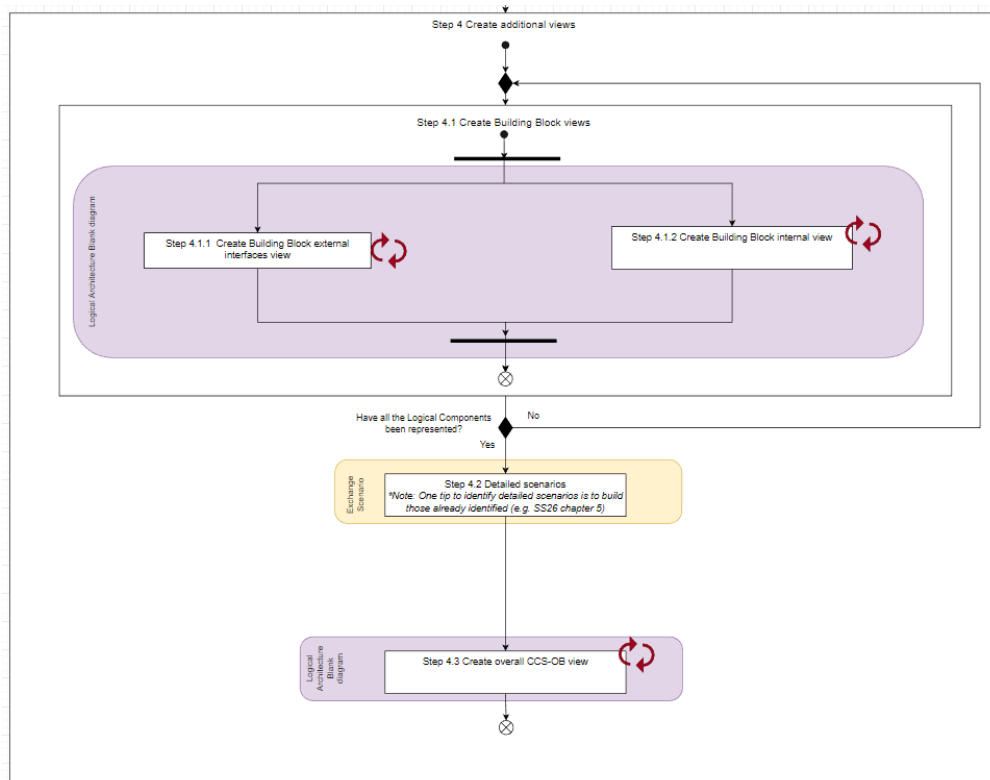


The following diagram shows the detailed workflow to be followed when modelling the Logical Architecture:



For readability, hereafter some figures zooming in each one of the sections in the workflow diagram above:





The workflow for the physical architecture will be provided in an upcoming release.

Each one of the steps is detailed in the attached document MBSE User Guide. [MBSE User Guide](#). In case the reader is not familiar with the Capella tool, the document MBSE User Guide offers a detailed explanation about how to perform each one of the steps listed below from a tool perspective.

3 Overall modelling rules

3-1, OCORA-1366, mandatory - Language

In order to ensure consistency within the project, British English is to be used at all times.
Refer to the [Cambridge Dictionary](#) for further information.

3-2, OCORA-1367, mandatory - Description links

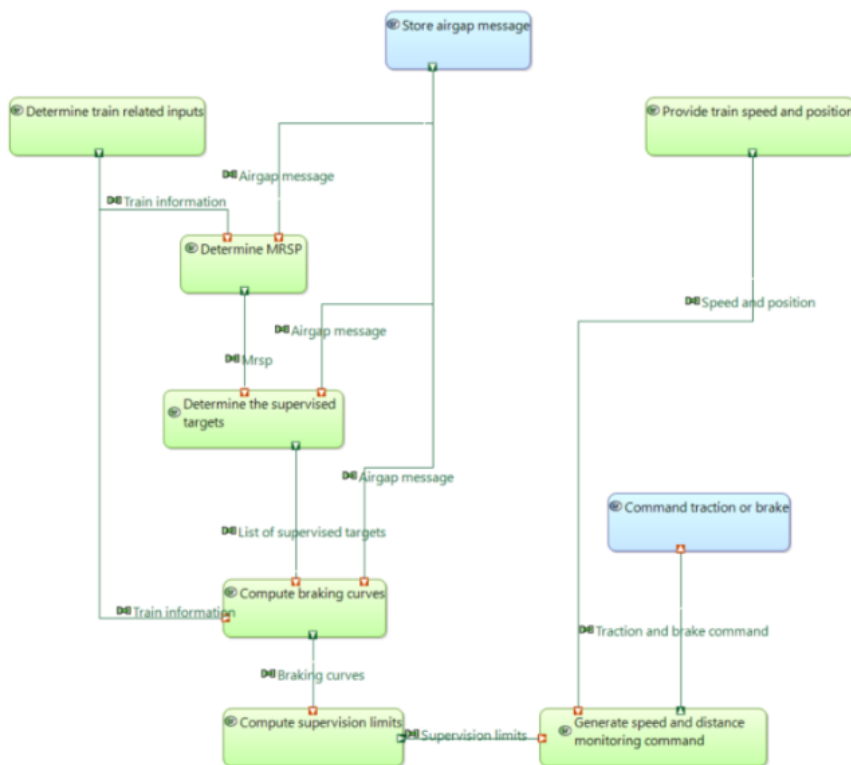
Description Links for model elements shall be used in all model element descriptions.

E.g. if a certain description refers to a defined model element, the name of the model element being referred to shall be copied (as a Description Link) and pasted (as a Link) in the corresponding description field.

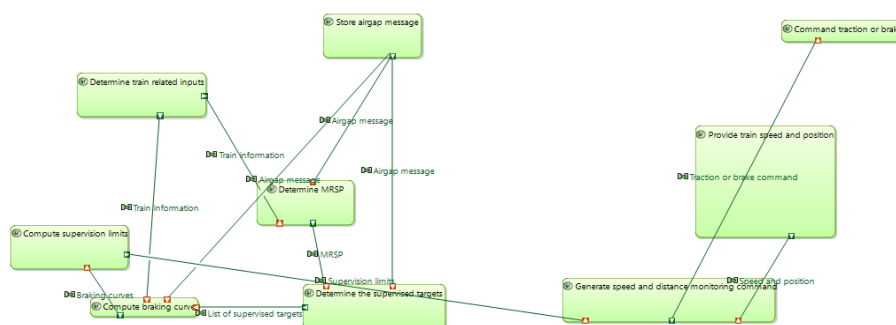
3-3, OCORA-7088, optional - Diagrams layout

The diagram layout and readability shall be optimised by avoiding lines crossing boxes, setting the same size for the diagram elements when possible or arranging the diagram elements in the most simple and logic possible way

- Example of good layout following the indications above:



- Example of bad layout (lines crossing boxes and other lines, functions with very different size, the arrangement does not follow



any logical order,...)

4 Modelling rules for System Analysis

4.1 Diagram types

4.1.1 Overall

4.1.1-1, OCORA-1354, mandatory - Diagram name

Each diagram shall be named according to following convention:

[<Layer letter>].[<Capella diagram type abbreviation>] [Name]

Layer letter: O=Operational, S=System, L=Logical, P=Physical

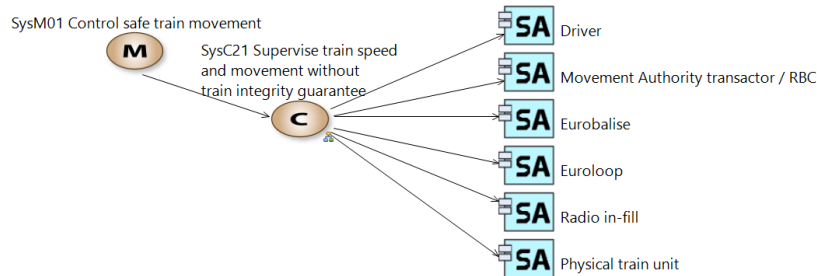
Capella diagram type abbreviation: abbreviation indicating the type of the diagram with the following possible values. It is provided by default by Capella:

- MCB - Mission Capabilities Blank
- CC - Contextual Capability
- SDFB - System Data Flow Blank
- SFBD - System Function Breakdown
- SAB - System Architecture Blank
- CSA - Contextual System Actors
- ES - System Exchange Scenario
- SDFB - System Functional Chain Description
- CDI - Contextual Component Detailed Interfaces
- STM - System State Machine
- CDB - Class Diagram Blank

Name: Free text. Check specific diagram type rules

4.1.2 Mission Capabilities Blank

In addition to the diagram rules, the modeller may check the rules which apply on the elements used to build this kind of diagram:
System Mission, System Capability, System Actor (in section 4.2 of the document)



4.1.2-2, OCORA-1368, mandatory - S.MCB Diagram content

Each diagram shall contain one single Mission and all the System Capabilities and System Actors derived from it.
No Constraints or other elements shall be included. If it is desired to represent these elements in a diagram, dedicated Mission Capabilities Blank diagram shall be created.

4.1.2-3, OCORA-1369, mandatory - S.MCB Diagram name (Full Mission view)

The diagram shall be named as the Mission it details.
Ex: SysM01: Control safe train movement

4.1.2-4, OCORA-1374, mandatory - Diagram name (Detailed Subset view)

The diagram shall carry the same name of the Mission that it details - free text explaining the purpose of the diagram.
Ex: SysM01: Control safe train movement - Preconditions

4.1.2-5, OCORA-1375, mandatory - System Mission representation

Each mission exploits **at least one** capability

AND

Each system capability is exploited by **at least one** mission

4.1.2-6, OCORA-1376, mandatory - System Capabilities representation

At least one system capability is visible on the diagram.

Note: The modeller is allowed to build a dedicated MCB diagram to focus on a specific aspect of the system (i.e. a specific subset of Capabilities), but this rule is still applicable.

4.1.2-7, OCORA-1370, mandatory - System Capabilities relationship

Only one single relationship of type generalisation, include or extend is allowed between two Capabilities (or none relation).

4.1.2-8, OCORA-1371, mandatory - System Capabilities relationships consistency

The set of relationships between system capabilities is logically consistent.

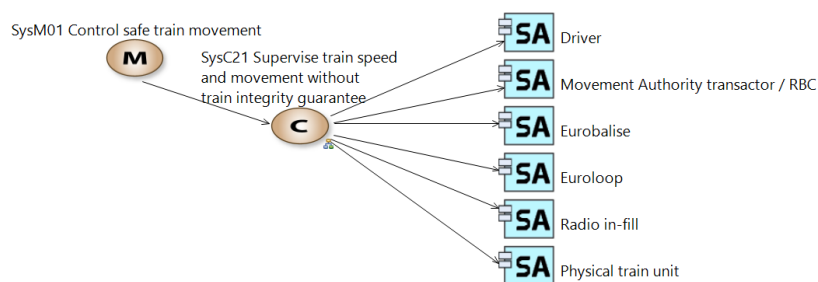
(this means avoiding issues like circular dependencies, making sure that include and extend relationships point in the correct direction, etc.)

