RCA

Reference CCS Architecture

An initiative of the ERTMS users group and the EULYNX consortium

Concept: Informal Architecture Overview

Document id: RCA.Doc.43
Version: Gamma.1
Date: 31.01.2020

Table of contents

List of figures

Version history

Gamma.1	31.01.2020	Bernhard Rytz	Ready for publication after review by the RCA core group

1. Introduction

1.1. Purpose of the document

This document gives an informal overview of the RCA architecture until the MBSE-generated specification can be released. The material in this document was previously contained in the document “RCA architecture overview”.

This document complements the documents:

- **RCA System Concept**: Essence of RCA rationale, goals, scope and system concept [RCA.Doc.15]
- **RCA Architectural Approach** [RCA.Doc.13]
- **RCA Concept Platform Independence** [RCA.Doc.11]

1.2. Terms and abbreviations

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

2. Informal Overview RCA architecture

This chapter provides an informal overview of the RCA architecture and complements the MBSE specifications providing the formal architectural specification.

2.1. Interface architecture

2.1.1. Overview diagram

This is the core of the RCA and describes the main components and especially their interfaces. These components and interfaces define the primary working structure of the RCA. Every interface that is marked with a red number shall be defined by an existing or new specification. The overall specifications shall fulfil the requirements of the end2end processes. The specified components are candidates for procurement units and, thus, “product candidates”. Different IMs may choose to use different granularities for procuring these components (see discussion in [RCA.Doc.13])

This architecture is derived by applying the decomposition principles and the layering principles to high-level functional architecture.

Figure 1 shows:

- in the background, the RCA layer model;
- RCA components (orange boxes = non-safe components, blue boxes = safe components, grey boxes = out-of-scope components);
- RCA interfaces (red boxes, orange boxes, red, green, blue, black arrows);
- in the centre of the safety-relevant components, we have the APS (“advanced protection system”) with its 3 components SL, SM, and OA. The APS can be thought of as replacing (the core of) the current interlockings;
- ERTMS/ETCS and EULYNX are sets of specifications that are reused for RCA. “Contribution EULYNX” means that some work of EULYNX will be reused for the generic functions;
- Note: interfaces with users are only shown roughly and need to be detailed at a later stage.

