Explanation to the RCA ARCH Processes

Dr. Lea Steurs, Dr. Oliver Lemke, Jorge Block

Content

Figures

1 RCA Modelling and Tooling (M&T) Setting

The RCA cluster M&T provides all services around tooling, processes and modelling rules for currently all other clusters in RCA working on the RCA model. Broader European usage of the M&T services is possible and under discussion. The current work mode setup between M&T and the clusters is presented in Figure 1. The tooling platform is the basis for all modelling work done in the different clusters. M&T provides the needed processes, modelling rules and enables core modelers. The core modelers are included in workgroups of different clusters and are there formalizing the model together with domain experts. While domain experts have only basic knowledge of the modelling rules, the core modelers are the experts providing modelling knowledge as a service. With such a centralised service, it is possible to achieve a common, overarching model as basis for discussions with industry. Besides the pure modelling rules, common processes like document management, configuration management, review, etc. need to be defined and will be delivered by the M&T cluster.

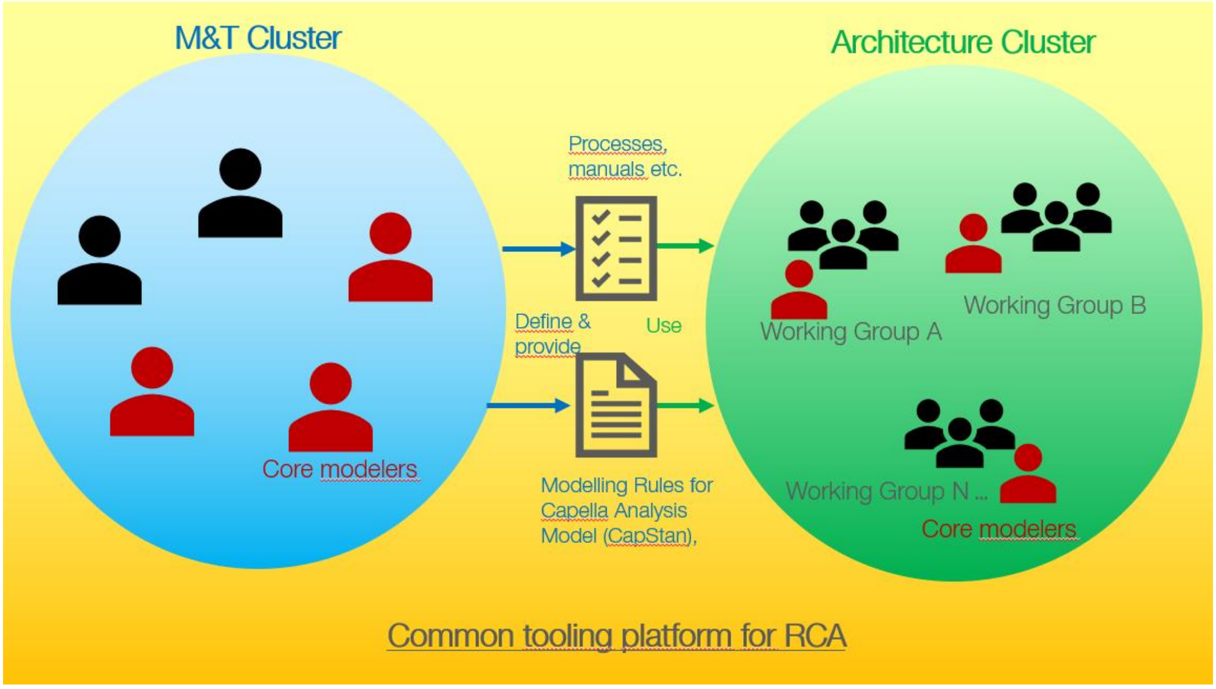


Figure 1 Setting of M&T as central service for all RCA clusters using the same tooling platform.

Currently about 4 FTE are working on processes and modelling rules, about 5 FTE on the tooling side including user support for RCA and EULYNX. Setting up a platform and a full set of modelling rules and processes took approximately 2 years and is still an ongoing effort (right now 190 subprocesses and 146 modelling viewpoints).