4.1.2-9, OCORA-1372, mandatory - System Actors representation

At least one actor (external to the system) shall be visible on the diagram

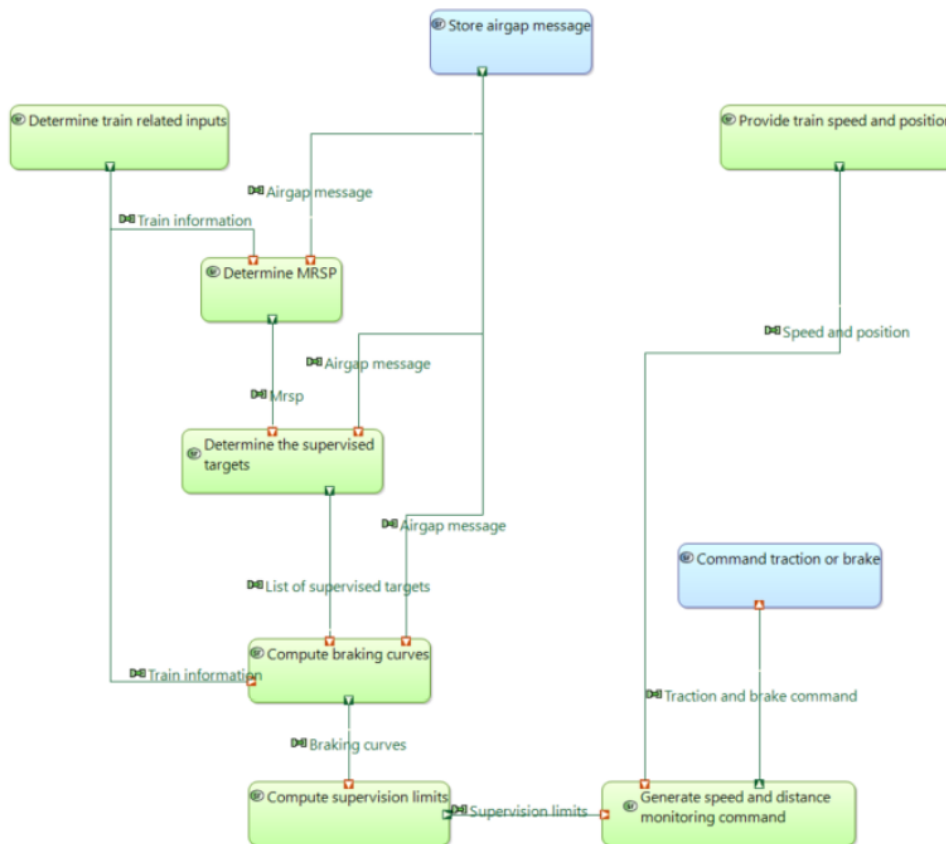
4.1.2-10, OCORA-1373, optional - S.MCB layout

Missions, Capabilities and Actors shall be arranged from left to right as shown below in order to fit possible extracts to PDF in A4 in portrait orientation or other similar formats



4.1.3 System Data Flow Blank

In addition to the diagram rules, the modeller may check the rules which apply on the elements used to build this kind of diagram:
System Function, Functional Exchange (in section 4.2 of the document)



4.1.3-2, OCORA-6332, mandatory - S.SDFB diagram content

The diagram shall contain all the Functions required to accomplish a System Capability. For readability or emphasis purpose, the content can be split into several diagrams showing different viewpoints.

Note: An optional reference to decide the right number of functions represented in a diagram for readability purpose is the ± 7 Rule (meaning that the ideal number of function is between 5 and 9)

4.1.3-3, OCORA-6331, mandatory - S.SDFB diagram name

The diagram shall carry the same name of the Capability that it details. In case there are several diagrams for one single Capability, free text will be added to indicate the diagram purpose.

Ex: SysC21 Supervise train speed without train integrity guarantee

4.1.3-4, OCORA-6330, optional - S.SDFB layout

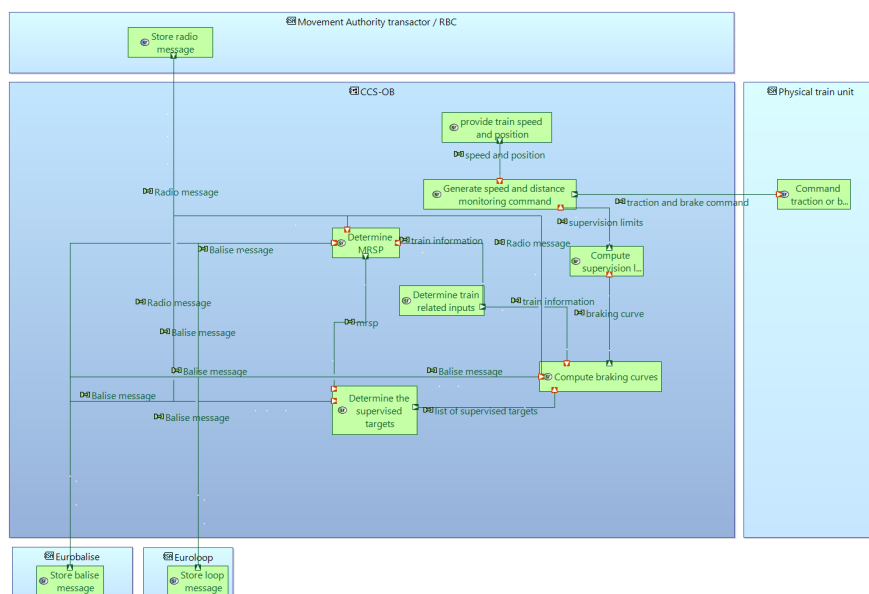
The System Functions shall be arranged in such a way that they fit possible extracts to PDF in A4 in portrait orientation or other similar formats. The diagram shall grow vertically, not horizontally.

4.1.3-5, OCORA-6329, mandatory - S.SDFB functions relationships

Each function represented in the diagram shall be linked to at least one function through an incoming or outgoing functional exchange.

4.1.4 System Architecture Blank

In addition to the diagram rules, the modeller may check the rules which apply on the elements used to build this kind of diagram:
System Function, System Actor, Functional Exchange (in section 4.2 of the document)



4.1.4-2, OCORA-6795, mandatory - S.SAB diagram content

The diagram shall contain the system of interest and at least one actor (external to the system).

4.1.4-3, OCORA-7089, mandatory - S.SAB viewpoints

The model shall contain a S.SAB diagram with the view of the full system and all the involved actors.

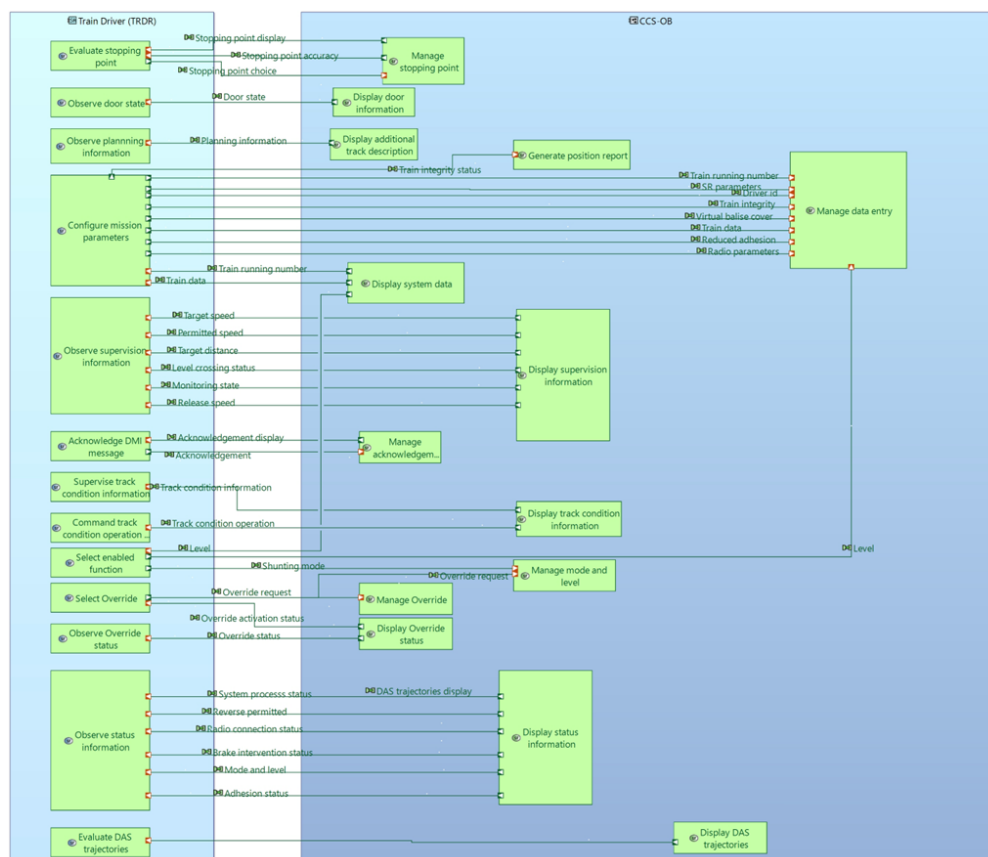
AND

The model shall contain one S.SAB diagram per capability, only with the functions and actors derived from that capability

In addition to that, the modeller has the freedom to add more S.SAB diagrams for specific views on his own criteria.

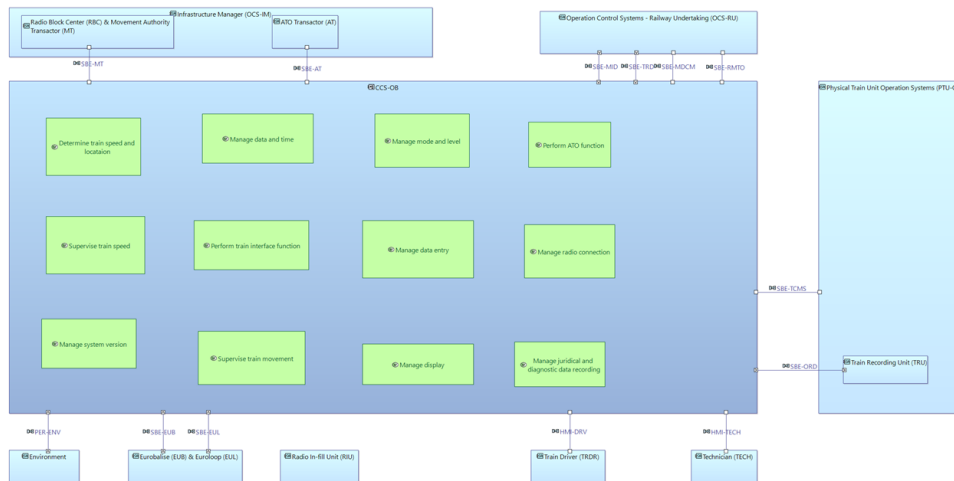
e.g.,

- S.SAB per Actor to show the exchanges between the system a this particular actor and to clarify the interface between the system an the actor.



S.SAB viewpoint example for the Train Driver actor

- S.SAB to cluster similar functions into a group and a S.SAB for each functional cluster to provide an overview of all system functions being part of this cluster,



S.SAB with Functional Clusters

Note: To support better readability of the overall S.SAB (diagram with the view of the full system), it shall be reduced to show the System, all actors and the exchanges between them (no functions).

4.1.4-4, OCORA-7091, mandatory - S.SAB Minimum number of functions per capability

Each capability shall have at least one actor function and one system function

4.1.4-5, OCORA-7092, mandatory - S.SAB functions relationships

Each function has to be connected to at least one other function via an in or out functional exchange (rule for previous diagram, SDFB)

4.1.4-6, OCORA-7093, optional - S.SAB number of internal functions

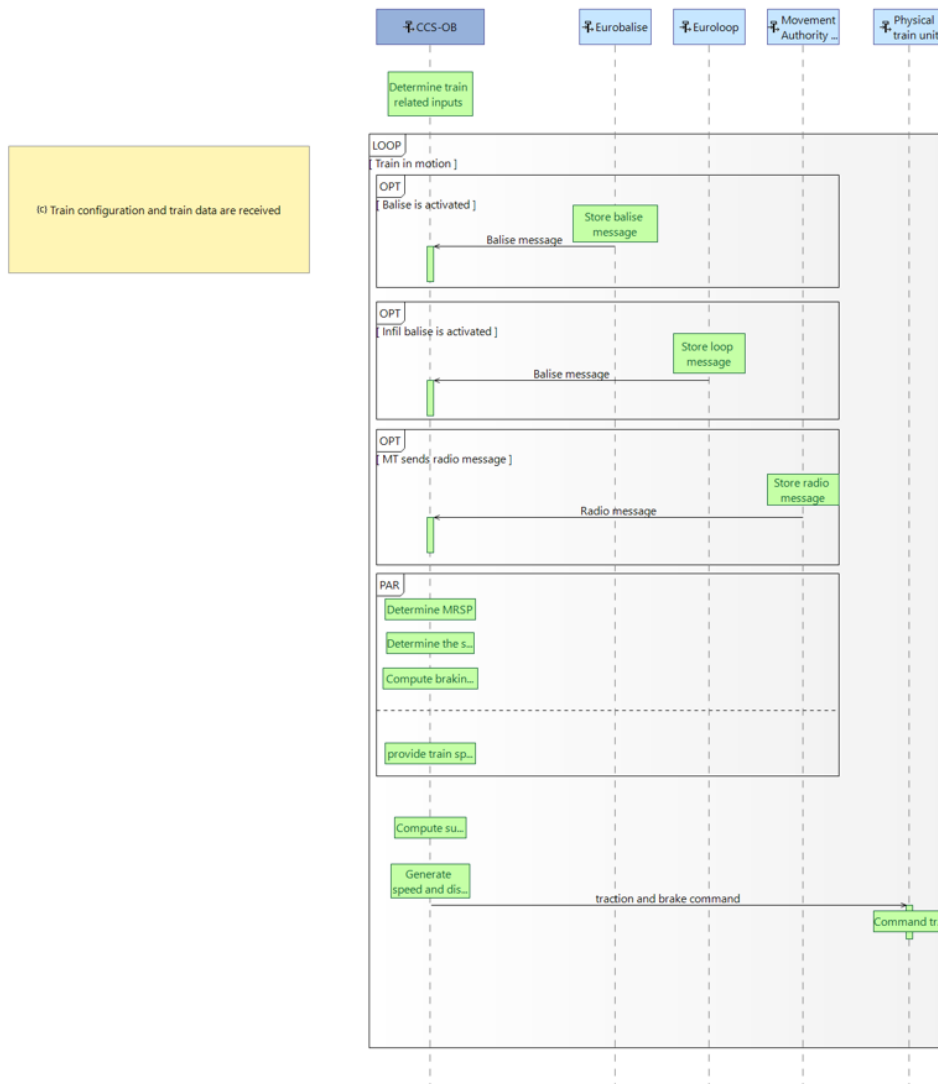
User shall try to limit the number of internal functions

4.1.4-7, OCORA-6803, mandatory - S.SAB diagram name

The overall diagram including all the system functions shall be named **[S.SAB] System CCS-OB architecture**. There might be other S.SAB diagrams displaying views showing reduced scopes. These additional views shall be named **[S.SAB] <free text>**, with a name clarifying the purpose of the view.

