RCA

Reference CCS Architecture

An initiative of the ERTMS users group and the EULYNX consortium

Concept: Architectural approach and System-of-systems Perspective

Document id: RCA.Doc.13
Version: Gamma.1
Date: 31.01.2020

Table of contents

Figures

Tables

Version history

Beta.1	26.8.2019	Bernhard Rytz	First publication after review by the RCA core group
Gamma.1	31.01.2020	Bernhard Rytz	Update (new document structure) after review by the RCA core group

1. Introduction

1.1. Purpose of the document

The publication of RCA has generated interest among railways and suppliers.
This document addresses the following questions:

- What kind of architecture does RCA try to describe (functional, logical, physical…) and why?
- What architectural design principles does RCA apply?
- How does RCA fit into existing architectures or with on-going initiatives? Does RCA lend itself to be a part of a system-of-systems approach?

This document complements the document:

- **RCA System Concept**: Essence of RCA rationale, goals, scope and system concept [RCA.Doc.15]

1.2. Relevance of architectural approach for RCA

If the goal of the chosen architectural approach in RCA is not well understood, RCA may be difficult to understand or may be lacking in content. Therefore, it is important to understand:

- the issues addressed by the RCA
- the issues that are (on purpose) omitted
- the principles on which design decisions are founded.

1.3. Relevance of system-of-systems perspective for RCA

Since RCA does not describe the complete architecture for an IM or the whole railway system, RCA will only be a “piece of the puzzle” and will need to be combined with other architectures. We will give a first overview of how this can be accomplished.

1.4. Relevant RCA documents

The RCA Documentation Plan [RCA.Doc.6] describes all published documents. The notation [Id] below refers to the document identifier in the documentation plan. The following documents are most relevant to get started with RCA.

- **RCA System Concept**: Essence of RCA rationale, goals, scope and system concept [RCA.Doc.15]
- **RCA Concept**: Informal Architecture Overview [RCA.Doc.43]
- **RCA System Architecture**: starting point for the model-based specification of RCA [RCA.Doc.35].

1.5. Terms and abbreviations

The terms and abbreviations are listed in the RCA Glossary [RCA.Doc.14].

1.6. Concepts and Examples in other domains

- SESAR (single European Sky ATM research) JU has published in [2] an overall architecture for ATM, taking a very broad perspective.
- System-of-Systems Engineering is an established discipline (see https://en.wikipedia.org/wiki/System_of_systems_engineering).
- The discipline Enterprise Architecture offers a perspective of applying architectural thinking to enterprises including also non-technical concepts, see https://en.wikipedia.org/wiki/Enterprise_architecture. For an application of EA (Enterprise Architecture) to railways, see [7].

- Model-based systems engineering (MBSE) is a methodology of systems engineering focusing on (architectural) models and capturing more than technical implementation decisions https://en.wikipedia.org/wiki/Model-based_systems_engineering.
- TOGAF is well-known architectural framework. RCA reuses some of the principles of TOGAF, see https://pubs.opengroup.org/architecture/togaf92-doc/arch/)

1.7. Reference material

- [1] Reference removed.
- [2] SESAR JU. “A proposal for the future architecture of the European airspace” https://www.sesarju.eu/sites/default/files/2019-05/AAS_FINAL_0.pdf.
- [3] ISO/IEC 42010:2011Systems and software engineering — Architecture description.
- [4] SysML is a systems modelling language (i.e., notation) derived from UML and the foundation for many system engineering methods. https://en.wikipedia.org/wiki/Systems_Modeling_Language.
- [5] Arcadia is a systems engineering method with a clear derivation process from need to solution. https://www.polarsys.org/capella/arcadia.html.
- [6] EULYNX Modelling Standard is available as part of the (public) documentation set of every EULYNX release. https://eulynx.eu/index.php/documents/documents-overview/published-documents/open-availability.
- [7] Network Rail “"Enterprise architecture within railway systems engineering" https://digital-library.theiet.org/content/journals/10.1049/iet-its.2018.5062?originator=ietauthorOffprint&identity=483079&timestamp=20200611101749&signature=397b95ccc00e2fca902d256f6a63d395&tinyUrl=http://ietdl.org/t/I9MA0.

2. Architectural Approach in RCA

RCA is a reference architecture, which leads to 2 important points:

- RCA is not a complete system architecture but provides a reference, i.e., a blueprint for building/procuring a concrete CCS for a specific IM.
- We will distinguish the architectural elements needed to define / develop RCA from the architectural elements needed when applying RCA.

This chapter will describe the architectural approach of RCA including:

- What is the understanding of the term "architecture" in RCA?
- What are the architectural definitions coming from the RCA process?
- Which architectural elements are necessary to define RCA?
- Which architectural elements are necessary to use / apply RCA?

2.1. Understanding of the term 'architecture' in RCA

The term "architecture" is frequently used, but with different perspectives. We apply the definitions from ISO/IEC 42010:2011 "Systems and software engineering — Architecture description".

The following concepts are central for the discussion (simplified from [3]):

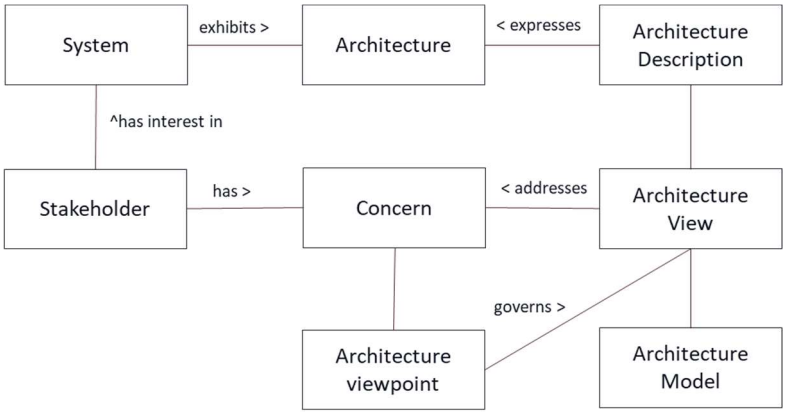


Figure 1. Elements of an architectural description

Important points here are:

- The distinction between the architecture of a system and the architecture description of a system.
- Architecture: fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution.
- Architecture description: work product used to express an architecture.
- Architecture viewpoints are used to define architectural views which address certain concerns of stakeholders with respect to the system.