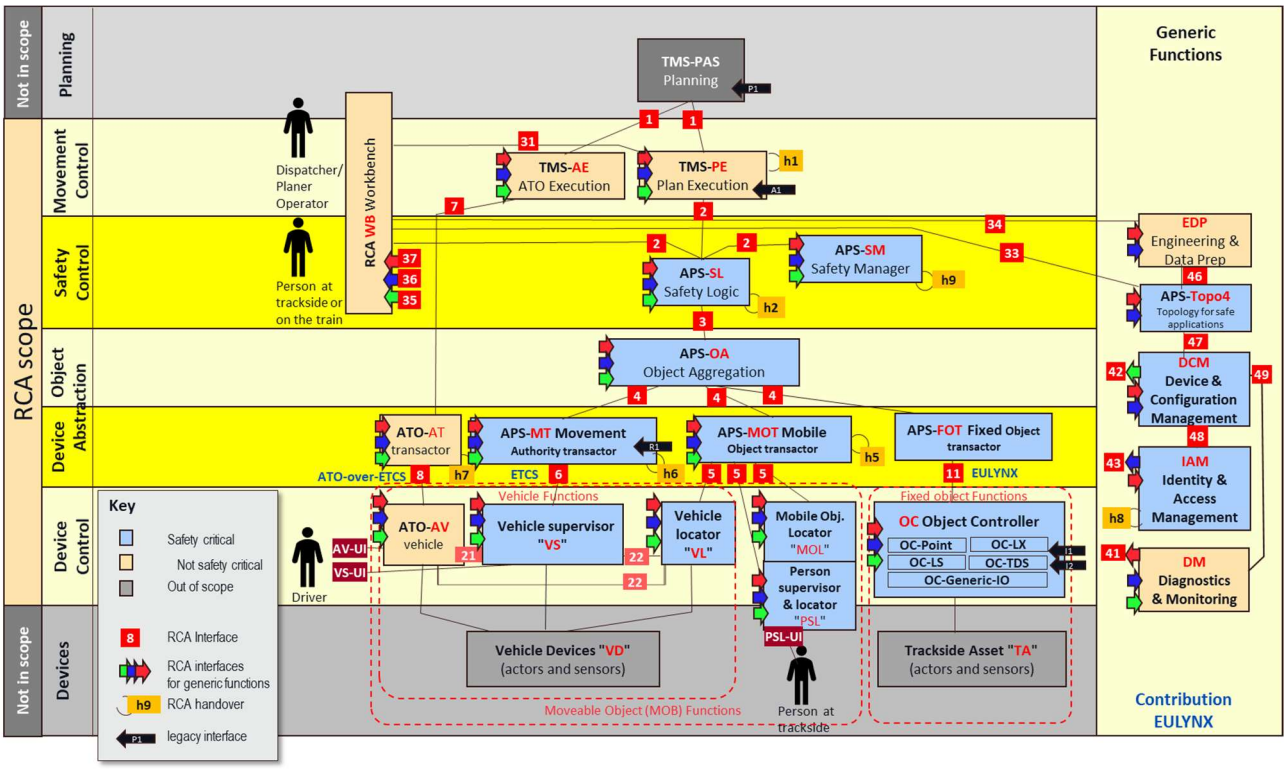

Figure 1. Interface architecture of RCA

2.1.2. Types / groups of interfaces

The following table shows how the interfaces have been grouped according to their characteristics

IDs	Type of interfaces
1-19	Control and Safety for multiple objects
h*	Handover interface (for multiple instances)
*-UI	Mandatory user interfaces
21-29	Command, Control, and Safety for one vehicle
31-39	API and interface “RCA Function”<>“Workbench”
41-49	Interfaces for cross cutting concerns and aspect concerns
P, A, I, R	Interfaces to legacy CCS, e.g., during migration

2.1.3. List of components

The following list provides a short overview. A description of the components can be found in the RCA System architecture [RCA.Doc.35].

ID	Name
TMS-PE	TMS (Traffic Management System)[1] Plan Execution and Control
TMS-AE	TMS[1] ATO Execution
APS-SL	APS (Advanced Protection System) Safety Logic
APS-SM	APS Safety Manager
WB	RCA Workbench
APS-OA	APS Object Aggregation
ATO-AT	ATO Transactor
APS-MT	APS Movement Authority Transactor
APS-MOT	APS Mobile Object Transactor
APS-FOT	APS Fixed Object Transactor
ATO-AV	ATO Vehicle
VS	Vehicle Supervisor
VL	Vehicle Locator
PSL	Person Supervisor & Locator
MOL	Mobile Object Locator
OC	Object Controller
DM	Diagnostics & Monitoring
EDP	Engineering / Data Preparation
APS-Topo4	APS Safe Topology System
DCM	Device & Configuration Management
IAM	Identity & Access Management
TMS-PAS	*TMS Planning*
VD	*Vehicle Devices*
TA	*Trackside Asset*

2.1.4. List of component-to-component interfaces

A description of the interfaces can be found in the RCA System architecture [RCA.Doc.35].

[1] The TMS prefix indicates that these components are attached to the TMS. Depending on the concrete configuration, they may be or not part of the TMS.

2.1.5. Where is the interlocking?

In RCA the functionality of today's interlocking and RBC has been allocated to several well-defined components with clear responsibilities. Figure 2 shows the architecture when the components with the "APS"-prefix are shown as an overall "APS" (Advanced Protection System).