2 The ARCH Process

The ARCADIA method already provides certain rules at a high level, but for detailed work, distributed over different groups, more guidance is needed. Especially in the case of RCA, where new stakeholder needs need to be assessed that lead to innovative products with influence on the whole railway system. To break down the complexity of such a new system and to obtain well defined requirements, strict modelling rules and a reliable process are key to success. In the case of RCA therefor the ARCH process based on the ARCADIA method was established. To not reinvent it bases on the existing work of DB. It is crucial, to achieve a common understanding of how certain features are expressed in the model. The ARCH process provides a clear work breakdown structure and rules to produce a consistent set of artifacts that ensures consistency of different approaches and help to get compliance on laws and regulations. It guarantees therefore, that the different groups working on the same model get to comparable results. The modelling rules also serve external experts, such as industry engineers, to exactly interpret the model. In Figure 2 the overall meta model of process and methodology as applied at RCA is shown. The following figures present different levels of the ARCH process and their link to the specific modelling rules as applied in RCA. The whole documentation is currently done in confluence, an easy-to-navigate documentation tool. In addition, a PDF export of the current status was generated to grant access to the modelling rules for everyone.

RCA agreed on using the ARCH process already used and provided by DB. Therefore some parts of the process still refer to DB internal information, but nevertheless the overall process is applicable to RCA. Cleaning up the ARCH process to a RCA process is ongoing work.

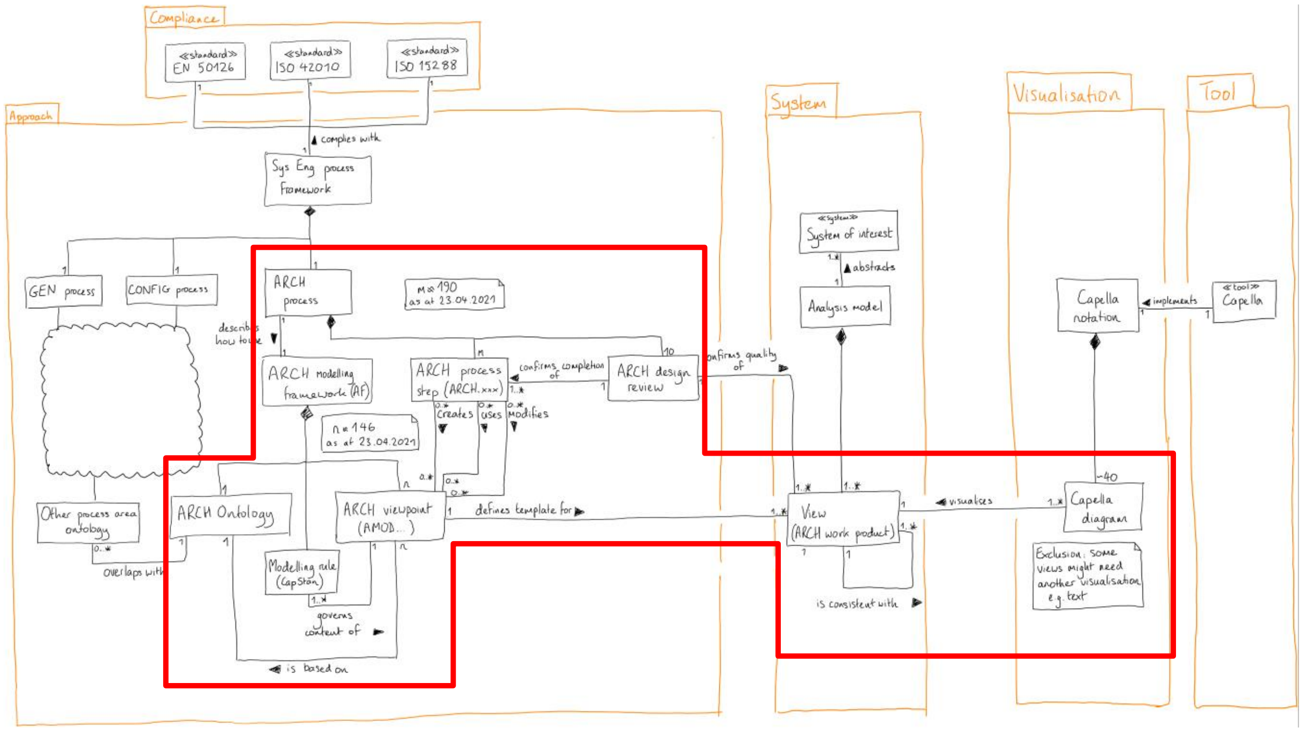


Figure 2 Meta model for process and methodology. The highlighted ARCH process as a core is further explained in Figure 3 & 4.