There are different architectural contexts that use quite different architectural modes. These contexts include enterprise architecture, systems engineering, software engineering.

RCA has a very specific target and therefore selects a small subset of architectural viewpoints for its description. We will therefore describe below which architectural viewpoints are important in RCA.

When integrating with other architectures (see also Chapter 4 "System-of-systems perspective in RCA"), it will be important to take into the account the fact that architectural descriptions may be quite different.

2.2. Simple architecture framework

We introduce here a simple reference schema (architecture framework) to facilitate the discussion about which architectural elements are addressed by RCA. The diagram shows:

- commonly used “layers” used in the disciplines EA (enterprise architecture) and SE (systems engineering).
- topics (such as requirements, Safety-View, RAM-View) being refined & allocated from top to bottom.
- the solution design results in the rectangle “component specifications”.

Since terminology in architecture has not been standardised yet, we have reused terms from SysML [4] and Arcadia [5].

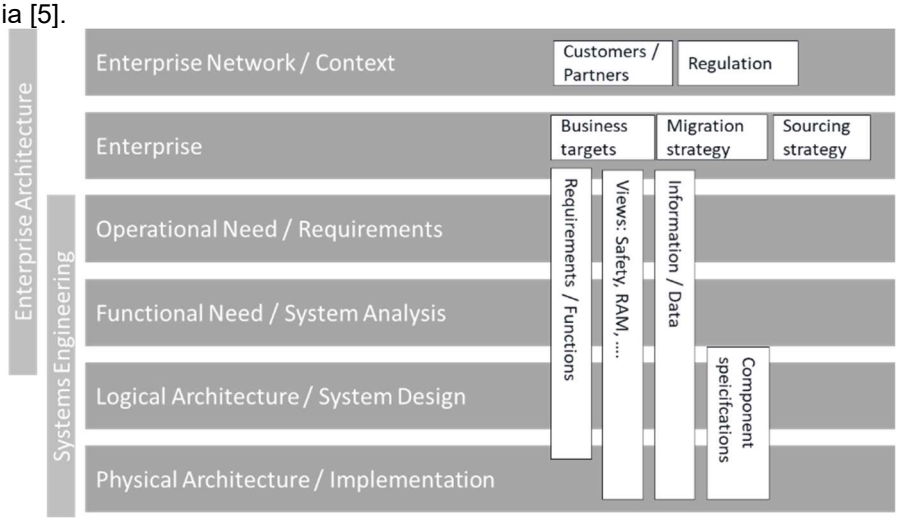


Figure 2. Reference schema for architectural elements

Note:

- physical architecture and implementation will be separate from a component supplier point of view. We keep them together here for simplicity.

2.3. Architectural modelling, methods and tooling

RCA is applying MBSE (model-based systems engineering) and a subset of SysML [4] as defined by the modelling standard of EULYNX [6]. Since the system scope and component complexity are different from EULYNX, some extensions to the modelling standard are being considered.

2.4. Deliverables from the RCA process

The main purpose / outcome of RCA are specifications for RCA components. These RCA component specifications can be used:

- by IMs as building blocks in designing / building a CCS system, and
- by suppliers as input to their product requirements.

The RCA components are not abstract functions, but buildable, procurable, runnable components (implementing, of course, the allocated functions). RCA components need not be physical boxes, but most components can also be implemented by “pure” software solutions (see also the topic “platform independence” in [RCA.Doc.11]).

The RCA components specifications consists therefore of two (largely) orthogonal aspects (see Chapter 4 in “RCA System concept” [RCA.Doc.15]):

- Interface architecture of RCA. This includes the following elements:
    - component decomposition
    - component behaviour definition

    - component interface definition, including information modelling
- Technical architecture of RCA. This includes:
    - the platform or runtime environment for the execution of a component
    - the communication stack providing the communication channel between components

The parts marked “Full spec” in the following diagram are the core of the architectural deliverable of RCA. Additionally, RCA will contain “Helper material”. Helper material is not necessarily sufficient to derive a system but contains design rationale and information that is reusable for a system derivation by an IM.

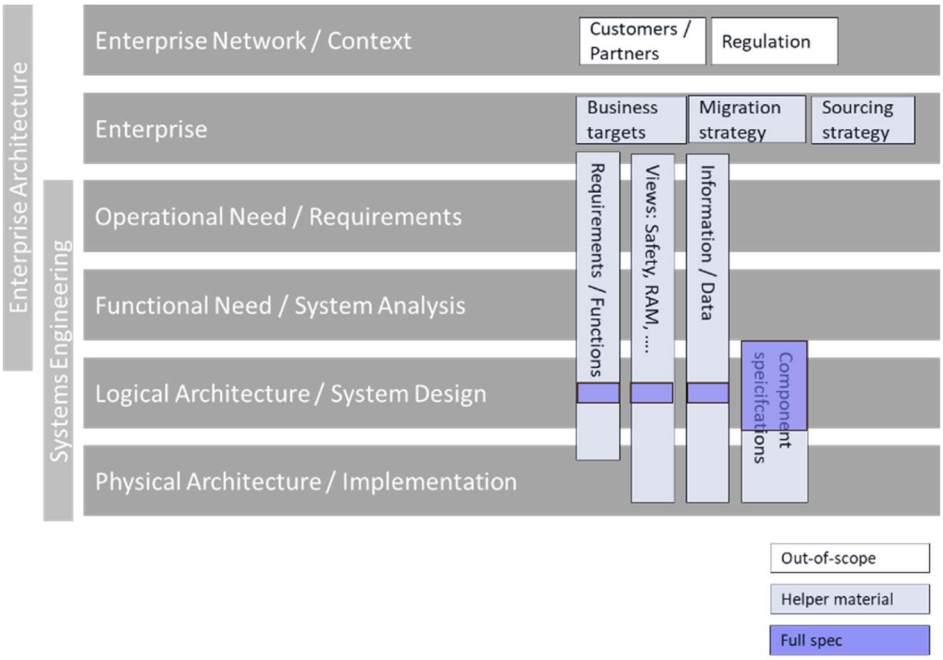


Figure 3. Architectural elements provided by RCA

2.5. Which architectural elements are necessary to define RCA?

In the design / specification process for RCA we use existing requirements and design inputs from several IMs.