4.1.5 Exchange Scenario

In addition to the diagram rules, the modeller may check the rules which apply on the elements used to build this kind of diagram:
System Function, Functional Exchange (in section 4.2 of the document)



4.1.5-2, OCORA-6804, mandatory - S.ES diagram name

The S.ES diagram shall be named [S.ES] <System Capability name>. In case several scenarios are required to describe a System Capability, meaningful free text will be added after the System Capability name to clarify the scenario scope.

4.1.5-3, OCORA-6797, mandatory - S.ES OPT fragment

Where OPT fragments are used, OPT fragments have a defined guard condition describing the logical condition under which the functional exchange takes place.

Note: Formal logical condition writing is not mandatory, but the guards conditions shall be true/false statements

4.1.5-4, OCORA-6798, mandatory - S.ES LOOP fragment

When a LOOP combined fragment is used, it shall have a guard condition that describes the logical condition under which the combined fragment takes place.

Optional: you can specify a minimum and maximum number of iterations of the loop between parentheses. The format is: (<min.>, <max.>) and it shall be added in a note attached to the LOOP combined fragment (a note because this capability is not supported by Capella). If not specified, a default range (0, *) is assumed.

4.1.5-5, OCORA-6799, mandatory - S.ES ALT fragment

Where ALT fragments are used, at least two operands are required. Each operand of the fragment has a defined guard condition describing the logical condition under which the functional exchange takes place.

4.1.5-6, OCORA-6796, mandatory - S.ES ALT fragment consistency

Where ALT fragments are used,

the set of guard conditions for the fragment covers all eventualities

AND

there are no logical intersections between guard conditions (they are mutually exclusive)

4.1.5-7, OCORA-6805, mandatory - Additional pre- and post- conditions

If the pre- and postconditions of the owning capability fully apply to the Exchange Scenario these shall be contained in the diagram.

Additional pre- and postconditions of the Exchange Scenario shall be specified in the related field in Capella, representing a refinement or more concrete conditions that apply to this specific scenario.

Preconditions will be represented in the upper left corner of the diagram. Capability preconditions will be represented over the specific preconditions

Postconditions will be represented in the bottom left corner of the diagram, at the end of the lifelines. Capability postconditions will be represented over the specific postconditions

4.1.5-8, OCORA-6806, mandatory - Not applicable pre- and post- conditions

Only pre- and post conditions applicable to the Exchange Scenario shall be represented in the diagram. (The System Capability may have pre- and/or post conditions which do not apply to the Exchange Scenario)

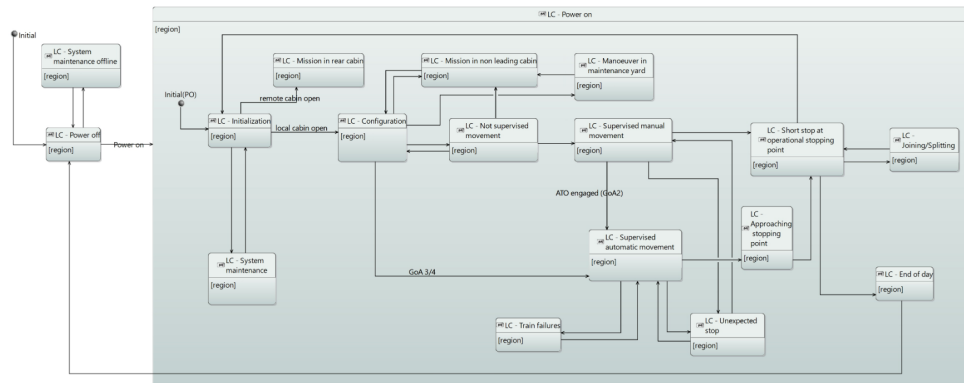
Any other additional pre- or postconditions providing context to the specific scenario shall also be specified in the related field in Capella.

Preconditions will be represented in the upper left corner of the diagram.

Postconditions will be represented in the bottom left corner of the diagram, at the end of the lifelines.

4.1.6 Mode State Machine

In addition to the diagram rules, the modeller may check the rules which apply on the elements used to build this kind of diagram:
ERTMS Mode, System Lifecycle Phase (in section 4.2 of the document)



4.1.6-2, OCORA-6800, mandatory - S.MSM consistency

There exists a single Initial state

AND

There exists at least one transition outgoing from each state (except for the Final state)

AND

There exists at least one transition incoming to each state (except for the Initial state)

AND

There exists at least one navigable path along transitions from the Initial state to the Final state, if a final state is defined.

4.1.6-3, OCORA-6801, mandatory - S.MSM transitions

All transitions have a definition of the conditions for the transition, selected from:

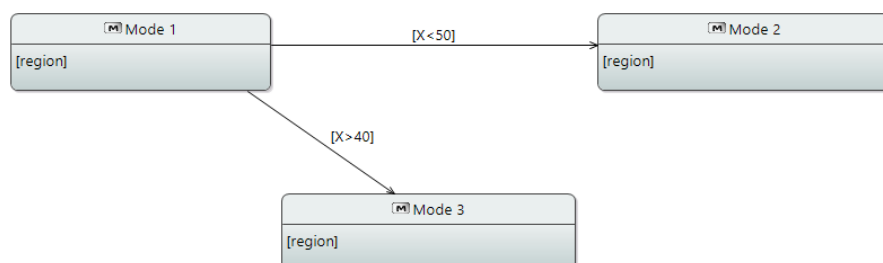
- Trigger definition, by referring to functional exchanges/exchange items/functions, with (optionally) guard conditions to qualify the trigger (preferred)
- Guard condition describing the trigger (non-preferred, for use where the triggers are not readily identifiable, especially for more abstract state machines)

4.1.6-4, OCORA-7141, mandatory - S.MSM transitions consistency

The set of transitions between states is logically consistent.

(this means avoiding issues like overlap of transitions, making sure that all the transition conditions are possible, etc)

Example of transitions overlap (when $40 < X < 50$):



Example of transition condition non possible: In the picture above, if X cannot be lower than 50

4.2 Elements

4.2.1 System Mission

In addition to the element rules, the modeler may check the rules which apply on the diagram where these elements are used: Mission Capabilities Blank (in section 4.1 of the document)

4.2.1-2, OCORA-1386, mandatory - Define the System Mission name

Naming convention:

Sentence case - the first letter of the first word in a sentence is capitalised.

The name of a system mission should begin with an active verb.

The verb should be specific enough to identify what the high-level goal for the system is.

In addition, the system mission name gets a prefix:

- SysM<unique integer number>: <Mission name>"

The usage of abbreviations should be well considered and should only be used in cases where the abbreviation is standardised (e.g. GoA).

4.2.1-3, OCORA-1387, mandatory - Define the System Mission description

Each mission has a meaningful description that includes a rational why it is a mission for the system.

4.2.2 System Capability

In addition to the element rules, the modeller may check the rules which apply on the diagram where these elements are used: Mission Capabilities Blank (in section 4.1 of the document)

4.2.2-2, OCORA-1377, mandatory - Define the System Capability name

Naming convention:

Sentence case - the first letter of the first word in a sentence is capitalised.

The name of a system capability should begin with an active verb.

The verb should be specific enough to identify what the actor wants from the system. Verbs such as "manage", "handle" or "process" are not specific enough and are not recommended.

In addition, the system capability name gets a prefix:

- SysC<capability number>: <system capability name>", e.g. "SysC1: Set point to position required by operational plan"
- The <capability number> has to be a unique integer number

The usage of abbreviations should be well considered and should only be used in cases where the abbreviation is standardized (e.g. GoA)

4.2.2-3, OCORA-1379, mandatory - System Capability Description

Detailed description of the Capability.

Can be expressed as a user story.

For example:

As <actor name>, the system helps me to <system capability> in order to carry out <mission>.

As <actor name>, I want to <receive benefit> from the system, when <precondition>, so that I can <mission>.

4.2.2-4, OCORA-1380, mandatory - System Capability Purpose Missions

Each System Capability shall be exploited by at least one Mission.

The relation can be established in the Mission Capabilities Blank diagram (Capability Exploitation relationship) or in the Mission properties.

4.2.2-5, OCORA-1381, mandatory - System Capability Involved Actors

Each capability has an involvement relationship (SysML equivalent: Association relationship) to **at least one** actor, except where a capability is included by other capabilities (in which case, a direct involvement with an actor is not necessary, but is allowed) or when a capability is subtyped (in that case, it inherits the actor from the supertype capability).

The relation can be established in the Mission Capabilities Blank diagram (Involved Actor relationship) or in the Capability properties.

4.2.2-6, OCORA-1384, mandatory - System Capability Preconditions

Conditions necessary for the Capability to be performed.

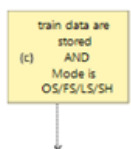
Must be selected from the list of Constraints and/or States in the System Analysis.

It can be expressed as:

- a constraint that evaluates to TRUE or,
- an entering state of the system.

A constraint is a Capella element which allow to formalize list of conditions carried by an element (function, a functional exchange).

E.g.:



4.2.2-7, OCORA-1385, mandatory - System Capability Postconditions

Conditions verified after the Capability has been performed.

Must be selected from the list of Constraints and/or States in the System Analysis.

It can be expressed as:

- a constraint that evaluates to TRUE or,
- an existing state of the system.

It is a measurable or observable result delivered by the system to an actor.

4.2.2-8, OCORA-6791, mandatory - System Capability GoA Level

Each System Capability shall be linked to at least one GoA level

4.2.2-9, OCORA-6794, mandatory - System Capability ERTMSLevel

Each System Capability shall be linked to at least one ERTMS level

4.2.2-10, OCORA-6792, mandatory - System Capability ERTMS Mode

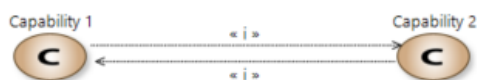
Each System Capability shall be linked to at least one ERTMS Mode

4.2.2-11, OCORA-6793, mandatory - System Capability Train Type

Each System Capability shall be linked to at least one Train Type (passenger, freight and/or construction)

4.2.2-12, OCORA-1383, mandatory - System Capabilities Logical Consistency

It shall be verified that the set of relationships between capabilities is **logically consistent** and allows for no unintended overlaps between the scopes of different system capabilities. Example of inconsistency: <<Includes>> circular reference



4.2.3 System Actor

In addition to the element rules, the modeller may check the rules which apply on the diagram where these elements are used: Mission Capabilities Blank, System Architecture Blank (in section 4.1 of the document)

4.2.3-2, OCORA-1388, mandatory - Define the System Actor name

Naming convention:

- Title Case - the first letter of each word is capitalised, words separated by space.
- Names should be unique

4.2.3-3, OCORA-1389, mandatory - System Actor Description

The description of a system actor focuses on the features that define them, not on its responsibilities or functions.

Guidance: a **brief summary** of responsibilities/behaviour is allowed but this should not read like a list of functions (because this information should be defined only through the allocation of functions to this actor, not duplicated in a text description)

4.2.3-4, OCORA-1390, optional - System Actor name and description compliance

System actor name and description is consistent with harmonised definitions from collaboration projects (RCA, S2R, OCORA etc.) or with international standards where applicable.

Note: Check References chapter to find related documents

4.2.4 System Function

In addition to the element rules, the modeller may check the rules which apply on the diagram where these elements are used: System Data Flow, System Architecture Blank, Exchange Scenario (in section 4.1 of the document)

4.2.4.1 Overall

4.2.4.1-1, OCORA-1392, mandatory - System Function name

Naming convention:

Sentence case - the first letter of the first word in a sentence is capitalised.

The name of a system function should begin with an active verb.