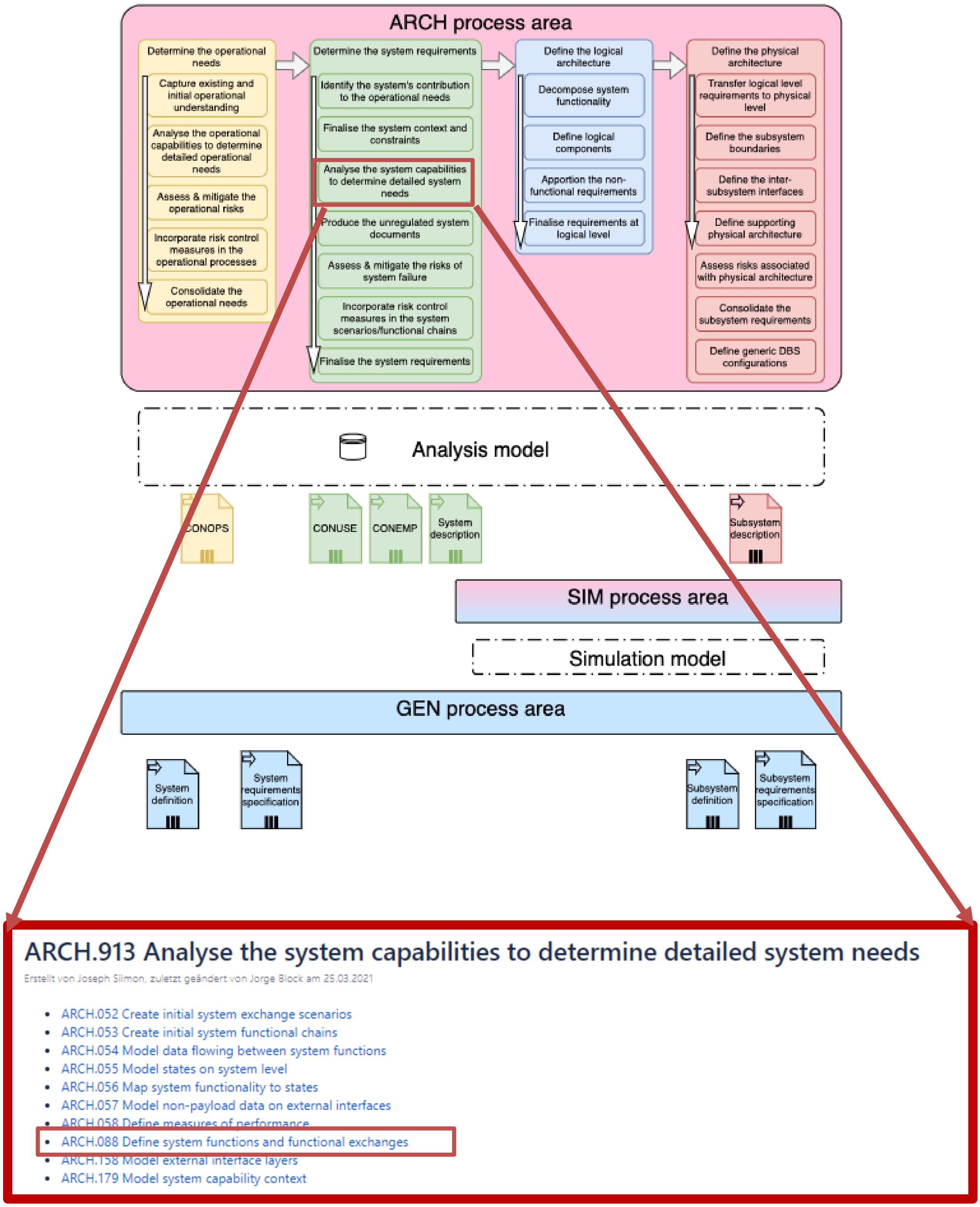


Figure 3 The ARCH process based on the ARCADIA method. For each box in the coloured columns above, there are multiple process steps defined in confluence.

ARCH.088 Define system functions and functional exchanges

Erstellt von Oscar Rubinstein, zuletzt geändert von Jorge Block am 26.03.2021

SM-2793 - Der Jira-Vorgang existiert nicht oder Sie sind nicht anzeigeberechtigt.

SM-4717 - Der Jira-Vorgang existiert nicht oder Sie sind nicht anzeigeberechtigt.