Regarding the “RCA components specs”, the RCA process works in 2 directions (see Fig. 4):

- “top–down” (from needs to solution):
  The RCA component specification is checked against the needs / design constraints. A new / different need may lead to an extension of the RCA component spec.
- “bottom–up” (from solution to needs):
  If an RCA component specification would be required to cover the sum of all existing needs / design constraints, the specification would become too complex, the opportunity for harmonisation would be missed, and the cost of products may be too high. Therefore, before considering an extension of the RCA component spec, an attempt at harmonising the requirements among IMs is necessary.

So, for a given existing requirement on which 2 or more railways diverge, 3 outcomes are possible:

1. The requirement is harmonised among railways (preferred)
2. The requirement can be handled by functionality / processes outside RCA components, the RCA components implement a “superset” of functionality to support the requirements.

3. The RCA component specification implements the sum of the requirements (expressed as variants).

The process needed to manage the 2 directions “top–down” and “bottom–up” will involve constant trade-offs and will require a clear governance. See also “Variability management” in RCA System concept” [RCA.Doc.15].

Regarding the helper material, the RCA process will document needs and design constraints that provide the rationale for the RCA component specification and which may be material for reuse in IM rollout programs.

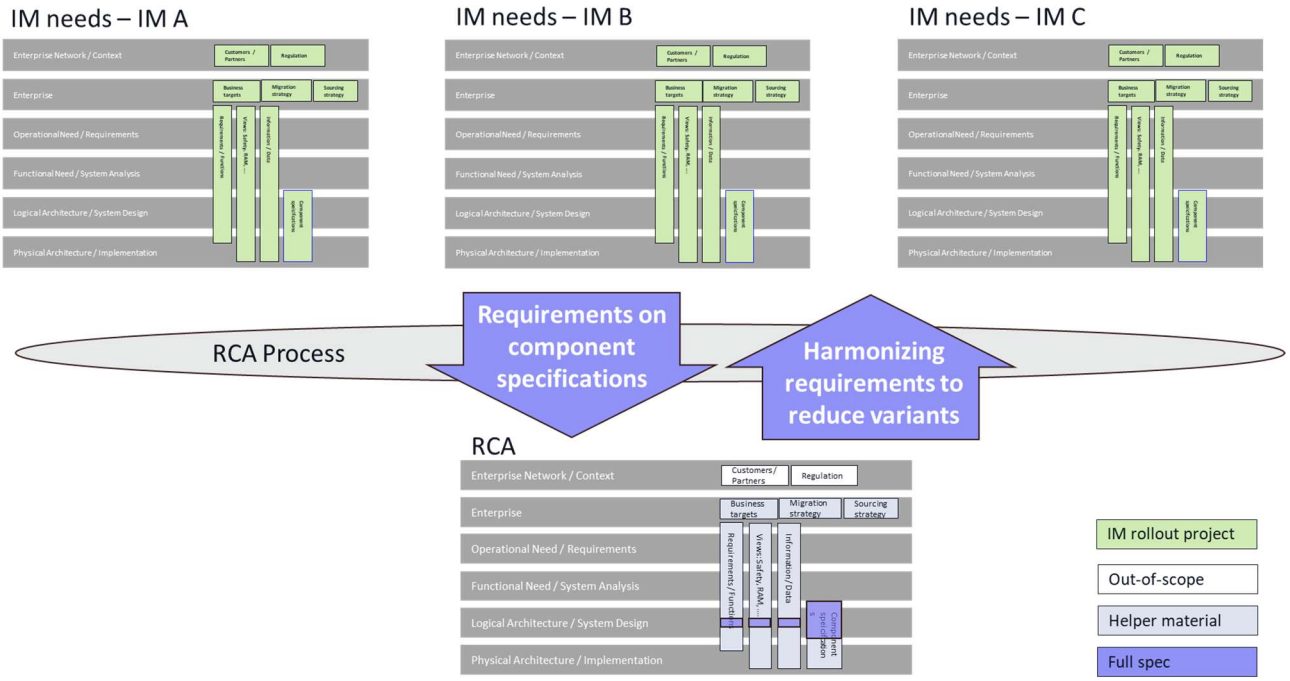


Figure 4. Architectural elements and how the RCA specification is defined

2.6. Which architectural elements are necessary to use / apply RCA?

Figure 5 shows how an IM rollout program will “import” RCA deliverables into their overall architecture and program. The RCA deliverables come in 2 categories:

- RCA components specifications:
  RCA specifications are to be used as tender templates to procure components. The RCA specs will also be used to plan integration, acceptance, linking to existing systems, etc.
- RCA “helpers”:
  These are requirements and design material that can be reused in the IM rollout program but will have to be extended and merged with the specific requirements material of each IM.

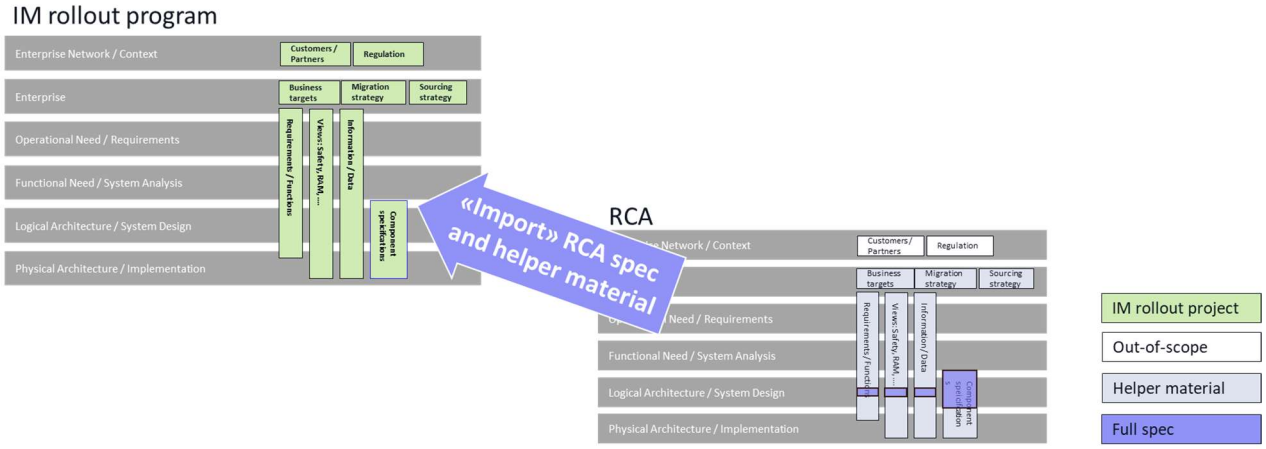


Figure 5. Architectural elements and how the RCA specification is applied

2.7. Where does the “harmonised” RCA component architecture come from?

The harmonised RCA is the result of applying functional and non-functional requirements of the participating IMs to the RCA interface architecture and checking for completeness / fit. RCA therefore contributes to the harmonisation of the ongoing programs of different IMs.