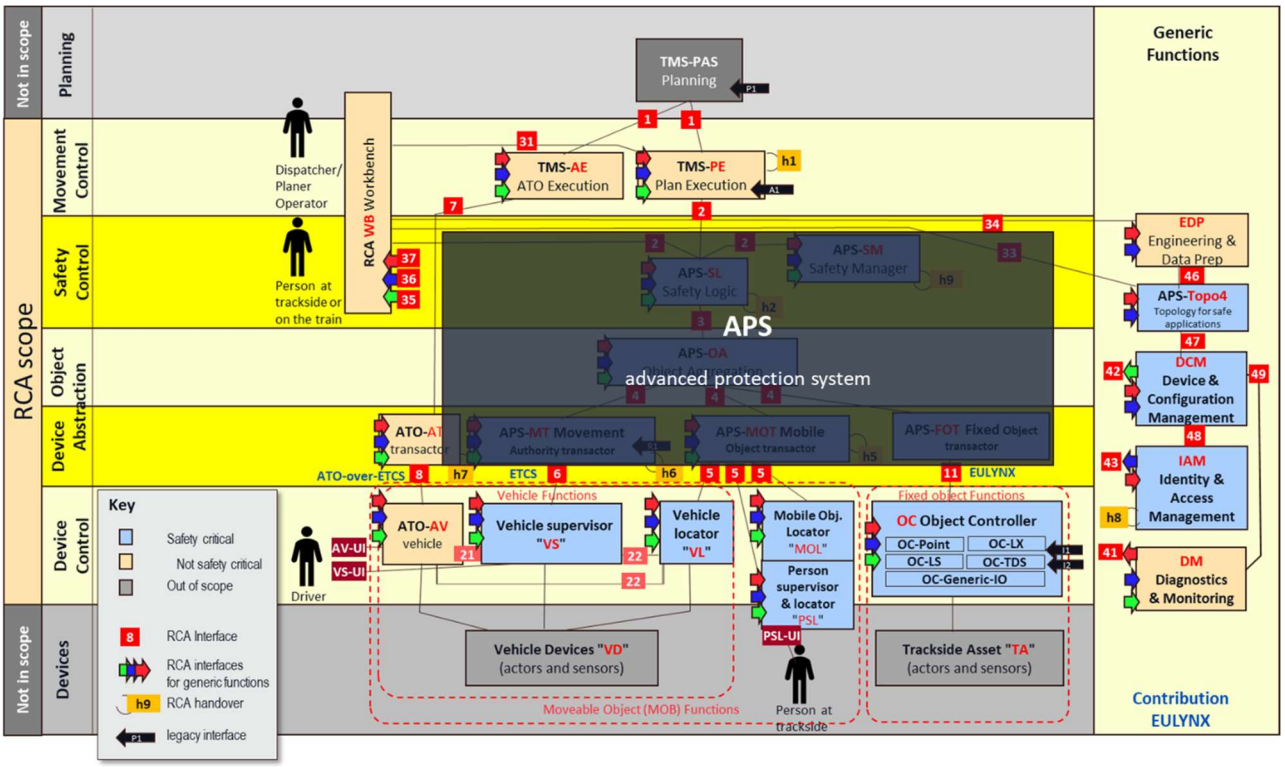

Figure 2. Where is the interlocking in RCA?

2.1.6. Important control loops

The control loops are the functional “highways” of a CCS. They help illustrate the interfaces and how they are used. The control loops do not specify any additional behaviour. They tend to be performance-critical and to drive non-functional requirements. The following diagram shows important control loops in RCA:

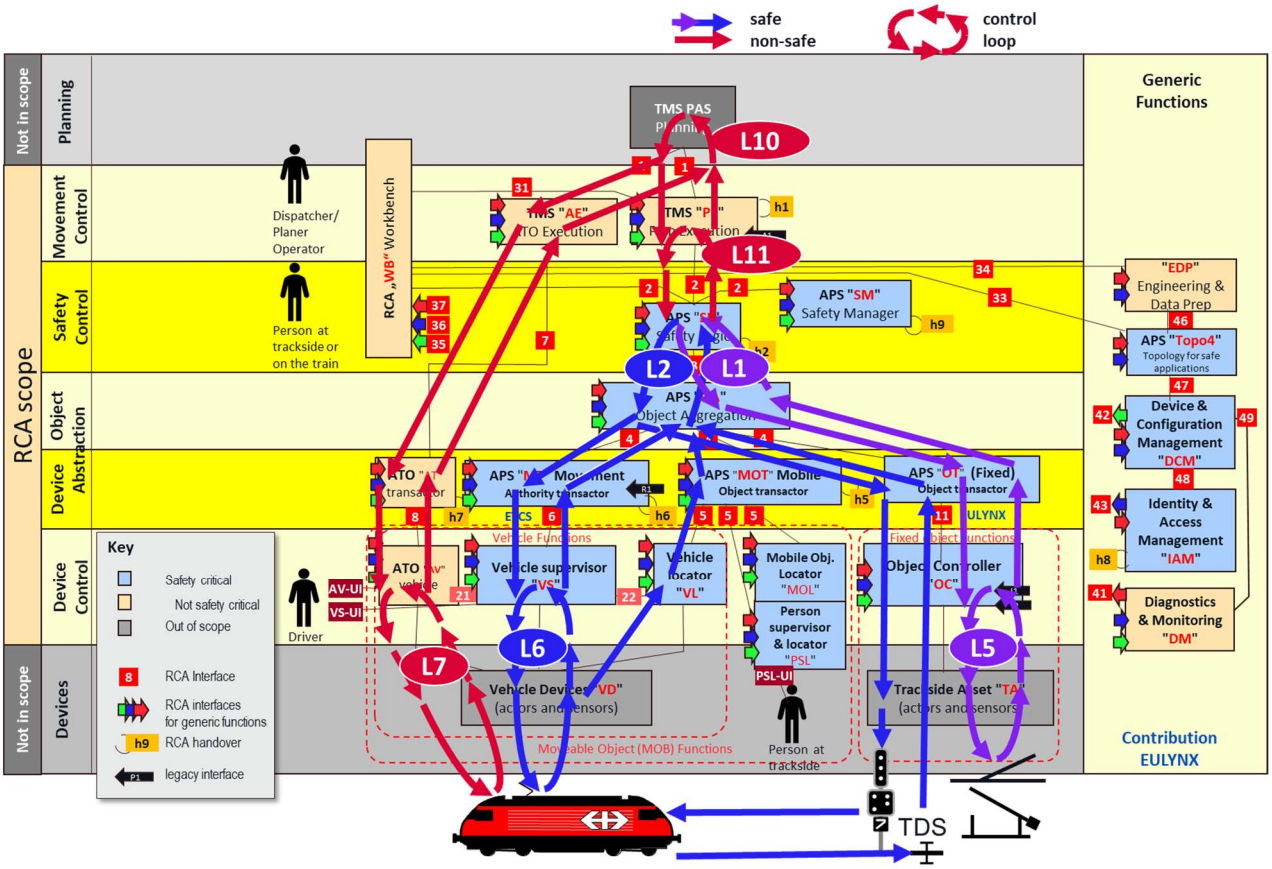

Figure 3. Important control-loops in RCA

The following table provides a short description of these control loops:

Id	Description
L1	Interplay between: • APS-SL prescribes state: trackside device in a certain allocation state; • “Real world” (as seen by the device controllers): has effective allocation state, which must comply with prescribed state. Downward: Demanded allocation state. Upward: Effective allocation state.
L2	Interplay between: • APS-SL prescribes state: Grant specific Movement Permission to a Moveable Object; • “Real world” (as seen by the device controllers): effective location, which must comply with prescribed state. Downward: Movement permission (permission to change location). Upward: Effective location of the Moveable Object in the real world. Note: There are multiple paths for detecting the location and granting the Movement Permission.
L5-L7	Local control loops, not directly in scope of RCA.

