Research Report
SIL4 Data Center

Version History

Version	Date	Changes
1.0	2021-10-06	

Authors and Contributors

Authors	Sonja Steffens	Siemens Mobility GmbH
	Frank Skowron	Deutsche Bahn AG
	Tom Suess	Siemens Mobility GmbH
	Oliver Mayer-Buschmann	Deutsche Bahn AG
	Frank Eschmann	Deutsche Bahn AG
	Patrick Marsch	Deutsche Bahn AG
	Niklas Schaffrath	Siemens Mobility GmbH
	Markus Wischy	Siemens Mobility GmbH
	Reza Anbarestani	Deutsche Bahn AG
Contributors	Prashant Pathak	Deutsche Bahn AG
	Andreas Armbrecht	Siemens Mobility GmbH
	Kathrin Baer	Siemens Mobility GmbH
	Frank Breitschaft	Deutsche Bahn AG
	Ibtihel Cherif	Deutsche Bahn AG
	Ramin Hedayati	Deutsche Bahn AG
	Alexander Heine	Deutsche Bahn AG
	Ulf Kittel	Siemens Mobility GmbH
	Michael Neumann	Deutsche Bahn AG
	Markus Pichler	Siemens Mobility GmbH
	Dirk Spenneberg	Deutsche Bahn AG
	Sandra St. Pierre	Siemens Mobility GmbH
	Constantin Wagner	Deutsche Bahn AG

Contents

1 Executive summary

Due to megatrends such as urbanization and an increased awareness for climate change, railway operators around the world have the urgent need to increase capacity, quality, and efficiency of rail operations. Driven by these megatrends, the railway sector is currently conducting one of its largest technology leaps in history, characterized by the introduction of a high degree of automation and various novel technologies in the rail system. As a basis for this, Deutsche Bahn and Siemens Mobility, within the sector initiative “Digital Rail for Germany”, are jointly investigating a key element of future rail operations – the SIL4 Data Center.

The collaboration has been undertaken in order to gain synergies from different points of view, experiences, and expertise, such as in the specification of major high-level requirements for a SIL4 Data Center including inputs from RCA/OCORA’s "White Paper on a Generic Safe Computing Platform for Railway Applications" (Version 1.01, December 2020), definitions by the EULYNX initiative, and already established standards for security and homologation. To assess the technical feasibility, an architectural design proposal has been created including a modularized approach towards a more flexible replacement of components within the system lifecycle, and the involvement of a multitude of different vendors, as well as a potential migration strategy. Furthermore, the collaboration seeks to describe how today's operational processes in rail operations will be impacted from a technical, financial and regulatory perspective.

The detailed analysis of the top-level objectives from the RCA/OCORA White Paper and EULYNX mentioned above shows that the complexity of system integration and homologation will grow extremely for a modular separation of applications, middleware, and hardware compared to EULYNX and existing SIL4 infrastructure setups. This research report provides an initial analysis on whether benefits through increased modularity, evolvability, and flexibility outweigh a significant surge in effort and complexity. Also, the critical consequences of the objectives for homologation, e.g., the need for recertification in the event of system changes, will be pointed out. An additional objective for geographical redundancy has been defined within the requirements analysis.

Based on high-level requirements, Deutsche Bahn and Siemens Mobility have developed a possible layered architectural design of a SIL4 Data Center. This specifically considers different application deployments over distributed data centers and different degrees of involvement of multiple vendors for one installation and is based on a common virtualization layer. This also includes an analysis of the dependencies between applications and the safety layer in order to achieve the best possible generic API design for modularity in the future, as envisioned by RCA/OCORA. As the SIL4 Data Center is a huge technological and conceptual leap, a potential migration strategy has been elaborated that is to ease the coexistence and operation of legacy systems next to RCA-compliant applications. Besides the engineering aspects, maintenance and update concepts which can be applied to a SIL4 Data Center are also described to achieve the highest possible availability of the overall system. Having multiple applications on one SIL4 Data Center, the aspects of operating applications with different safety integrity levels must be considered. Also concepts of geographical redundancy, including their potentials and challenges for additional availability and safety, are discussed. Ultimately, the report highlights how the SIL4 Data Center can be embedded into the cybersecurity architecture specified by the cybersecurity working group of the EU research project X2Rail-3.

Integration and testing of the SIL4 Data Center and its applications are particularly challenging tasks, especially for a safety-relevant system with components from multiple vendors. The report provides a recommendation for an integration concept that differentiates between vertical integration of hardware and software and horizontal integration of different applications. The concept also considers the different roles and responsibilities for integration and testing. An overall integration and testing responsibility is relevant for the approval process. Whoever is taking over overall responsibility for the whole system and how a potential split of responsibilities between operator and multiple vendors can be achieved still needs to be defined, as envisioned by the RCA/OCORA objective.

For both Deutsche Bahn and Siemens Mobility, it has been crucial not only to look at the technical implications of a SIL4 Data Center but also to develop an understanding for the overall changes triggered by this technological approach and how those affect the complete process chain throughout the system lifecycle. Therefore, this research report also includes an analysis of the impact of operational changes compared to current technological approaches, including a rough financial evaluation and a compilation of further aspects regarding the composition of the total cost of ownership. The main finding is that the increased level of complexity will lead to substantial ramp-up costs, but the overall business case for the SIL4 Data Center approach along the full technology lifecycle appears positive. In addition, it is to be noted that some costs are currently difficult to evaluate, especially those related to multivendor integration.

The lifecycle perspective also leads to an outline for a different maintenance strategy, as various lifecycles of the individual parts need to be considered when utilizing COTS components in a SIL4 Data Center. Along with this aspect, there is also the assessment of different levels of combining and operating components from multiple vendors involved and the implications of such a setup.

A key challenge in the data center approach is that also revolutionary technological approaches need to remain compliant with existing safety regulations, as it is imperative that the safe system design be certified by the authorities. Therefore, this research report includes an outline of relevant safety standards and how these must be applied in the context of homologation of a SIL4 Data Center and impacts of the application of safety standards on the overall system design and definition of interfaces are highlighted. It is important to minimize the set of requirements from safe application layers (SRACS), and to enable changes to non-safety-relevant software and hardware components without affecting the upper system layers.

Throughout their collaboration, Deutsche Bahn and Siemens Mobility realized that certain aspects around the SIL4 Data Center concept have not been defined yet. This holds especially true for the independency between modular applications on the one side and the safety layer solutions on the other. In this respect, it seems that overall integration and homologation costs may be higher than envisioned by RCA/OCORA. Additionally, the technical support for clear evidence of any influences between the different parts to enable the definition of clear responsibilities is currently unsolved. Another open point is the high degree of flexibility regarding multiple vendors of modular systems, which is in contradiction to the usual aim to define few generic system variants that can be eligible for type approval to be reused in multiple installations. In addition, it must be noted that cross-vendor integration of safety-relevant parts within an installation is still unsolved in today's setups.

Detailed dependencies and interactions between different applications, especially from a performance and time behavior perspective, are currently not defined. Because of this, there are no specific requirements for the design of safety layers regarding the simultaneous execution of several applications. Moreover, it is important to point out that cloud-like approaches, e.g., dynamic resource management, directly affect the safety case of safety layers and cannot easily be fulfilled by non-safety-relevant software for cloud infrastructures.

Within the research collaboration of Deutsche Bahn and Siemens Mobility, the idea of a SIL4 Data Center has been pushed forward significantly. The objectives described in the RCA/OCORA White Paper have been refined and analyzed under the aspects of feasibility and practical implementation. Design alternatives, potential pitfalls, high effort drivers have also been identified. This research report will be the foundation for refining the SIL4 Data Center.

2 Introduction

2.1 Background

Less traffic, less congestion, less particulate matter – and more people and more goods on the rails: Driven by these trends the rail sector in Europe is currently conducting one of its largest technological leaps into the digital future. The sector initiative Digitale Schiene Deutschland is taking advantage of this opportunity and bringing future technologies into the rail system. This benefits not only passengers, but also the climate and Germany as a business location. And all this without having to construct a single new track.

The foundation for this is being laid with the fundamental modernization and digitization of the infrastructure through the consistent introduction of digital control and safety technology. To achieve a far-reaching digitization of the railway system, DB is currently working with other European railways and industry partners within the Digitale Schiene Deutschland towards introducing, for example, the following:

- AI-based real-time dispatching of rail operations throughout Germany
- automation of rail operations up to what is known as Grade of Automation 4 (GoA4)
- a new architecture for command, control and signaling (CCS) that enables train operation with minimal distance through an ETCS Level 3 moving block approach
- fully automated incidence prevention, mitigation, and resolution.

Overall, a significant improvement in capacity, and efficiency of the railway system will be achieved, all of which are requirements for more traffic on rails and strengthening of the railway as the climate friendly mode of transport. To implement the aforementioned points, it will not only be necessary to introduce various technologies such as artificial intelligence (AI), advanced sensing, high-precision localization, etc., into the rail sector. Furthermore, a novel computation and connectivity fundament will have to be established to support the applications that will drive future rail operations.

As one key element of the fundament for future rail operations, Siemens Mobility (SMO) and Deutsche Bahn (DB) have jointly collaborated on the concept of a **SIL4 Data Center** for safety-critical trackside applications up to SIL4. Such applications would, for instance, include the further evolution of the European Train Control System (ETCS) and digital interlockings, e.g., in form of the Reference CCS Architecture (RCA), and additional safety-critical applications that will be introduced in the context of highly automated rail operations, such as automated incidence management. In this case, the term "SIL4 Data Center" corresponds to the concept of distributed computing platforms that are based on a modular separation of applications, middleware, and hardware for reasons that are detailed in the following.

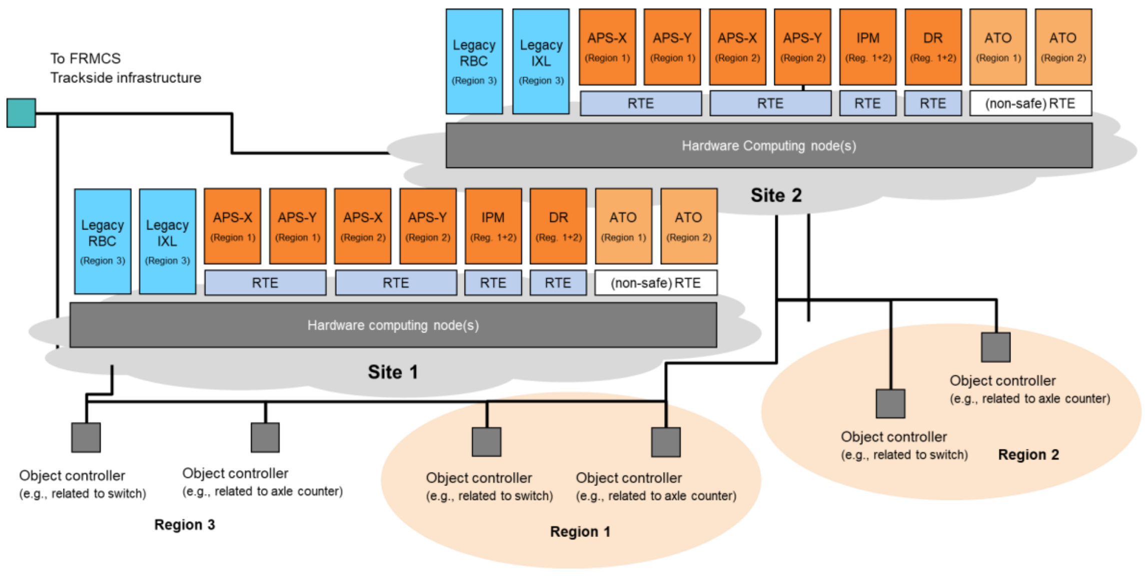


Figure 1 High deployment example for a SIL4 Data Center with geographical redundancy

For illustration purposes, a possible (highly simplified and abstracted) exemplary deployment of a SIL4 Data Center setup is shown in the figure above. This consists of two pools of computer hardware in two distinct data centers (referred to as "Site 1" and "Site 2" here).

The application examples shown in the figure include the following:

- the possible future Advanced Protection System (APS, basically an architectural evolution of ETCS/RBC trackside functions), which, in the example, is split into subsystems (here called APS-X and APS-Y), and where one bundle of application instances (APS-X and APS-Y) jointly serve one region of object controllers (related to switches or axle counters), serving regions 1 and 2
- applications Incidence Prevention Management (IPM) and Digital Register (DR), which both are in this example assumed to serve both regions
- Automatic Train Operation (ATO), which is typically seen as non-safety-relevant (at least for GoA2 operation), but which may for reasons of synergy also be deployed on the same pools of computing nodes, albeit likely on a simplified and non-safe runtime environment (RTE)
- a legacy installation with "Interlocking" functionality, serving region 3
- a legacy installation with "RBC" functionality, serving region 3

For maximum availability, the application instances are geo-redundantly deployed in both data center sites. In general, one can see in Figure 1 that multiple application instances may run on a common RTE instance, or have individual RTE instances, as elaborated in detail in section 4.

This work has taken input from related activities of the railway initiatives RCA and OCORA on a Safe Computing Platform (for on-board and trackside equipment) OCORA-40-004-Gamma-CP-Whitepaper [3] but has gone further in depth towards possible specific implementations of SIL4 Data Centers for trackside CCS applications, and also complemented the work from RCA and OCORA via a detailed study of possible migration steps from today's application and platform approaches to the envisioned target computing architecture.

2.2 High-level objectives of a Safe Computing Platform in SIL4 Data Center context

Clearly, the key objectives for a SIL4 Data Center, being the focus of the collaboration among SMO and DB, are very much related to those of a generic Safe Computing Platform as covered in RCA and OCORA. For this reason, the following table lists the objectives as identified in RCA and OCORA-40-004-Gamma-CP-Whitepaper [3] with a ranking and additional comments by DB and SMO, specifically with a focus on trackside SIL4 Data Centers and also incorporating findings from the collaboration. In section 3.1, these objectives are then evaluated from a technical feasibility perspective.

No. in RCA /OCORA	**Objective OCORA-40-004-Gamma-CP-Whitepaper [3]**	**Importance and comments by SMO and DB, specifically focused on trackside SIL4 Data Centers**
1	**Meet safety and real-time requirements of CCS (and similar) railway applications** The platform shall meet safety requirements of applications up to safety integrity level (SIL) 4, e.g., acc. to EN 50126, EN 50128 and EN 50129, and support applications with real-time characteristics (e.g., overall processing cycles in the order of 10-100ms).	**Very high** Clearly, it is a main requirement of the Safe Computing Platform to support applications up to SIL 4. From DB’s perspective, it is also important to stress that the Safe Computing Platform should not only focus on CCS needs, but also explicitly support additional safety-relevant applications that will be introduced in the context of digitalized rail operations (e.g., digital map support, incidence management, etc.).

No. in RCA /OCORA	Objective OCORA-40-004-Gamma-CP-Whitepaper [3]	Importance and comments by SMO and DB, specifically focused on trackside SIL4 Data Centers
2	**Respect diverse lifecycles of business logic, runtime environment, and hardware.** The platform shall be partitioned with respect to the different lifecycles of business logic, runtime environment, and hardware. The platform shall support fully independent lifecycle handling, i.e., with minimal dependencies.	**Very high** This objective is clearly seen as a main paradigm for the design of a SIL4 Data Center (and Safe Computing Platform in general). From SMO's perspective, it is also very important to consider the lifecycles of the legacy solutions when migrating into the SIL4 Data Center.
3	**Open market to new players** The platform shall open the market to new, non-rail-oriented software and tooling companies. They shall be able to become involved in functional application development without providing their own platform safety mechanisms (e.g., related to safe communication, fault tolerance implementation, etc.).	**Very high** This objective is seen as particularly important especially by DB. However, the processes need to be considered to ensure that different vendors' applications can be integrated and run together. SMO is in general in favor of any competition that takes place on a level playing field.
4	**Minimize total cost of ownership (TCO)** The platform shall minimize the total cost of ownership, i.e., the overall lifecycle cost.	**Very high** A minimization of the TCO is clearly essential for the entire rail sector. From SMO's perspective, it is important to stress, however, that it cannot be taken for granted that, e.g., modularity based on standardized interfaces inherently leads to a reduction of the TCO, particularly considering integration efforts – this has to be evaluated in detail and the right level of modularity and vendor multiplicity has to be found.
5	**Vendor independence** Different vendors shall be able to provide functional applications, computing platforms and development tools, respectively, without a vendor lock-in. It shall be possible to purchase hardware directly from different vendors throughout the lifespan of the software. The platform shall build on existing HW/SW solutions, stimulating competition among vendors and allowing them to shine with their specific expertise and distinctive solution features.	**Very high**
9	**Facilitation of application development** The platform shall use an open, well-documented application model and programming interface, facilitating that third parties develop applications.	**Very high** Especially from DB's perspective, this point is seen as rather essential, also because digitalized rail operations will require the introduction of new safety-critical applications (see above). It is important to stress that the stated application model and programming interface must be standardized.
10	**Modularity** The platform shall allow for a modular safety certification process, using pre-certified components leading to a dramatically simplified and shortened full system certification process. An evolution or update of the platform shall not	**Very high** A certain degree of modularity is, of course, necessary to accommodate the afore-mentioned diverse life cycles of application, middleware, and hardware. From SMO's perspective, it is nevertheless essential to determine the right degree of modularity considering the associated integration complexity as well.

No. in RCA /OCORA	Objective OCORA-40-004-Gamma-CP-Whitepaper [3]	Importance and comments by SMO and DB, specifically focused on trackside SIL4 Data Centers
	require a new E2E homologation of application and platform, as detailed in Section 7.	
11	**Encapsulated, transparent fault tolerance mechanism** The platform shall transparently encapsulate the safety and fault tolerance mechanisms. Vendors may offer different (new) approaches to safety and fault tolerance as they become available on the market – solution agnostic and future-proof.	**Very high** From the perspective of SMO and DB, an encapsulated, transparent fault tolerance mechanism is actually not an objective in itself, but rather a prerequisite to achieve the aforementioned objective of modular separation of applications, middleware, and hardware.
7	**Migratable and portable business logic** The business logic is considered a significant system asset, being the component with the longest lifetime. It must hence be portable to different computing platform evolutions. We here further differentiate: • Migratability for legacy applications: It should be decently easy to migrate legacy applications to the new platform. • Portability for new applications: Applications running on the platform should be portable to any other vendor's or evolved version of the platform.	**High** From the perspective of SMO and DB, it is rather straightforward to expect that by modularly separating the application, middleware, and hardware, applications should be portable from one platform to another (though it is understood that this will always come with some integration effort). What appears also important and has hence seen special emphasis in the collaboration among SMO and DB, is to provide a migration path from legacy solutions (both applications and platform approaches) into the same common data center. To that end, the common basic requirements for the porting of legacy solutions onto the same virtual computing nodes need to be defined.
15	**Support for running multiple applications (also with different SIL levels) on one physical platform** It shall be possible to run multiple applications, possibly with different SIL levels, on a single physical platform to reduce cost, space, power dissipation, etc., and simplify certification, maintenance, system evolution, etc.	**High** For data center deployments, which have been in the focus of the collaboration among SMO and DB, one may argue that mixed SIL support is not that critical as for on-board installations, as one could afford to have platform realizations that are focused on fewer (or one) applications. However, it is important to stress that each safety concept of each solution for a safety layer shall allow that any other software (with any SIL level) can run on the same physical platform. Hence, the platform should inherently always support mixed SIL.
14	**Centralization** The platform shall allow to centralize applications physically in a safe data center to simplify lifecycle management, reduce TCO by means of simplified, optimized operations, and benefit from increased availability and optimized resource usage.	**High** Centralization is seen as paramount by DB, as it has a strong impact on TCO and OPEX reduction, see also section 6.
12	**Scalability** The platform shall be highly scalable, i.e., it should by design be able to support an arbitrary number of applications and arbitrary number of compute nodes	**High** From the perspective of SMO and DB, scalability and centralization are strongly related to each other, i.e., scalability (particularly in the number of computing resources and the number of applications supported by one platform) is a prerequisite toward centralization.

No. in RCA /OCORA	Objective OCORA-40-004-Gamma-CP-Whitepaper [3]	Importance and comments by SMO and DB, specifically focused on trackside SIL4 Data Centers
13	**Flexible usage of compute resources** The platform shall enable a flexible mapping of business logic to compute resources (e.g., such that the platform can be expanded while applications are running, and that business logic can be re-mapped when compute nodes fail). It shall be able to leverage advances in computing technology (i.e., when better compute nodes are available, it shall be possible to assign more instances of business logic to the compute nodes).	**High** A flexible usage of computing resources is seen as important by DB, particularly for the following use cases: • It shall be possible to change the association of application and RTE instances to computing nodes during runtime, e.g., to perform planned hardware maintenance or replace/upgrade hardware computing nodes, involving manual reconfiguration. • The platform shall be able to automatically respond to failures in hardware, e.g., by creating a new replica of an affected application (and a new RTE instance) on a spare hardware computing node. This must happen sufficiently fast to prevent an application from running in an undetermined mode for too long (e.g., 2oo2). From SMO's perspective, the system complexity and cost to address these objectives can likely be strongly reduced if the need for manual intervention is accepted in both cases. It is also noted that the second use case above may be addressed via local redundancy or geographical redundancy.
New	**Native support of geographical redundancy** The platform shall support that the instances of individual application functions are geographically distributed for the purpose of highest availability and resilience, especially in the context of centralization.	**High** This objective was not listed in the OCORA-40-013-Gamma-CP-Requirements [4] but identified and added in the context of the collaboration among SMO and DB. It shall be noted that it will only be supported among geographically distributed RTEs from the same vendor.
8	**System evolvability** The platform shall be open to extensions (in the sense of additional system services that are added over time, e.g., related to FRMCS). Adding new functionalities shall be possible with minimal to no changes to existing applications (though these may naturally not be able to leverage the new functionalities).	**High** System evolvability is obviously an important objective, but it is assumed that this may be achieved rather easily through a suitable system design.
6	**Industrial readiness** It shall be possible to procure a platform as off-the-shelf solution supported by an open and dynamic market. The solutions shall be mature (e.g., reliability proven in field) and backed by effective acceptance and integrated logistical support (e.g., maintenance service, tooling, availability of spare parts).	**Modest** From the perspective of SMO and DB, there is a certain risk attached to this requirement: Naturally, the introduction of a safe computing platform with a clear separation between applications, middleware, and hardware is a major technology leap for the rail sector, and this will clearly require some years of technological development and prototyping. If too much emphasis is now placed on having "mature" solutions available, there is a risk of compromising the overall vision.

2.3 Expected benefits of the envisioned SIL4 Data Center

In summary, SMO and DB expect the following benefits from the SIL4 Data Center concept investigated in the project:

- It shall provide a future-proof basis for the application needs of future safety-critical railway applications, in particular allowing to decouple the different lifecycles of applications, middleware, and hardware (also allowing for better obsolescence management) and to maximally utilize commercial-off-the-shelf (COTS) components and leverage the latest trends in the IT sector, for instance regarding virtualization.
- It shall allow for a higher degree of centralization of application functionality, in order to save on the number of data centers needed for future rail operation, and to consequently save CAPEX and OPEX.
- It shall provide the means for geographical redundancy that enable higher availability of railway applications overall.
- Altogether, it shall provide a well-reduced total cost of ownership for the data center infrastructure for future railway operations.

For the abovementioned expected benefits of the SIL4 Data Center concept, an initial rough quantitative cost analysis has been conducted in the collaboration, as described in section 6.1.