- This "RCA interface overview" does not show a pure functional structure since the defined decomposition principles (see chapter 3.2) are already applied. They combine aspects of the functional, system, lifecycle, and technical architectures, to support the scoping of real products (HW, MW, SW) that make sense as standard products for the railways. The decomposition rules follow different architectural quality attributes, e.g., exchangeability and independence, difference of lifespan or lifecycle type, isolation of safety cases (“modular safety”), no combination of functionality that is not always used together, etc.
- In general, the split that leads to the "RCA components" describes interfaces at positions in the CCS automation pyramid[1] where different products are able to work together on a standardised basis. However, these interfaces can also be used within a product or product cluster. There could be a requirement in a tender to have the flexibility to react to unexpected lifecycle events (e.g., discontinued products, strong price hikes) by changing the supplier or product type only for RCA components at a later stage in the lifecycle. How many RCA interfaces are used is decided by the individual railways during the design of the lifecycle strategy for their CCS architecture, their migration programs and their tenders.

2.8. How the interface architecture is linked to procurement options

By way of example, Figure 6 shows that IMs can choose different procurement strategies for an RCA-based system, these procurement strategies lead to different integration needs.

[1] https://en.wikipedia.org/wiki/IEC_62264

Granularity of procuring RCA components:

- Upper left: The RCA specification defines interfaces for the RCA components. These interfaces include “functional” interactions with other components and interactions with technical building blocks, including runtime environments and communication stacks;
- Upper right: an IM can choose to procure in a fine-granular way, isolating all RCA components in its procurements;
- Lower left and right: an IM can choose to bundle RCA components in larger procurement units. Whether the internal interfaces are part of the procurement requirements is up to the IM.

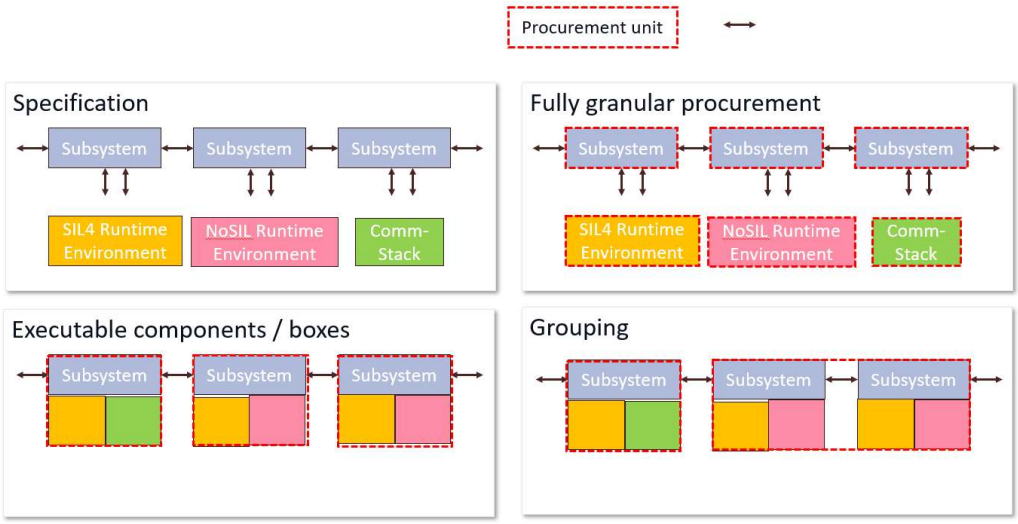


Figure 6. Different options for “procurement granularity”

Implications of fine- vs coarse-granular procurement:

- With fine-granular procurement the task of integration becomes more important. Integration can be performed by an IM or can be tasked out to a third party;
- When procuring bundled components, the IMs should make the “internal” interfaces between RCA components a part of their requirements to a) ensure future exchangeability and b) help drive harmonised specifications;
- An IM may procure in an even more fine-granular way, by splitting up an RCA component. In this case, the IM must define their own specifications for the new interfaces. These interfaces could be candidates for inclusion in RCA.

2.9. Relation of RCA to “harmonised operational processes”

Full harmonisation of operational processes is not in scope of the RCA. When changing to a new architecture for the future railway system in large migration programs, some harmonisation of operational processes becomes possible (compare to standardised DMI, etc.).

RCA has excluded full harmonisation of operational processes from its scope because the processes have too many dependencies, such as national laws; national affordability of the same safety, performance or availability targets; processes are “programmed” into hundreds of products and thousands of installed systems; processes are part of thousands of safety cases, etc.

Migrating to the RCA will support the harmonisation of operational processes. However, the harmonisation methodology even in this case needs certain steps and depends on the initial situation of the IM and the structure and duration of the migration.

3. Architectural design principles for RCA

3.1. General architectural principles for "RCA-based" systems

The following table gives an overview of the currently identified principles. Some of these principles need more work to be applicable by the RCA group or the Working Groups.

Principle	Rationale	Consequences
Target state requirements and migration requirements have equal priority in RCA = RCA design for a "pure" future state and includes necessary mechanisms for migration	ERTMS/ETCS deployment lessons learned - importance of considering operations and degraded mode, being stuck with a supplier-specific solution due to lack of common interfaces - importance of agreeing on a target state and functional needs and then considering migration. Starting from the current state and working forward leads to incremental development, which may not provide an optimum solution and is likely to lead to solutions with a national flavour.	- Architecture supports several migration (and sourcing) strategies. - IMs must define their requirements for a migration towards RCA into the RCA process.
In RCA every function is only designed once (no large set of functional alternatives in the target state)	In a target state with "pure" ETCS "L3+" there is no objective reason for functional diversity. Diversity has led to poor performance / cost and poor innovation.	- Harmonisation of requirements has to take place. - If needed, variability specified through configuration / parametrisation. - A certain degree of variability can be achieved by using different configurations of RCA components. - Note: this does not require total harmonisation of operational processes.
Minimum trackside assets possible	Trackside assets exist in higher numbers, in harsher conditions and are more difficult to access. Reducing them reduces LC-cost / function and increases reliability (environment, MTTR).	Safe and reliable alternatives have to be developed (e.g. localisation, train integrity moved from trackside to vehicle).
Modularity: Exchangeability of components	Key to ensuring keeping LC-cost down and ensuring evolvability.	Requires careful interface specifications. Requires interfacing techniques for up- and downward compatibility, e.g., capability-based protocols.