Goal	Identify and capture functions and exchanges that are needed for a particular capability
Requirements met by this process step	See ARCH.913 Analyse the system capabilities to determine detailed system needs
Inputs	Individual needs identified during ARCH.052 Create initial system exchange scenarios and ARCH.053 Create initial system functional chains. (this activity should run in parallel with those two activities) System functional chains from collaborative projects (optional) System exchange scenarios from collaborative projects (optional)
Outputs	AMOD-056 System functions and exchanges (single system capability)
Methodology	As a exchange scenario or functional chain is built, functions are arranged in a sequence. Functions/functional exchanges can be reused if appropriate, but if no existing function/exchange meets the need at that point in the scenario/process, then a new function needs to be created. That is what this activity is for. As a part of this step, the rules and the guidelines for the definition of system functions should be followed defined within 2. Modelling Rules for System Analysis#ModRules_VE_SysFunction. Additional information can be found here Pattern for defining system functions. Each system function associated with operational states of the system under consideration ("core function") will then be assigned to one of five categories representing the main functions of a control system (control, actuate, indicate, sense, observe). The types of resulting functional exchanges between these core functions shall also be limited to those defined in the 2. Modelling Rules for System Analysis#ModRules_VE_SysFuncExchange. If functions already created previously before the ARCH process has been completely introduced, this functions should be always be reused/renamed, in preference to creating new ones, **if it is appropriate**. If relevant system functions and functional exchanges are already defined at an international level (e.g. RCA), these should be referred to and reused if appropriate.
Tools and non-human resources	Team for Capella
Cardinality	Once per system capability.
Completion criteria	The output diagram conforms to its modelling rules. The set of functions and functional exchanges identified is safe enough to try.
Design review	ARCH.R.3 - System feature review
Step done by (Responsible)	System architect
Provides input to/assists (Contributes)	Inclusive OR: • Subsystem architect • Expert for GoA4 railway operation • Cross-cutting engineer • Engineer
Uses outputs (Informed)	RAMS manager RAMS engineer Verification & validation manager

Figure 4 Example of a process description on the lowest level (ARCH.088 in Figure 3).

AMOD-056 System functions and exchanges (single system capability)

Erstellt von Saurabh Pandey [X], zuletzt geändert von Jorge Block am 22.04.2021

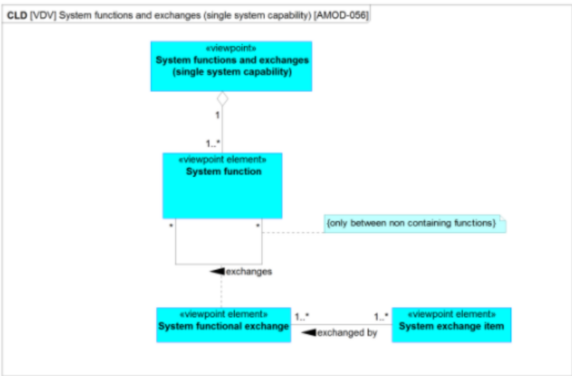
Created by	ARCH.088 Define system functions and functional exchanges
Concerns	Define the functions needed to achieve a particular system capability Define the functional exchanges needed to achieve a particular system capability
Modified by	ARCH.052 Create initial system exchange scenarios ARCH.053 Create initial system functional chains
Used by	ARCH.065 Assess safety risks of functional chain ARCH.070 Assess information security risks of functional chain ARCH.075 Assess operational performance risks of functional chain
Viewpoint	**In Capella this corresponds to a SDFB diagram.** **Mandatory Viewpoint:** 
Viewpoint modelling rules	2. Modelling Rules for System Analysis#ModRules_VP_SDFB
Element modelling rules	2. Modelling Rules for System Analysis#ModRules_VE_SysFuncExchange 2. Modelling Rules for System Analysis#ModRules_VS_SysFunction
Specific view modelling rules	2. Modelling Rules for System Analysis#ModRules_VS_AMOD-056

Figure 5 Example of a viewpoint definition created for process step ARCH.088 presented in Figure 4. Here also the correspoding modelling rules are linked.