2.4 SMO motivation to collaborate on the SIL4 Data Center

For SMO it is of tremendous importance to collaborate on concepts for the future of rail transportation. This holds particularly true in the case of a potential architecture for safe rail applications. As a leading innovator of the industry, it has been the clear motivation of SMO to collaborate on the SIL4 Data Center to align the own view with that from DB, specifically regarding the following:

- technical overall architecture based on the SMO experience in the context of defining such an architecture for a SIL4 Data Center and developing such a needed software-based safety platform as runtime environment
- identifying affected cross functionalities and corresponding interfaces to be defined as standard for an overall data center solution with different vendors involved
- identifying the affected interfaces which need to be defined as standard for an overall data center solution with different vendors involved
- identifying needed changes and open points in the process of homologation for a system/installation consisting of modular parts supplied by different vendors
- identifying needed changes and open points in the process of integration of a system/installation consisting of modular parts supplied by different vendors
- identifying required changes and open points throughout the complete lifecycle of a SIL4 Data Center and how operational processes are impacted

In the context of this alignment, a particular emphasis has been placed on the following:

- the relevant requirements for basic migration of any legacy solution into a common SIL4 Data Center
- the development of new applications (e.g., as defined by RCA)

2.5 DB motivation to collaborate on the SIL4 Data Center

While for DB the necessity and pursuit of a safe computing platform approach with a modular separation of application, middleware, and hardware is unquestionable, it is clear that this bears substantial challenges. It is hence essential for DB to collaborate with suppliers such as SMO at an early conceptual stage in order to:

- determine if there are any potential showstoppers regarding the vision of the railways
- obtain an early understanding of which requirements are associated to which effort and cost

- see how the safe computing platform approach can possibly be adjusted such as to maximally leverage standardized developments of suppliers
- develop migration strategies from today's platform approaches to the target architecture

While DB sees a strong merit in defining a common high-level safe computing platform approach for both on-board and data center deployments, it has been important for DB to focus the project on the SIL4 Data Center to go further into detail into particular requirements and challenges, and consequently also different implementation approaches, related to data center deployments.

3 Objective evaluation and technical requirements

In this chapter, some terms are introduced, that will be explained later in section 4. They are also briefly defined in Terms and Abbreviations, see section 10.1.

3.1 Evaluation of the high-level objectives from RCA/OCORA

In this section, the high-level objectives from RCA and OCORA, as described before, are evaluated from a technical perspective.

3.1.1 Evaluation of objective No. 1 – Meet safety and real-time requirements

Objective recapped: The platform shall meet safety requirements of applications up to safety integrity level (SIL) 4, e.g., acc. to EN 50126, EN 50128 and EN 50129, and support applications with real-time characteristics (e.g., overall processing cycles in the order of 10-100 ms).

This objective appears realizable from a basic technical view. For technical details regarding the RTE, see section 4.10**.**

Specific technical challenges regarding time behavior are:

- **Replica cycle time**
  For the synchronicity of the application replicas running on different computing nodes jittering effects and network latency times (especially in huge network configurations in case of geographical redundancy) need to be considered. By this cycle times and reaction times in small ranges as e.g., 10ms are not possible. For this see also section 4.1.3.
- **Application processing time**
  These processing cycles significantly determine the overall processing time of a sequence of application invocations. As within a replica, the communication can be direct (unvoted); this can hence be improved when these applications are bundled within a replica so that the overall application processing time is (mostly) only depending once on the processing cycle.
- **Application reaction time**
  The minimum possible reaction time of an individual application replica depends on the named cycle time.
  Detailed information regarding the time behavior of an RCA application (APS*) and especially the time-relevant criteria for inter-application communication is not available today.
  So this topic cannot be evaluated in deeper detail. Therefore, a target corridor for the time behavior of an RCA application needs to be defined.

3.1.2 Evaluation of objective No. 2 – Respect diverse lifecycles

Objective recapped: The platform shall be partitioned with respect to the different lifecycles of business logic, runtime environment, and hardware. The platform shall support fully independent lifecycle handling, i.e., with minimal dependencies.

The desired support of diverse lifecycles can be achieved by strictly considering the following:

- a defined system architecture with defined interfaces between all involved parts (COTS hardware, virtualization, RTE, application) considering the individual lifecycle aspects
- defined processes for the overall integration and validation of changed parts within the installation (e.g., new COTS hardware, IT security patches, new engineering data, new application software, etc.)

The architecture and processes must be elaborated and fixed before any device or system will be commissioned and put into service.

Specifically, for the RTE, the requirement “separation of the safety layer from the non-safety-relevant parts” is proposed to handle diverse lifecycles in the context of flexible maintenance:

- short lifecycles for non-SIL software of the RTE, especially for the IT security mechanism (to install patches in short time periods)
- longer lifecycles for the safety-relevant parts of the RTE as the safety layer and the safety-relevant applications

3.1.3 Evaluation of objective No. 3 – Open market to new players

Objective recapped: The platform shall open the market to new, non-rail-oriented software and tooling companies. They shall be able to become involved in functional application development without providing their own platform safety mechanisms (e.g., related to safe communication, fault tolerance implementation, etc.).

One key point to stress in this respect is that the RTE only ensures that applications are running safely on the COTS hardware – the safety of the implemented functionality of the application itself is irrelevant to the RTE. For the development of safety-relevant applications, new application vendors therefore need the know-how and competence regarding the standards in any case, e.g., CENELEC, even if they do not need to bother about platform aspects such as composite fail-safety, etc.

In addition, new application vendors must also provide the safety case for an application running on an RTE. Not only will it cover the safety aspect of the application's own functionality, but also the part of integrating the application on the RTE and validating the fulfillment of the SRACs provided by the RTE. This requires a separate certification or even an approval process.

Hence, any new application vendor will, of course, have to consider if its own business case is still positive, especially given the end-to-end integration and certification effort that this player would have to bear.

The situation may be different for novel safety platform vendors from outside the rail sector. Naturally, they would have to acquire the know-how of developing CENELEC-conformant platforms but in this case may leverage experience from providing platform solutions for safety-critical applications in the automotive and aviation fields, and they would not have to bear the end-to-end integration and certification effort.

3.1.4 Evaluation of objective No. 4 – Minimize total cost of ownership

Objective recapped: The platform shall minimize the total cost of ownership, i.e., the overall lifecycle cost.

For the "total" view of the cost situation, it is necessary to distinguish between different aspects:

- Centralized COTS-based data center
- operational benefits
- development costs for RTEs and applications
- integration costs
- new system concepts that need major redevelopment (e.g., APS*)
- project timelines

The complexity and dependencies to integrate multiple applications on several RTEs with vendor multiplicity will be significantly higher than today in EULYNX. This must be taken into account for the business case, and it is hard to evaluate at this early stage if the desired reduction of the total cost of ownership (TCO) can be achieved.

Besides the obvious development efforts, there will be a significant impact on the overall processes regarding development, assessment, homologation, rollout (planning, procurement, installation, and maintenance), and integration including the related test and tooling environment. The respective change process to set up the necessary environment will require high priority and attention to ensure positive outcomes of the SIL4 Data Center architecture over its lifetime.

In section 6**,** the business aspects will be discussed in more detail.

3.1.5 Evaluation of objective No. 5 – Vendor independence

Objective recapped: Different vendors shall be able to provide functional applications, computing platforms and development tools, respectively, without a vendor lock-in. It shall be possible to purchase hardware directly from different vendors throughout the lifespan of the software. The platform shall build on existing HW/SW solutions, stimulating competition among vendors and allowing them to shine with their specific expertise and distinctive solution features.

The modular architecture of the data center with standardized interfaces enables the involvement of several vendors. For detailed information regarding vendor multiplicity and vendor independence, see section 6.4.

3.1.6 Evaluation of objective No. 6 – industrial readiness

Objective recapped: It shall be possible to procure a platform as off-the-shelf solution supported by an open and dynamic market. The solutions shall be mature (e.g., reliability proven in field) and backed by effective acceptance and integrated logistical support (e.g., maintenance service, tooling, availability of spare parts).

The hardware components of the generic computing platform can be commercial-off-the-shelf solutions, due to a huge amount of other use cases in other domains using the same industrial COTS hardware technology. On the other hand, RTEs and also applications for safety-relevant rail applications as “off-the-shelf solution supported by an open and dynamic market” may be more challenging.

The knowledge, competence and experience which is needed to provide such an RTE for SIL4 applications running on COTS hardware is only available in a few companies. Even with standardized interfaces of an RTE, the RTE solutions themselves will even in the future be a very "domain-specific" topic. Additionally, the market for installations of such RTE solutions is not comparable to other industrial use cases (with an exponentially higher number of installed solutions). Therefore, it will not be possible to get in a "commercial-off-the-shelf" situation together with industrial readiness of those products.

3.1.7 Evaluation of objective No. 7 – Migratable and portable business logic

Objective recapped: The business logic is considered a significant system asset, being the component with the longest lifetime. It must hence be portable to different computing platform evolutions. We here further differentiate:

- *Migratability for legacy applications: It should be decently easy to migrate legacy applications to the new platform.*
- *Portability for new applications: Applications running on the platform should be portable to any other vendor’s or evolved version of the platform.*

The business logic is considered as a significant system asset because it is the component with the longest lifetime.

It must hence be portable to future and/or different computing platform evolutions.

We further differentiate between:

3.1.7.1 Migratability for legacy solutions

To migrate an existing legacy application onto the same computing nodes as an RCA installation, the legacy solution (= legacy application + legacy safety layer) needs to be adapted specifically to fulfill the basic common requirements for running a legacy installation within the same data center as RCA installations.

To that end, it is important to define the standardized interfaces and common requirements from the view of common data center handling to make even legacy solutions behave comparably.

For possible migration concepts of legacy applications, see section 4.6**.**

3.1.7.2 Portability for new applications

An application which is newly developed based on one specific RTE solution depends on this RTE solution regarding the following:

- the SRACS of the specific RTE solution
- the development tools (T3 tools for safety measures required by the specific RTE solution)
- the test environment for vertical integration of the application running on the specific RTE
- the application-related configuration data required by the specific RTE solution

All these aspects are basic conditions for application validation.

To that end, the porting of an application from RTE solution 1 to RTE solution 2 will require a non-trivial effort in terms of application adaptation, integration, and validation.

For details regarding the API of an RTE, see section 4.5.

3.1.8 Evaluation of objective No. 8 – System evolvability

Objective recapped: The platform shall be open to extensions (in the sense of additional system services that are added over time, e.g., related to FRMCS). Adding new functionalities shall be possible with minimal to no changes to existing applications (though these may naturally not be able to leverage the new functionalities).

Regarding "openness to extensions", one should differentiate whether the extensions relate to safety-relevant functionality or not:

- **Safety-relevant extensions** (e.g., new safe communication protocols) are likely tightly bound to the RTE and its internal interfaces. "Openness" can here be reflected in the RTE architecture, e.g., regarding a suitable abstraction between communication protocols and other RTE functions but can of course only be leveraged by the RTE vendor, who would have to provide each extension themselves. "Openness" beyond vendors would, at least for safety-related functions, require the publication of the complete internal architecture of the RTE, which is naturally undesirable.
- **Non-safety-relevant** extensions (e.g., FRMCS protocols and services) can likely be implemented separately from the RTE and can hence evolve independently from the RTE.

3.1.9 Evaluation of objective No. 9 – Facilitation of application development

Objective recapped: The platform shall use an open, well-documented application model and programming interface, facilitating that third parties develop applications.

"Open and well-documented" does not mean "standardized" per se.

For the benefit of "harmonized tool chains for application development (for all RTE solutions in the same way)", the requirements for such tool chains need to be provided and clearly defined as a standard.

Otherwise, each RTE vendor will define and use their own solutions. This might be well-documented but will not become a standard.

Such RTE solution-specific tool chains would not be an obstacle for enabling third parties in developing applications, but the same application would have to be developed in different ways for the different RTE solutions.

It is hence recommended that tool chains for application development are standardized to a reasonable extent (see also section 6.5), specifically considering the large range of application needs that should be supported by such a standard.

3.1.10 Evaluation of objective No. 10 – Modularity

Objective recapped: The platform shall allow for a modular safety certification process, using pre-certified components leading to a dramatically simplified and shortened full system certification process. An evolution or update of the platform shall not require a new E2E homologation of application and platform, as detailed in Section 7.

The "modular safety certification process" itself is well known and established but will get very challenging when considering the question "who takes over the responsibility for the overall system" (also considering the integration of multiple application subsystems into an overall system, not only the integration of one application onto the computing platform).

Also, the complexity does not come from modularization, but from the different parties and roles involved in the integration and certification process:

- The split of a safety-relevant system into small pieces delivered by different parties leads to a split in safety responsibility with a missing definition of the overall responsibility for integration and safety.
- Each interface between safety-relevant parts needs to be integrated for safety certification.
- The SRACS of each individual application must be handled with an overall view, even if there is no definition of the overall responsibility for safety.
- RTE-specific SRACS of different RTE solutions must be handled by the safety applications (running on the RTE).

The split of a huge system into small pieces leads to the delivery of small pieces for which the overall responsibility for integration must be defined. Nevertheless, the complete system needs to fulfill the requirements for safety certification (e.g., EBA Bund Anlage 17 [5]).

The experience with EULYNX shows the unsolved situation for the overall certification of such a system provided by modular pieces (interlocking logic/object controllers) by different vendors.
This unsolved situation with the missing "cross-vendor responsibility" today shows the complexity even in a relatively simple modularization as defined by EULYNX for splitting a system into two types of subsystems (interlocking logic and object controllers).

The complexity of integration and certification for the modular approach of an RCA-based architecture is exponentially higher than the current situation in EULYNX.

3.1.11 Evaluation of objective No. 11 – Encapsulated, transparent fault tolerance mechanism.

Objective recapped: The platform shall transparently encapsulate the safety and fault tolerance mechanisms. Vendors may offer different (new) approaches to safety and fault tolerance as they become available on the market – solution agnostic and future-proof.

The basic safety principles (hardware separation, diversity, redundancy, voting) are established for decades and used/realized in different combinations and variants by all rail companies. In this context, it is seen as unlikely that entirely novel fault tolerance approaches will be introduced.

It is important to stress that the specific details of such safety approaches will not “become available on the market”, because of the following reasons:

- It is “core knowledge” of the inventor of the RTE solution (and very often protected by patents), that is treated as confidential know how.
- It is a core part of the safety certification, and therefore cannot be easily exchanged.

Overall, it is regarded as possible that the platform transparently encapsulates the safety and fault tolerance mechanisms, and this would also allow that the detailed mechanisms remain proprietary to the RTE vendor.

3.1.12 Evaluation of objective No. 12 – Scalability

Objective recapped: The platform shall be highly scalable, i.e., it should by design be able to support an arbitrary number of applications and arbitrary number of compute nodes.

Scalability can be distinguished in different steps:

- static: foreseen in the architectural design (e.g., avoiding fixed limits), but scaling up needs new integration and validation
- dynamic: possible scalability at runtime, e.g., based on the current processing or memory load

The static scalability that avoids fixed limits can be fulfilled. From a very simple point of view, each application is running (in replicas) on its own RTE and the amount of RTEs required for an installation depends on the number of applications.

The most important technical requirements to run "an arbitrary number of applications" on the RTE will include aspects about:

- time-critical dependencies between different applications belonging to one installation
- bundling of different applications within the same replica
- performance-relevant quantity amounts (number of incoming/outgoing messages per application cycle), maximum allowed reaction time of each application

This input is not yet defined in detail in the current RCA stage.

The dynamic scalability at runtime is discussed in more detail in section 3.1.13.

3.1.13 Evaluation of objective No. 13 – Flexible usage of compute resources

Objective recapped: The platform shall enable a flexible mapping of business logic to compute resources (e.g., such that the platform can be expanded while applications are running, and that business logic can be re-mapped when compute nodes fail). It shall be able to leverage advances in computing technology (i.e., when better compute nodes are available, it shall be possible to assign more instances of business logic to the compute nodes).

The usage of separate compute resources (e.g., three separate hardware computing nodes for the 2oo3 principle) is a basic rule for safety argumentation in safety layers.

So far, the individual parts of a safety layer are mapped to the used hardware computing resource in a static way, e.g., by defined configuration data for the safety layer.

This configuration data is part of the validation of the system and cannot be changed without re-validating this data.

In this way, “static” does not mean “inflexible” – the static configuration can naturally be changed and re-validated.

This means that flexibility is provided, but in a static way, and it is not changed during the runtime of the system.

For a “dynamic flexibility during runtime” in the usage of compute resources, the following is necessary:

- The responsibility for fulfillment of the safety conditions (e.g., safety layer is running as 2oo3 on three separate hardware computing elements) must be defined.

- The responsibility for fulfillment of the availability conditions (e.g., new computing resources shall provide the needed CPU power, performance, and time behavior for the needed workload) must be defined.
- The responsibility for the fulfillments of installation rules from the view of installation consistency must be defined (e.g., installation of the correct parts of safety layer and application onto the new computing resources used).
- The network infrastructure must be considered to enable dynamically added new computing resources to communicate to connected systems. To that end, network components such as switches/routers must be reconfigured.

Such a dynamic reconfiguration or orchestration of computing resources would fall within the responsibility of the system's virtualization layer and be part of the safety case. On the other hand, the virtualization layer should not belong to the safety-relevant part due to it is complexity and flexibility in use.

To automatically solve such a dynamic flexibility during runtime of a safety-related system by the computing platform itself is not easy to achieve in today’s view due to safety- and availability- related aspects which need to be reliably met for SIL4. For this, see also section 8.8.

3.1.14 Evaluation of objective No. 14 – Centralization

Objective recapped: The platform shall allow to centralize applications physically in a safe data center to simplify lifecycle management, reduce TCO by means of simplified, optimized operations, and benefit from increased availability and optimized resource usage.

Centralization does not directly depend on the usage of a new RTE and COTS-based hardware, and in principle the centralization of SIL4 logics is already possible today with the EULYNX architecture.

An additional benefit of centralization is the "unification" of the SIL4 products, which means they run together within a centralized SIL4 Data Center on the same standardized COTS hardware.

On the other hand, a failure of one data center would have an unacceptably high impact. Therefore, centralization inherently imposes the need to have geographically distributed, redundant data centers.

To gain this benefit of "unification", it is important to:

- define a common system architecture with standardized interfaces between all involved parts (rail products and infrastructure parts as IT security, installation/update, and diagnostics)
- define the parts that can be commonly used as a basic layer for all applications and run on the same hardware. These are e.g., virtualization and load operating system (Load OS).
- define the standardized infrastructure elements for IT security, installation/update, and diagnostics

For details, see section 4.

3.1.15 Evaluation of objective No. 15 – Support for running multiple applications

Objective recapped: It shall be possible to run multiple applications, possibly with different SIL levels, on a single physical platform to reduce cost, space, power dissipation, etc., and simplify certification, maintenance, system evolution, etc.

There are three different situations to be distinguished:

- multiple applications with the same SIL level (SIL4 only): Each application runs either on one common RTE instance, or on their own, separate RTE instances.
- multiple applications with different SIL levels (SIL4 + less than SIL4): Applications with different SIL levels should run on separate RTE instances. The SIL level of the RTE instance could be adjusted to the SIL level of the respective application.

- mixture of SIL4 and non-SIL applications: The non-SIL application can run without any RTE, while the SIL4 application runs strictly separated on their RTE.

The safety concept of each solution shall completely cover all aspects to achieve the necessary SIL level in their own responsibility for their own scope, independently of any other software that runs on the same hardware.

This also implies that non-SIL applications that shall run on an RTE will have extensive additional development (if practicable at all) and validation effort because they also need to fulfill the safety requirements imposed by the safety layers.

This topic will be further elaborated in section 7.

3.1.16 Evaluation of objective No. "New" – Native support of geographical redundancy

Objective recapped: The platform shall support that the instances of individual application functions are geographically distributed for the purpose of highest availability and resilience.

Geographical redundancy supports the centralization on fewer data centers while fulfilling availability constraints even in catastrophic situations (e.g., flooding, earthquakes).

The basic redundancy concept for "geographical redundancy" (by distributing the computing nodes involved onto different geographical sites) can be achieved by an RTE or by a legacy safety layer.

Further information can be found in section 4.11.7.**Fehler! Verweisquelle konnte nicht gefunden werden.**

3.2 EULYNX-related requirements

In this section, the specific requirements of the EULYNX project relevant for a data center are introduced. First, it should be mentioned that the EULYNX requirements focus on the behavior, the interfaces, and the non-functional requirements of the subsystems.

3.2.1 Communication interfaces SCI*, SDI*, and SMI*

The communication interfaces SCI*, SDI*, and SMI* as defined by EULYNX shall be additionally supported by the RTEs and by the legacy solutions running in the data center.

With this capability, a RTE or legacy solution could be part of a safety-relevant solution in this context. But there might exist realizations of the Maintenance and Data Management (MDM) EULYNX subsystem that run in a non-safety-related context on top of the virtualization layer (i.e., without an RTE). These would only need to implement the relevant diagnostic and management interfaces for this type of subsystem, namely the SDI* and SMI*.

To that end, it shall not make a difference if the communication partner is running within the same data center or located at any other place outside of the data center.

3.2.2 Upcoming EULYNX standardization for IT security, installation and update

If EULYNX also defines in the future standards for cross-functionalities such as IT security and installation/update with the associated infrastructure elements, then this shall be defined according to the solution which is needed within a data center.

The new aspect (compared to the EULYNX architecture today) would be that all the processes and interfaces include the aspect of "an arbitrary number of applications is running on RTEs together on same hardware". In EULYNX, the situation today is "one product is running on one hardware device (without mixture of multiple applications and safety layers on the same hardware)."

3.2.3 Common requirements regarding geographical redundancy

The requirements and architecture for the highest level of availability in case of a high grade of centralization shall be defined by EULYNX according to the solution for rail products running within a data center. That could be considered to reach the highest level of availability for relevant stations or lines.

3.2.4 Cross-vendor integration and overall responsibility to solve

To date, the topic "cross-vendor integration and overall responsibility" has not been implemented in a real-world installation for the newly defined standard interfaces, e.g., for the interfaces between a centralized interlocking logic and decentralized object controllers. On the PoC level, some very successful tests have been performed that demonstrated the correctness of the technical approach.

Achieving the objective of "vendor independence" is therefore challenging.

For additional details, see section 5.7.

3.3 Security requirements

3.3.1 Legal requirements

The NIS directive [6] and the Cybersecurity Act of the European Union [7] are driving the security requirements for critical infrastructure as rail automation.

The NIS Directive is implemented in national law in the EU member states (as the “IT-Sicherheitsgesetz” in Germany). These laws require that operators of critical infrastructure such as rail automation systems set up an IT security management system (ISMS, e.g., as defined in ISO 27001), conduct regular external audits, report incidents, and implement state-of-the-art security.

3.3.2 EU harmonization

Over a period of four years, the cybersecurity working group of the EU research project Shift2Rail has created a consensus for IT security interoperability among the key European stakeholder (operators and solution providers) for rail automation systems.

The first consensus was to use the international industrial security standards of IEC 62443 [10]. All books were analyzed and found to be fully applicable to rail automation and not in conflict with existing CENELEC standards in this domain.

The outcome of the high-level and dozens of detailed risk assessments was that security level 3 is required for an acceptable risk for all major zones of the rail automation system. Security level 2 is required for external zones (behind a "data diode") as external diagnostics.

In early 2021, the cybersecurity working group released a set of work products that are the foundation of interoperable security according to IEC 62443-3-3 SL3. The documents define a generic security architecture and three protection profiles with technical requirements: for trackside components, on-board components, and FRMCS/ACS components.

The output of X2Rail cybersecurity workgroup of Shift2Rail is made available for further detailing for standardization groups as UNISIG, ERA and EULYNX.

3.3.3 Security requirements for components within the SIL4 Data Center

Security requirements shall be derived from existing international standards as far as possible (e.g., IEC 62443, CENELEC TS 50701).