Principle	Rationale	Consequences
Scalable to different needs (e.g. achieving safety levels by choosing the right (number of) devices)	Ensure broad applicability of RCA-based systems.	Need to consider several target configurations.
Functions are allocated to Software. Software is separated from Hardware	- In many cases, SW has better quality- / price-ratio. - SW is easier to evolve.	- The SW / HW interface has to be specified[2]. - Procedures for updating SW securely are needed.
Communication (carrier, lower-level protocols) is exchangeable	Key to ensuring keeping LC-cost down and ensuring evolvability.	- Functional and communication aspects are separated in SW. - Communication is properly layered.
Slim SIL 4 components: No combined implementation of business processes and safety functions	Development on higher SIL levels is disproportionately more expensive. Safety related functions kept to a minimum.	Architecture needs to provide the corresponding interfaces. These interfaces may be critical from an NFR-point of view.
High ability to automate functions (“transactional completeness”)	The architecture must allow (not necessarily enforce) full automation to support corresponding goals by implementers.	The functional architecture must not rely on “miracle” functions but describe functional blocks, that can (at least in principle) be automated[3].
Interfaces are upward and downward compatible	Evolvability (see also modularity)	Profile- / capability-based interfaces
Modular safety: Ability to isolate safety cases and homologation	Evolvability	- Design of components (interfaces) must allow isolation of safety cases. - Early evaluation of “modular safety” concepts with assessors necessary
“Core” systems and their interfaces are “process-agnostic”	Ensure broad applicability of RCA-based systems.	E.g., Business processes only implemented in the TMS, other layers are process independent and focus on the basic “physics” of rail control and command.
“Cutting edge” technology is properly isolated, i.e., integrable, but not mandatory	The speed of technological maturation is difficult to predict. The architecture must be open for foreseeable evolutions while allowing implementations “here and now”.	E.g., Flexibility concerning the connection and mixture of localisation devices

Table 1. Architectural Principles for RCA

[2] In some cases, an integrated stack may be procured, at the cost of having lock-in between SW and HW.

[3] RCA focuses on interfaces. To provide context for the interfaces, components/blocks/subsystems and behaviour will have to be described to some extent.

3.2. Rules for splitting functions into components

These rules are rough guidelines that help to provoke thought about the functional break-down.

Reason to split	Explanation
Different NFRs (non-functional requirements like safety, RAM or security)	NFRs drive the needs for the SW development process and for the deployment configuration. Packing functions with very different requirements together may lead to waste / over-design.
The right split creates simple interfaces and self-sufficient systems	The classic modularity principle: high cohesion / low coupling.
Independent lifecycle • Independent installation timing or installation process • Independent change • Independent lifespan	Separated functions may help upgradability (e.g., need for retesting).
Usage only in a part of all installations	Possibility not to deploy a function at all.
Necessary hardware topology (e.g., regional)	Available devices (computing power, storage capacity) and / or communication access to the devices and / or realms of responsibility (e.g., in a vehicle) can make this necessary.
Different markets or supplier types	“Unusual” bundles may lead to difficult / inefficient procurements.
Smaller components make it easier for more vendors	Lower market entrance hurdles BUT additional integration work.
Split of homologations and safety cases	Reduce procedural effort.
Split the work / Division of labour	Ability to distribute / parallelise work.
Reuse potential	Possibility to reuse important existing work.

Table 2. Rules for splitting components in RCA

3.3. Layering principle for the architecture

The RCA is divided into architectural “layers”. Layers play an important role in structuring architectures and have successfully been used in computer architecture, communication architecture, etc.
In RCA, every layer has special types of blocks, and may have special design rules for interfaces (generic for all blocks in the layer) and especially special “abstraction levels”. Example: On the device-control layer a function will know about the type of hardware (point, level crossing, TDS) it controls. On the safety-control layer “objects” are only known by their abstract hardware-independent capabilities (e.g., “trafficability on a node-edge-model”).
RCA components are assigned to exactly one layer.