3 Explanation of ARCH process layers in Capella

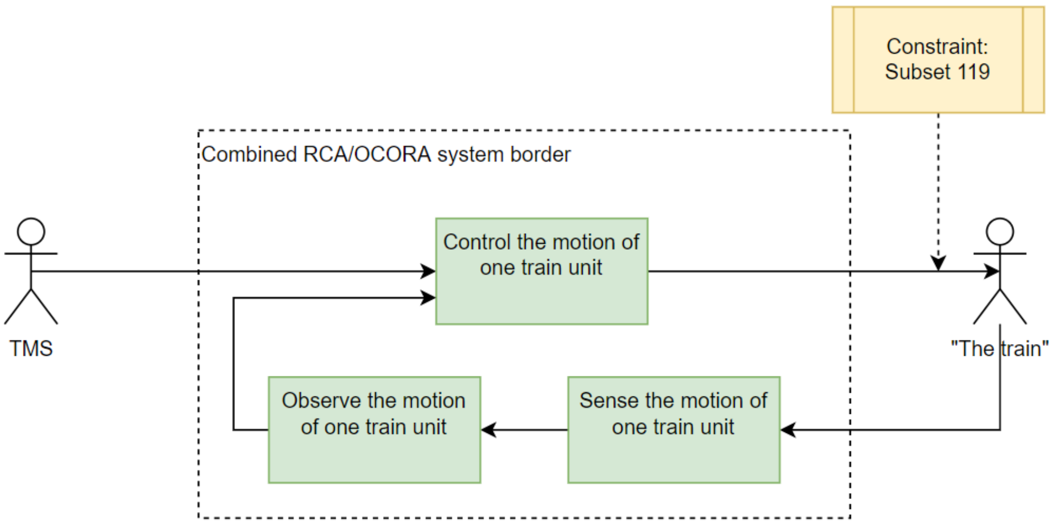
The ARCADIA method does not attempt to provide a full blown process for specific domains such as RCA. It provides a method for a functional analysis to derive a system architecture. All domain specific needs have to be defined by the domain experts. Therefore, M&T provides with the ARCH process such a framework.

In the following, an example per layer is presented. Especially in examples including interfaces to the Traffic Management System (TMS) and the train onboard CCS system (OCORA) the overall complexity becomes visible. The operational layer was decided to be defined by each infrastructure manager, therefore is not part of the RCA work package and not further explained here.

3.1 System architecture (SA) layer

This layer represents the system needs or "Definition of work statement" - and does not show any solutions, just what the task is that needs to be fulfilled by the corresponding system.

Example



In this example, that roughly translates to:

- We need a system that interacts with TMS and the train
- The system needs to control the motion of that train
- The system must be able to sense and observe the motion of that train
- The observation must be fed back to form a control loop
- The interface to the train must be compliant to subset 119. The interface to TMS is not constraint (we can design whatever we want: morse code, fax, homing pigeon).

The SA layer does not yet mention any solution concepts and should be agnostic as possible. The above example could be implemented by todays electronic interlockings, trackside signals and legacy train control systems like PZB or ETCS..

Resulting required decision points between OCORA and RCA as an example for the overall complexity to be handled:

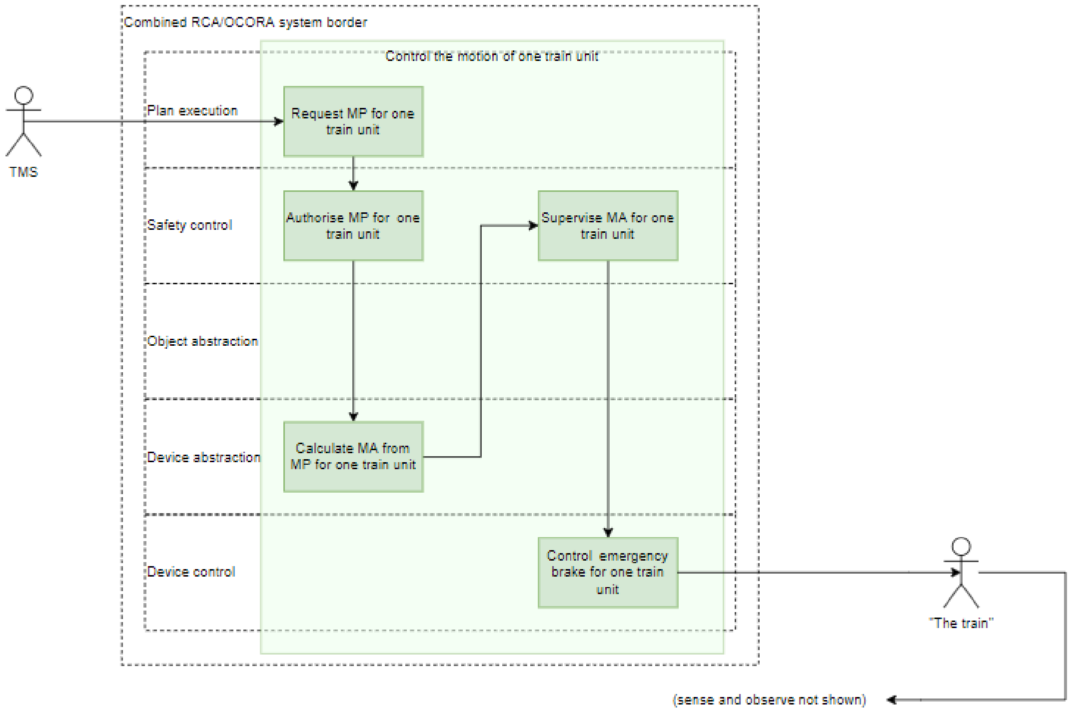
- System border and actors need to be agreed on: what is in and what is out of our combined system and to what are we talking to the outside?
- Functions (and system capabilities - not shown here) for the entire system (what does our combined system do to deliver the needs from the stakeholders?)
- RAMSS parameter (how safe, secure and well does the entire system need to do this?)