4.2.4.1-2, OCORA-1393, mandatory - System Function description

Function Description - describe what the function is for, such that the following boxes can be ticked:

- The description explain what the function will produce (output)
- (optional) The description explains why the inputs are required
- (optional) The description explains how the inputs may be used to produce the outputs

Pattern: This function <verb>

4.2.4.2 Functions allocated to the System

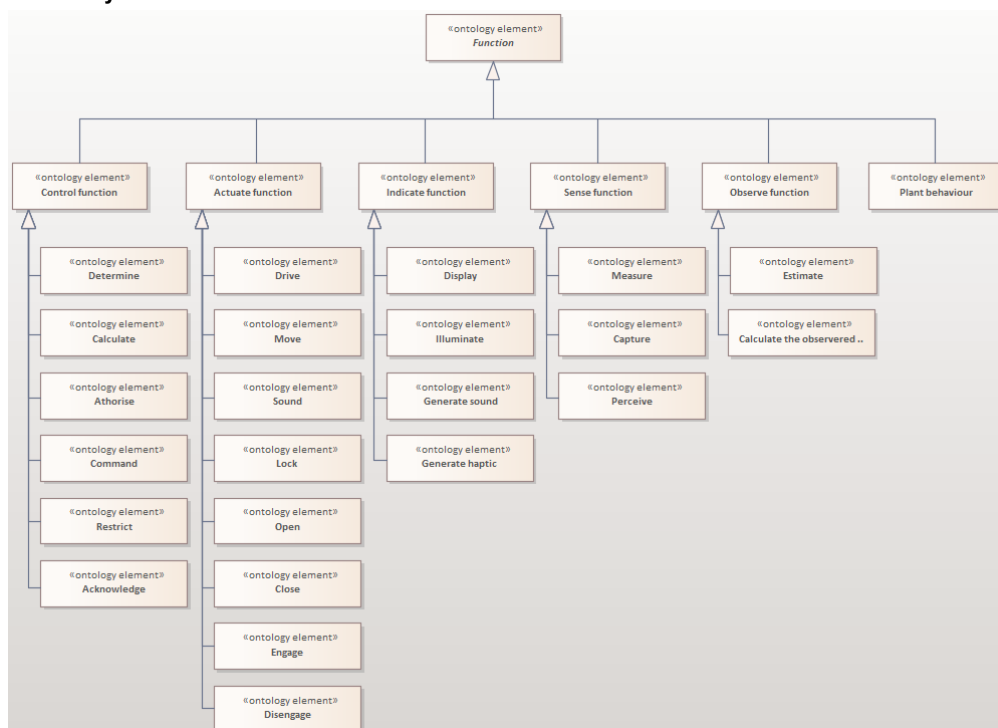
4.2.4.2-1, OCORA-1394, optional - Control System Principle

Each core system function should be named and described in such a way that it conforms to one of the categories of control system function. A "core" system function is a function that is active when the system is in the OPERATIONAL state (or equivalent).

All functions are members of **one and only one** of the following categories, and are located in the corresponding function package in the model:

- Control
- Actuate
- Indicate
- Sense
- Observe

Taxonomy:



Source: RCA - Methods for definition of functions

The main five taxonomy elements (control, actuate, indicate, sense and observe), or their specialisations are to be used for naming system functions.

If neither the main functions nor their specialisations fit to the function being created, it is possible to use verbs not defined in the above-depicted taxonomy.

In such a case, the usage of a previously undefined verb must be justified and coordinated with the parties responsible for the modelling rules.

Exception: for HMI only sense and observe functions are permitted to merge at the system boundary (in this case, the name of the function may contain two verbs Sense+observe)

In case the proposed taxonomy does not include a representative verb for the purpose of the function, alternative verbs are allowed.

4.2.4.2-2, OCORA-1391, optional - External Functional Exchanges

Functions at system level are supposed to have at least one functional exchange that crosses the system boundary.

Exception: A "control" or "observe" function may have only internal functional exchanges if at least one functional exchange feeds to an **indicate** function. (Rationale: the states produced by the "control" or "observe" function are **needed** outside the system and reach the actor via the "indicate" function, so there is an external need for the information produced)

4.2.4.3 Functions allocated to Actors

4.2.4.3-1, OCORA-6802, mandatory - Actor functions necessity

Actor functions are defined wherever some behaviour is expected from an actor in order to reach the end conditions of a system capability

4.2.5 Functional Exchanges

In addition to the element rules, the modeller may check the rules which apply on the diagram where these elements are used: System Data Flow, System Architecture Blank, Exchange Scenario (in section 4.1 of the document)

4.2.5-2, OCORA-6810, mandatory - Functional exchange name

Naming convention:

Sentence case - the first letter of the first word in a sentence is capitalised.

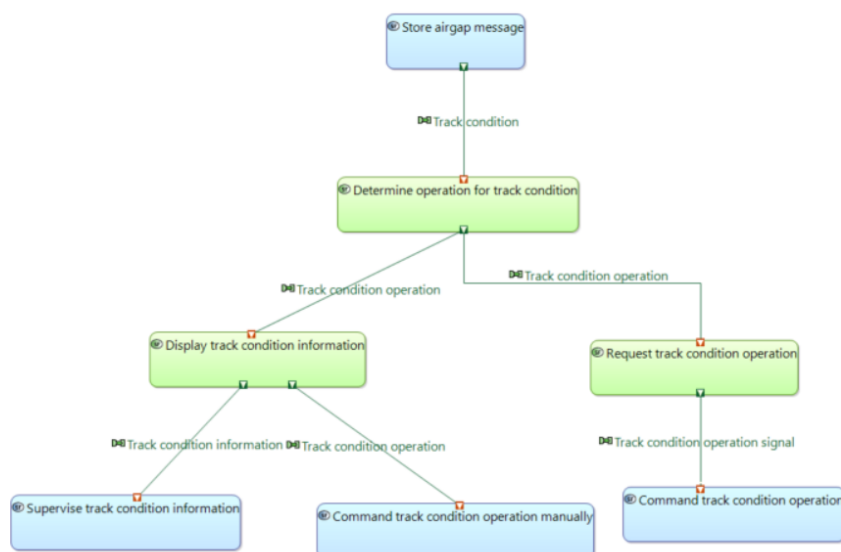
4.2.5-3, OCORA-6811, mandatory - Various outgoing functional exchanges

WHERE a function outputs more than one separate **exchange type**, each exchange type exits the function by a separate output port.

Example:

In the diagram below, the function "Determine operation for track condition" outputs one single exchange type (even though there are 2 functional exchanges because it is received by 2 functions)

On the other hand, the function "Display track condition information" outputs 2 different exchange types, and therefore 2 output ports are needed

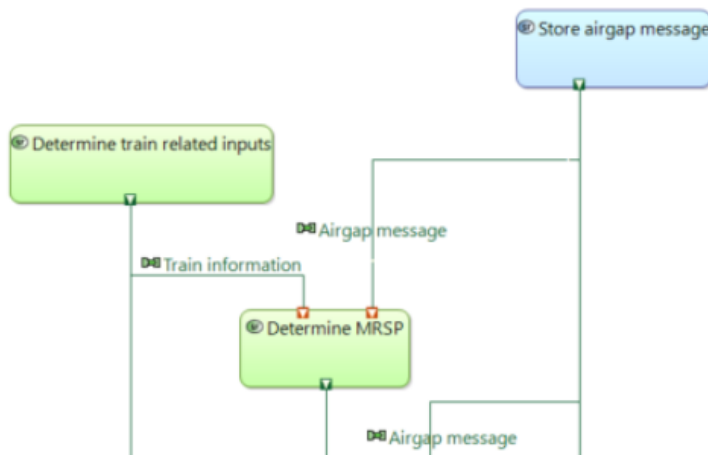


4.2.5-4, OCORA-6812, mandatory - Various incoming functional exchanges

WHERE a function takes as inputs more than one separate **type** of state (that is, a completely separate type of functional exchange), each separate type of state enters the function by a separate input port.

Example:

In the diagram below, the 2 inputs receive by the function "Determine MRSP" are different, therefore they shall be linked to the function through 2 different input ports



4.2.5-5, OCORA-6807, mandatory - Various instances of outgoing functional exchanges

WHERE a function produces more than one **instance of the same type** of a functional exchange, all the instances of this type of functional exchange exit the function via the **same** output port.

(see rule [OCORA-6811 - Various outgoing functional exchanges](#))

4.2.5-6, OCORA-6808, mandatory - Various instances of incoming functional exchanges

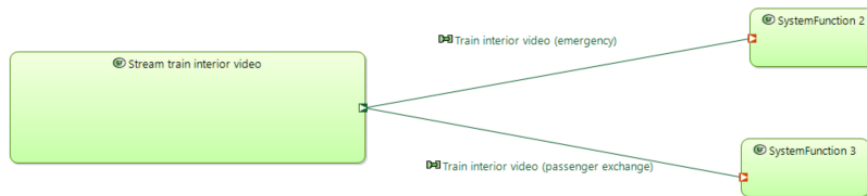
WHERE a function takes more than one **instance of the same type** of a functional exchange as inputs, all the instances of this type of functional exchange enter the function via the **same** input port.

(see rule [OCORA-6812 - Various incoming functional exchanges](#))

4.2.5-7, OCORA-6809, optional - Multiple instances name

Multiple instances of the same type of functional exchange must have identical names **but** it is permissible to add a unique suffix to them if this improves the descriptiveness of the model

Example:



4.2.6 System Lifecycle Phases

In addition to the element rules, the modeller may check the rules which apply on the diagram where these elements are used: Mode State Machine (in section 4.1 of the document)

4.2.6-2, OCORA-1401, mandatory - System Capabilities Lifecycle Phase - Name

The name of the Lifecycle Phases shall start by the Prefix LC

4.2.7 ERTMS Modes

In addition to the element rules, the modeller may check the rules which apply on the diagram where these elements are used: Mode State Machine (in section 4.1 of the document)

4.2.7-2, OCORA-1402, mandatory - System Capabilities Modes and States- Name

The name of the ERTMS shall start by the prefix ERTMS

4.2.8 State Transitions

In addition to the element rules, the modeller may check the rules which apply on the diagram where these elements are used: Mode State Machine (in section 4.1 of the document)

4.2.8-2, OCORA-7068, mandatory - State Transitions Description

Every State Transition shall have a Guard indicating the condition to perform the change of state. Name, summary, Effects and State Transition Realizations are optional fields.

5 Modelling rules for Transition from System Analysis to Logical Architecture

5-1, OCORA-8309, mandatory - Transition of validated elements

Only elements which have already been validated shall be transitioned from the System Analysis to the Logical Architecture

5-2, OCORA-8312, mandatory - Transition settings

Before performing a transition, the default transition settings shall be modified. All the Business Criteria shall be switched to normal. All the other parameters shall remain with their default values. The final settings are shown in the following picture:

Difference Categories

Set up difference categories in order to define what differences and elements are shown.
 - Filtered differences are never shown.
 - If there are focused differences, then only those are shown (unless filtered).