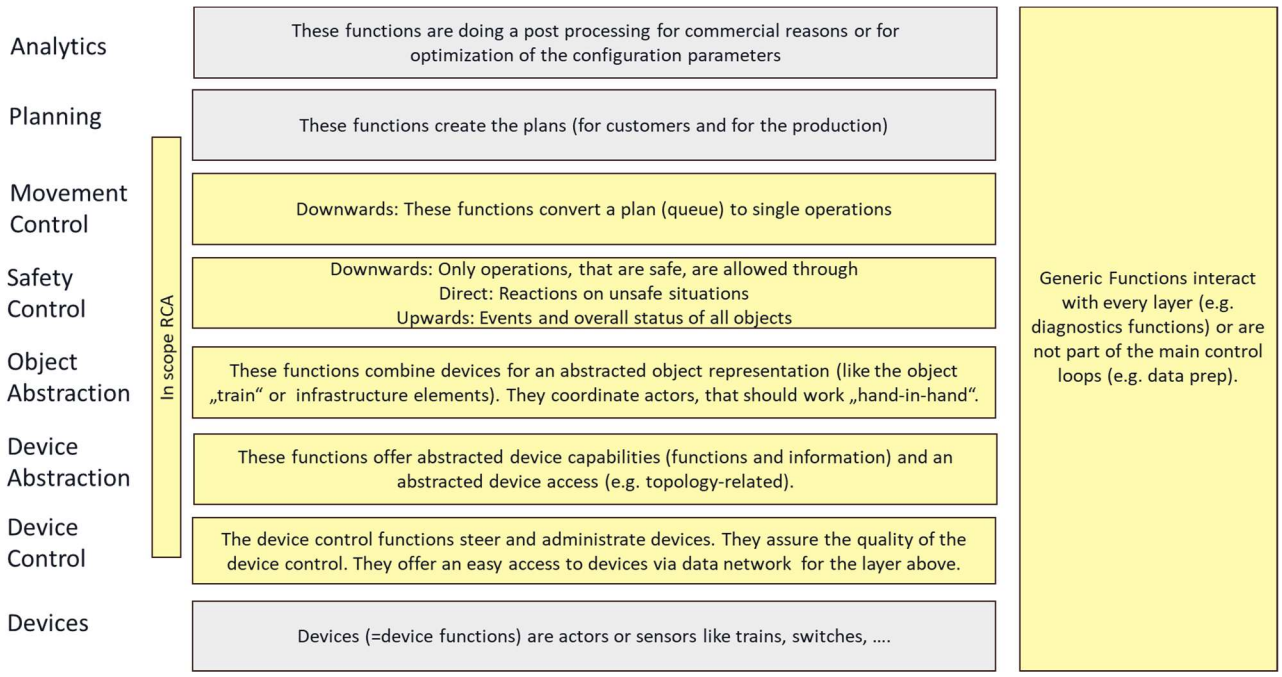


Figure 7. Overview architectural layers

The following table describes the layers in more detail:

Layer name	Interface upwards	Interface downwards	Type of functions on the layer	Layer characteristics
Planning		Operation Plan	These functions create the plans (for customers and for the production)	
Movement Control	**“Execution State” =** Progress of the plan execution **“Operational Status”** = States and actual capabilities of all **objects** = **trackside assets** (“**TA**” like tracks, switches, crossings….), all **moveable objects** (“**MOB**” like trains, Persons, obstacles).	**Object control requests (OCR)** (update movement permission, control object, change object status, etc.)	These functions implement the operation plan by issuing single object-control requests when the condition regarding the current operational status are met These OCRs can, for example, change a switch position or update a movement permission.	“Process independent”. No implicit assumptions for process rules on this layer. Every process- or company-specific rule should be implemented on a higher architecture layer (planning). The operation plan (and delivered static descriptions for process rules), that is converted to device requests shall fully describe the process, that must be executed - also in degraded modes. Layers from here downwards are “real-time” layers. The functional design has to take the system implementation into account (high workload, some hundred thousand events per second for a small network, so efficient simple functions have to be designed, decision processes shall be fast). This layer is very important for the migration. The functional interfaces to legacy systems must be designed and shall be taken into account in the functions for the target architecture. All interfaces down from this level are asynchronous (cut up functional sequences, queueing, locking, necessary for scalability). All functions of this layer shall be designed for a maximum autarchy duration with stepwise degrading modes when the planning system is offline. This and higher layers shall support the most simple functionality in the lower safety

Layer name	Interface upwards	Interface downwards	Type of functions on the layer	Layer characteristics
				layers by relieving lower layers from functionality that is not safety critical. Lower layers shall only have “check and driver” functionality.
Safety Control	**Operational Status**	**Object** control commands for the control of MOB/TA	These functions check requests from upper layers or users: Do they lead to a safe state of the production? If yes, then they are executed. They also check events and overall status of all objects and invoke emergency reactions for unsafe situations.	On this layer no hardware-, asset-, asset-layout-, track-layout, train-type-, track-usage-condition- or train-capability-specific requirements shall exist. All these attributes are described as abstract object attributes that shall fit together for a safety check function. Functions shall be implemented that allow a generic safety case for a specific parameter- and rule-set that is set for a whole network. The parameter- and rule-sets shall allow the definition of different safety targets and principles without ever changing the implemented system.
Object Abstraction	Addressable abstracted **objects** with functions and attributes	Actor coordination: Coordinated **device** control commands for the control of MOB/TA	These functions combine devices for an abstracted object representation. They co-ordinate devices (actors) for the execution of object-control commands, which should work “hand-in-hand”.	Devices are handled on this layer in a generic way. They are described only by their generic attributes and capabilities (functions, methods), not as hardware models or hardware types. Aggregating devices to objects (e.g., to trains or train-components) is done by a rule-interpreter for configurable rules, where each has its own verification (extendibility). An object is defined by one or more attributes (like ID, start-position, end-position, length, type, etc.) and devices deliver one or more of these attributes together with an object ID, to whom they belong. The execution of object-control commands can influence more than one device inside of the moveable object and in parallel on the track.
Device Abstraction	Abstract device capabilities. Abstract device addressing.	System-specific protocols	These functions offer abstracted device capabilities (functions and information) and an abstracted device access (e.g., topology-related).	The downward interfaces (and on the layer below) shall implement the “modular safety” strategy which reduces the effort to integrate new pre-certified devices to a minimum. This shall be achieved by a small and stateless protocol (small capability protocol modules) and fully testable behaviour on both sides of the interfaces. This layer shall allow different device models to be treated as the same generic device-type.
Device Control	(safe) Device status	(safe) Device management and control	The device-control functions steer and administrate devices. They assure the quality of the device control. They offer easy access to devices via data network for the layer above.	

Layer name	Interface upwards	Interface downwards	Type of functions on the layer	Layer characteristics
Generic functions	Information for Management systems	Remote management control or prepared static data	Generic functions interact with every layer (e.g., diagnostics) or are not part of the main control loops (e.g., data prep).	This layer contains cross-cutting-concern functions and aspect-concern-functions. The design of these functions shall be done as a flexible service layer.
Devices	System-specific interfaces		Devices (=device functions) are actors or sensors like in trains, switches, ….	

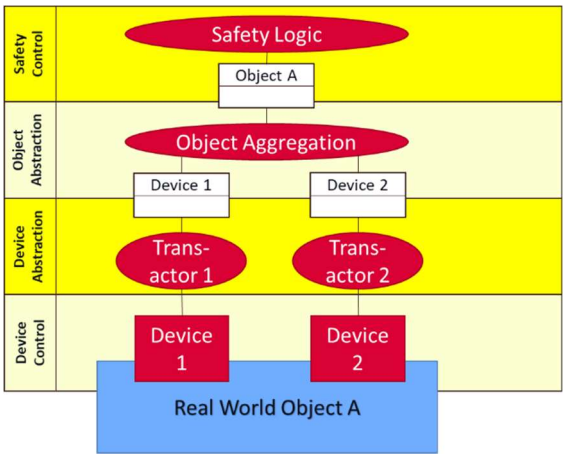
Table 3. RCA layers and their characteristics

3.4. Concepts of Objects and Devices

To understand some of the architectural decisions in RCA, it is helpful to discuss the role of "Objects" and "Devices".