Id	Description
L10	Interplay between: • TMS-PAS, where an operation plan is established (and frequently updated / optimised); • TMS-AE and TMS-PE executing the plan as faithfully as possible and updating TMS-PAS with “real-world” information by providing an execution status.
L11	Interplay between: • TMS-PE, decomposing the operation plan from PAS into single requests for changing trackside device allocation state (L1) and for granting movement permission (L2), issued when all the preconditions regarding “device allocation state” and “location of the moveable objects” are met; • APS-SL, ensuring TMS-PE requests are safely executable and controlling their execution, updating TMS-PE with the “effective device allocation state” and the “location of the moveable objects”. Downward: Requests for changing trackside device allocation state (L1) and for granting movement permission (L2). Upward: “effective device allocation state” and “location of the moveable objects”.

Note: The signals shown in this diagram may be needed at the border (border to another supervision system or to non-supervised area) of an RCA-based system.

2.2. RCA technical architecture

The RCA component architecture takes into account technical considerations (such as dealing with objects and devices) but focuses mostly on functional or logical aspects, i.e., without considering where and how the components actually run. In this chapter we outline the technical aspects needed to deploy and run an RCA-based system.

We divide the technical architecture into 3 perspectives:

- **Deployment architecture**: what computation, what data will reside on what kind of resource nodes at which locations connected with which communication infrastructure;
- **Runtime environment**: architecture of each node: HW, OS, SW;
- **Communication architecture**: how exchange of information (on different levels, e.g., between components and between nodes) is implemented (as a stack).

We use the term “node” to denote a physical resource to which functionality in the form of a component can be mapped. The following diagram shows:

- RCA components (as described in the interface architecture) are deployed on nodes. Not shown: several components can be deployed to the same node;
- The nodes include a “Runtime Environment” and a “Comm-Stack”. The interfaces towards the components are standardised (as “RTE and “CA” (“Comm-API”)); see the document “RCA Beta – Platform independence” for more details;
- Communication between nodes happens physically over a set of communication protocols (“CP”). Logically the components have specific interfaces as described in the interface architecture;
- Note: for safety-relevant functionality mechanisms such as voting, guaranteed delivery, etc. are necessary. We expect some of these mechanisms to be provided in the runtime environment and / or Comm-Stack. This normally entails that the component must obey specific requirements while interacting with the underlying systems;
- Note: important non-functional requirements such as performance and availability are strongly driven by the features and overall architecture of the nodes;
- Note: it is likely that there will be two or more types of runtime environments, depending on required safety level or other non-functional characteristics;

- Not shown: in RCA the nodes may be virtualised or even be “running in the cloud”;
- Not shown: several components may share the same node.

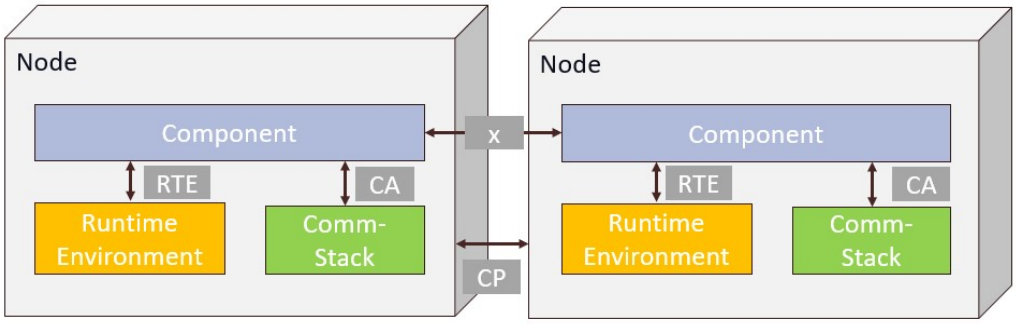

Figure 4. RCA components are based on an independent platform

See also the document “RCA Concept Platform Independence” [RCA.Doc.11]