Category	Normal	Filtered	Focused
Basic			
Property differences	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Moves	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Order differences	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Additional elements on the left	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Additional elements on the right	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Merge process			
Merged differences	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Ignored differences	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Semantic criteria			
Actor	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Component	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Function	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Functional Chain	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Capability	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Data	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Interface	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Exchange Item	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Modes / States	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Functional Exchange	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Component Exchange	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Physical Link	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Part	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Function Port	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Component Port	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Physical Port	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Generalization	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Component Functional Allocation	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Component Exchange Allocation	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Component Exchange Functional Exchange Allocation	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Deployment Link	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Property Values	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Viewpoint elements	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Business criteria			
Name changes	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Summary changes	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Description changes	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Remove 'References modifications' on elements realized by many elements	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Remove 'Addition of Elements' on elements realizing many elements	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Applied property values	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Empty packages	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Attributes not automatically updated	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
References not automatically updated	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Component Allocation of parent functions	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Reset

Apply OK Cancel

*Note: The settings can be modified in the "Decide what differences must be shown according to various criteria" icon from the transition window, as shown below:

Functional Transition

Synthesis

- Playground (160)
 - Playground (115)
 - Logical Architecture (59)
 - Logical Functions (5)
 - Root Logical Function (5)
 - Test D (1)
 - FunctionalExchange 5
 - Test 2
 - Test 4
 - Capabilities (38)
 - Interfaces
 - Data
 - Structure (12)
 - REC Catalog (55)
 - Library Dependencies
 - ProgressStatus (6)
 - EXTENSIONS (35)
 - [projectApproach] SingletonComponents

Candidate model

Resulting model

Details

Cancel Apply OK

5-3, OCORA-8310, mandatory - Delete a transitioned element

If an element from the System Analysis which had already been transitioned has to be deleted, the corresponding transitioned element in the Logical Architecture shall be deleted manually (Deletions cannot be propagated automatically)

6 Modelling rules for Logical Architecture

6.1 Diagram types

6.1.1 Overall

6.1.1-1, OCORA-7577, mandatory - Diagram name (Logical Architecture)

Each diagram shall be named according to following convention:

[<Layer letter>].[<Capella diagram type abbreviation>] [Name]

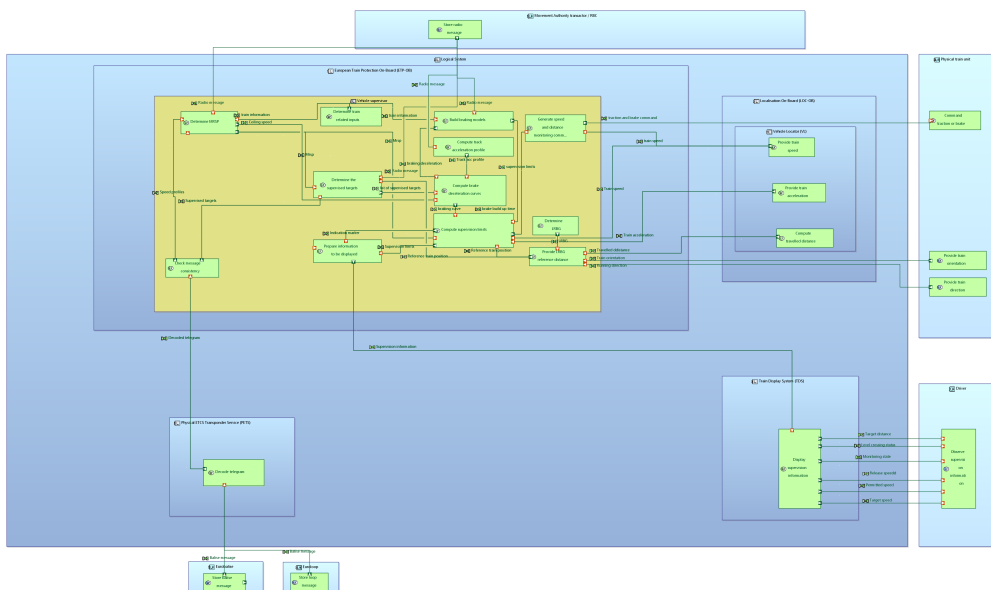
Layer letter: O=Operational, S=System, L=Logical, P=Physical

Capella diagram type abbreviation: abbreviation indicating the type of the diagram with the following possible values. It is provided by default by Capella:

- CRB - Capability Realisation Blank
- LDFB - Logical Data Flow Blank
- LFBD - Logical Function Breakdown
- LAB - Logical Architecture Blank
- L.ES - Logical Exchange Scenario (Default in Capella [ES])
- LDFB - Logical Functional Chain Description
- STM - Logical State Machine (Default in Capella [MSM])

Name: Free text. Check specific diagram type rules

6.1.2 Logical Architecture Blank



6.1.2-1, OCORA-7580, mandatory - L.LAB diagram content

The diagram shall contain the system of interest and at least one Logical Component (internal to the system).

Note:

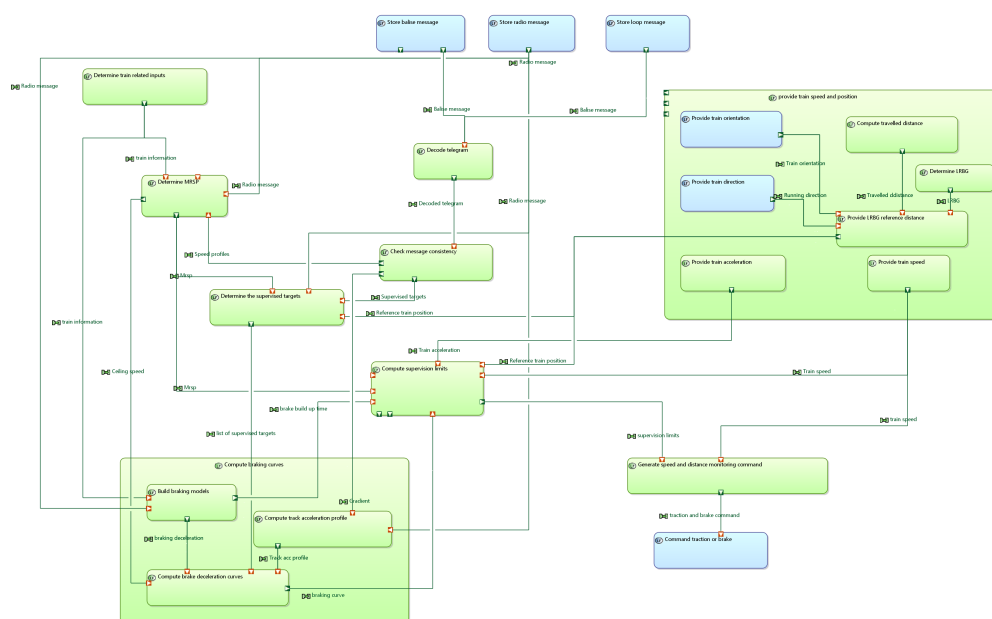
Creating a L.LAB for each logical component showing the exchanges with other components and the actors helps to breakdown a complex system into smaller readable parts.

In certain cases, an additional L.LAB, focusing on an actor showing its exchanges with the system, may also be created.

6.1.2-2, OCORA-7581, mandatory - L.LAB functions relationships

Each function shall be connected to at least one other function via an in or out functional exchange.

6.1.3 Logical Dataflow Blank



6.1.3-1, OCORA-7578, optional - L.LDFB layout

The Logical Functions shall be arranged in such a way that they fit possible extracts to PDF in A4 in portrait orientation or other similar formats. The diagram shall grow vertically, not horizontally.

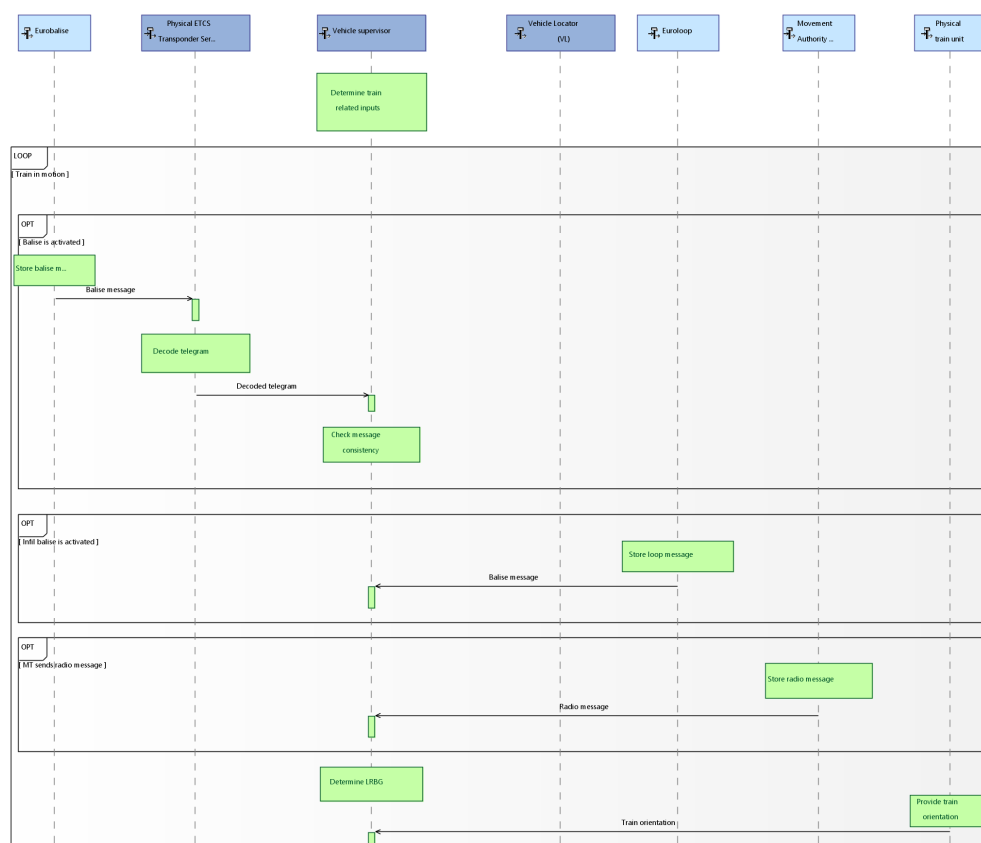
6.1.3-2, OCORA-7579, mandatory - L.LDFB functions relationships

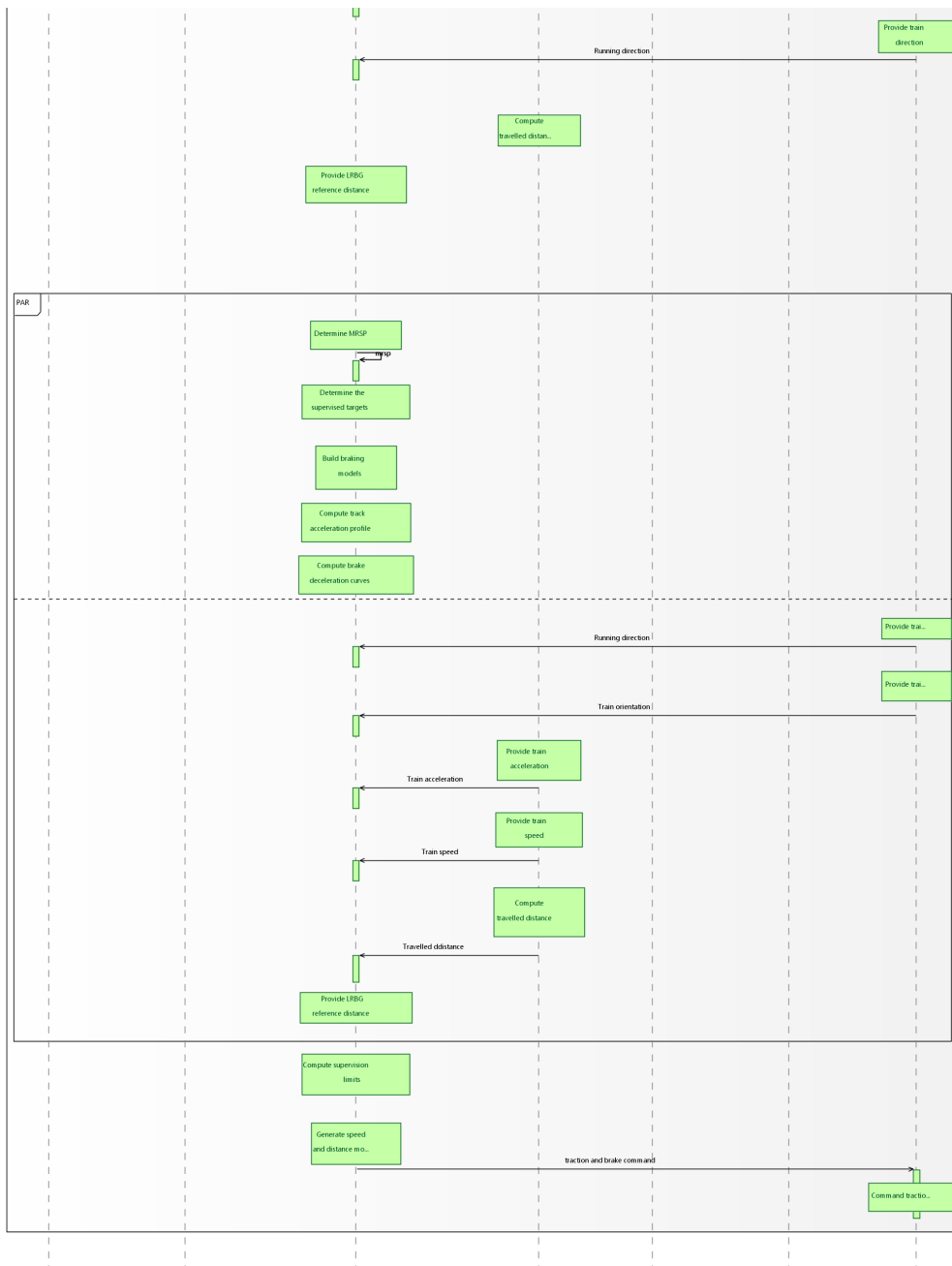
Each function represented in the diagram shall be linked to at least one function through an incoming or outgoing functional exchange.