The diagram shows that:

- Objects are monitoring and managing 1..* devices;
- "Object Aggregation" combines multiple device information to a single object representation of the "real world object".



Examples include:

- Train with ETCS-OBU at the front and a localization tag at the rear and / or using trackside TDS;
- Movement permission to the ETCS-OBU and / or to a trackside signal (e.g. border signal);
- Level crossing with separate device for intersection scanning or warning lights;
- New devices in the future.

Object Aggregation is an important function, because:

- Object Aggregation provides an **abstract view** of controlled objects to the upper layer
- How objects are connected with devices is hidden from the upper layers Advantages:
  - Changes for new device combinations only affect the OA and not the upper layers → Similar to Hardware-Abstraction Layer in Operating System;
  - Object Aggregation can support many different combinations of devices;
  - Migration from old to new can be encapsulated in Object Aggregation.
- The problem of aggregation is independent of the problem of the safety logic. Advantages:
  - Good "separation of concerns" (→ Reducing complexity of each problem);
  - Independent lifecycle of independent changing logic;
- This follows the proven pattern of an "layered architecture" and an "automation pyramid", where **each layer solves a specific problem**.

3.5. **Allocation of functions to layers**

The following diagram shows the allocation of (simplified) functions to the layers of the RCA. For each function there will be one or more RCA components (based on the defined decomposition principles).

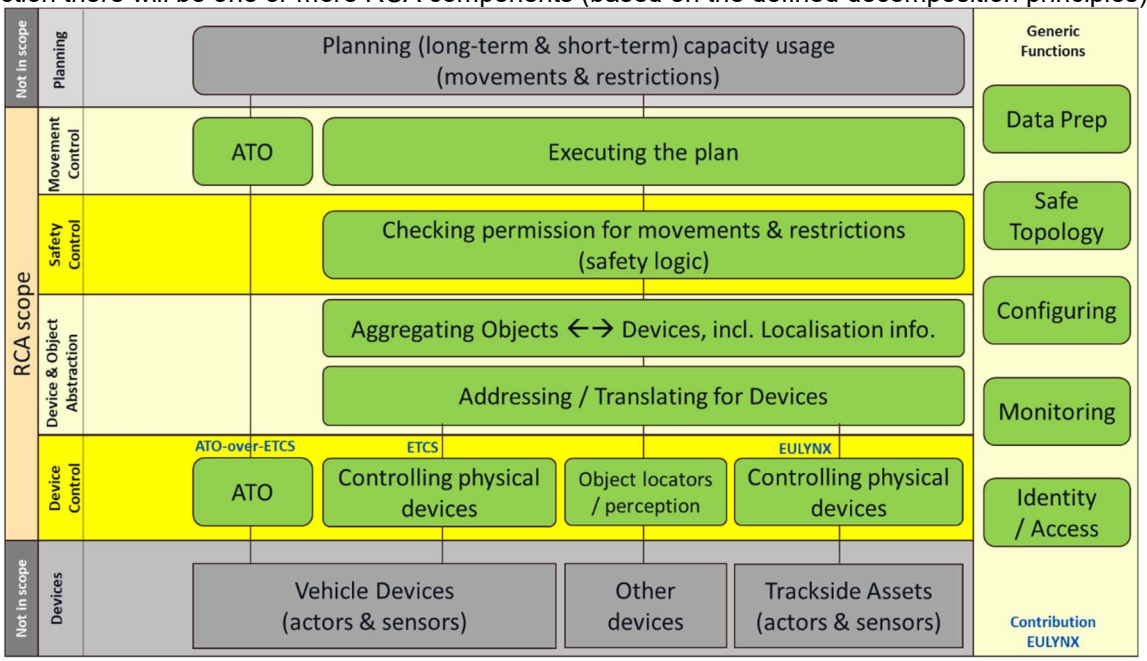


Figure 8. Overview allocation of functions to layers

4. System-of-systems perspective in RCA

This chapter will describe the system-of-systems perspective of RCA including:

- How do we use the system-of-systems perspective in this document?
- How can an RCA-based system be integrated in a systems-of-systems perspective of the railways?

4.1. System-of-systems perspective in this document

The concept “system-of-systems” is very broadly defined (see https://en.wikipedia.org/wiki/System_of_systems). In the context of RCA, the components of RCA can be seen as systems and, thus, RCA is a system of systems. In this document, however, we take a higher-level perspective, looking at the “railway system” as a system-of-systems, of which the RCA may be a part (i.e., a system).

Relation to “Enterprise Architecture”: EA is a form of system-of-systems approach, often focusing on one enterprise (including its interfaces). Enterprise Architecture deals not only with technical systems but also with “soft systems” such as goals, organisations, processes, capabilities.

As an example from another domain, in [2] the SESAR JU proposes an overall architecture for the European airspace, including governance, regulatory, organization and technical systems issues. Figure 9 shows the proposed services in this context.

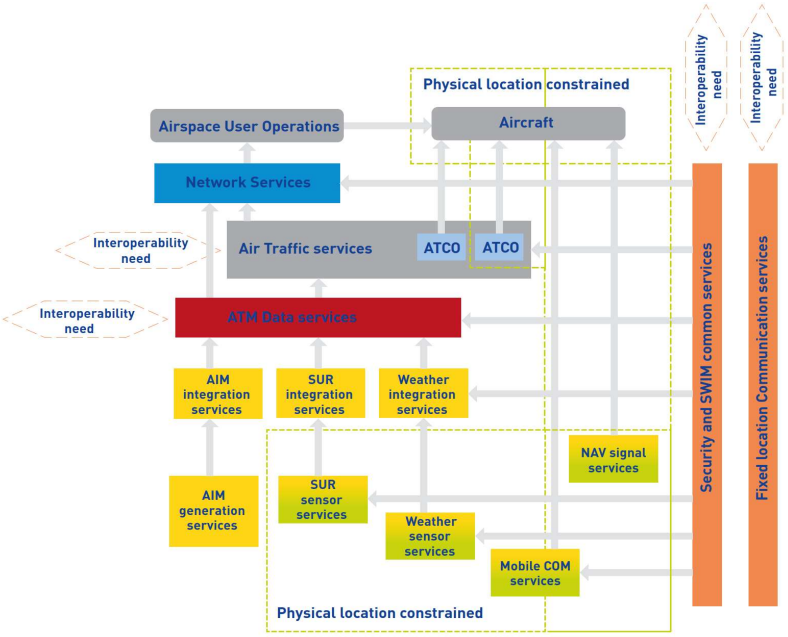


Figure 9. Example: Architecture of SESAR

In the telecoms sector, there are models (such as eTom[4]) which provide a shared reference architecture (without being binding).