The components (e.g., physical devices with data processing and communication interfaces) shall implement the requirements stated in the X2Rail-3 protection profile "trackside components".

If components are used on-board, the X2Rail-3 protection profile "on-board components" shall be used.

The CSA Cloud Controls Matrix (CCM) [8] is a cybersecurity control framework for cloud computing, which is composed of 197 control objectives that are structured in 17 domains covering all key aspects of cloud technology. It can be used as a tool for the systematic assessment of a cloud implementation and provides guidance on which security controls should be implemented by which actor within the cloud supply chain.

3.3.4 Organizational compliance

Information security policies and guidelines in organizations need to be met in the form of processes adopted by organization of a CISO. There could be different governance models (e.g., three lines of defense) which affect the development process. Roles and responsibilities are necessary to be addressed in early stages of project for the matter of risk assessment, risk communication, risk treatment, and risk acceptance.

4 Architecture

4.1 Basic architecture

4.1.1 High-level system architecture

The computing platform is based on the general system architecture that has been proposed in the RCA Architecture [1].

The main differences are in nomenclature due to a slightly different viewpoint on the architecture:

- The hardware together with the load operating system and the virtualization layer form the **Generic Computing Platform**, which can be used for safe and non-safe applications.
- The Generic Computing Platform together with the Runtime Environment (RTE) form the **safe computing platform**, on which the safe applications are run.
- The virtualization layer is considered a separate top layer of the Generic Computing Platform. In contrast to the current RCA/OCORA architecture, it is not seen as part of RTE, since it is more related to the data center infrastructure, hosting multiple different RTEs and legacy systems. The virtualization layer might be sourced as an independent software solution.

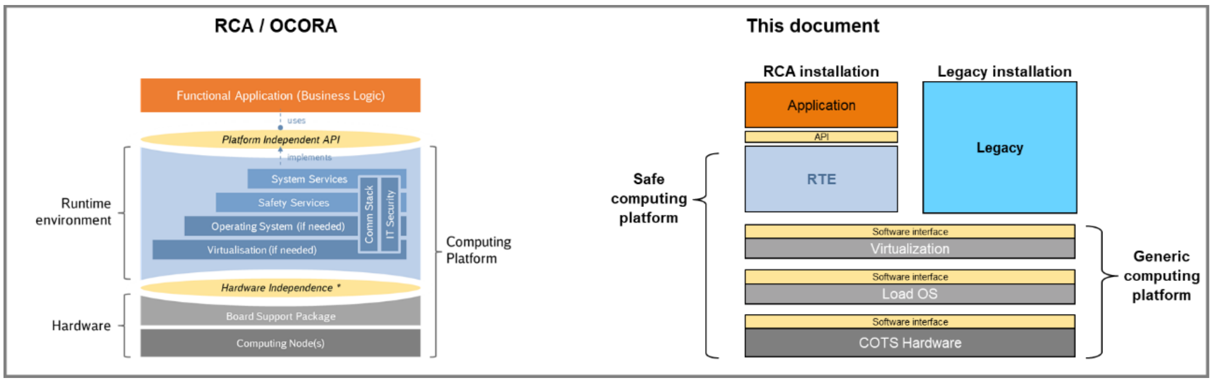


Figure 2 Computing platform

It should be stressed that, unlike the point of view in the RCA/OCORA White Paper, it has been noticed in the collaboration among SMO and DB that at least for SIL4 Data Center deployments there would be a strong merit in also standardizing interfaces between RTE, virtualization, load operating system and COTS hardware, as shown in Figure 2.

4.1.2 Basic RTE architecture

The basic RTE architecture is proposed as follows:

- A safety-relevant application APP-1 is running in at least two application replicas APP-1[1] and APP-1[2] in parallel (synchronously) as cyclic processes (see section 4.1.3). For increased availability, the instantiation of additional application replicas shall be possible. The number of replicas and the distribution of the replicas on the hardware computing nodes depend on the SIL requirements of the application and the safety concept of the RTE, e.g., if the 2oo3 or 2x2oo2 principle is used. The RTE and application replicas are running distributed on separate hardware computing nodes (see section 4.10).
- All replicas of an application are running with the same application software.
- The outputs of the individual replicas APP-1[1] and APP-1[2] are compared by a safe voter of the RTE.
- After successful voting, the RTE-internal safe message is handed over to a protocol gateway of the RTE, which forwards the data via the defined communication protocol to the connected systems.

- A safe clock component of the RTE creates the triggers for the cyclic working of the application replicas.

All RTE implementations (provided by different vendors) shall be agnostic to the application logic regarding:

- the safety concept and redundancy concept (local and geographical redundancy) of their own running installation
- the safe, redundant, and secure communication to the connected systems

The RTE design shall support the required flexibility to ensure the following:

- system evolvability, e.g., to add additional protocol gateways (as their own architecture elements) for further communication protocols
- combined usage of different communication protocols (each provided by a separate, protocol-specific gateway)

Figure 3 below shows an application APP-1 running as two replicas [1/2]. The communication to the connected system(s) is done with redundant communication, i.e., two transport channels [A/B]

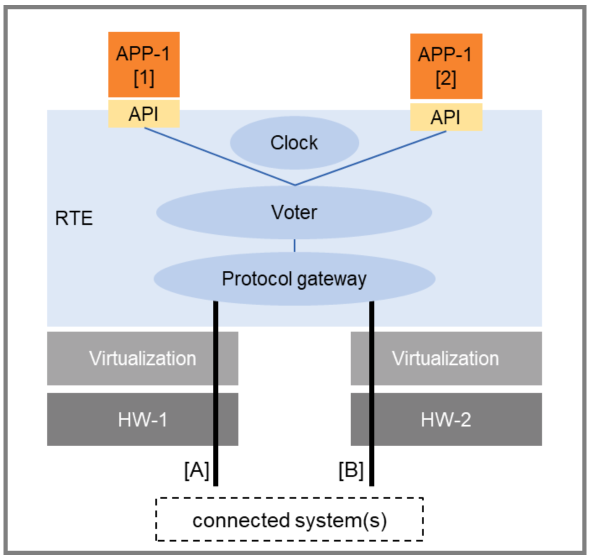


Figure 3 Basic RTE architecture

The connected system can be installed on the same computing nodes or on other computing nodes, it can be an application within the same installation or an application within another installation.

4.1.3 Parallel synchronous cyclic application replicas

For application replicas which are running distributed on several computing nodes, a synchronized cyclic working principle is recommended to achieve a synchronously running system as a basis for stable voting of the outputs created by the replicas running in parallel.

In an IP-based architecture – considering geographical redundancy – network latency times and jitter effects in the behavior of the running software components need to be tolerated by the RTE in a way that achieves a stably running system.

Therefore, it is not possible from today's perspective to use application replica cycle times in case of very small time ranges as e.g., < 100ms. The specific limit for the lowest possible cycle time depends on the network behavior and on the technical solution of the RTE in handling of such influences, such as jittering effects and network latency times.

The trigger for the synchronous start of work of all application replicas must be created by a safe monotonous clock. This trigger needs to be created for all replicas at the same time.

The cycle time of an application is used for all associated replicas and shall be configurable for each application. That depends on:

- the functionality of the application (e.g., driven by such safety-relevant requirements as “safe reaction within time x“)
- the message processing time of the application
  This depends very much on the number of incoming messages per time, e.g., driven by the size of the controlled area of the installation (e.g., small interlocking, large interlocking).

The defined cycle time shall consider even worst-case scenarios in terms of performance, time behavior and message processing of the application.

Cyclic working means that all messages to be processed by the application are buffered until the next working cycle begins. At the start of the working cycle, all buffered messages are considered for processing by the application.

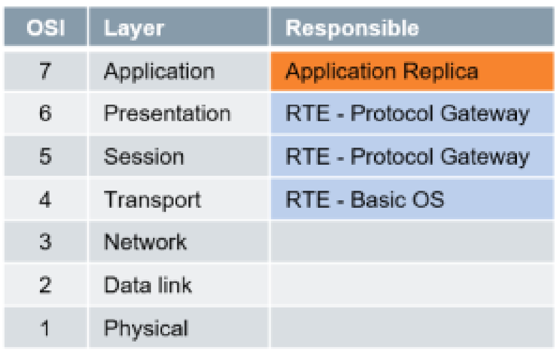
This message buffering leads to "delay" times in the processing of the buffered messages. As an average value, the delay time is half the defined cycle time.

4.1.4 Protocol gateways

The application (running as several replicas on the RTE) shall be abstracted from the communication protocols which are required for communication to the connected systems:

- The data transfer to the connected systems shall be provided by protocol gateways of the RTE.
- A protocol gateway shall handle all required safety and redundancy aspects (for safe and redundant communication), transparent for the application.
- The content/payload of messages exchanged end-to-end between applications shall be completely transparent for the protocol gateway of the RTE.
- Each communication protocol for communication to other systems provided by other vendors needs to be defined as a vendor-independent standard (e.g., SCI*).
- The internal RTE communication between the application replicas and the RTE components (clock, voter, protocol gateway) is defined by the RTE itself; this safety relevant internal RTE communication is a basic part of the safety concept of the RTE.
- The application receives and sends content/payload without knowledge about the communication protocol.
- The RTE architecture shall enable an easy extension regarding additional communication protocols by additional protocol gateways.
- For each application, it shall be possible to decide by application specific RTE configuration about the usage of the specific one or more communication protocol gateways.

The following table shows the corresponding (responsible) part within an RCA-based installation for the individual OSI layers:



OSI	Layer	Responsible
7	Application	Application Replica
6	Presentation	RTE - Protocol Gateway
5	Session	RTE - Protocol Gateway
4	Transport	RTE - Basic OS
3	Network	
2	Data link	
1	Physical	

Table 1 Responsible parts for the individual OSI layers

Each safety-relevant protocol needs to be realized by the RTE, e.g., as a "**safe protocol gateway**" as part of the safety layer of the RTE.

Such a safe protocol gateway belongs technically close to the RTE because it needs a safety layer to be a "safe" protocol and it needs to participate on the RTE internal communication to receive messages from the internal RTE voting and send messages to the internal RTE application replicas.

A safe protocol gateway of an RTE is responsible for safe and redundant communication. It is one of the safe "endpoints" in terms of safety-relevant communication.

The communication stack and IT security mechanism shall be realized within the basic operating system (Basic OS) of the RTE. The communication and security functions shall be provided to the safe protocol gateways via standard application programming interfaces (e.g., network socket API). This keeps the safe protocol gateways independent from the communication and security functions and follows the principle of separating safety and security.

For solutions to transfer safety-relevant communication via non-safety-relevant protocols, it is possible to use non-safety-relevant communication protocols independently of the RTE. These non-safety-relevant protocols are handled as a "black channel".

The Figure 4 below shows an example of how safety-relevant protocols could be provided by the RTE and non-safety-relevant (example here: FRMCS) protocols could be provided as a "service function" by an independent vendor. The application APP-1 is running in three replicas [1/2/3]. Each application replica creates the application payload for communication with its counterpart at the communication endpoint on the far right (not shown in the figure) and provides this payload via the API. The output of the application replicas is processed by the voter and then handed over to the protocol gateway of the RTE.

The RTE ultimately also provides the lower layer protocol stack (for instance TLS/TCP/IP) for communication with the service function residing outside the RTE. The service function decodes the TLS/TCP/IP, adds the "FRMCS layer", and provides the TLS/TCP/IP layers underneath again, before the communication leaves the service function.

It should be noted that Figure 4 is highly simplified, and more complex constellations are conceivable. For instance, there could be multiple instances of service functions (e.g., related to independent communication paths). In addition, the figure only shows the application user plane and omits the fact that the so-called "FRMCS Service Client" that would ideally be provided by the service function also has to establish a control plane connection to a so-called "FRMCS Service Server", which is not shown in the figure. Additionally, the redundancy concept for such a communication needs to be defined to achieve a highly available system even in the case of a hardware failure.

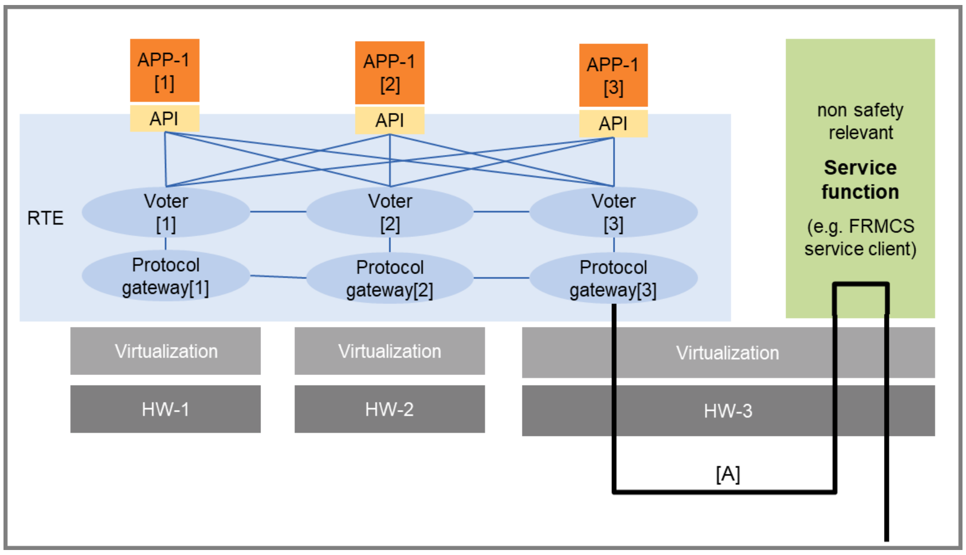


Figure 4 Architecture for connection of a non safety relevant service function

Figure 5 below shows how the individual parts (application, RTE protocol gateway, service function) on both sides of the end-to-end communication are involved in communication.

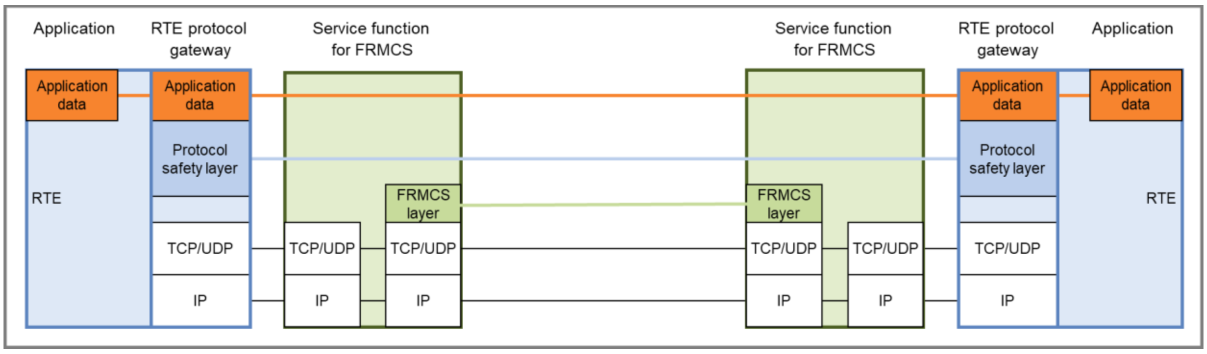


Figure 5 Example: FRMCS protocol as non safety relevant service function

4.2 Flexibility in SIL4 Data Centers

Figure 6 below shows the principle of how the required flexibility in the architecture for several installations with different SIL levels running in the same data center on the same hardware computing nodes can be realized.

To achieve the goal of "centralization of multiple installations (provided by different vendors) together on the same hardware computing nodes", a unified solution for virtualization within one data center is necessary, i.e., every installation is running in one or multiple virtual machine(s).

It shall be possible to migrate existing legacy applications (e.g., today's business logic for DB's digital interlockings (DSTW)) into the same data center as newly developed applications.

To that end, it is necessary to define the common basic requirements, which shall be fulfilled by both RCA and legacy installation types in the same way to achieve a common behavior in the data center.

Example in the figure below:

- The applications APP-1 and APP-2 are running as an application bundle on RTE-1 with the 2oo3 principle in three instances [1/2/3] distributed on three hardware computing nodes. For the aspect "bundling of applications", see section 4.4.4.
- The application APP-3 is running on RTE-2 with the 2oo3 principle in three instances [1/2/3] distributed on three hardware computing nodes.

- The legacy installation Legacy Inst-1 is running with the 2oo3 principle in three instances [1/2/3] distributed on three hardware computing nodes.
  For details, see also section 4.6.
- On each hardware computing node, diagnostic software is running to provide diagnostic data related to the underlying parts, such as virtualization and COTS hardware.

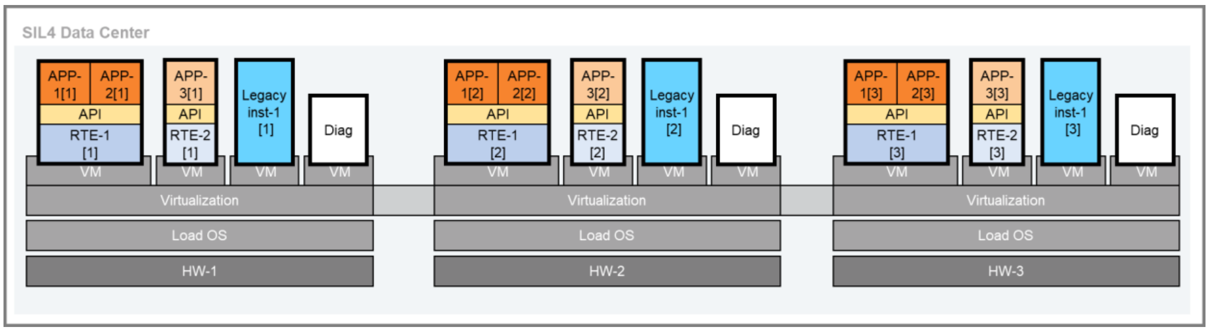


Figure 6 Scalability: several different installations running on the same computing nodes

To integrate several installations running with different RTE solutions, each in their own virtual machine, the requirements for the virtualization solution need to be defined, and each safety layer (RTE or legacy) needs to include those requirements in their own safety concept. These requirements for the virtualization shall cover the needs of all safety layers because a safety layer must not have "stronger" requirements regarding the behavior (and quality of behavior) of the virtualization solution.

An open point is where to place the hardware abstraction. On the one hand, it is appreciable to have the hardware abstraction implemented in the virtualization layer as much as possible. On the other hand, this would lead to a full emulation of the hardware, which is not practicable. So, further investigations should be made regarding which hardware aspects should be abstracted in the virtualization layer and which in the RTE.

The solution for the virtualization must be suitable for the applications' requirements and their required time behavior, which to date has not been extensively analyzed.

Another challenge is the orchestration of the virtualization, i.e., which installation will run on which processors and gets which system resources (memory, interface). To that end, a separate orchestration solution will be necessary to orchestrate and configure the virtualization layer. This should be investigated further in more detail.

4.3 Support of different lifecycles for products and sub-products

To flexibly handle individual parts with different lifecycles, these parts shall be handled as separate sub-products with modular validation/assessment/approval (depending on the SIL level of the sub-product).

To that end, especially as part of minimizing maintenance efforts, it is essential to separate the non-safety-relevant parts (e.g., for IT security) in the best way possible from the safety-relevant parts (e.g., safety layer).

Figure 7 below shows the principle of separating the products into sub-products, according to the related SIL levels and types (specific/generic).

Interfaces between the sub-products themselves are defined by and the responsibility of the product vendor.

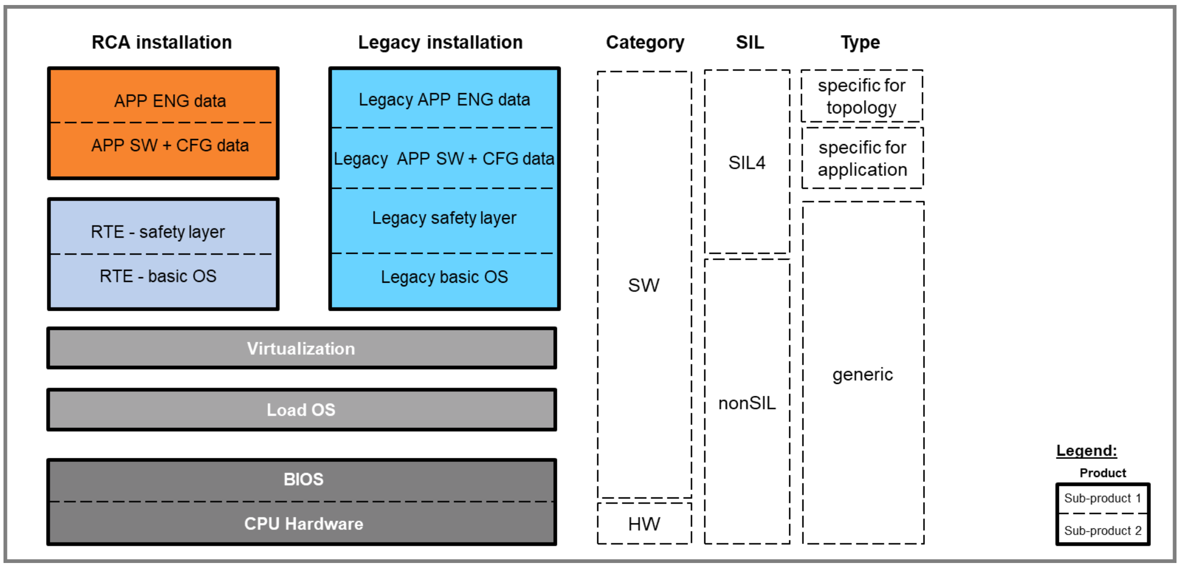


Figure 7 Products and sub-products

4.4 Architecture within an RCA installation

Each individual APS* of an RCA-based installation (e.g., system with the functionality "Frankfurt interlocking") is running on its own RTE (with its own voter, clock, protocol gateway). Different APS* may run on different RTE solutions from different vendors.

The communication between the individual APS* within the same installation is done by the involved RTEs that the APS* are running on (except for the application layer, which is transparent to the RTEs).

Note:
It is expected that RCA will specify (or at least recommend) suitable safe communication protocols for communication between different APS*.

4.4.1 RCA installation with different RTE solutions for APS*

Figure 8 below shows four applications (APP-1/2/3/4), each running on an RTE from a different vendor (RTE-1/2/3/4).

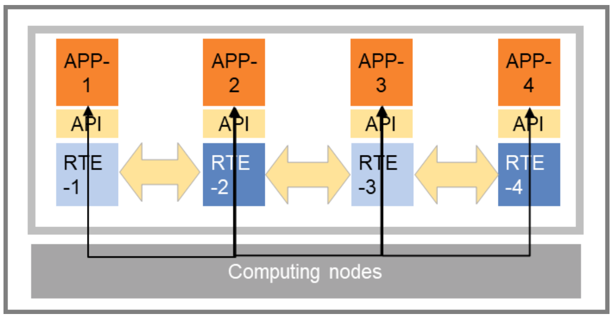


Figure 8 Installation with applications each on own RTE

In such an installation, the RTEs would apply the safe communication protocol for communication between different APS* that is to be specified/recommended by RCA.