6.1.3-3, OCORA-8024, mandatory - L.LDFB redundant functional exchanges

When there are redundant functional exchanges because of the functions breakdown, representing the high and the low level function, only the low level functional exchanges shall be displayed. The high level ones shall be hidden.

6.1.4 Logical Exchange Scenario





6.1.4-1, OCORA-7583, mandatory - L.ES OPT fragment

Where OPT fragments are used, OPT fragments have a defined guard condition describing the logical condition under which the functional exchange takes place.

Note: Formal logical condition writing is not mandatory, but the guards conditions shall be true/false statements

6.1.4-2, OCORA-7584, mandatory - L.ES LOOP fragment

When a LOOP combined fragment is used, it shall have a guard condition that describes the logical condition under which the combined fragment takes place.

Optional: you can specify a minimum and maximum number of iterations of the loop between parentheses. The format is: (<min.>, <max.>) and it shall be added in a note attached to the LOOP combined fragment (a note because this capability is not supported by Capella). If not specified, a default range (0, *) is assumed.

6.1.4-3, OCORA-7585, mandatory - L.ES ALT fragment

Where ALT fragments are used, at least two operands are required. Each operand of the fragment has a defined guard condition describing the logical condition under which the functional exchange takes place.

6.1.4-4, OCORA-7582, mandatory - L.ES ALT fragment consistency

Where ALT fragments are used,

the set of guard conditions for the fragment covers all eventualities

AND

there are no logical intersections between guard conditions (they are mutually exclusive)

6.2 Elements

6.2.1 Logical Function

6.2.1.1 Overall

6.2.1.1-1, OCORA-7549, mandatory - Logical Function name

Naming convention:

All function names are written using sentence case capitalisation. The first word in the name of the function will begin with a capital letter and the remaining words will be lower case.

Example: Compute braking curves profile

6.2.1.1-2, OCORA-8035, mandatory - Logical Function definition

A Logical Function shall describe a behaviour, not a constraint or a non functional requirement

6.2.1.1-3, OCORA-7551, mandatory - Leaf Logical Function allocation

All leaf-level logical functions have been **allocated** to

one of the proposed logical components

or

one of the actors

6.2.1.1-4, OCORA-7553, mandatory - Leaf Logical Function maximum refinement

The Logical Functions shall describe an abstract behaviour, not the solution space (i.e. the Physical Architecture)

6.2.1.1-5, OCORA-7552, mandatory - Logical Function description

Function Description - describe what the function is for, such that the following boxes can be ticked:

- The description explain what the function will produce (output)
- (optional) The description explains why the inputs are required
- (optional) The description explains how the inputs may be used to produce the outputs

Pattern: This function <verb>

Note: see rule  OCORA-1393

6.2.1.2 Logical Function allocated to Logical Component

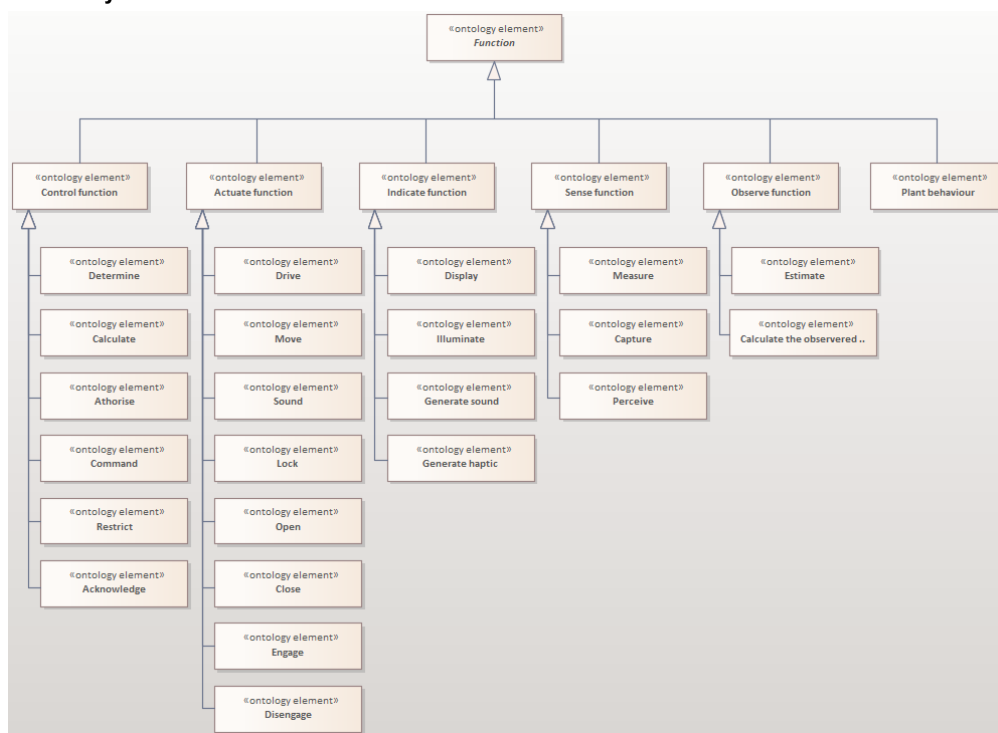
6.2.1.2-1, OCORA-7557, optional - Control system principle Each core logical function should be named and described...

Control system principle Each core logical function should be named and described in such a way that it conforms to one of the categories of control function. A "core" function is a function that is active when the system is in the OPERATIONAL state (or equivalent).

All functions are members of **one and only one** of the following categories:

- Control
- Actuate
- Indicate
- Sense
- Observe

Taxonomy:



6.2.1.3 Logical Function allocated to Logical Actor

6.2.1.3-1, OCORA-7558, optional - Logical Functions (actor) merge

Functions allocated to logical actors are permitted to merge multiple items from the control system pattern, where used; in this case, the name of the logical actor function may contain several verbs (e.g. Sense+observe or Control+actuate).

Note: This rule may be applied for instance to improve diagrams readability or to reduce modelling effort without added value

6.2.1.3-2, OCORA-7559, mandatory - Logical Functions (actor) purpose

Functions allocated to logical actors must describe the relevant behavior of the actors towards the system logical architecture. This means that the inputs actors shall provide to the system and the outputs they expect to obtain from the system must be unambiguous.

6.2.2 Functional Exchange

6.2.2-1, OCORA-7560, mandatory - Functional exchanges applicable rules

Same rules defined for Functional Exchanges at System level are applicable at the Logical Architecture (see section 4.2.5 of this document)

6.2.3 Logical Actor

6.2.3-1, OCORA-7561, mandatory - Logical Actor consistency with System layer

Each logical actor has an identical name to the corresponding system actor
and

The set of logical actors is identical to the set of system actors
and

Each logical actor has an identical description to the corresponding system actor

6.2.3-2, OCORA-7562, mandatory - Logical Actor traceability to System layer

Each logical actor traces to one and only one system actor

6.2.4 Logical Component

6.2.4-1, OCORA-7563, mandatory - Logical Component name

Naming convention:

Title Case - the first letter of each word is capitalised, words separated by space.

Name should be singular

The last word of the name should be a noun (e.g. Sensor, Actuator, Observer, Executor, Controller, Indicator)

The usage of abbreviations should be well considered and should only be used in cases where the abbreviation is either standardised (e.g. ETCS, PRM, GoA) or when it improves readability as in the case of known defined abbreviation within the project (e.g. DPS).

6.2.4-2, OCORA-8036, mandatory - Logical Component definition

The Logical Component shall describe a set of behaviour, not a physical component of the actual solution

6.2.4-3, OCORA-7570, mandatory - Logical Component ports direction

Logical component ports have direction consistent with the functional exchanges they support - this means either

Where a component exchange supports functional exchanges in both directions, the component port has direction INOUT

exclusive or

Where a component exchange supports only incoming functional exchanges, the component port has direction IN

exclusive or

Where a component exchange supports only outgoing functional exchanges, the component port has direction OUT

6.2.5 Logical Component Exchange

6.2.5-1, OCORA-7571, mandatory - Logical Component Exchange (LC-Actor) name

Naming convention:

All component exchanges between logical components and an external actor crossing the system boundary are named in accordance with the following naming convention (pattern):

Prefix for interface to a non human actor: CI_

Prefix for interface to a non human actor standardized by OCORA: SCI_

Prefix for interface to a human actor: HMI_

The ActorName should be written for component exchanges in PascalCase - All words capitalised and run together. First letter is capitalised.

Each logical component exchange with the same name should have an unique number as part of the name (e.g. one component exchange on system level now splitted into two because the target is not the system anymore but two logical components.)
If there are more then one component exchanges for one interface (interface layer) each component exchange should end with the layer information: _<InterfaceLayerName> (But please do not include the word "layer" in the layer name)

Guidance:

logical component exchange name = CI_ / SCI_ / HMI_ ActorName>_<unique number starting at 01>_<InterfaceLayerName>

Examples:

CI_PlanningSystem_01_Application

SCI_PlanningSystem_02_Application

6.2.5-2, OCORA-7573, mandatory - Logical Component Exchange (LC-LC) name

All component exchanges between logical components inside the system boundary are named in accordance with the following naming convention (pattern):

C <unique number>

Note:

The standard mechanism of Capella is used which provides an automatic generated name and number for the created component exchanges.

Example:

C 03

C 19

6.2.5-3, OCORA-7572, mandatory - Logical Component allocation of Functional Exchanges

Where a functional exchange crosses the boundary of the allocating logical component A

the functional exchange is assigned to the logical component exchange

and

the function port is assigned to a logical component port on A (automatically done by capella when functional exchange is assigned to component exchange)

and

the assigned logical component port is connected to a logical component port on another logical component B

and

logical component B hosts the function at the other end of the functional exchange

Corollary: When functions are re-allocated to other logical components, function port allocations to component ports must be revisited to ensure all three conditions above are still met. The tool will not always stop the user from leaving an invalid port allocation after moving a function.

6.3 Views

6.3.1 Overall

6.3.1-1, OCORA-7590, mandatory - Views automatic synchronisation

The diagrams generated in the following views shall enable in the tool (Capella) the automatic synchronisation functionality:

- Logical Component external interfaces view
- Logical Component internal view
- Overall CCS-OB

On all other diagrams the automatic synchronisation functionality shall be disabled.

6.3.2 Capability Logical Architecture

6.3.2-1, OCORA-7595, mandatory - Capability Logical Architecture name

The diagram shall carry the same name of the Capability that it details. In case there are several diagrams for one single Capability, free text will be added to indicate the diagram purpose.

Ex: SysC21 Supervise train speed without train integrity guarantee

6.3.2-2, OCORA-7591, mandatory - Leaf-functions visibility

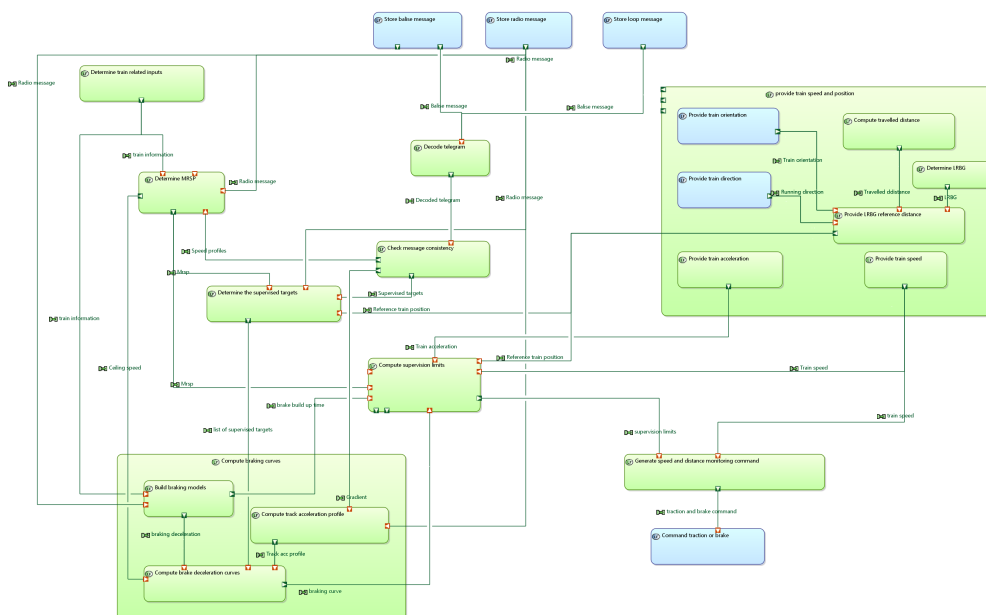
All relevant leaf-level functions are visible. Parent functions are hidden.