In the railway sector, there seems to be no widely adapted, shared, overarching reference architecture (yet). It is not the purpose of RCA to provide such a model. From the RCA perspective it would be helpful to have such models and RCA will help map / integrate RCA into such an overarching architecture. There is a proposal in Shift2Rail IPX to start working on such an architecture.

[4] https://en.wikipedia.org/wiki/Business_Process_Framework_(eTOM)

4.2. Combining RCA with other systems into a system-of-systems of railways

Figure 10 shows that RCA is already preparing integration into a system-of-systems:

- EULYNX, RCA and OCORA each have a clear specification scope that is complementary among the 3 initiatives (no overlaps). To efficiently implement the overall CCS functionality, the functionality has to be apportioned and some concepts and interfaces have to be jointly prepared.
- The ERA TSI CSS is shown behind RCA and OCORA, since RCA and OCORA act on the foundation / background of the existing TSIs to ensure interoperability. When needed, CR (change requests) will be submitted to the defined process.

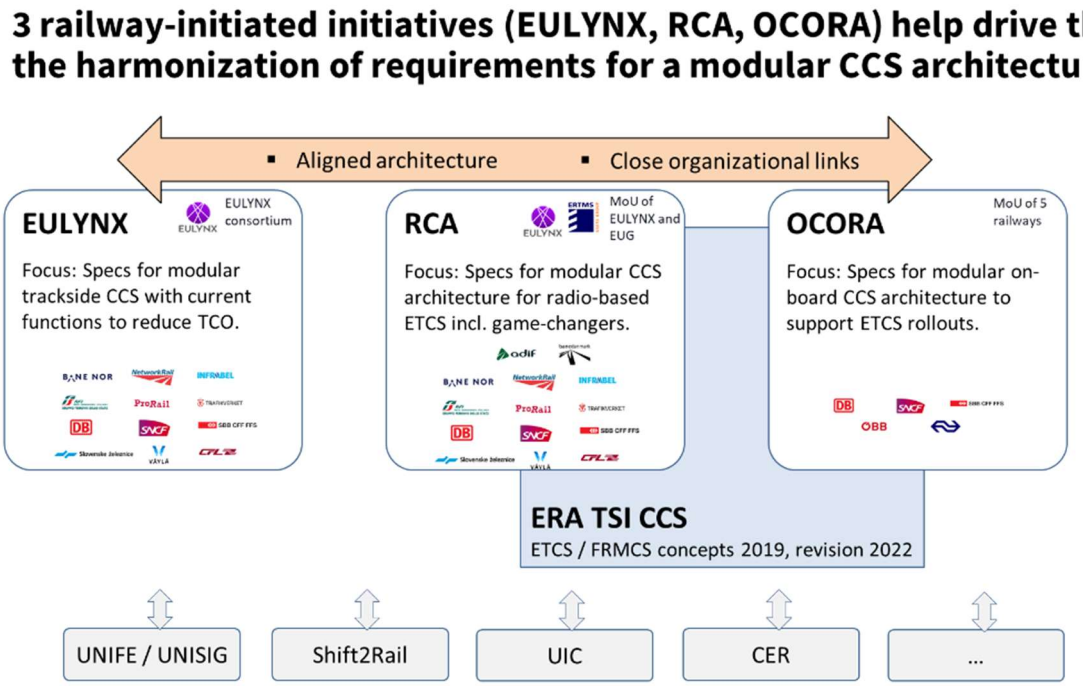


Figure 10. Relation of EULYNX, RCA and OCORA

4.3. Important criteria for integration into a system-of-systems

For architectures / systems to be easily integrable, the following characteristics are important:

- Very clear and explicit scope:
  - what elements (needs, functions, component) are in scope and will be defined by the architecture?
  - dealing with the system border: ownership of interfaces (definitions, exported conditions, assumptions);
- Explicit (and mostly shared) architectural principles: what are the architectural goals (modularity, reuse, evolvability, …) and what criteria drive the architecture process?
- Clean and precise modelling: applying a well-defined modelling standard, using tools that support an open process;
- Open process: open availability of all architectural artefacts, easy access to the artefacts (tools), transparent change-management process.

RCA adheres to these criteria in its development process.

5. Summary and outlook

The focus of RCA is to provide component specifications based on harmonised requirements for use as tender templates by IMs planning ETCS rollouts.

RCA is therefore only an (important) piece of the overall system / architecture puzzle: in the CCS domain it covers most logical component aspects on the trackside. Other architectural aspects of the CCS domain and architectural aspects of other railway domains must be integrated in a system-of-systems perspective.

The current approach of RCA with a clear scope will facilitate the integration of RCA.

Next steps:

- The European Commission and ERA are working on how to address the overall CCS architecture and its evolution.
- RCA, OCORA and EULYNX will continue to align their respective architectural scope.
- The S2R project Linx4Rail, starting 2020, will provide an opportunity to link RCA, OCORA and EULYNX work with S2R activities.

6. Architectural FAQ

6.1. What kind of architecture does RCA provide?

In a nutshell:

- Component spec:
  - RCA interface architecture: consists of a logical architecture of components with defined interfaces, including component behaviour, component states, message behaviour (sequences), message payload, data model for the message payload.
  - RCA technical architecture: consists of a logical/physical architecture describing the interaction of applications (the components of the interface architecture) with a platform (a component including physical aspects) over a well-defined API (see also “Platform Independence”).
- Helper material: includes material providing the rationale for the component specification and focusing on the architectural aspects on the level of “functional need / system analysis”. This includes the definition of end-to-end functions and their allocation to components.

6.2. Why does RCA start at the “bottom” (components)?

As shown in Sections 2.5 ”Which architectural elements are necessary to define RCA?” and in 2.7 ”Where does the “harmonised” RCA component architecture come from?”, the process does not start at the bottom, but goes back and forth between needs (top) and solution (bottom). Many years of IT experience has shown that “top-down / waterfall” approaches (e.g., do a full requirement & functional analysis before thinking about the solution design) is wasteful and often leads to over-generic, unworkable architectures.