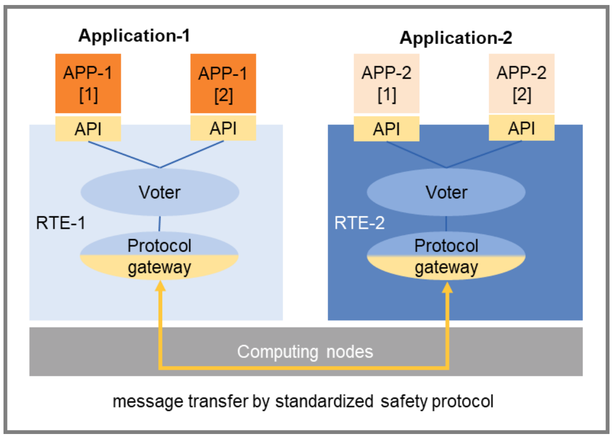


Figure 9 Inter-application communication by different RTE

If multiple, different RTE solutions were used in one installation, the integrity check by the RTEs (see section 4.10.7) cannot cover the overall view of the installation. To that end, it must be evaluated in the overall safety case how to handle the issue of "overall integrity of an installation with different RTEs".

Note:
The existing standard protocols for communication between systems (e.g., RaSTA protocol for all the SCI* interfaces of EULYNX) are designed for client-server-like communications. RaSTA is explicitly not recommended for use in system internal communication due to missing flexibility as peer-to-peer protocol and restriction regarding non-SIL participants. Therefore, another protocol must be found in RCA or designed for the purpose of safe communication between RTEs.

Different RTE solutions (e.g., with 2oo3 principle, 2x2oo2 principle) lead to different distributions onto the hardware computing nodes.

Figure 10 below shows exemplary the different usage of the hardware computing nodes:

- APP-1 is running on RTE-1 with the 2oo3 principle: three instances [1/2/3] on three hardware computing nodes HW-1/2/3 with redundant communication [A/B] via HW-1 and HW-2
- APP-2 is running on RTE-2 with the 2x2oo2 principle: four instances [1/2/3/4] on four hardware computing nodes HW-1/2/3/4 with redundant communication [A/B] via HW-1 and HW-4

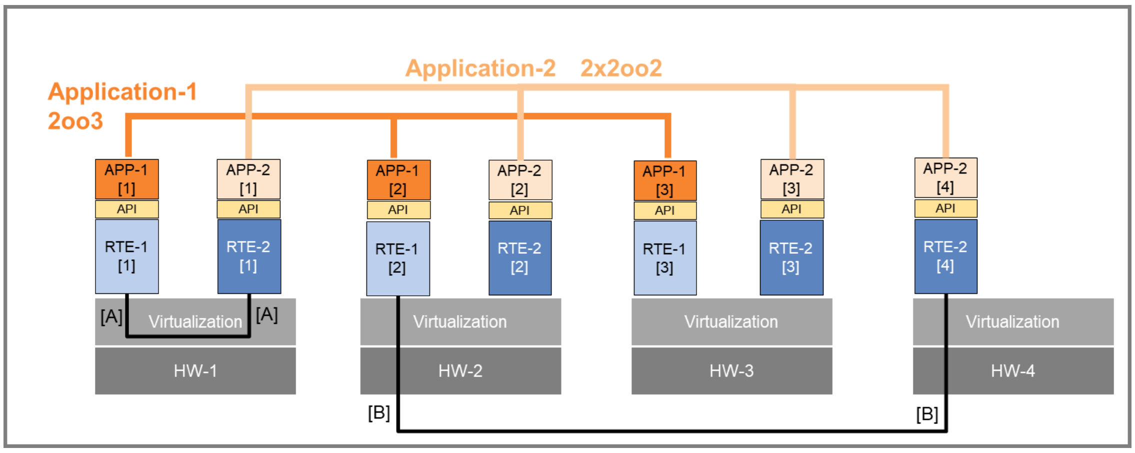


Figure 10 Different usage of hardware computing nodes

4.4.2 RCA installation with the same RTE solution for all APS*

Figure 11 below shows four applications (APP-1/2/3/4), each running on an instance of the same RTE solution.

In this case, the RTE vendor may use the safe communication protocols specified/recommended by RCA for communication between different APS* subsystems, as in the previous case. Alternatively, the vendor may also apply proprietary safe communication protocols, if – for instance – this is superior from a performance, timing, or efficiency perspective.

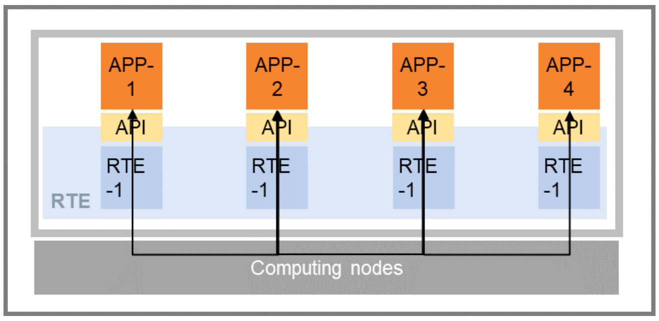


Figure 11 Installation with applications on same RTE

For this installation, communication for the data exchange between applications is achieved using an RTE-specific message transfer (tunneling gateway) for communication between the individual RTEs.

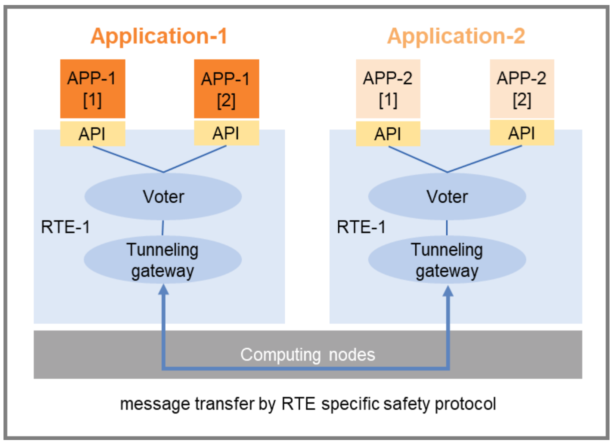


Figure 12 Inter-application communication within same RTE

4.4.3 Communication between applications

For communication between applications, it must be considered that message distribution delays are caused by fixed work cycles of application scheduling schemes. For the aspect "message delay of incoming messages", it does not matter if both applications are running on the same hardware computing nodes or distributed. The root cause for the delay is the cyclic execution of the applications, as messages can be processed only in the next application cycle, independent of the network latency.

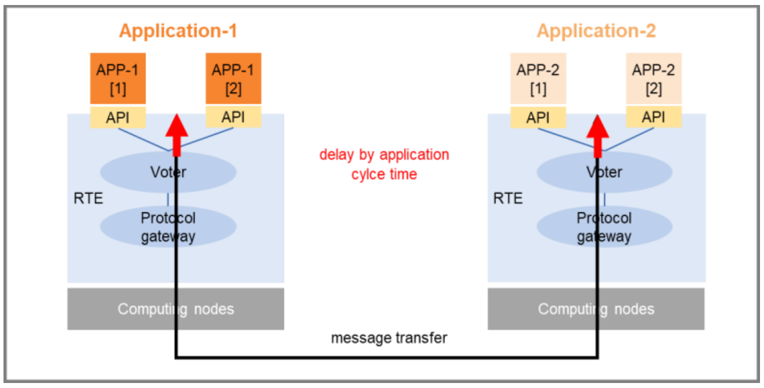


Figure 13 Inter-application communication

4.4.4 Bundling of applications for time-critical inter-application communication

Therefore, for very time-critical inter-application communication, it is necessary to bundle these applications within the same application replica. The goal of this application bundling is that the time-critical inter-application communication does not need to go through the voting mechanism, but rather stays in the bundle within the different replicas.

For this application bundling, the RTE must provide additional services called "Application Manager" (AppMan). This AppMan within the RTE shall provide all the required services to run several time-critical applications within same application replica.

<u>Example:</u>
The following Figure 14 shows two applications (APP-1 and APP-2) running as a bundle in each replica.
Incoming messages are distributed to both replicas and are forwarded within the bundle to APP-2. APP-1 and APP-2 communicate within the bundle, and APP-1 creates the output which is forwarded to the voter.
The bundle-internal, inter-application communication is not visible to the voter.

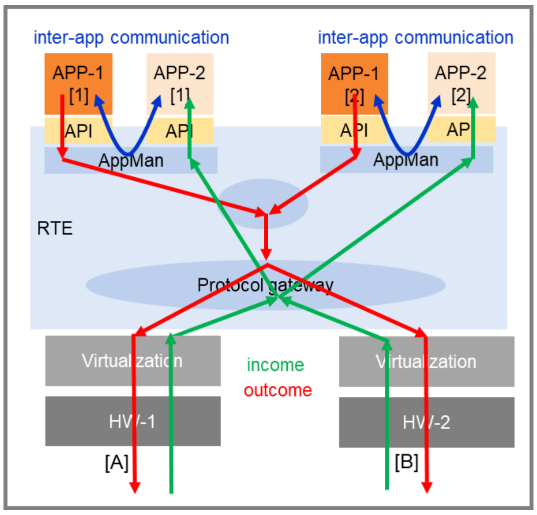


Figure 14 Bundling of applications

The required services to handle the cross-dependencies for interaction between bundled applications depend on:

- time behavior of the individual applications
- time-critical interactions between bundled applications
- time behavior of the complete bundle (= complete replica), cycle time (worst-case scenarios!)
- application priorities (in the start-up phase, in running-mode)
- message priorities for interactions within the bundle
- message priorities for incoming and outgoing messages
- handling of failure scenarios (only a completely running bundle is useful: how to handle partial failures of individual parts of the bundle)

Specific details about these application-related interdependencies were not included in the scope of this project and require additional investigation.

4.4.5 Application communication to connected rail systems

Other rail systems (e.g., object controllers) need to be connected to exactly one APS application. But it is possible that several other rail systems are connected to the same APS application.

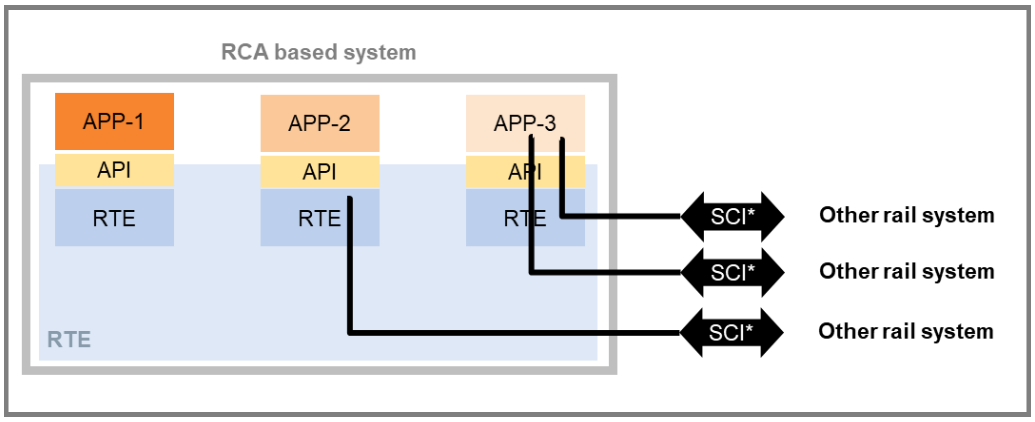


Figure 15 Communication to connected rail systems

4.5 API of the RTE

The API should be designed in a way that applications can be ported from one RTE implementation to the implementation of another vendor without the need of redevelopment. Nevertheless, it will need a recompilation, reintegration, and adaption of the SRACs to the new RTE. Also, the test cases need to be reconsidered and possibly updated.

The following parts of the API very closely depend on the specific safety solution of the RTE (which is not defined in detail for RTE solutions):

4.5.1 Specific safety measures for the application software running as application replicas

Dedicated safety mechanisms need to be provided and implemented by the application replica that are executed at runtime.

The specific safety mechanisms depend on the specific safety concept of the RTE solution and do not directly affect the logic of the application. Such safety mechanisms may be e.g., memory checks, self-checks, generation of hash values over processed activities.

Such safety mechanisms need to be added to the application source code e.g., by RTE-related tools (category T3) to fulfill the associated SRACS of the RTE.

These required safety mechanisms will be different for each RTE solution.

4.5.2 Specific development tools for application development

Each RTE solution will require the use of dedicated tools for the development of an application (e.g., compiler, tool for adding safety mechanisms to the source code).

4.5.3 Specific test kit for application development and validation

Each RTE solution will provide specific RTE parts as test kit for testing of the application.

This test kit provides means of support for simpler testing and original RTE parts for the vertical integration of the application onto the original RTE.

4.5.4 Specific CFG and ENG data for RTE usage by an application

Each RTE solution will define specific configuration (CFG) data (for the generic use of the RTE) and engineering (ENG) data (for the specific use of the RTE in specific installations).

This safety-relevant data depends on the RTE solution and needs to be set by the application (as the RTE user).

Examples of such data:

- RTE configuration data for the specific running (2oo3, 2x2oo2, etc.) mode on the hardware computing nodes
- RTE configuration data for the required optional RTE parts, such as communication protocol gateways
- RTE configuration data for the specific application (e.g., application-related cycle time)
- RTE engineering data for the specific communication connections (to the connected systems)

4.5.5 Specific data handling tools for safety relevant RTE CFG and ENG data

RTE-related tools may be helpful for handling safety-relevant CFG and ENG data.

The data format of this data is defined by the RTE itself, though it would be helpful to standardize it to a certain extent for improved portability of applications across RTE solutions.

To optimally handle such data, the following may be useful:

- performing the necessary checks (syntax, consistency) offline using a dedicated tool to avoid checks on the RTE
- converting such data offline from a readable format (e.g., text file) into a binary format to avoid "text parsing" on the RTE

The toolchain and process for safe conversion from a readable format (e.g., text file) into a binary format requires validation.

4.5.6 Specific SRACS of the RTE

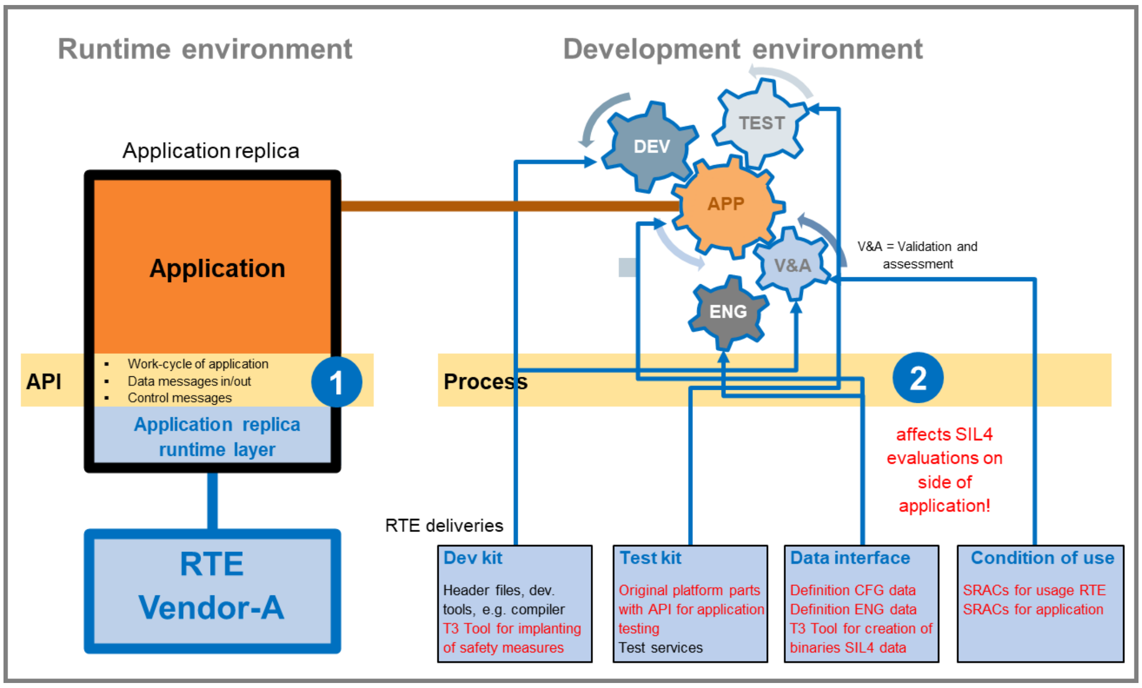
Each RTE solution will provide specific SRACS.

So far, SRACs are highly dependent on the specific RTE solution. To enable portability of applications across different RTE implementations, without major redevelopment efforts, it would be desirable to standardize SRACs to some extent. This is currently seen as challenging and requires additional investigation.

4.5.7 RTE deliveries to the application

The following Figure 16 shows the different kinds of "RTE deliveries" which shall be provided by each RTE vendor to the application vendor.

These deliveries depend on the specific RTE solution and are required for development, integration, and validation of the application.



Runtime environment
Development environment
Application replica
Application
API
Work-cycle of application
Data messages in/out
Control messages
1
Application replica runtime layer
RTE Vendor-A
DEV
TEST
APP
V&A
ENG
V&A = Validation and assessment
Process
2
affects SIL4 evaluations on side of application!
RTE deliveries
Dev kit
Header files, dev. tools, e.g. compiler
T3 Tool for implanting of safety measures
Test kit
Original platform parts with API for application testing
Test services
Data interface
Definition CFG data
Definition ENG data
T3 Tool for creation of binaries SIL4 data
Condition of use
SRACs for usage RTE
SRACs for application

Figure 16 RTE solution specific parts of API

(1) = operative API within the RTE

(2) = process interface for development, integration, and validation of the application

4.5.8 Abstraction of the application from RTE solution-specific safety activities

To achieve a completely "RTE-independent" application source code, an additional abstraction between "pure application logic" (application source code) and "executable application for a specific RTE" would be necessary.

This would mean to separate the RTE-related application activities:

- use of RTE-related T3 tools for software generation
- use of original RTE software parts for vertical integration "application is running on an RTE"
- safety evaluation of fulfillment of the RTE SRACS by the application
- creation of the RTE configuration data as defined by the application

An application APP-1 running on two different RTE solutions (RTE-1 and RTE-2):

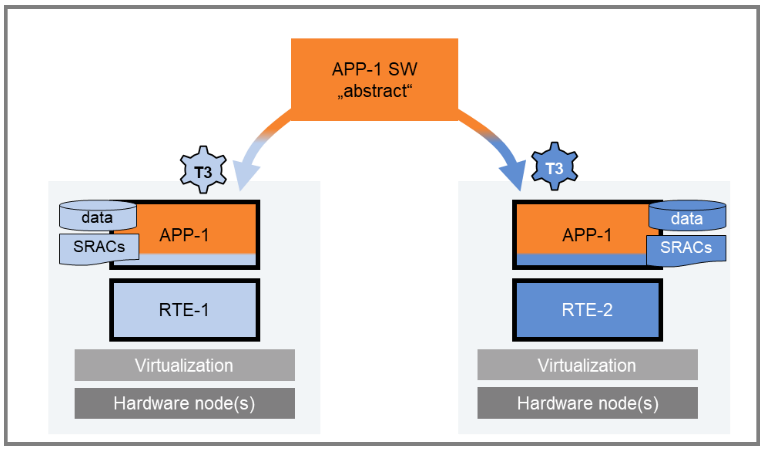


Figure 17 Abstraction of the application

This abstraction leads to a shift of responsibility for all RTE-related application activities to achieve a certification for "application is running correctly on an RTE".

Not only the vertical integration with the RTE but also the horizontal integration with neighboring applications is shifted to another role because abstracted applications cannot be integrated in a real context with abstracted neighboring applications.

This role needs to assume complete responsibility for:

- porting of abstract applications to the RTE
- vertical integration "APP-1 is running on RTE-x"
- horizontal integration “APP-1 with neighboring applications”
- overall validation

Figure 18 below shows the basic situation in the case of abstracted applications for a simple exemplary installation with two applications, each on a different RTE.

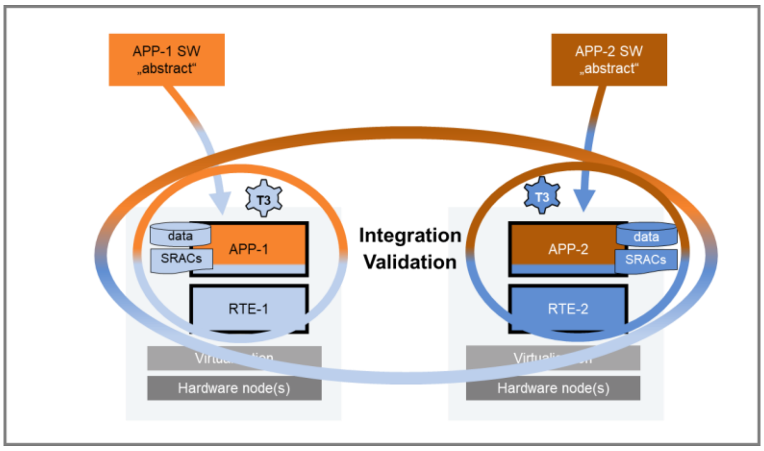


Figure 18 Porting of abstract applications on RTE

4.5.9 Summary

The opening of the market for application development by multiple vendors (which do not have an RTE in their own portfolio) is very much simplified by defining a generic API.

However, the RTE API can only provide basic support and the remaining specifics will lead to a remaining RTE-related effort for development (example: SRACs), configuration (example: replica configuration), testing (example: SRAC-related tests), building (linking with the RTE), integration (on the RTE), and certification (not for the application's business logic but for the RTE specifics) so that a delivered application cannot be freely exchanged between the RTEs.

It should be analyzed whether this restriction would not be weighted too high: to manage efforts, it can be envisaged to deploy an application only on certain RTE types. However, this would impose a restriction for cases like those stated in sections 4.5.4 and 4.5.5. In practice, this will likely mean restricting the number of different RTE types to a small number such as two.

An alternative analysis result could be an RTE abstraction layer in each application as depicted in section 4.5.8.

4.6 Migration concept for legacy solutions

To achieve a common data center, it is important to define the relevant interfaces which must be provided by a legacy installation and relevant requirements for basic aspects to be fulfilled by a migrated legacy solution.

4.6.1 Interfaces for legacy installations

To bring a legacy installation (safety layer, application, and data) into the same data center as an RCA-based system, the legacy installation shall fulfill the same basic interfaces as defined for an RCA-based installation.

The main interfaces to migrate a legacy installation are:

- interface to COTS hardware considering CPU architecture and performance
- interface to the virtualization considering the required time behavior of the virtualization and the required safety-relevant behavior of the virtualization (e.g., reliability regarding the mapping of the running SIL4 software distributed on several different hardware computing nodes)
- interface to the load operating system for the common handling of software and data updates in the data center
- interfaces to the infrastructure for IT security
- interfaces to the infrastructure for installation and update
- interfaces to the central diagnostic system

Out of scope are the details about the legacy installation's internal architecture, e.g., the interface between legacy application and legacy safety layer.

This is completely the responsibility of the owner/vendor of the legacy installation.