3.2 Logical Architecture (LA) layer

This layer is used "Defining subproblems, introduction of basic solution concepts and logical building blocks" – and shows, which basic ideas and concepts are used to define small "bricks" of solutions that can be used to build an architecture upon.

LA layer **does not yet define a system architecture, but a library of building blocks to construct an architecture from.** Also, the LA layer does not yet define, if a solution is realized e.g. on the track or onboard side. There are many splitting criteria to build logical building blocks, but an important one is a layer architecture of layers with a specific purpose, e.g. safety control is a layer that decides authoritatively about the state changes of a controlled object.

Example



In the represented example case, that roughly translates to (only shown on the control function):

- The system function splits up into 5 logical functions according to the purposes of the layers. One layer can be used multiple times.

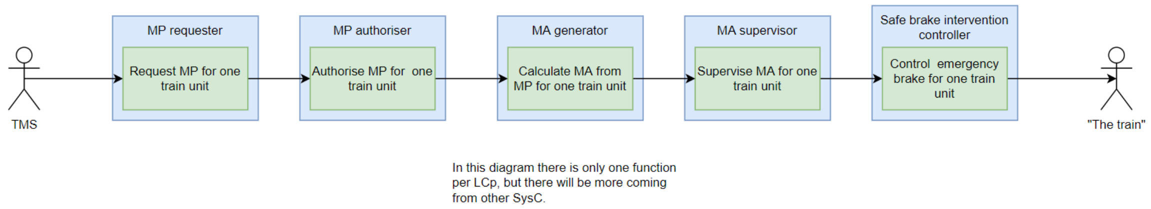
- First the Movement Permission (MP) is introduced as concept; the MP is requested by the plan execution function (as it executes the operational plan coming from TMS)
- Then the MP is authoritatively allowed or denied by a safety control function – meaning, there is nobody else allowing or denying a MP
- Then the MP is translated to a Movement Authority (MA) in the abstraction layer
- Then the driving is authoritatively supervised against MA, this again is a safety layer function
- In case of a violation of the MA a safety reaction is triggered and commanded to the vehicle by the device control layer

- We can use all the nice concepts we think of: MPs, MAs, etc. all those solution concepts are fully valid here and determine the splitting of a system function

Logical functions are also the elements that will carry the specification of their behavior. This can be done by natural language (not recommended) or by semi-formal or formal means of specification. Also non-functional requirements are derived for the functions (e.g. accuracy, latency, failure rates...).

Functions are then allocated to logical components. These logical components are:

- *not subsystem* but small providers of services
- not instantiated, so there is only 1 per system
- not allocated to the deployment location (trackside, OB or cloud...)