6.3.2-3, OCORA-7592, mandatory - Capability Logical Architecture content

The content displayed in the diagram is compliant with the following points:

- Showing all logical components which are part of a realised capability and all functions allocated to this logical components, the allocations of functional exchanges to component exchanges for functional exchanges crossing the border of the logical components.
- The filter 'Hide Functional Exchanges' and 'Show Exchange Items on Component Exchanges' is activated

6.3.3 Capability Logical Dataflow



6.3.3-1, OCORA-7594, mandatory - Capability Logical Dataflow name

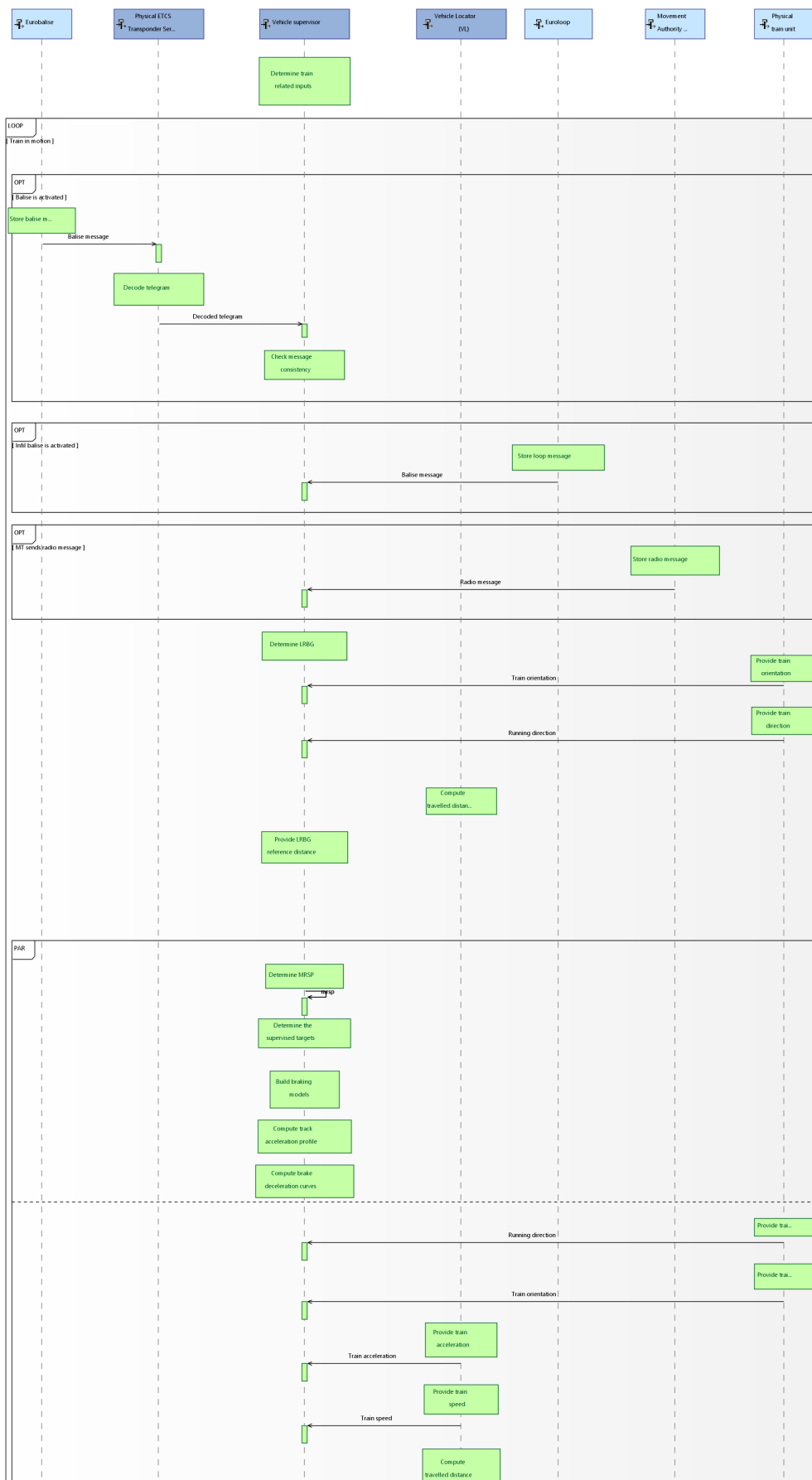
The diagram shall carry the same name of the Capability that it details. In case there are several diagrams for one single Capability, free text will be added to indicate the diagram purpose.

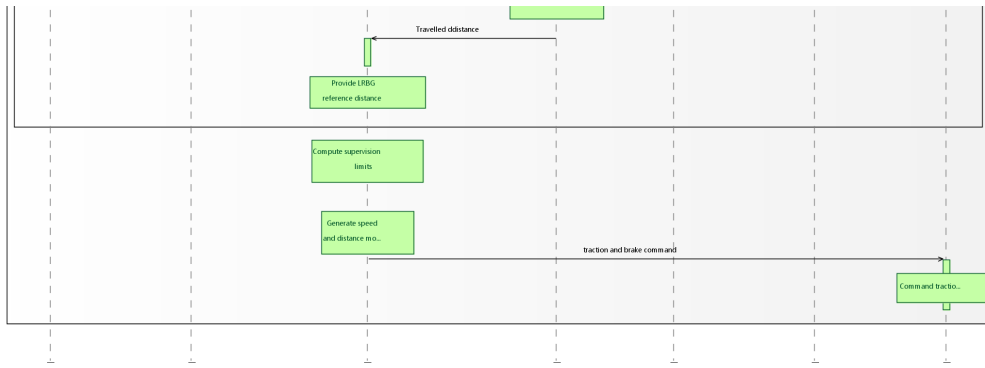
Ex: SysC21 Supervise train speed without train integrity guarantee

6.3.3-2, OCORA-7593, mandatory - Capability Logical Dataflow content

The diagram displays all functions AND functional exchanges involved in the fulfilment of the owning capability, including functions allocated to actors.

6.3.4 Capability Exchange Scenario





6.3.4-1, OCORA-7616, mandatory - Capability Exchange Scenario name

The diagram shall carry the same name of the Capability that it details. In case there are several diagrams for one single Capability, free text will be added to indicate the diagram purpose.

Ex: SysC21 Supervise train speed without train integrity guarantee

6.3.4-2, OCORA-7617, mandatory - Capability Exchange Scenario content

The diagram displays all functions AND functional exchanges involved in the fulfilment of the owning capability, including functions allocated to actors.

6.3.4-3, OCORA-7615, mandatory - Capability Exchange Scenario obligatory nature

The Capability Exchange Scenario is mandatory only in case there is at least one exchange between two different Logical Components.

If there are no internal exchanges within the System or they all happened inside the same Logical Component, the Capability Exchange Scenario is optional and it is up to the modeller to decide whether to include it or not.

6.3.5 Building block external interfaces view

**Note: A building block is a sourceable unit of the CCS on-board system (hardware and/or software), having standardised functionality, standardised performance (RAM), standardised safety (including Tolerable Functional Failure Rate [TFFR], Safety Integrity Level [SIL] and Safety Related Application Conditions [SRAC]), standardised security and standardised interfaces towards other building blocks and/or external systems.*

Combined HW/SW Building blocks exclusively communicating with each other through the CCS Communication Network (CCN) using standardise interfaces.

SW Building blocks, deployed on an instance of the generic (safe) computing platform, shall communicate with each other through the standardised Platform Independent API. Communication with computing platform external systems will be realised by the computing platform (integrating with the CCN).

Building blocks are modifiable / adaptable and therefore portable / replaceable, without impacting other building blocks.

**Note 2: This view is built by using a Logical Architecture Blank diagram*

6.3.5-2, OCORA-7623, mandatory - Building Block external interfaces view name

The diagram shall carry the same name of the Building Block that it details plus the suffix -external interfaces view.

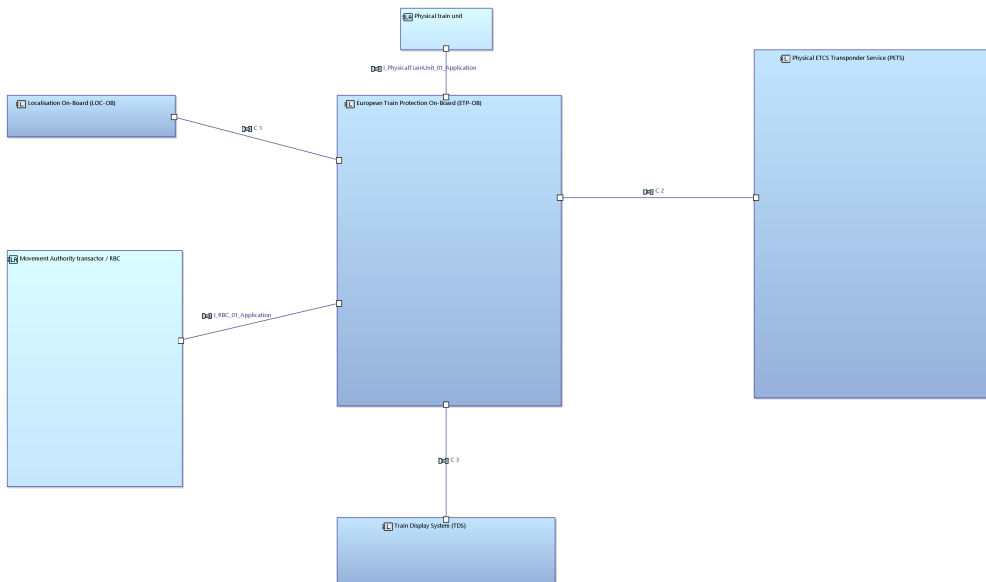
Ex: European Train Protection On-Board - external interfaces view

6.3.5-3, OCORA-7624, mandatory - Building Block external interfaces view content

The diagram displays all the Logical Component Exchanges related to the Building Block. The rest (those which do not apply to the Building Block subject of the view) shall remain hidden.

Logical Components which do not interface with the Building Block subject of the view are hidden.

All functions and functional exchanges are hidden.



6.3.6 Building block internal view

*Note: A building block is a sourceable unit of the CCS on-board system (hardware and/or software), having standardised functionality, standardised performance (RAM), standardised safety (including Tolerable Functional Failure Rate [TFFR], Safety Integrity Level [SIL] and Safety Related Application Conditions [SRAC]), standardised security and standardised interfaces towards other building blocks and/or external systems.

Combined HW/SW Building blocks exclusively communicating with each other through the CCS Communication Network (CCN) using standardise interfaces.

SW Building blocks, deployed on an instance of the generic (safe) computing platform, shall communicate with each other through the standardised Platform Independent API. Communication with computing platform external systems will be realised by the computing platform (integrating with the CCN).

Building blocks are modifiable / adaptable and therefore portable / replaceable, without impacting other building blocks.

*Note 2: This view is built by using a Logical Architecture Blank diagram

6.3.6-2, OCORA-7618, mandatory - Building Block internal view name

The diagram shall carry the same name of the Building Block that it details plus the suffix -internal view.

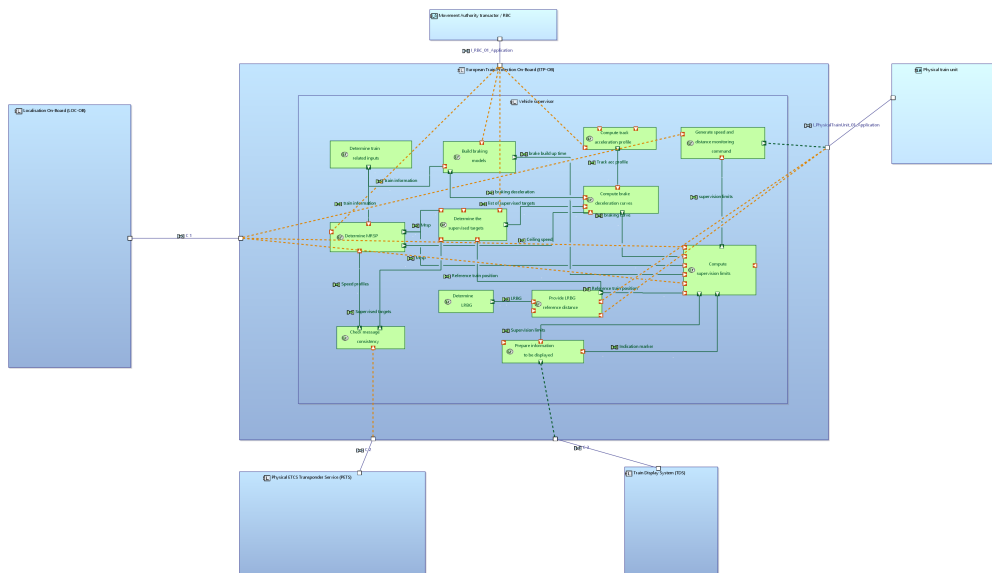
Ex: European Train Protection On-Board - internal view

6.3.6-3, OCORA-7619, mandatory - Building Block internal view content

The diagram displays all leaf functions allocated to the Building Block detailed in the view AND their incoming and outgoing functional exchanges. All other functions and functional exchanges are hidden.