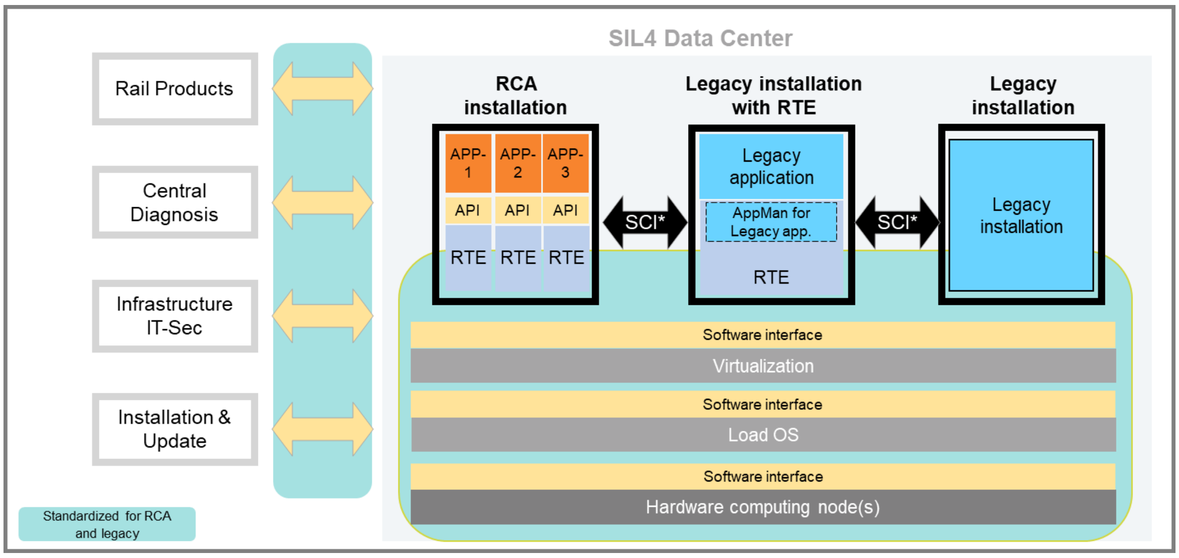


Figure 19 Standardization for migration of legacy solutions

4.6.2 Requirements for legacy installations

Important requirements for the migration of a legacy system are:

- The RTE and the safety layer of the legacy installation shall provide the same flexibility and scalability regarding the usage of the hardware computing nodes.
- The legacy installation shall allow that unused CPU resources are used by any other software (of any safety integrity level).
- The legacy installation shall fulfill all the common interfaces (see section 4.6.1 above).

4.6.3 Variants of legacy installations

For migrating a legacy installation to the SIL4 Data Center, two variants are possible:

1. The legacy installation is a solution with an RTE and a legacy application-specific Application Manager (AppMan).
   In this variant, the same RTE is used as for RCA-based systems, but for running the legacy application, a specific Application Manager is necessary.
   All services required by the legacy application are provided by the Application Manager for the legacy application.
   This Application Manager is a kind of "legacy operating system" for the legacy application.
   In this variant, the internal system interface is not suitable for the generic API of the RTE.
   The legacy installation is provided as complete package by the same vendor.
2. The legacy installation is a complete standalone solution independent from the RTE.

Both variants are provided as a complete package by the same vendor, and the preferred variant for the specific solution will depend on the specific situation of the legacy installation.

For the owner of the data center, it shall not make a difference which legacy installation variant is installed; the basic behavior of the installations on the computing nodes shall be the same for all variants.

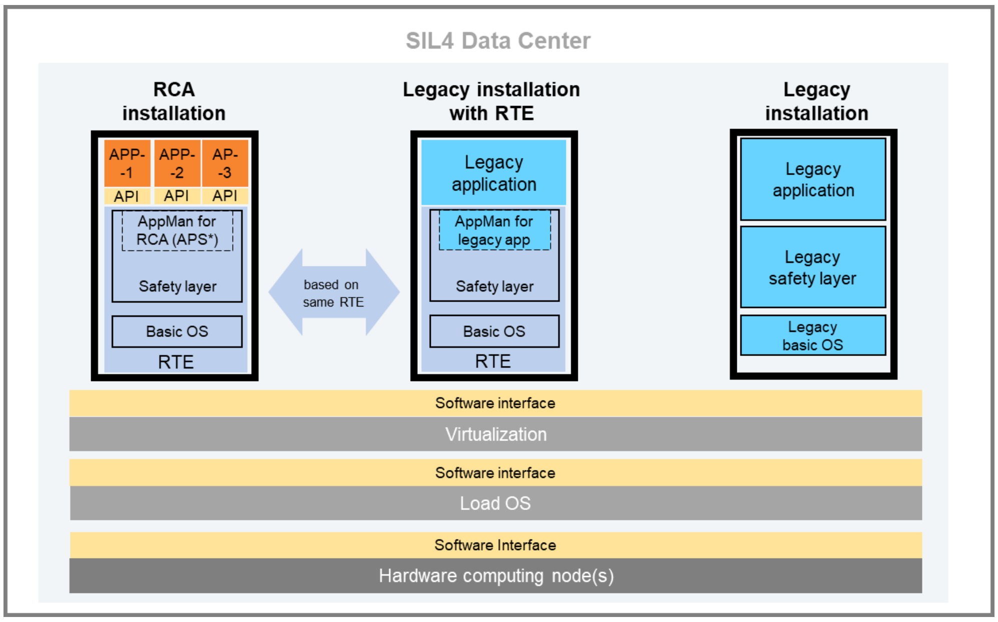


Figure 20 Variants of legacy solutions

4.7 Scalability

The following aspects of the SIL4 Data Center need to be flexibly scalable.

Regarding the applications:

- The application cycle time shall be configurable for each application individually (in the range of 100 ms to 400 ms).
- The number of applications running together as a bundle shall not be limited.

Regarding communication:

- The specific usage of external communication protocols (e.g., RaSTA connections via a RaSTA protocol gateway) shall be flexibly scalable for each individual application.
- Communication protocol-related parameters shall be configurable per communication connection (e.g., the time period for RaSTA heartbeat messages).
- Redundancy aspects (for redundant communication) shall be solved transparently for the application and be flexibly scalable regarding the number of used transport channels (e.g., four transport channels in case of geographical redundancy).

Regarding basic installations:

- The redundancy concept shall be configurable for each installation regarding local and geographical redundancy.
- The number of installations running on the same hardware computing nodes (each installation in its own virtual machine from a hardware and virtualization layer point of view) shall be flexibly scalable.

4.8 Technical flexibility in maintenance

This section describes the required technical flexibility to allow efficient maintenance of the individual parts of a data center according to the individual lifecycles.

Besides the technical possibilities, the operating expenses resulting from the specific maintenance strategy need to be considered for any decision. This will be discussed in section 6.3.

The maintenance of parts with short lifecycles (e.g., COTS hardware, implemented IT security mechanism in a non-safe layer) shall be possible during the runtime of installations without the need of stopping the system.

To that end, two aspects need to be taken into consideration:

1. system distribution:
   Each safety-relevant installed system (e.g., Frankfurt interlocking) is running in several instances distributed on several hardware computing nodes with hardware redundancy.
2. Resource sharing:
   The individual instances of different safety-relevant installations (e.g., Frankfurt interlocking and Berlin interlocking) can run together on the same hardware computing nodes.

Figure 21 below shows an exemplary situation where two installations are running together on the same hardware computing nodes.

The parts **highlighted in green** are the "non-safety-relevant parts with short lifecycles".

For these parts, maintenance shall be possible during system runtime via a gradual maintenance process for the individual green areas, i.e., instance by instance.

This flexibility needs to be permitted from the point of view of the safety concepts of the RTEs; for more information, see section 4.10.6.

The parts **highlighted in yellow** are the "safety-relevant parts".

For these parts, it is not possible to make changes in a running installation. Each installation needs to be stopped completely (all three instances on HW-1/2/3) for maintenance.

Even in a configuration with geographical redundancy, the complete installation (running as several instances on different geographical sites for geographical redundancy) is affected by each update activity.

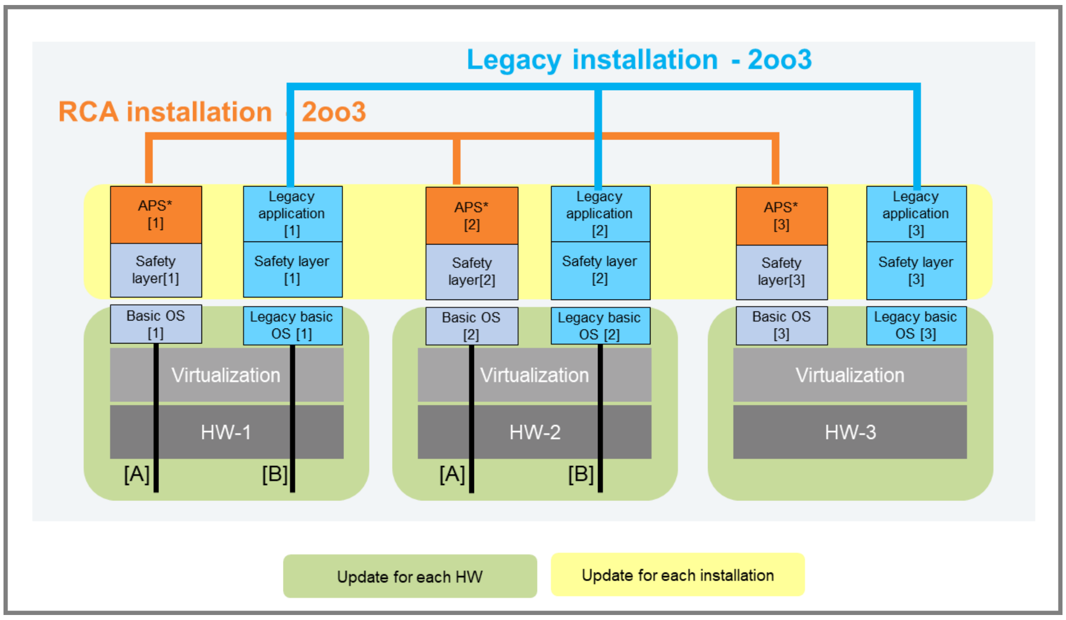


Figure 21 Flexibility in update for maintenance

4.8.1 Maintenance and qualification of COTS hardware

For lean maintenance of the hardware computing nodes, it shall be possible to replace hardware computing nodes hardware by hardware for a running system at runtime with full safety responsibility.

To that end, it shall be possible (= permitted from a system point of view) to run a system on a combination of different versions of hardware computing nodes, i.e., all the software layers on it (virtualization, RTE) shall allow to use different types/versions of hardware computing nodes within the system context.

It must be considered that all used hardware versions need to be "qualified" for usage within the specific generic system. This hardware qualification is necessary to ensure that the new hardware version can be used without any impact on the availability (performance, time behavior) of the systems running on it.
Hardware qualification needs to be done for each environment variant as a combination of virtualization version-x and RTE version-y.

For hardware maintenance, an overall configuration management is necessary to define "which hardware version is qualified for which environment (=virtualization version x and RTE version y)". For availability reasons, the usage of a specific hardware version is only allowed in environments according to the hardware qualification for the specific environment.

Changes in the hardware of a specific hardware computing node shall not have any impact on the behavior of software layers running on it.

4.8.2 Maintenance and qualification of virtualization software

For lean maintenance of the virtualization software, it shall be possible to update the virtualization software one node at a time for a running system at runtime with full safety responsibility.

To that end, it shall be possible (=permitted from a system point of view) to have different versions of the virtualization software running on the different hardware computing nodes, i.e., all the RTEs on it shall allow to use different versions of virtualization software on the individual hardware computing nodes within the system context.

It must be considered that all used virtualization software versions need to be "qualified" for usage within the specific generic system. This software qualification is necessary to ensure that the new virtualization software can be used without any impact on the availability (performance, time behavior) of the systems running on it.
Software qualification for the virtualization software needs to be done for each environment variant as a combination of virtualization software version-x and RTE version-y.

Changes in the virtualization software on a specific hardware computing node shall not have any impact on the behavior of the RTEs running on it.

4.8.3 Maintenance of the RTE – basic operating system (non-safety-relevant)

For lean maintenance of the basic operating system of the RTE (e.g., IT security patches), it shall be possible to update the basic operating system of the RTE one node at a time for the running system at runtime with safety responsibility.

To that end, it shall be possible (=permitted from an RTE point of view) to have different versions of the basic operating system running on different hardware computing nodes within the RTE context. The associated RTE shall allow to use different versions of its own basic operating system on the individual hardware computing nodes within the RTE context.

It must be considered that all used basic operating system software versions need to be "qualified" for usage within the specific installation. This software qualification is necessary to ensure that the new basic operating system software can be used without any impact on the availability (performance, time behavior) of the installations running on it.
Software qualification for the basic operating system software needs to be done for each environment variant as a combination of basic operating system version-x and safety layer version-y of the associated RTE.

Changes in the behavior of the basic operating system on a specific hardware computing node shall not have any impact on the behavior of the RTE safety layer running on it.

4.8.4 Maintenance of the RTE – safety layer and applications (safety-relevant)

For the safety-relevant parts of an installation (RTE safety layer and applications), maintenance always affects the complete installation, i.e., all involved instances (RTE safety layer instances, application replicas).

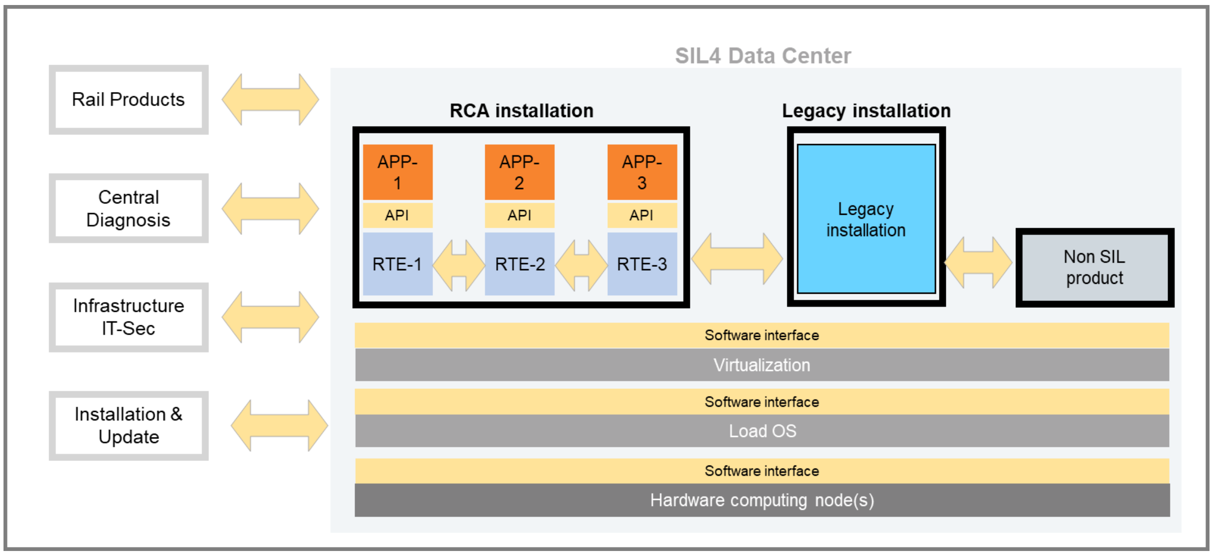


Figure 22 System interfaces

From a safety point of view, all redundantly running parts (RTE safety layer instances, application replicas) within an individual installation of an application running on an RTE shall have an identical version, meaning all RTE safety layer instances have the same version and all application replicas have the same version.

The precondition for maintenance is the overall integration of all involved parts (all involved installations of individual applications running on the RTEs).

Note:
The voter of the RTE safety layer cannot vote the outgoing messages of application replicas with different versions as they result in different application replica behavior.

4.8.5 Maintenance of legacy installations

For legacy installations, the maintenance situation is defined as follows:

The vendor of the legacy installation provides a new version of the legacy installation and it is the responsibility of the legacy vendor that all required changes are included in the delivery package.

Even for legacy systems, the maintenance of non-safety-relevant parts (e.g., IT security patches) shall be done in a lean way without affecting safety-relevant parts.

4.8.6 Commissioning of new installations

The commissioning of a new installation on hardware computing nodes which are already in use for running installations shall be possible without any impact on the running installations.

The virtualization layer and the virtual machines need to be configured in a way that any changes in the system configuration (e.g., setup of additional virtual machines) do not affect the existing virtual machines. This applies especially to performance degradations due to additional virtual machines (e.g., processor load, memory access time, network latency, and bandwidth).

4.9 System interfaces

All relevant interfaces shall be identified because these interfaces need to be defined as standard to achieve a vendor-independent common architecture for the complete data center.

Figure 22 shows the interfaces which need to be defined as standard:

The need for standardization of the interfaces between the RTEs depends on the decision of whether the RTEs within one installation shall be from only one vendor or multiple vendors (see section 4.4.1 and 4.4.2).

4.9.1 Interfaces between different rail installations

The interfaces between RCA installations shall be defined as standard in the same way as SCI* for coupling to legacy installations (i.e., including safe communication protocols).

The same standard SCI* interfaces shall be used independently of the location of the systems. This includes the following two scenarios:

- both systems are running within the same data center
- both systems are located at different locations

The communication between installations via SCI* interfaces is independent from the installation's internal communication between the individual applications, i.e., if all applications are running on the same RTE solution or on different RTE solutions.

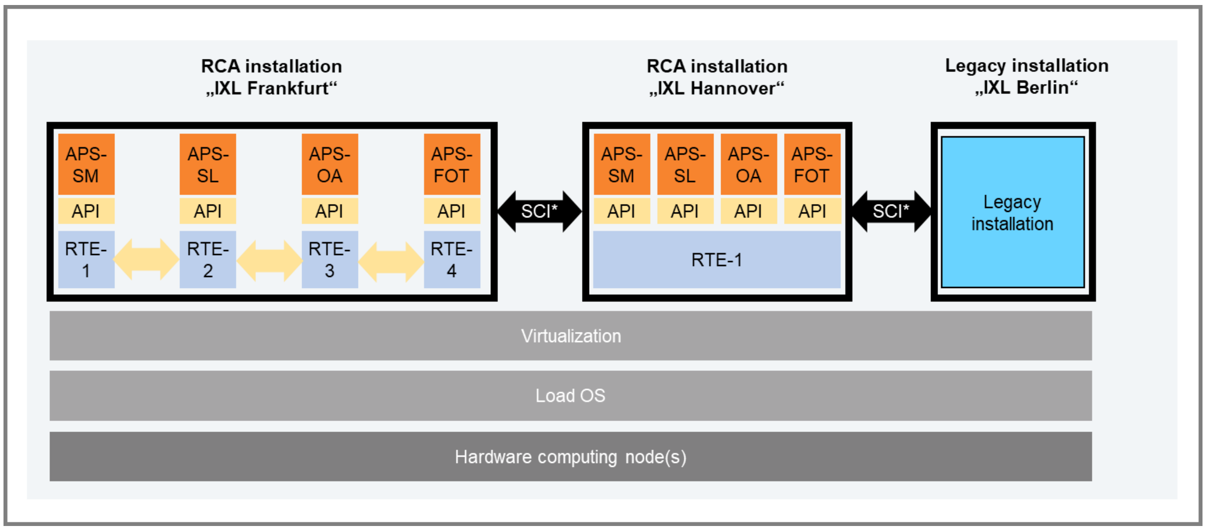


Figure 23 Interfaces between rail systems

4.9.2 Interfaces for installation and update

It shall be possible to install and update the rail systems within the SIL4 Data Center remotely by means of an installation and update tool operated by a maintainer.

To that end, the security aspects need to be considered.

The mechanism for installation and update of an installed system shall be transparent for the application, i.e., this process and its interfaces shall not touch the application.

On each individual hardware computing node of the target system, the load operating system (installed on the hardware) will be responsible for installation and update. Specifically, it performs the following:

- CPU setup to prepare the COTS hardware for remote access
- remote installation and update of the virtualization solution and the virtual machines
- remote transfer and storage of the installation into the virtual machines
- start-up of the installations by the installation itself (detailed start-up mechanism must be implemented within the installation)

The interfaces between the infrastructure for installation and update, the load operating system, and the affected software parts (virtualization, RTE, legacy installation) shall be defined as standard to achieve a vendor-independent solution for the complete data center.

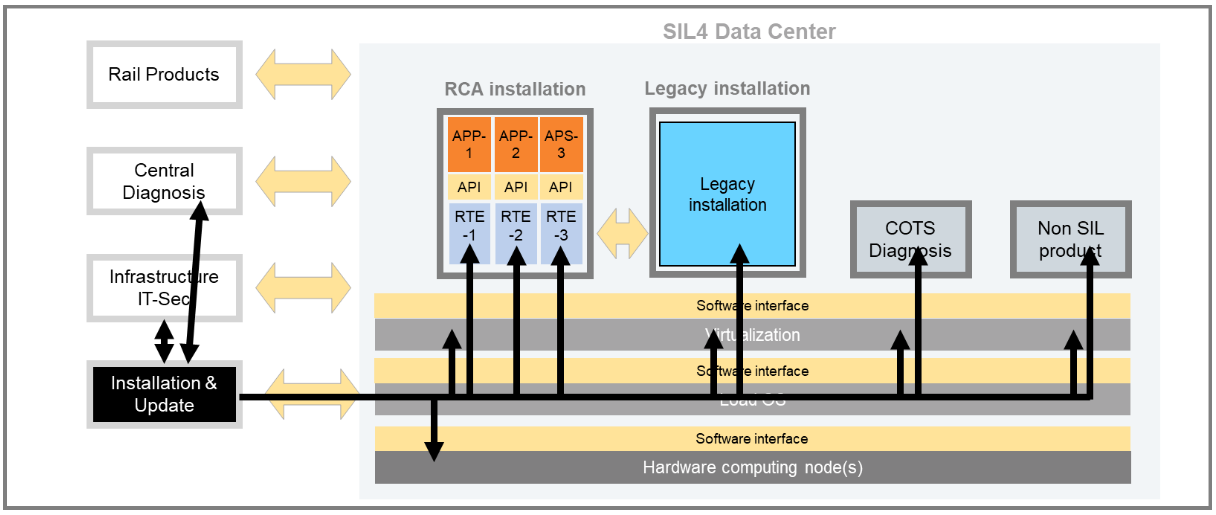


Figure 24 Interfaces for installation and update

The installation and update process does not include the integrity checks at system start-up and runtime. This is part of the integrity checks described in section 4.10.7.

4.9.3 Interfaces for IT security

Each single installation running in the data center is required to fulfill the IT security requirements (the requirements of the protection profile "trackside component" of X2Rail-3).

Central services for IT security are provided by the central infrastructure for IT security. As per X2Rail-3 IT-Security Architecture, the central infrastructure for IT security consists of the mandatory and highly recommended shared security services with interfaces based on international standards.

X2Rail-3 defined the protocols used for the interfaces of the shared security services as follows (refer to section 4.12):

- System-wide time synchronization (TIME): NTPv4 (RFC 5905)
- Central logging (LOG): syslog-ng (i.e., syslog over TLS) (RFC 5425)
- Security and Event Management System (SIEM): syslog-ng (RFC 5425)
- Certificate Management (PKI): CMP (RFC 4210)
- Identity and Access Management (IAM): SCIMv2 over TLS (RFC 7655)
- Backup (BKP): rsync over SSH
- Remote SW Update (SMU): OPC UA SC & HTTPS
- Asset Inventory (INV): OPC UA SC

The interfaces for access to the COTS hardware (e.g., Trusted Platform Module) need to be defined as standard for all kind of installations (RCA/legacy).