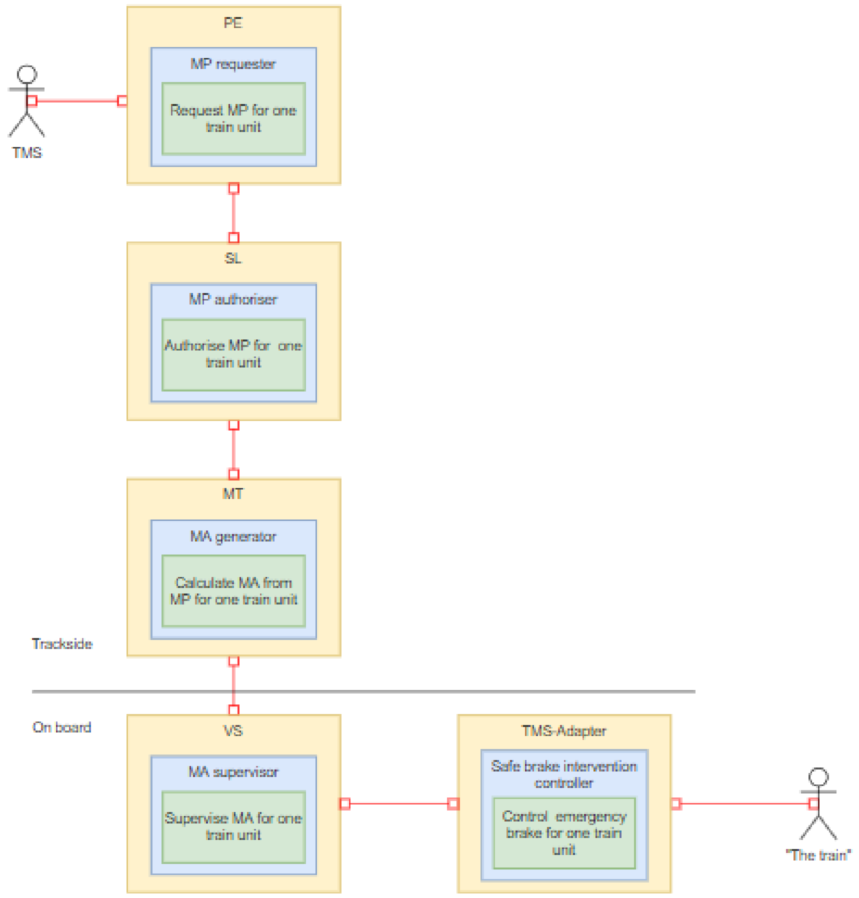


3.3 Physical Architecture (PA) layer

"Do system architecture, define subsystems and interfaces" - in this layer, the "real" subsystem architecture is created. **Only from here** an allocation to trackside/on board is possible, also in RCA-terms an allocation of functionality to domains should happen here. Usually, multiple physical architectures are possible based on the **same** logical architecture.