The diagram displays all the Logical Component Exchanges related to the Building Block. The rest (those which do not apply to the Building Block subject of the view) shall remain hidden.

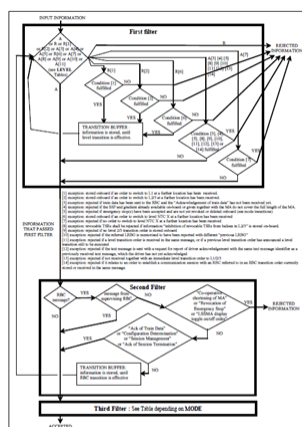

Logical Components which do not interface with the Building Block subject of the view are hidden.

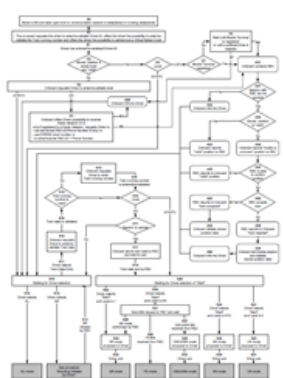


6.3.7 Detailed Scenario

6.3.7-1, OCORA-7625, mandatory - Implementation of Sequence Diagrams from Subset 026

The following Sequence Diagrams from subset 026 shall be implemented in the CCS-OB Logical Architecture:

Extract	Chapter	Description	POLARION link
 <p>Figure 3: schematic representation of the filtering of received information.</p>	subset 026 §4.8.1	<p>Acceptance of received information</p> <p>3 stages for evaluating the received information from balise or radio and assess if it is accepted or rejected</p> <p>The sequence contains ALT fragment in order to model the conditional path</p> <p>This sequence can be model in a dedicated diagram and called in a larger one in a REF fragment</p>	 <p>OCORA-5287 - Introduction</p>



subset
026
§5.4.4

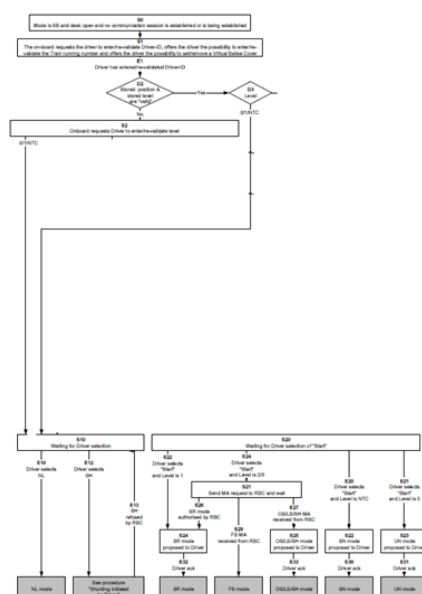
Start of mission

This flowchart describes each path which conduct to the beginning of a mission

We can describe it in one operational scenario with ALT fragment which includes the following path

- NL initiated by driver
- SH initiated by driver
- SR mode
- FS mode
- OS/LS/SH mode
- SN mode
- UN mode

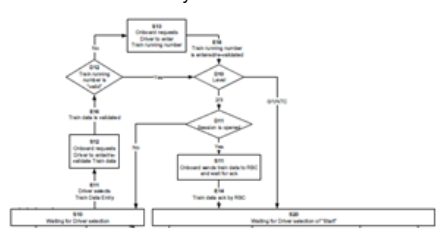
and some REF fragment for specific sequence of the scenario : data entry, connection to RBC



Refers also to ERA DMI §11.7.2 start up, §11.7.3 main window

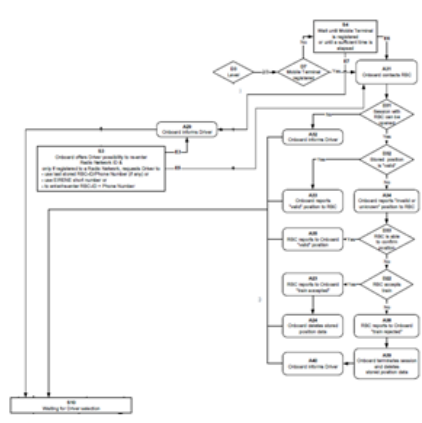
we can identify the following sequence diagram which can be called as a REF in a operational scenario

Process data entry

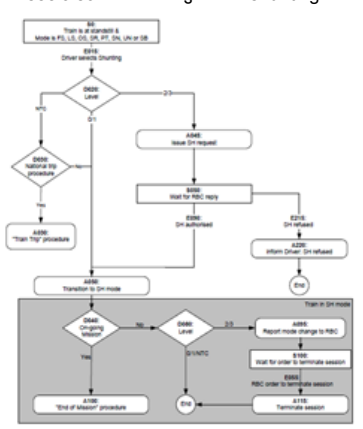


see also sequence of data check ERA DMI §10.3.4.7

Process RBC connection












- Shunting initiated by driver : cf §5.6
<https://trace.sbb.ch/polarion/redirect/project/OCORA/workitem?id=OCORA-5693>
 see also ERA DMI §11.7.4 shunting







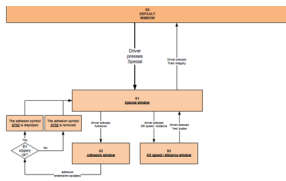
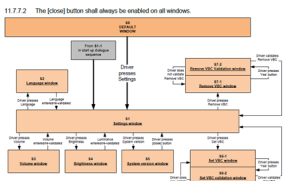


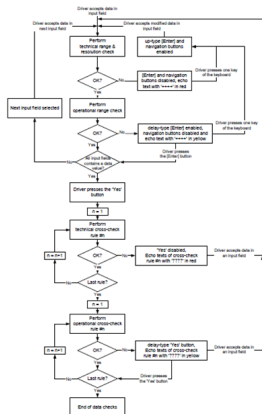
others sequence can be captured :

- Loss of an already open connection
- Driver re-enter data entry before start of mission (2 times in the process)
- Describes the possibilities list in §5.4.5.3

no figure	subset 026 §5.5	Procedure end of mission Entry to mode considered as a and of mission End of mission procedure	 OCORA-5668 - Procedure End of Mission
no figure	subset 026 §5.7	shunting with order from trackside with different condition : for the current location / for a further location	 OCORA-5723 - Entry in Shunting with Order from Trackside
no figure	subset 026 §5.8	Procedure override	 OCORA-5748 - Procedure Override
no figure	subset 026 §5.9	Procedure on sight	 OCORA-5783 - Procedure On-Sight
no figure	subset 026 §5.10	Level transition	 OCORA-5822 - Level Transitions
no figure	subset 026	Procedure train trip	 OCORA-5947

	§5.11		- Procedure Train Trip
no figure	subset 026 §5.12	change of train orientation	 OCORA-5979 - Change of Train Orientation
no figure	subset 026 §5.13	train reversing	 OCORA-6014 - Train Reversing
no figure	subset 026 §5.14	joining splitting	 OCORA-6023 - Joining / Splitting
no figure	subset 026 §5.15	RBC handover	 OCORA-6041 - RBC/RBC Handover
no figure	subset 026 §5.16	Procedure passing a non protected level crossing	 OCORA-6103 - Procedure passing a non protected Level Crossing
no figure	subset 026 §5.17	Changing Train Data from sources different from the driver	 OCORA-6116 - Changing Train Data from sources different from the driver
no figure	subset 026 §5.18.2	Passing a powerless section with panto to be lowered	 OCORA-6145 - Passing a powerless section with pantograph to be lowered
no figure	subset 026 §5.18.3	Passing a powerless section with main power switch to be switch off	 OCORA-6160 - Passing a powerless section with main power switch to be switched off
no figure	subset 026 §5.18.4	passing a non stopping area	 OCORA-6172 - Passing a non stopping area
no figure	subset 026 §5.18.5	passing a radio hole	 OCORA-6179 - Passing a radio hole

no figure	subset 026 §5.18.6	passing a air tightness area	 OCORA-6183 - Passing an “air tightness” area
no figure	subset 026 §5.18.7	inhibition of a special brake	 OCORA-6195 - Inhibition of a defined type of brake
no figure	subset 026 §5.18.8	advinsing a tunnel stopping area	 OCORA-6204 - Advising a tunnel stopping area
no figure	subset 026 §5.18.9	souding the horn	 OCORA-6212 - Sounding the horn
no figure	subset 026 §5.18.10	changing the traction system	 OCORA-6218 - Changing the traction system
no figure	subset 026 §5.19	limited supervision	 OCORA-6227 - Procedure Limited Supervision
no figure	subset 026 §5.20	generation of track condition to an external function	
 <small>Figure 140 - Special window dialogue sequence</small>	ERA DMI 11.7.6	special window	
 <small>Figure 141 - Settings window dialogue sequence</small>	ERA DMI 11.7.6	settings window	

[illegible]

6.3.7-2, OCORA-7626, mandatory - Implementation of State Machines from Subset 026

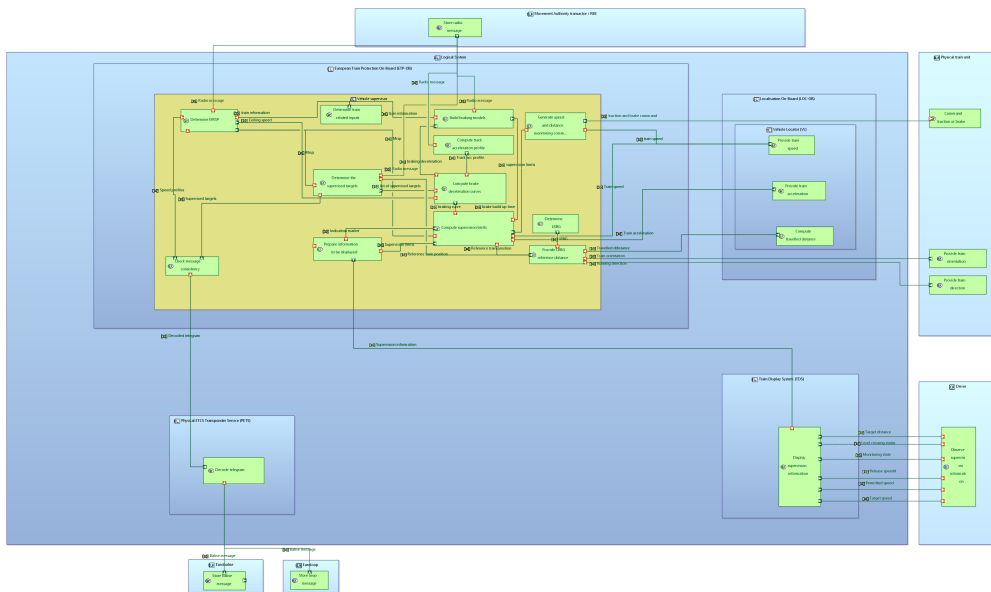
The following State Machine Diagrams from subset 026 shall be implemented in the CCS-OB Logical Architecture:

[illegible]

6.3.7-3, OCORA-7627, optional - Additional detailed scenarios

The modeller may add further views of detailed scenarios if needed

6.3.8 Overall CCS-OB



*Note: This view is built by using a Logical Architecture Blank diagram

6.3.8-1, OCORA-7620, mandatory - Overall CCS-OB view name

The diagram shall be named CCS-OB Logical Architecture

6.3.8-2, OCORA-7621, mandatory - Overall CCS-OB view content

The diagram shall display all the Logical Components and Logical Component Exchanges of the CCS-OB system. Functions and Functional Exchanges shall be hidden.

6.3.8-3, OCORA-7622, optional - Overall CCS-OB additional views

The modeller may add further views of the whole CCS-OB system if needed

7 Modelling rules for Physical Architecture

This section will be provided in an upcoming release.

8 Model review

8-1, OCORA-7087, mandatory - Model review

The model shall be reviewed by the different OCORA partners (SBB, SNCF, DB, NS) before publishing model content in official OCORA Releases. The purpose of the review is to validate:

- The model content
- The model format and its compliance with the set of rules defined in this Guideline

8-2, OCORA-7090, optional - Review System Capabilities relationships consistency

The reviewers shall check the consistency in the System Capabilities relationships since the tool does not include automatic validation rules for this purpose