A possible solution is the unified trust anchor API for the Linux operating system libuta, refer to https://github.com/siemens/libuta for further information)

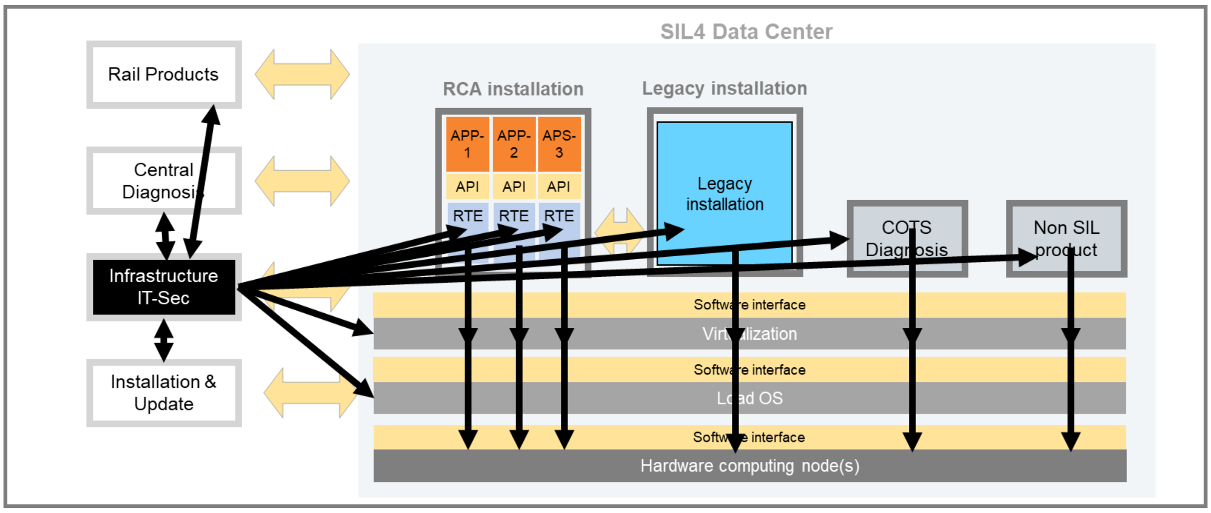


Figure 25 Interfaces for IT security

4.9.4 Interfaces to central diagnostic system

All software layers within the data center provide their own diagnostic data to the central diagnostic system.

The central diagnostic system collects the diagnostic data of all software layers running in all computing nodes at all existing geographical sites (Figure 26).

4.10 Safety and availability

4.10.1 RTE safety measures for application replicas

All required safety measures for the deployment of the application replicas fall within the responsibility of the RTE but are not completely encapsulated.

Depending on the specific solution of the RTE, different mechanisms and RTE-related tools need to be considered and might generate code that instruments the application.

This shall not have any impact on the functional logic of the application itself, but it affects the processes on side of the application for development, integration, and validation of the application. To mitigate or even eliminate this, a concept of abstraction layer (see section 4.5.8) should be foreseen.

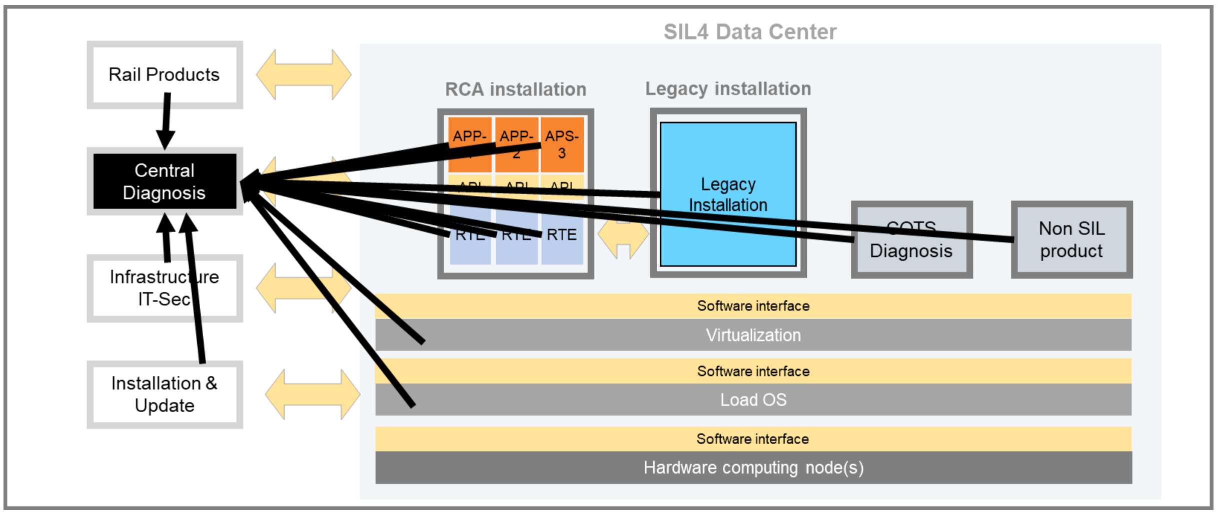


Figure 26 Interfaces for diagnosis

4.10.2 Availability of application replicas

The RTE shall be scalable regarding the number of application replicas to increase the availability of the installation.

The safety-related requirement "at least two application replicas on two different hardware computing nodes" can be enhanced to include the availability aspect: "at least two application replicas on the same hardware computing node for availability (in addition to the replicas on different hardware computing nodes)".

Each transitory failure of an individual replica shall be repaired automatically by the RTE as fast as possible.

The goal is to achieve a system with an unreduced number of running replicas under all circumstances.

Note:
Effects on individual application replicas can be caused by e.g., bit dumpers, influences by non-safety-relevant software layers (e.g., virtualization).

4.10.3 Safe and available RTE components

All required safety measures for achieving safe voting, a safe clock and safe protocol gateways fall within the responsibility of the RTE itself and transparent for the application.

Any transitory influence on an individual software component shall not lead to static failure of the software component; it shall be repaired automatically by the RTE as fast as possible.

4.10.4 Safe internal RTE communication protocol

The safety measures for the internal RTE communication protocol for communication between the RTE components itself fall within the responsibility of the RTE and are transparent for the application.

4.10.5 Separation of safety-relevant and non-safety-relevant parts

The safety argumentation of the RTE shall define a clear separation between the safety-relevant parts (safety layer, application) and the non-safety-relevant parts (basic operating system incl. security layer, virtualization).

Motivation: Allow a lean process of handling changes in non-safety-relevant parts with shorter lifecycles (e.g., IT security patches) without impacting the assessments of the safety-relevant parts.

4.10.6 Flexible usage of non-safety-relevant parts

The safety argumentation of the RTE shall allow flexible usage of the non-safety-relevant parts.

For flexible maintenance of the running installation, the RTE instances on the hardware computing nodes shall be permitted to run on different (integrated, of course) versions of the non-safety-relevant parts as (the RTE's) own basic operating system, hardware, and virtualization.

Motivation: Enable a lean maintenance process for the non-safety-relevant parts instance by instance without the need for a complete stop of the running installation.

4.10.7 Integrity check for safety-relevant parts of an installation

The integrity of all involved safety-relevant parts of an installed system (RTE safety layer instances, application replicas) running distributed on separate computing nodes shall be ensured by the RTE.

This integrity shall be continuously ensured by the RTE for the complete lifecycle of the system – independent of the configuration of local/geographical redundancy.

Motivation: The configuration of an installation can be affected by non-safety-relevant parts, e.g., by the upload mechanism for software maintenance. To that end, any effect on integrity shall be identified safely and continuously by the RTE.

Note:
If an installation (e.g., Frankfurt interlocking) consists of several different RTE solutions, this integrity check cannot be done by the RTEs (because each RTE has only a reduced scope within the installation). For the scenario "several RTE solutions in one installation", the issue of integrity checks is not clarified today.

4.10.8 Persistent saving of application data by the RTE

The RTE shall provide a functionality for persistent saving of application data in case an entire system restart occurs.

Note:
If an installation (e.g., Frankfurt interlocking) consists of several different RTE solutions, persistent data saving is solved individually by the RTE solutions.

This needs to be considered from an overall point of view to ensure that the saved data is consistent (from the overall point of view onto the complete installation with different RTEs involved).

4.10.9 Safe strictly monotonous clock

The RTE shall provide a safe monotonous clock for cyclic triggering of the application replicas. Note:
If an installation (e.g., Frankfurt interlocking) consists of several different RTE solutions, each RTE solution provides its own monotonous clock for its application replicas.

There is no synchronization of the clocks between different RTE solutions.

This needs to be considered in terms of the time behavior of the applications. It is not possible to synchronize the application running cycles of different RTE solutions.

4.10.10 Example: Application in 2oo3 mode with six replicas for increased availability

The situation of this example is as follows (as shown in Figure 27 below):

- One APS application APS-1 is running in six replicas [1] to [6].
- The replicas are distributed on the hardware devices HW-1 to HW-3 for the 2oo3 principle (safety and availability).
- Each hardware device has two replicas (availability).
- The communication to external systems is redundant:
  Channel A is provided by the RTE on HW-1, channel B is provided by the RTE on HW-2.
- The third hardware is for availability on a hardware-device level.
  If one hardware device fails, the system can continue to run safely on the remaining two hardware devices.
- The second replica on each hardware is for increased availability within one hardware.
  If one replica fails, the other replica continues to run, and the system continues to run in 2oo3 mode on a hardware level.

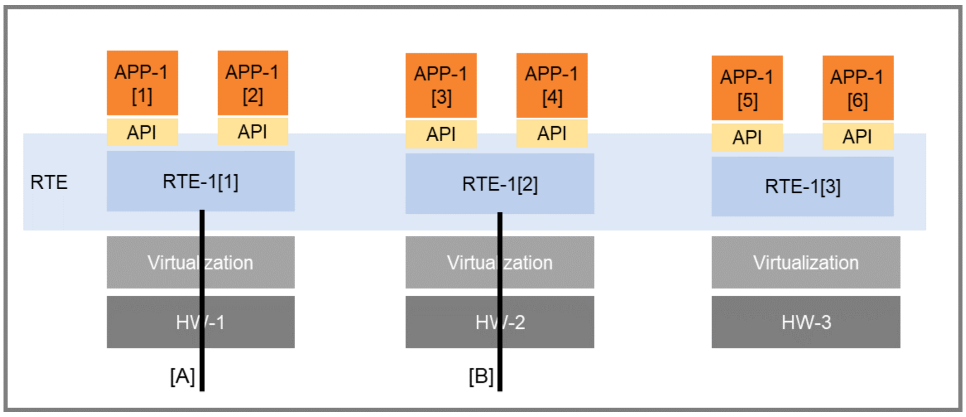


Figure 27 Example: 2oo3 principle with six application replicas

The view into the RTE for this exemplary use case is as follows:

The RTE parts (voter, protocol gateway) are running synchronously on each hardware device.

For redundant communication to external systems, the protocol gateways on two hardware devices are "active"; the third protocol gateway on HW-3 is required for safe running in case of reduced availability (e.g., if HW-1 fails).

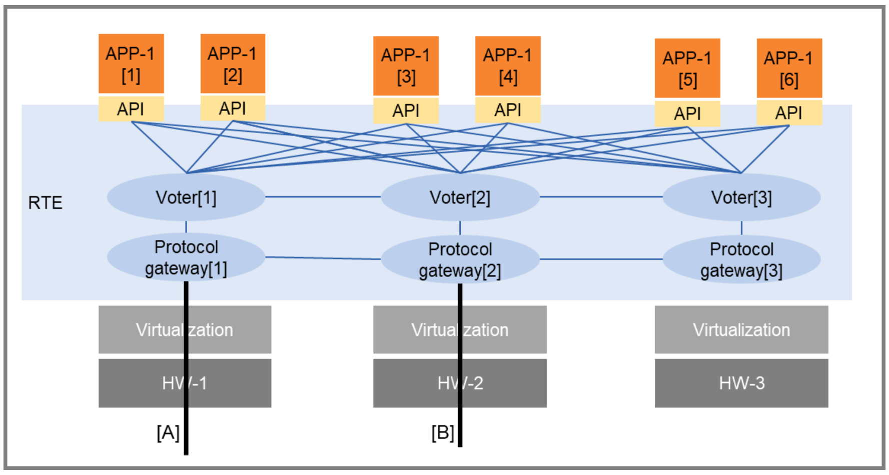


Figure 28 RTE view for 2oo3 principle with six application replicas

4.10.10.1 Scenario 1: hardware failure in 2oo3 mode with six replicas

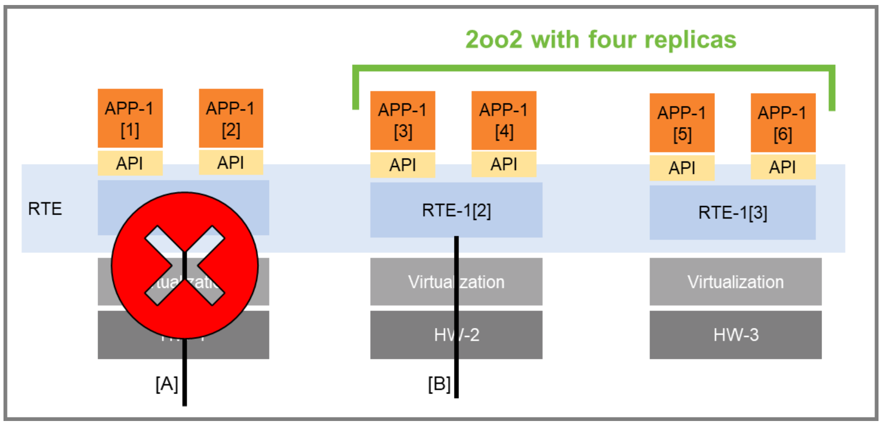


Figure 29 Availability scenario 1: hardware failure

If one COTS hardware fails completely, the application is running in 2oo2 mode with four replicas.

It depends on the failed hardware if communication to the connected systems is affected or not, i.e., if communication is still redundant or with reduced availability.

4.10.10.2 Scenario 2: replica failure in 2oo3 mode with six replicas

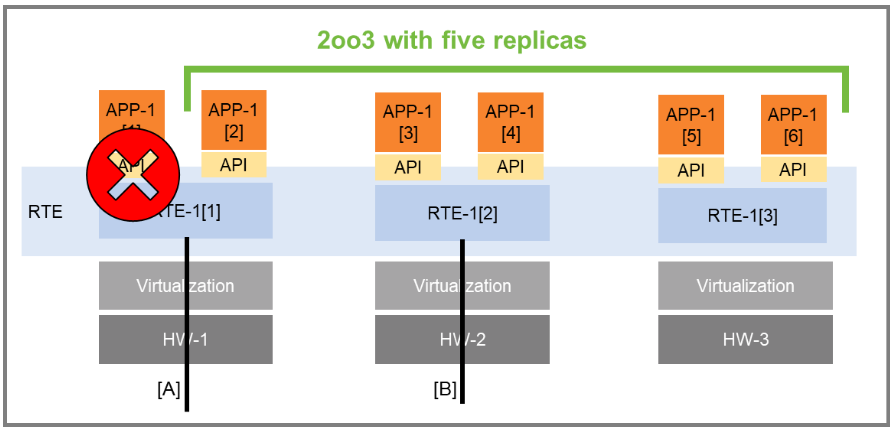


Figure 30 Availability scenario 2: replica failure

If one replica fails, the application is running in 2oo3 mode with five replicas.

The communication to external systems is not affected (is still redundant).

4.10.10.3 Scenario 3: hardware failure and replica failure in 2oo3 mode with six replicas

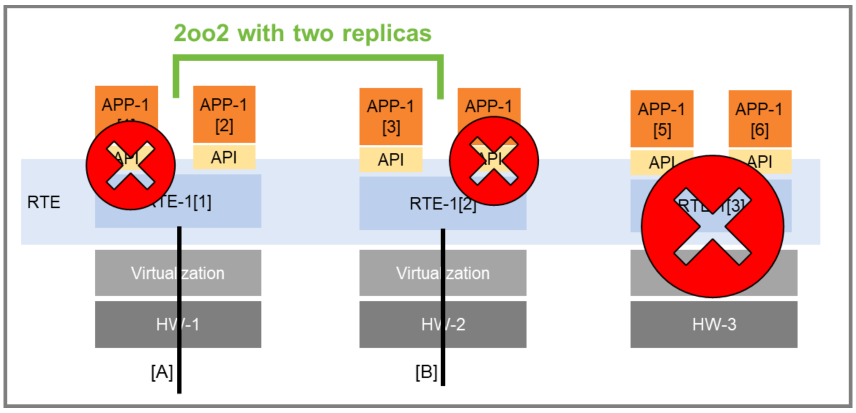


Figure 31 Availability scenario 3: several failures, system without further redundancy

The application is running in 2oo2 mode with redundant communication (depending on which hardware has failed).

4.10.11 Example: APS application in 2x2oo2 mode with eight replicas for increased availability

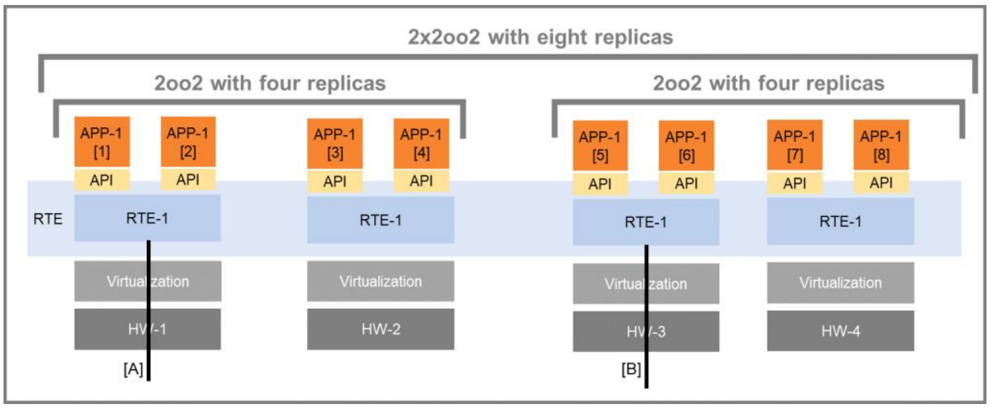


Figure 32 Example: 2x2oo2 principle with eight application replicas

The application is running in 2x2oo2 mode with eight replicas.

Each 2oo2 pair is responsible for one channel of the redundant communication [A/B].

4.11 Geographical redundancy

Railway systems are part the critical infrastructure, and they must be protected against failure. There are several events which may lead to a failure of a complete data center site, such as fire, natural disasters (flood, storm), or even terrorist attacks. Geographical redundancy can limit the effect of such events and ensure continuous operation.

Redundancy is an essential principle for implementing high-availability architectures and IT infrastructures. Redundancy generally refers to the multiple presence of similar objects. These can be technical components, information, services, or even personnel. In high-availability environments, redundancy is usually characterized by the fact that more resources are available than are necessary to fulfill the specified functions of an IT system. Redundancy is not only important for availability; composite fail-safety is also one of the three fail-safety principles of EN 50129.

Composite fail-safety is already applied to the desired SIL4 platform in local scope: Applications like APS are made of equal, redundant replicas which form e.g., 2x2oo2 or 2oo3 fail-safe systems. The basic idea of geographical redundancy can be based on this approach and applies the same pattern.

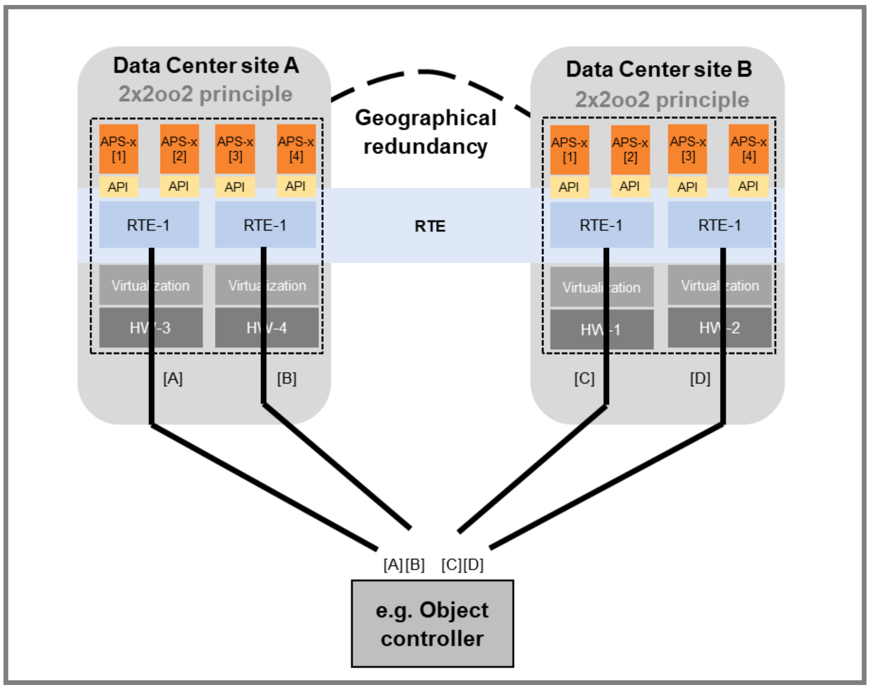


Figure 33 Example: 2x2oo2 principle with geographical redundancy by two sites

4.11.1 Instant "switchover"

By using this design, the whole system remains 100% in operation even if one site fails (the same case as with local redundancy where one replica fails). "Switchover" time is zero; it is even wrong to speak of switchover here:

- Replicas are synchronized at every point in time.
- Replicas at the non-failed site continue their operation; the RTE continues to vote (less, but enough) replicated results.
- Non-failed network links to/from decentralized systems continue to link; the RTE continues to vote (less, but enough) replicated input/output messages.

4.11.2 System integrity over all involved computing nodes

The individual RTE instances and application replicas of the system – running on separate computing nodes with geographical redundancy – need to be consistent regarding the installed versions, i.e., each RTE instance needs to be the same version, each application replica needs to be the same version.

Changing the RTE and/or application replicas for maintenance purposes (e.g., update to a new software version) needs to be done for the complete system at the same time. It is not possible (from a technical point of view) and allowed (from a safety point of view) to have different versions of RTE and/or application replicas running together as one system.

For safety purposes, the RTE shall ensure by implemented check mechanism that a failure in handling of the update rules cannot lead to an inconsistent system running with an unauthorized combination of different versions of RTE and/or application replicas. Such a failure in update management shall be identified by the RTE in a safe manner.

For each maintenance scenario regarding software/data of the RTE and/or application, all involved instances of RTE and application replicas need to be considered; geographical redundancy does not mean that system changes in software/data can be done "on the fly" in a hot-running SIL4 system.

4.11.3 Network parameters

With a remote connection, network parameters (e.g., latency, packet failure rate, bandwidth) must fulfill the same quality criteria as in the local case because all platform-relevant processes, e.g., voting, must communicate not only locally but also remotely. This requires a performant backbone network.

4.11.4 Residual availability (replicas)

A geographically distributed system is available with the same rate as a system with the same number of replicas at a local site (not considering remote network availability). In both cases, a failed replica will not let the whole system fail, but the system has a reduced residual availability. This is tolerable for a local system because repair of a failed replica is a manageable process (spare stock), and the time to repair is relatively short, so the temporary decrease in availability is acceptable.

With geographical redundancy, if a failure was caused by a failure of a complete site, this situation cannot be remedied in the short term, as it takes longer (but is less likely). Therefore, each geographical site needs to be equipped in a manner that it fulfills the availability requirements without the failed site. In other words: whereas with local redundancy a 2x2oo2 (or 2oo3) system is sufficient, geographical redundancy requires more replicas considering the added decentralized site(s): for the case of two sites two times 2x2oo2 or two times 2oo3.

4.11.5 Residual availability (remote communication)

Decentralized systems have redundant network links and disjunct routes to platform-hosted applications like APS. Even if the usual two links to a locally redundant system would – with geographical redundancy with two sites – simply be torn apart (used as one for each site), this would still fulfill the required redundancy from a global, geographical redundancy point of view. But again, if one site fails, there is no more redundancy for communication, so the residual availability is reduced below the accepted value. With the same reasoning as for the residual replica availability above, this also entails increasing the link redundancy to include the number of geographical sites. Most railway protocols (e.g., RaSTA, ETCS Subset-098) are already prepared such that the communication stacks can be configured for more links, transparent to the application(s) using them.

4.11.6 Number of sites

Building geographical redundancy from two sites is the minimum, yet sufficient configuration.