2.2.1. Deployment architecture

The specific deployment architecture is not specified by RCA, except the technical implementation of functional interfaces and the standard runtime environment for applications. The concrete deployment architecture depends on local decisions by the implementing IM.
We strive for a standardised runtime environment for each RCA component. This means that an RCA component development can focus on functional aspects, with most technical and some non-functional aspects being delegated to the runtime environment.

2.2.2. Runtime environment

The following diagram shows the important principle of separation of functional SW (the components), the OS (with common middleware such as safety functions) and the hardware. These elements are loosely coupled over well-defined interfaces and allow a) the components to be separately procured from the computing infrastructure and b) the computing infrastructure to be shared among several components.

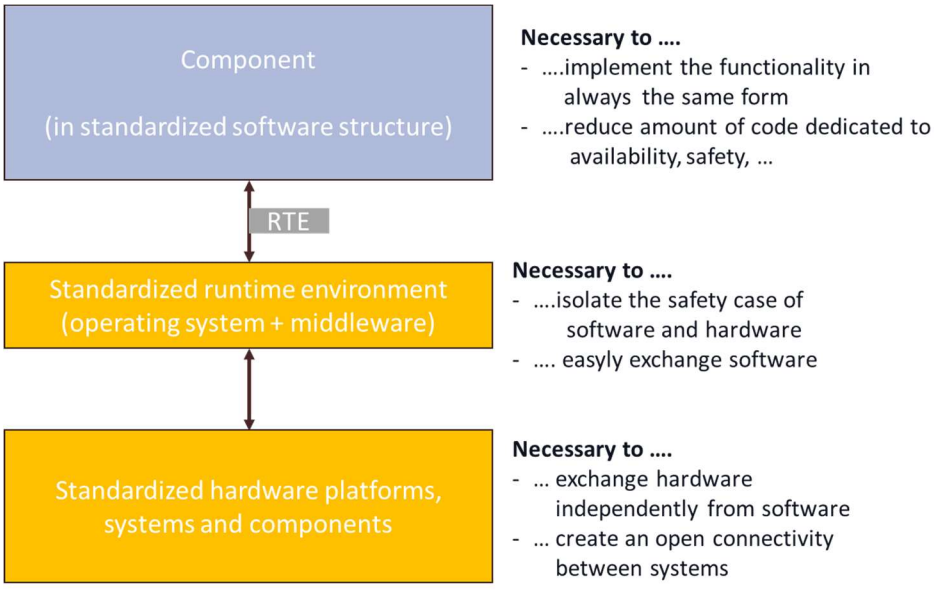

Figure 5. Elements of platform independence

See also the document “RCA Concept Platform Independence” [RCA.Doc.11]

An even more ambitious example is given in the Figure 6 (target for the programme smartrail 4.0):

- Modularised platform, distinguishing safe and unsafe components (“apps”);
- Using the same platform on all devices (from data centre to small personal devices).