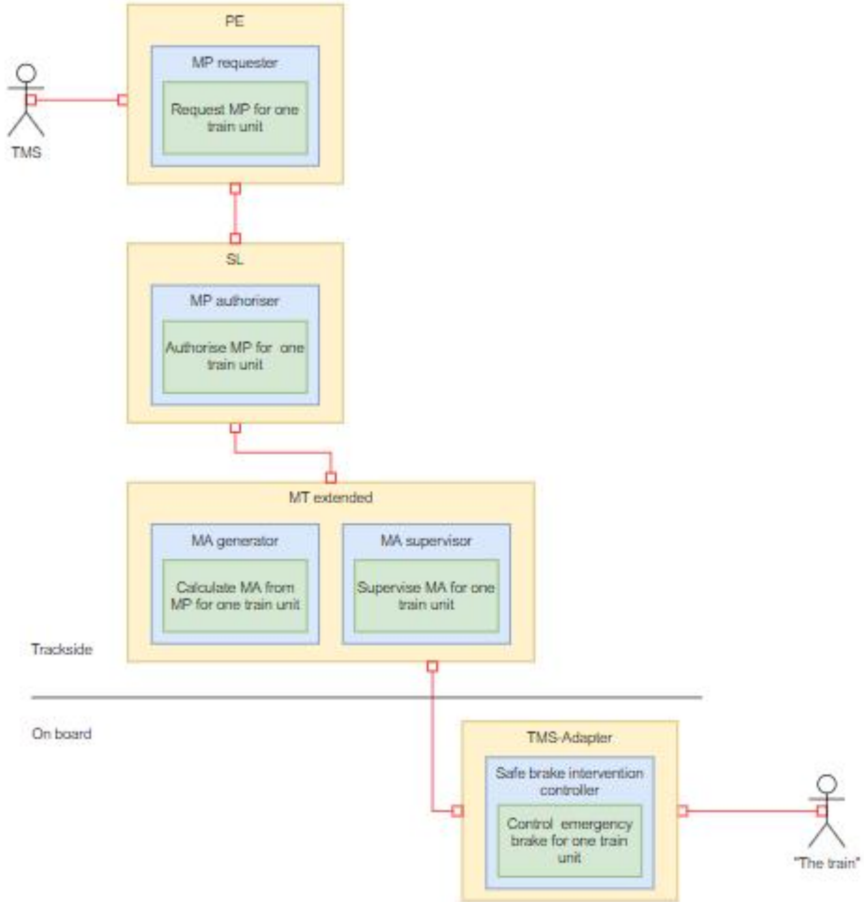
Example 1

The first example shows, how the logical components are allocated to subsystems in a classical way creating roughly the existing ETCS SS026.



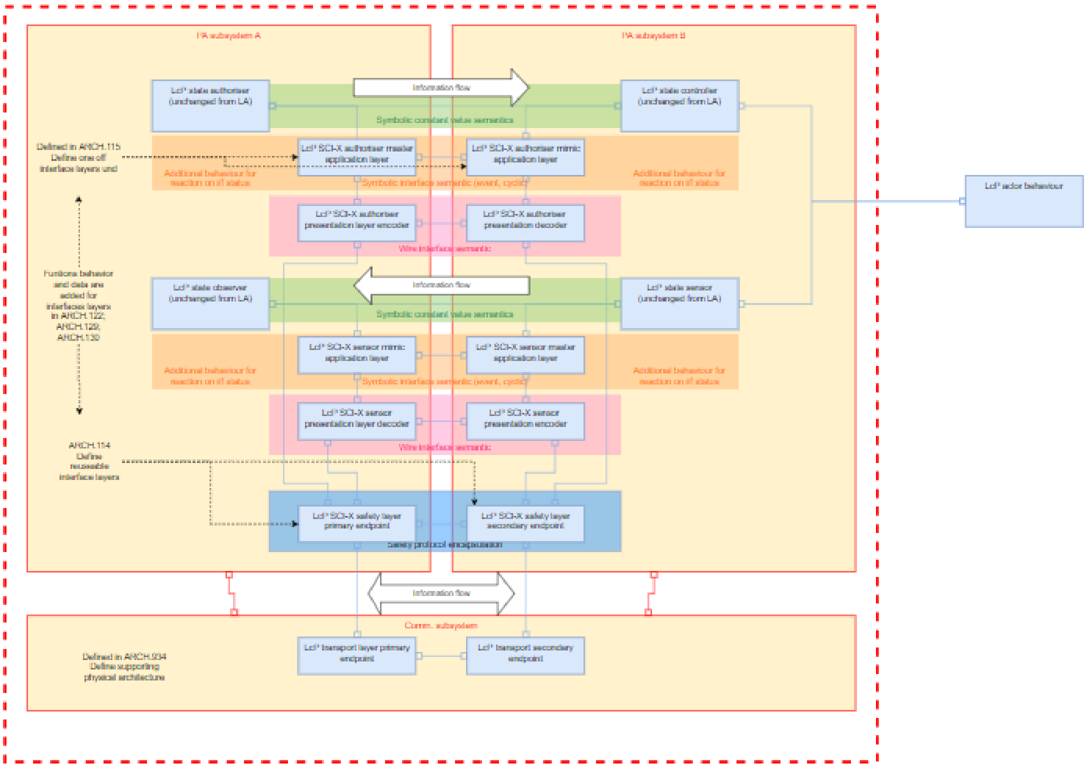
Example 2

The second example shows an architecture that was at once proposed for a virtual EVC implementation on the trackside. In this architecture, only brake commands are transmitted to the train (we are aware, that this is not wanted, but just used as example here). But the interesting fact is that both architectures are based on the same logical components



After the logical components have been allocated to the subsystems, all interfaces and their extended behavior have to be specified. This means that also behavior has to be added that takes the existence of interfaces into account, e.g. the fact that they can fail or need to be initialised.

This will create a lot of work and requires deep knowledge of the subsystems and in any case needs to be done in the domains. In the end, on this layer all requirements will come together and are the basis for generating specification documents.



This layer defines:

- which individual subsystems are in the architecture
- by which standardised interfaces are these subsystems connected (full stack, at least so many layers as are needed to define the interface on FFFIS level)
- the supporting architecture, as shown in the example. This is a key factor because it brings in an entirely new domain of computation and communication systems which heavily influence the architecture of the deployable subsystems.