Also, three sites could be beneficial, as they mean fewer total replicas (example: two sites with 2x2oo2 systems = eight replicas in total versus three sites with 2oo2 systems = six replicas in total). They still fulfill the residual availability requirement. Whether to decide for two-site or three-site redundancy will depend on considerations such as:

- costs per replica per site
- costs per redundant link per site
- costs per data center per site
- staff availability/distribution
- manageability of the split-brain problem per site (see section 4.11.7 below)

4.11.7 Split brain

Geographical redundancy has another challenge: With a remote connection, it must be considered that each site is working correctly but their connection, needed for synchronization, fails. The sites can detect this, but without further knowledge, it is not clear which site takes over the master role, and as the two sites can no longer communicate, they cannot perform a master election. But this is required because all decentralized systems are still connected to the sites and will accept and answer requests. This case is called "split brain".

As a result, such a case can lead to an unsafe situation overall. For a two-site system, there is no system-inherent way to solve the split-brain problem (no majority). An external solution will be required: safely detect the situation (connection loss) and perform an external master election. The detection and solution are time-critical to both avoid an unsafe situation and minimize the operational impact. This can be solved by a manual intervention, or an automatic intervention by another safe system. An external solution needs to be placed on a third location to achieve the required availability in case of an outage of the two-site system.

A three-site system can solve this situation by its asymmetric design: The isolated single site can detect the situation and would automatically disapprove itself. The two sites still connected can perform a master election. Note, however, that a subsequent failure of the communication between the two remaining sites would mean the automatic disapproval of the remaining system as again the split-brain problem would occur or would require the same complex solution as in a two-site system.

4.12 Security architecture

The security architecture defines a common security infrastructure, also called shared security services. These shared security services were defined in the EU research project X2Rail-3 in work package Cyber Security (see D8.2-2 - Generic cybersecurity architecture and shared security services – publication pending) and are based on the international standard IEC 62443 [10].

Below figure shows the shared security services of X2Rail and their respective relations. In particular it shows the central rail automation security zone, the back-office security service and adjacent systems, with central and trackside equipment.

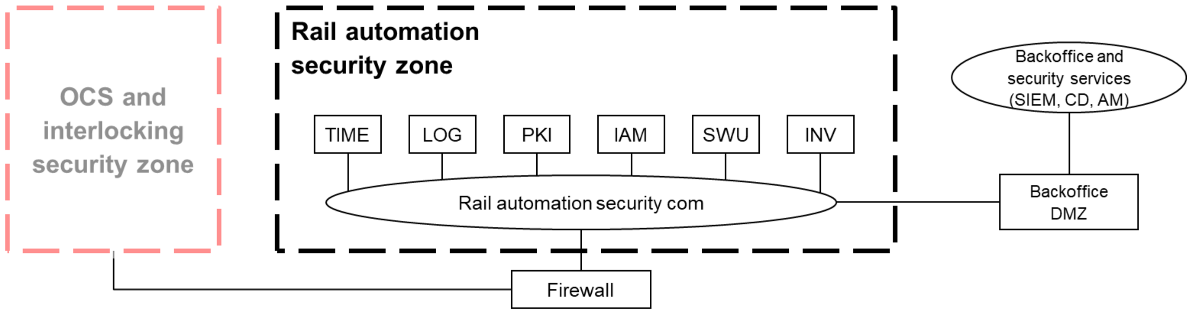

OCS and interlocking security zone
Rail automation security zone
TIME
LOG
PKI
IAM
SWU
INV
Rail automation security com
Backoffice and security services (SIEM, CD, AM)
Backoffice DMZ
Firewall

Figure 34 Shared security services and relations to trackside infrastructure

For high availability of the shared security services, it is recommended to install multiple instances of key security services in geographically separated locations. These geographically redundant installations can be co-located with the geographically redundant SIL4 data centers. A total failure of one data center should not affect the functionality of the shared security services for components in other data centers. In that case the security services are provided by another data center.

The shared security services are defined by the following mandatory and highly recommended services:

Mandatory services:

- System-wide time service (TIME)
- Central logging (LOG)
- Identity and Access Management (IAM)
- Backup (BKP)
- Asset inventory (INV)
- Intrusion detection/continuous security monitoring (IDS)
- Security Incident and Event Management (SIEM)

Highly recommended services:

- Public Key Management (PKI)
- SW Update (SWU)

The following chapters describe the shared security services briefly (excerpt from X2Rail-3 D8.2-2 Generic cybersecurity architecture and shared security services – publication pending):

4.12.1 System-wide time service (TIME)

A system-wide synchronized time is needed to analyze audit logs (e.g., by a SIEM system), backup and restore functions and to evaluate the validity of certificates.
System-wide time synchronization is achieved by a central time service that acts as a common time source for all components requiring time synchronization (e.g., signaling, security, networking components).
The system-wide time synchronization function typically can synchronize its time with an upstream time source (e.g., from a customer time source or a global navigation satellite system). This upstream time source could be behind a demilitarized zone (DMZ).

The selected protocol of X2Rail-3 for time synchronization is NTPv4 (RFC 5905). This is also applicable for systems deployed in data centers.

It is typically implemented on the operating system level of the RTE and provided to applications via operating system level APIs.

4.12.2 Central log service (LOG)

A central log service is a service that collects audit records (logs) from all components which generate logs and compiles them in a system-wide, time-correlated audit trail.
CR 2.8 of IEC 62443-4-2 mandates that security-relevant logs are generated for the following categories: access control, request errors, control system events, backup and restore events, configuration changes and audit log events.

Note:
There are different log content and log destinations:

- Security related logs: content as mentioned above, destination: SIEM
- Continuous monitoring logs: content logs from IDS (HIDS, NIDS), destination: SIEM

The central logging service can provide the time-correlated audit trail to other services (e.g., diagnostic or SIEM systems).

The selected protocol of X2Rail-3 for central logging is syslog-ng / syslog over TLS (RFC 5425). This is also applicable for systems deployed in data centers.

It is typically implemented on the operating system level of the RTE and provided to applications via operating system level APIs.

4.12.3 Security Incident and Event Management (SIEM)

A Security Incident and Event Management (SIEM) service provides real-time analysis of security alerts generated from the system-wide audit log (including logs from network devices e.g., firewalls, routers and IDS).

A SIEM service can be hosted in the security operation center of the railway operator or outsourced to another party. Therefore, the rail automation solution should send the security related logs to the SIEM service. This is realized through an interface of the signaling log service.

The selected protocol of X2Rail-3 from LOG to SIEM is syslog-ng / syslog over TLS (RFC 5425). This is also applicable for systems deployed in data centers.

This interface is solely implemented through the central log service.

4.12.4 Public Key Management (PKI)

A public key infrastructure is used to create, manage, distribute, use, store and revoke digital certificates. Digital certificates in combination with a PKI facilitates the authentication of communication partners and the cryptographic protection of their network communication.

Examples for PKI usage include but are not limited to:

- KMC online key distribution (UNISIG 137)
  Device to device communication authentication
  Human user authentication (e.g., for diagnostic access)
- A public key management service is required to automate the process for certificate management and make use of a product supplier or asset owner “root of trust”.

A PKI consists typically of a (local) registration authority (RA/LRA) and a certificate authority (CA). If a PKI already exists at a railway operators' site, the SIL4 Data Centers’ PKI shall be able to integrate with that existing PKI.

The operation of a PKI, especially of a certificate authority, shall be performed according to commonly accepted best practices (refer to [9]). A common requirement for the central PKI installation is SL 3 or SL 4 depending on the respective requirements.

A PKI is not explicitly required by IEC 62443-3-3 or 4-2, although key management for larger systems (e.g., hundreds of components) is only manageable with an automated key management system such as a PKI.

Therefore, the conclusion is that a PKI shall be part of the shared security services.

The selected protocol of X2Rail-3 for certificate management is CMP (RFC 4210). This is also applicable for systems deployed in data centers.

It is typically implemented on the security layer of the RTE and provided to applications via operating system level APIs.

4.12.5 Identity and Access Management (IAM)

An Identity and Access Management (IAM) service manages accounts of human and possibly technical users (e.g., services, processes, machines), the relationship of accounts to groups, and the assignment of authorizations.

A central IAM service reliefs single components from implementing account, identifier and authenticator management. An IAM service could also provide single sign-on as an additional functionality.

In larger installations, the IAM service can be part of a larger federated system. Identities can be located and managed in different enterprise and security domains; however, authentication is realized using common identifiers. The central IAM service for the SIL4 Data Center should therefore be able to integrate into a federated IAM system (e.g., a corporate directory).

A local IAM for the SIL4 Data Center would not integrate into a corporate directory. Users need then to be managed on both systems (no synchronization as with a federated IAM).

IAM services for signaling use cases should be restricted to interactive access only (e.g., for diagnostic or maintenance access) and should not adversely impact essential functions (e.g., access to train control user interfaces).

Note:
It is advisable, as a fallback, to have access to a local management system, even when the central IAM service is not reachable.

There are two main scenarios: token-based access and token-less access.

Token-based systems use a single sign-on services where a (human) authenticates and obtains a token to access the resource. This is widely used for web-based user interface access (e.g., web applications using single sign-on).

Token-less access is used where tokens are not (yet) supported (e.g., current OPC UA servers). In order to support unified account and identifier management without tokens, the resource (e.g., an OPC UA server of a signaling device) needs to contact the shared security service IAM to complete the authentication and authorization e.g., by requesting the permissions of the authenticated user from the IAM.

The selected protocol of X2Rail-3 for identity and access management (token-based) is OAuth 2.0 using OpenID Connect and SAML 2.0. This is also applicable for systems deployed in data centers.

It is typically implemented for graphical user interfaces to enable single sign-on.

The selected protocol of X2Rail-3 for identity and access management (token-less) is SCIMv2 over TLS (RFC 7655). This is also applicable for systems deployed in data centers.

It is typically implemented on the security layer of the RTE and provided to applications via operating system level APIs.

4.12.6 Software update

Regular security patches are required to fix vulnerabilities and keep the installed system in a secure state. The IEC 62443-3-3 and -4-2 standards do not define the upgrade procedure (e.g., local or from remote). Therefore, updating components of an installed system can be done manually for each component. However, a centralized update service increases the reliability and reduces the time for updating a system drastically.

Although software update is technically feasible, the certification of a safety-related system poses several process challenges that need to be addressed with all stakeholders (manufacturers, operators, and assessors).

The selected protocols of X2Rail-3 for remote software update are OPC UA SC (for controlling software update process) and HTTPS (for download of software packages). These are also applicable for the SIL4 Data Center to allow a controlled and coordinated software update and are typically implemented on the security layer of the RTE and provided to applications via operating system level APIs.

4.12.7 Backup

A backup service stores the required software and its configuration of devices. The stored artefacts are used to recover components after a disruption or failure. A centralized, automated backup service simplifies the tasks for control system backup, backup verification, backup automation and recovery and reconstitution.

There are several options for a backup strategy. The scope of a backup should contain all data that is needed to restore a rail automation solution after a security incident (e.g., resulting in compromised devices).

Therefore, all data (software and configuration) of all devices involved should be part of the backup.

A backup should be created after every change (software or configuration). It is advisable to practice restoration of devices from backup during maintenance intervals.

For higher availability and increased security, the backup data can be kept on offline media (and in a different location). These measures including the retention policy should be aligned with the railway operator's corporate IT strategy.

Backup data has to be integrity protected and ownership authenticated.

Encryption of backup data is needed only in case it contains confidential or human user specific data.

For the integrity, authentication and encryption mechanism, the used keys should use the public key infrastructure of the SIL4 Data Center.

The selected protocol of X2Rail-3 for backup is rsync over SSH. This is also applicable for systems deployed in data centers.

It is typically implemented on the operating system layer of the RTE and provided to applications via operating system level APIs (e.g., to restore).

4.12.8 Asset inventory

An asset inventory service stores information of the installed components of a rail automation system for asset management.

Typically, the following component specific data is stored in an asset inventory system: serial number, hardware versions, software versions, configuration version, installation location.

Note:
The asset information should be also available for SIEM operator (either by co-location of the two services or by an interface from SIEM to INV)

The selected protocol of X2Rail-3 for asset inventory is OPC UA SC. This is also applicable for systems deployed in data centers.

It is typically implemented on the security layer of the RTE and provided to applications via operating system level APIs.

4.12.9 Intrusion detection / continuous monitoring

The intrusion detection / continuous monitoring service detects, characterizes and reports security breaches. This can be realized by collecting essential information from a rail automation system at strategic locations. Typical implementations include network intrusion detection systems, which analyze the network traffic from various areas of the communication network.

If abnormal behavior is detected, an alarm shall be generated and sent to the SIEM service.

The selected protocol of X2rail-3 for interfacing from the network tap with intrusion detection is Pcap-NG over TLS. This is also applicable for systems deployed in data centers.

This interface is implemented in network taps.

4.12.10 Other recommendations from X2Rail-3 cyber security working group

Protection from malicious code can be achieved through several different technical solutions: antivirus, whitelisting and anomaly detection.

X2Rail-3 recommends using whitelisting for protection from malicious code for rail automation devices (refer to chapter 5.10.1 of D8.2-2 of X2Rail-3).

X2Rail-3 states that antivirus solutions for rail automation products are not the right choice for protection from malicious code. Rail automation devices are rather static in nature, and therefore frequent signature updates have a negative impact on the availability of safety systems. The likelihood of false positives can impact safety functions. Furthermore, antivirus solutions are computing-resource intensive and may lead to resource exhaustion and additionally impact safety functions).

5 Integration and testing

Integration and testing follow a gradual approach, starting from a small number of components to a larger number of components. This, of course, includes hardware and software. The environment of the components under test needs to be simulated in some way, if components are not available or it is not useful to use real components (e.g., object controllers, trains). Integration needs to be repeated if changes are made to individual parts.

In today's legacy systems, the responsibility for the generic overall system and specific installation comes from a single source, because it is delivered as one complete package by one single supplier. All necessary integrations of all the individually assessed modular parts (safety layer, business application, COTS hardware) are integrated into a generic topology-independent system with generic type approval. For a specific customer installation, an approved generic system is provided together with installation-specific approved engineering data (for the topology). Involved subcontractors (e.g., for developing individual software parts) are in the responsibility of the legacy vendor and are not visible to the customer.

Within the modular RCA approach, a defined "safe computing platform" aims to be open and will allow multiple vendors to run their safe computing applications on it. This means having multiple safety-related products coming from multiple vendors.

This is a major change in the responsibility for integration testing: it imposes that the overall integration and testing responsibility will move to the infrastructure manager with an increased quantity as the granularity has also increased: more, smaller applications must be integrated. This will likely impose a change in the integration and test processes and their extent as more work will have to be carried out by the infrastructure manager that was previously done by the application vendors. This situation as such is not new (compare interlocking application and RBC application from different vendors).

The RCA architecture is designed to be topology independent. This might simplify the integration of the overall installation (i.e., together with the specific engineering data for one site), but on the other hand, it will most likely increase the integration complexity of the preceding layers.

This chapter provides a high-level overview of the possible integration steps in a modular system architecture. It elaborates the integration dependencies, i.e., which integration steps are necessary depending on the specific changed component. A change within an individual system part might imply further actions, definition of additional test cases, etc. that are not covered in this report in detail.

5.1 Generic type approval of legacy systems

To achieve a generic type approval for generic customer installation variants, legacy system vendors like SMO have established an overall process for integration and testing. The main principle here is the clear differentiation and separation of the defined generic system variant that can be re-used in all installations of a customer and the specific engineering data sets for the individual installations of the customer. This corresponds to the separation of generic software development and application data development as specified in EN 50128.

Important conditions for this lean process for the approval of specific installations are:

- The functionality and external interfaces of the system variant must be clearly defined.
- The functionality of the system must cover 100% of the customer requirements (to enable the re-use in every installation).
- The approval for the system variant must be a "type approval".

The main objective of this approach is a very lean and efficient process for the usage of the approved generic system variant in customer installations.

Without a type approval for the generic system for a customer, intensive testing is necessary for each customer installation, generating additional effort. Another effect is that re-use possible between different installations becomes practically impossible (functional interlocking test cases for the Frankfurt interlocking cannot easily be re-used for functional testing of the Berlin interlocking).

For legacy systems, this situation is solved by each legacy vendor using their best method, transparent to the customer.
The legacy vendor provides a complete package for a customer installation; vendor-internal processes for the handling of system tests and data tests are not relevant for the customer.

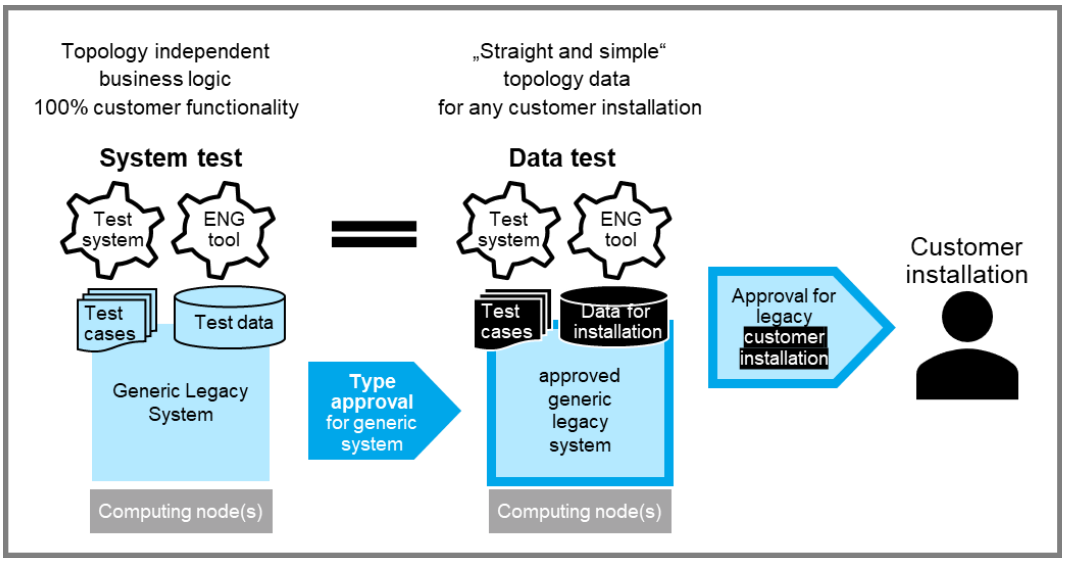

Topology independent business logic 100% customer functionality
„Straight and simple" topology data for any customer installation
System test
Data test
Test system
ENG tool
Test cases
Test data
Generic Legacy System
Computing node(s)
Type approval for generic system
Test system
ENG tool
Test cases
Data for installation
approved generic legacy system
Computing node(s)
Approval for legacy customer installation
Customer installation

Figure 35 System test and data test for legacy solutions

For additional details, see section 5.6.

5.2 Installation-related system variants" (RCA) versus "generic system"

The basic condition to achieve a type approval for a generic system would be the definition of a system variant which can be used in exactly the same way (without any change in the system variant) in a series of installations.

The benefit of a generic system variant with a generic type approval is that all system-related activities (overall integration, safety case, risk analysis, RAMS) need to be done only once and then this approved system variant can be used in a lean way for a series of installations.

5.2.1 Today's situation

Figure 36 below shows an example of the situation for a generic system which is used in two installations.

- The functional scope of the system is growing with each installation from (F1+F2) to (F1+F2+F3), the system covers all needed functionalities.
- The vendor situation is the same for installation 1 and installation 2.
- A generic "Testbahnhof" (test station) is possible to achieve a generic type approval for the generic system. The "Testbahnhof" is growing based on system functionality.
- Functional enhancement of the system (e.g., adding of functionality F3) is available for all installations; it would be easily possible to update installation 1 with this functionality F3.
- Bug fixing in the system is available for all installations (one fix can be re-used easily for several installations without the need for new integration).

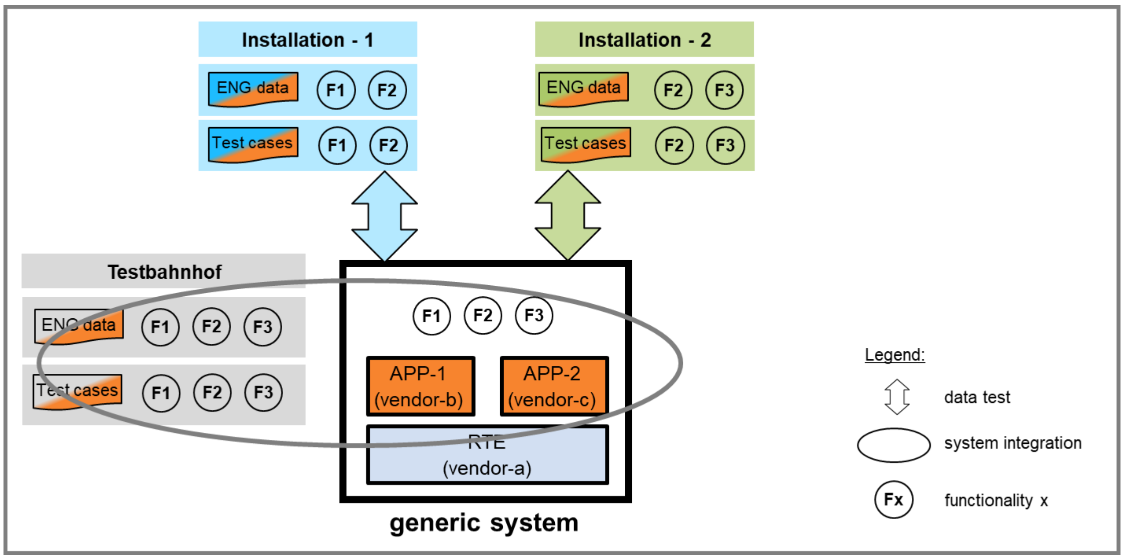


Figure 36 System integration and data testing

5.2.2 New situation with vendor multiplicity

RCA is defining a more fine-grained architecture: classical safety logic-related products like interlocking (route protection) and RBC (train protection) will be replaced by a composition of building blocks (RCA components). That will bring new challenges for type approvals as additional types will be added to the two current types interlocking and RBC.

Furthermore, the multiplicity of combinations will increase (different combinations of APS applications running on different RTE solutions, provided by different vendors). This means each such combination must be integrated and certified.

Figure 37 below shows an example of the situation for two combinations (each with a simplified content of one RTE and two applications) with differences in:

- the application vendors involved (APP-2 by vendor c and vendor d)
- the RTE vendors involved (with different safety principles (RTE with 2x2oo2 versus 2oo3)

Due to the differences in both combinations, it is not easy to use a kind of generic "Testbahnhof" for both integrations, which means each integration must be done individually in its own context of the specific combination with reduced re-use of test environment and test cases.

If APP-1 is changed (e.g., due to bug fixing or functional enhancement), it needs to be integrated both in system variant 1 and system variant 2.

The approach of "flexibility for each installation" is therefore quite the opposite of the intention of "generic type approval for a generic system variant". The individual modular parts (RTEs and applications) can of course be re-used for several installations, but from an overall system integration point of view re-use between different installations is limited.