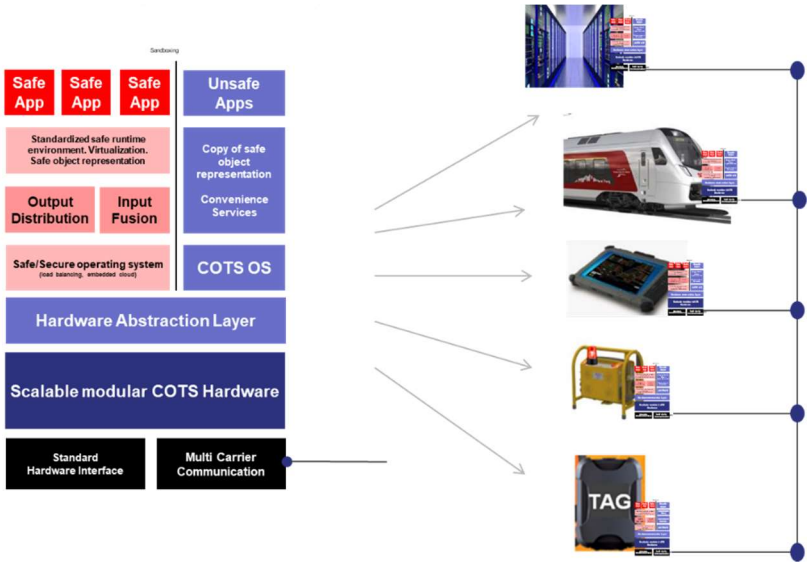

Figure 6. Example of platform independence

2.2.3. Communication architecture

RCA specifies communication over the “whole stack”, i.e., not only on the “logical” level between components, but also all the way down to physical details, allowing the exchangeability of logical components as well as physical nodes (“boxes”). This includes integrability between nodes from different suppliers.
The following diagram shows the principle of an RCA communication stack linking logical components and physical nodes:

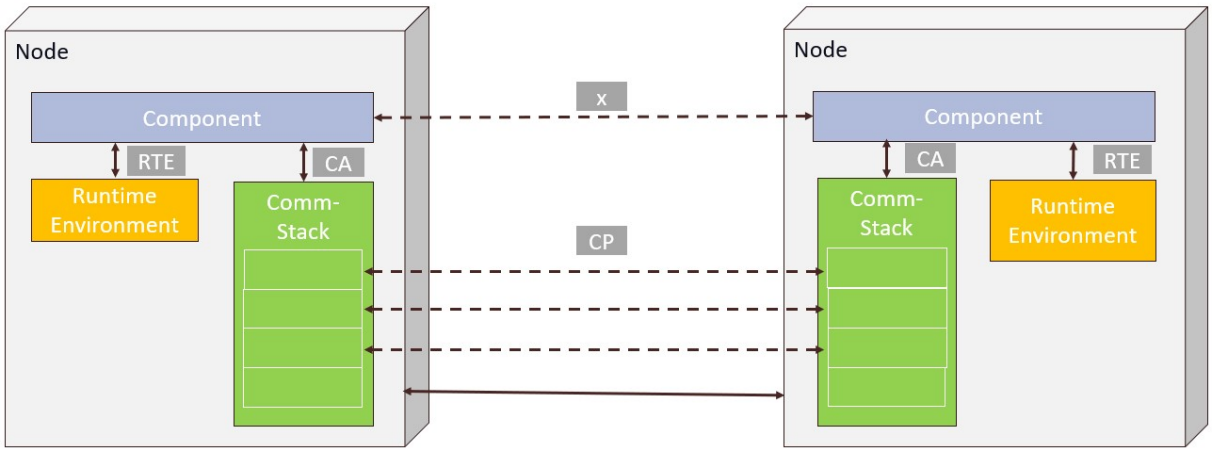

Figure 7. Structure of the communication architecture

The interfaces RTE, CA and CP are described above. The interface “x” stands for a specific interface between 2 components, as described in 2.1.4 “List of component-to-component interfaces”. The “x” stands for the “application-level”, abstracted communication between 2 components, physically the communication happens over the “Comm”-Stack.
For the RCA communication stack, we use the following principles:

- Upper layers are independent of lower layers (this includes “bearer independence”);
- Ideally one stack can be used for all components / nodes; at the same time, it is possible to use more than one stack in an RCA-based system;

- The stack includes functions such as transport of packets, sessions, security layer, safety layer, application protocols (including protocols for versioning, upward / downward compatibility);
- The stack is to be based on (a selection of) existing and open protocols (to ensure long-term support).
- When components are co-located on the same node, simplifications at the lower levels are possible.