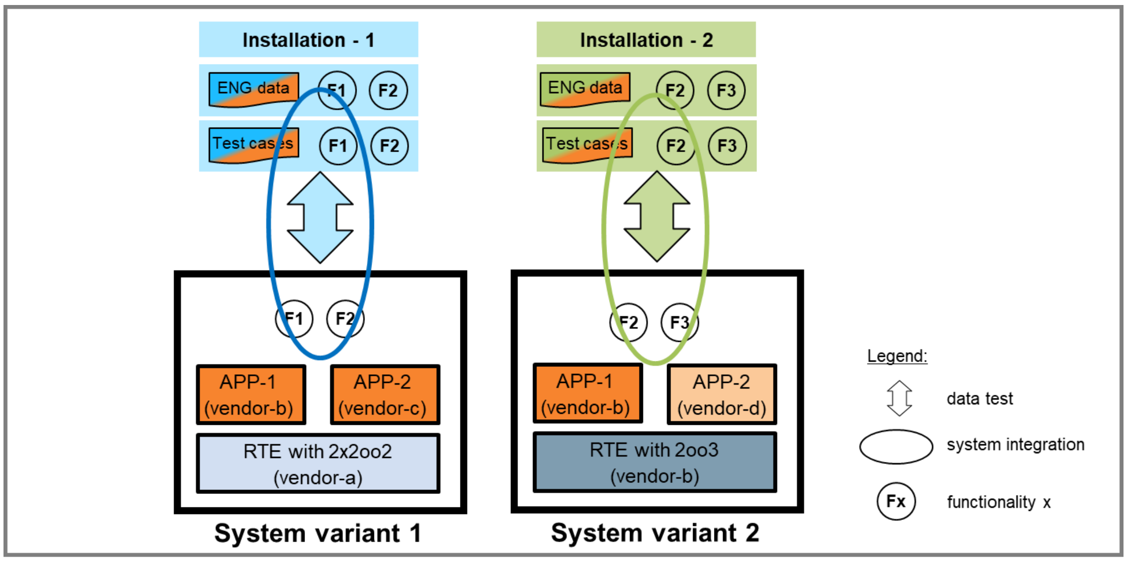


Figure 37 System variants with different vendors involved

5.2.3 Conclusion

The effort for achieving a "type approval for a generic system" (with 100% scope covered by a “Testbahnhof”) is a worthwhile investment if the approved generic system is re-used sufficiently often without a change in a series of installations.

So, it must be evaluated and decided from a cost perspective, whether it is better to:

- keep the vendor situation stable (restricted to a minimum of combinations) to achieve a stable system definition with stable processes and a test environment for integration and re-use of a series of installations
- accept vendor multiplicity and therefore increased integration effort because there is limited re-use between different installations
  To restrict these effects of multiplicity, a small number of combinations could be chosen right from the beginning.
- establish standardized test procedures and environments which allow for flexible vendor configurations at limited additional integration effort

For further details in the context of "automated overall integration testing of a system variant", see section 5.8.

5.3 Vertical and horizontal integrations

Typically, the first integration steps are the integrations of the individual layers COTS hardware / virtualization / RTE / application:

- virtualization integrated with the hardware computing node → virtual computing node
- RTE integrated with the virtual computing node → safe computing platform
- application (software + data) integrated with the safe computing platform → installation (e.g., Frankfurt interlocking)

The integration of these layers is called vertical integration in this document.

After successful vertical integration of an application on the safe computing platform, parallel execution and communication between the different applications are integrated and tested. This integration is called horizontal integration in this document.

Figure 38 below shows the different kinds of vertical and horizontal integrations in the different systems (RCA/legacy).

The RCA system in the example consists of:

- three applications (APP-1/2/3) each running within an own RTE-1/2/3
- three applications (APP-4/5/6) running together as bundle on RTE-4

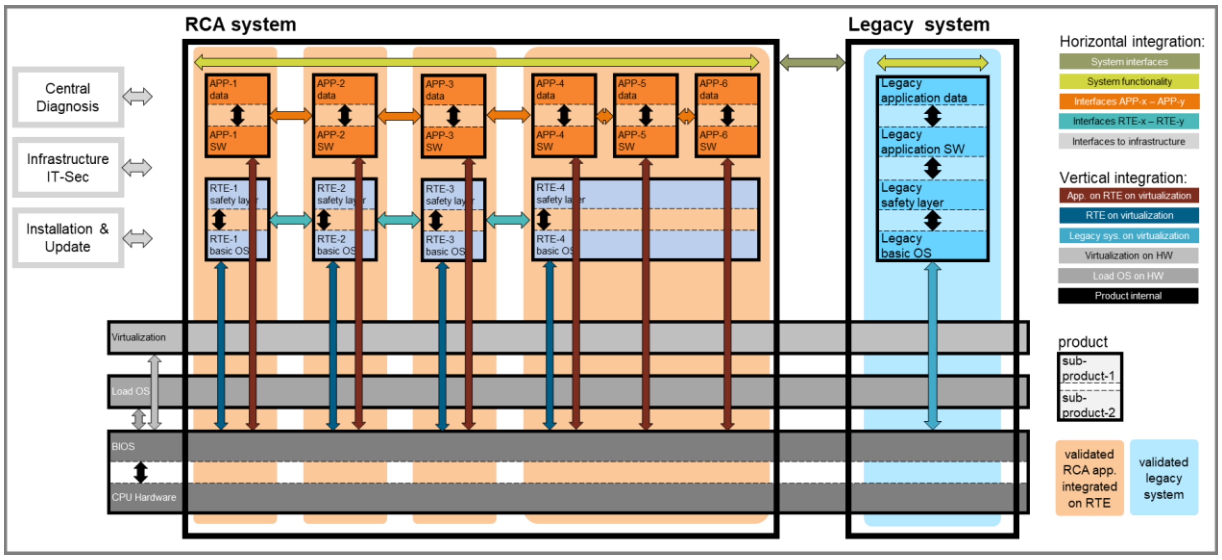


Figure 38 Overview – vertical and horizontal integrations

<u>Note:</u>
In the current EULYNX architecture, vertical integration (application running on safety layer and hardware) is completely the responsibility of the vendors of the safety logic and object controllers and transparent for the customer. Horizontal integration between the safety logic and object controllers must be done "across the participating vendors".

All these vertical and horizontal integrations are indispensable for the commissioning of an installation. In the context of maintenance, the need for integration depends on the specific change.

Some integrations are safety-relevant and the basis for validation. Others are not safety-relevant (but needed to ensure the required availability) and the basis for qualification of non-safety-relevant parts in the installation.

New or changed parts of an installation may only be used if the new part has been integrated with the relevant other parts.

Only integrated combinations of all the involved parts may run in an installation.

<u>Example:</u>
It is not allowed to use a new version of the virtualization solution if it is not integrated before with the affected installations and the used hardware computing nodes.

An integration is necessary for every change to avoid an unexpected impact on availability – a non-integrated piece of software or hardware may never be installed in a running rail product with full safety responsibility.

The way forward is not to skip integration as such but to define the best and most efficient way for lean and fully automated integration of any change.

5.3.1 Integrations for the scenario "new COTS hardware"

Each COTS hardware version that will be used needs to be qualified for all installations in which this new hardware version is to be used.

For this hardware qualification, it must be shown that all the software layers of the installation are running as expected on this new hardware version.

The main scope of this qualification is to test performance, time behavior, and reaction time.

Particularly, some functions close to the hardware (e.g., Trusted Platform Module access for IT security, update of BIOS) need to be integrated.

Figure 39 below shows the required integrations when changing the hardware (highlighted yellow).

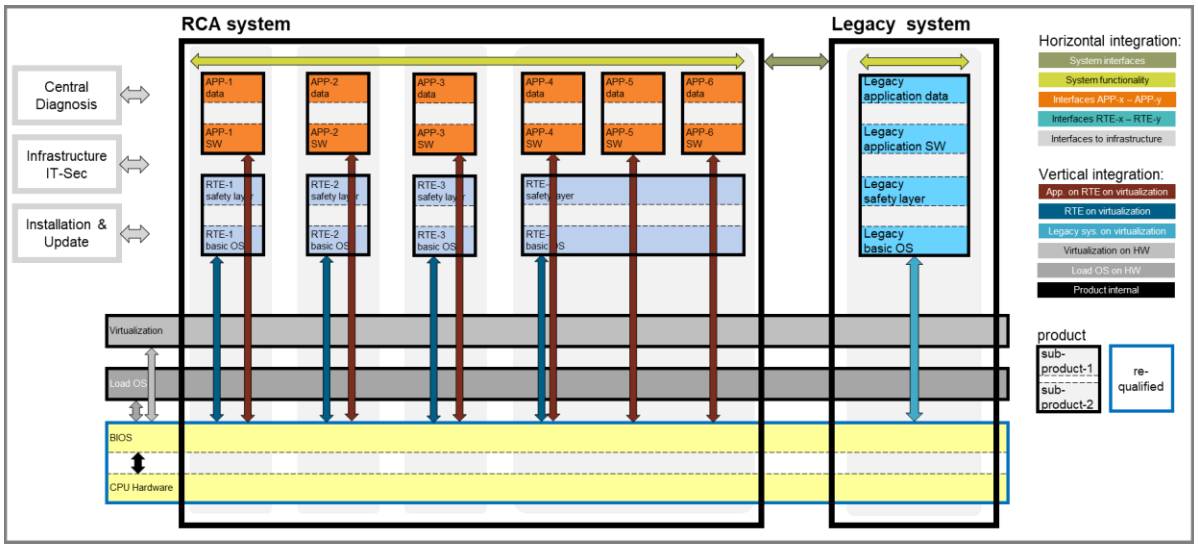


Figure 39 Integrations for change scenario “new COTS hardware”

5.3.2 Integrations for the scenario "new load OS"

Each version of the load operating system needs to be integrated on all used hardware computing nodes and all installations.

The goal is to have a common solution on all used hardware nodes to handle hardware spares in the context of remote installation and update.

If there is a new load operating system version (for the initial upload of first software packages), this load operating system needs to be integrated with the COTS hardware and all the software layers involved in the context of "installation and update of software on the hardware computing nodes." This is a two-step approach, in which the first step is the update of the load operating system itself, and the second step is the update of the upper-layer (e.g., RTE, application) and lower-layer software components (e.g., virtualization, BIOS) by the load operating system.

The load operating system needs to be installed (and maintained) even on inactive hardware spare parts to allow remote installation and update later, when the spare part will be used.

Figure 40 below shows the required integrations when changing the load operating system (highlighted yellow).

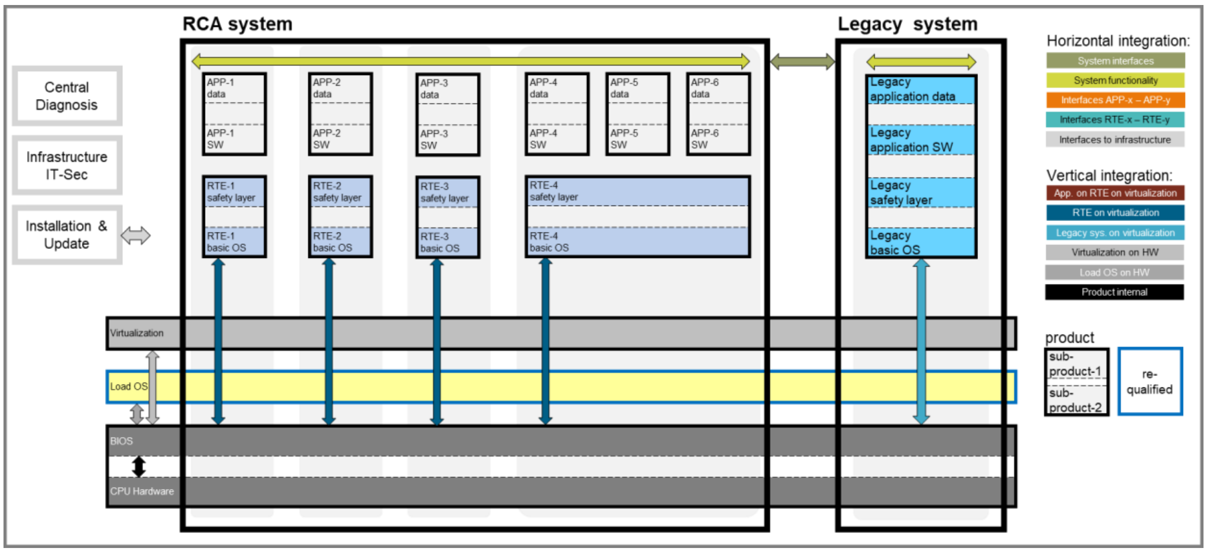


Figure 40 Integrations for scenario “new load OS”

5.3.3 Integrations for the scenario "new virtualization"

Each version of the virtualization solution needs to be integrated on all used hardware computing nodes and all installations where this new virtualization solution is to be used.
The goal is to have a consistent version of the virtualization solution version within the entire data center. This means that a new version is to be gradually qualified and installed for all existing (running) installations.
For this software qualification, it must be shown that all the software layers of all installations running within the virtualization are running as expected on this new virtualization version.
The main scope of this qualification is to test performance, time behavior, and reaction time.

Note:
Whether performance testing by the virtualization is sufficient, or if additional performance testing by the installations themselves is required, is yet to be clarified. The goal is to ensure that the performance and time behavior of the COTS platform are sufficient for the installations running on it.

Figure 41 below shows the required integrations when changing the virtualization (highlighted yellow).

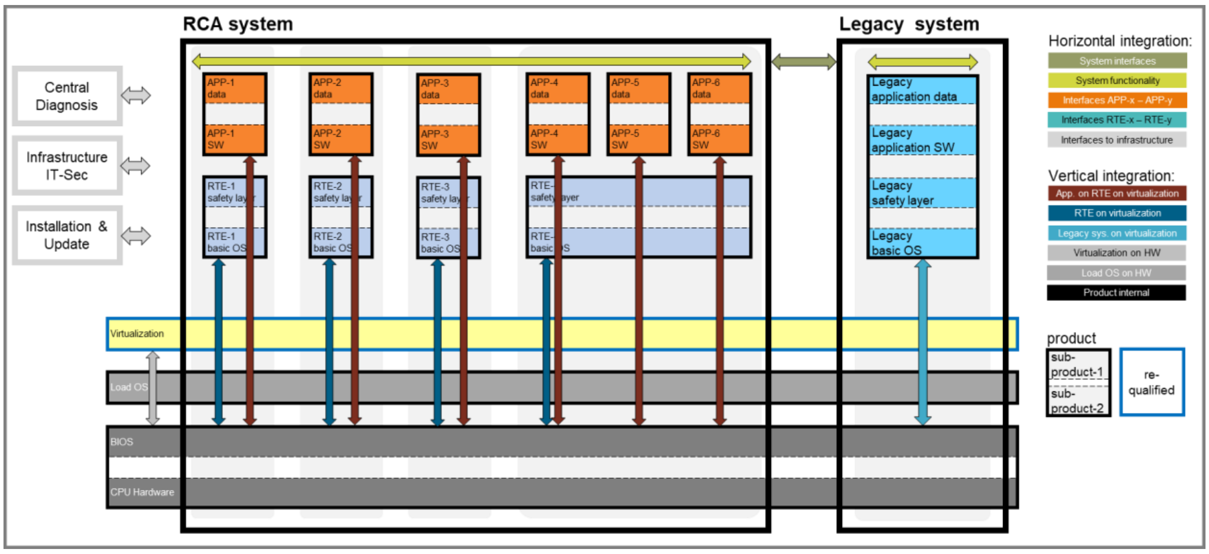


Figure 41 Integrations for scenario “new virtualization”

5.3.4 Integrations for the scenario "new basic OS of RTE"

A new basic operating system leads to the creation of a new version of the RTE. The product-internal integration of the RTE's basic operating system with the RTE safety layer is the responsibility of the RTE vendor.
The new (re-validated) RTE version needs to be integrated with the virtual computing node (vertical integration) and with the neighboring RTEs (horizontal integration).
The affected application APP-3 needs to be integrated with the new version of RTE-3 (vertical integration).
The need for integration with infrastructure components depends on the specific change in the new RTE version.

Figure 42 below shows the required integrations when changing RTE-3's basic operating system (highlighted yellow).

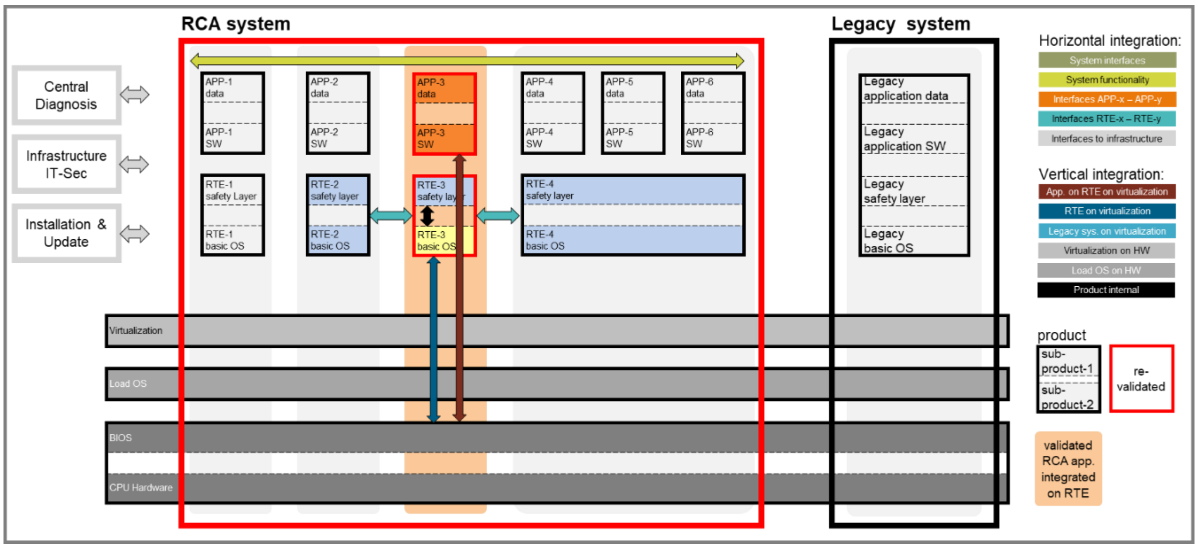


Figure 42 Integrations for scenario "new basic OS of RTE"

5.3.5 Integrations for the scenario "new safety layer of RTE"

A new safety layer leads to the creation of a new version of the RTE. The product-internal integration of the RTE's safety layer with the RTE's basic operating system is the responsibility of the RTE vendor.
The new (re-validated) RTE version needs to be integrated with the virtual computing node (vertical integration) and with the neighboring RTEs (horizontal integration).
The affected application APP-3 needs to be integrated with the new version of RTE-3 (vertical integration).
The need for integration with infrastructure components depends on the specific change in the new RTE version.

Figure 43 below shows the required integrations when changing the RTE-3's safety layer (highlighted in yellow).

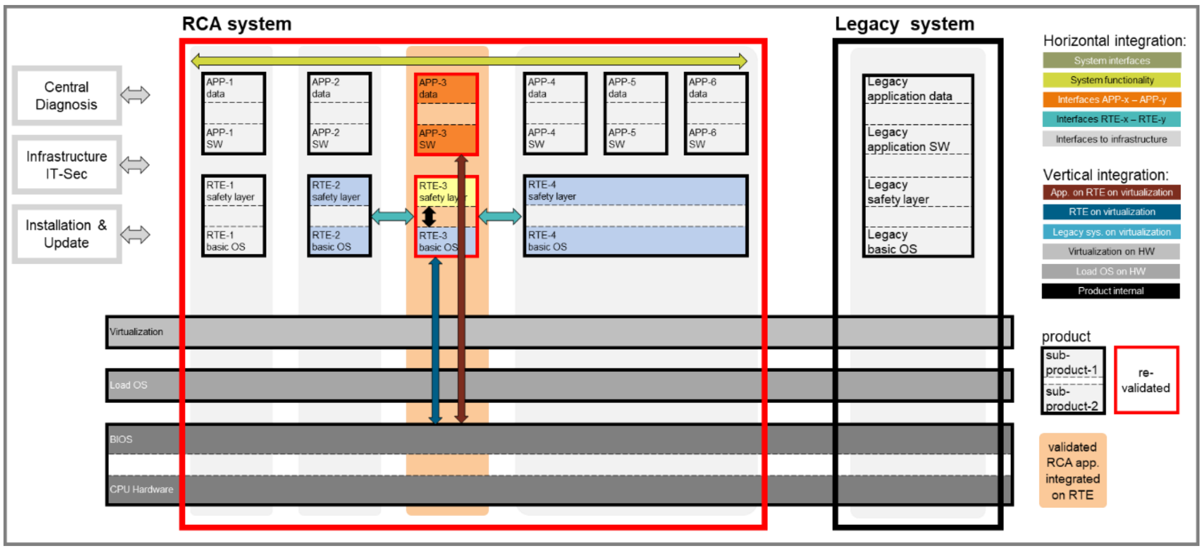


Figure 43 Integrations for scenario “new safety layer of RTE”

5.3.6 Integrations for the scenario "new RCA application"

A new version of an RCA application software leads to the creation of a new version of the application. The product-internal integration of the application software with the application data is the responsibility of the application vendor.
The new (re-validated) application version needs to be integrated with the RTE running on the virtual computing node (vertical integration) and with the neighboring applications (horizontal integration). The need for integration with infrastructure components depends on the specific change in the new application software version.

Figure 44 below shows the required integrations when changing the RCA application APP-3 (highlighted yellow).

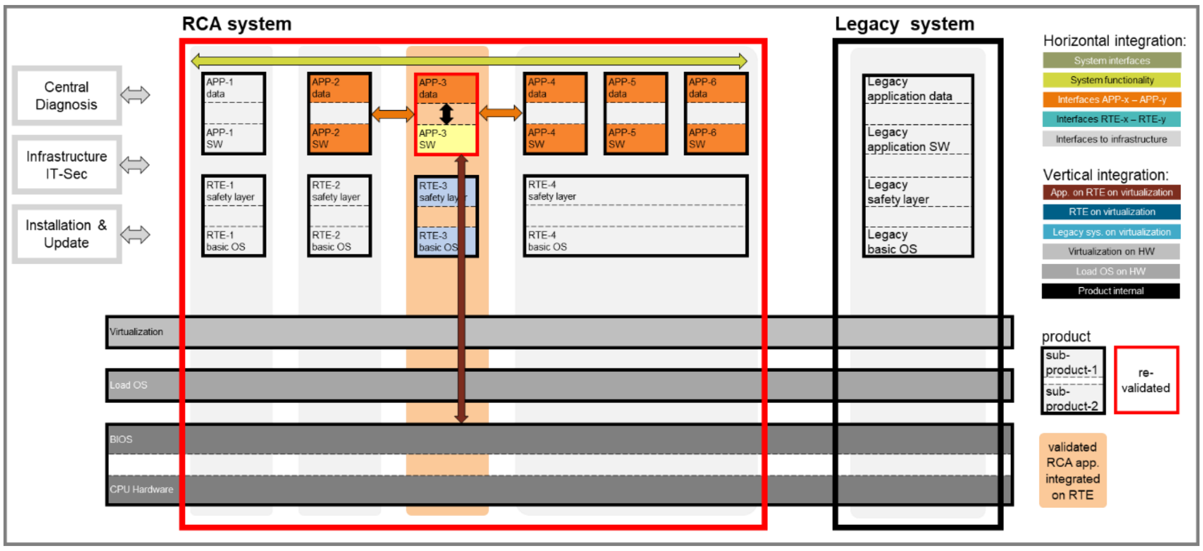


Figure 44 Integrations for scenario “new RCA application”

5.3.7 Integrations for the scenario "new safety layer of RTE with application bundle"

In this case, the modular applications APP-4 / APP-5 / APP-6 are running together in an application bundle". Under this circumstance, the vertical integration of the individual modular applications does not yield a clear result about the application behavior, as integration with the RTE is does not apply to the complete application replica. Vertical integration with the RTE and horizontal integration of the whole bundle needs to be done together with all involved applications (of the bundle). To that end, all suppliers (of RTE-4 and all applications APP-4 / APP-5 / APP-6) must be involved.

Figure 45 below shows the required integrations when changing RTE-4's safety layer (highlighted yellow).

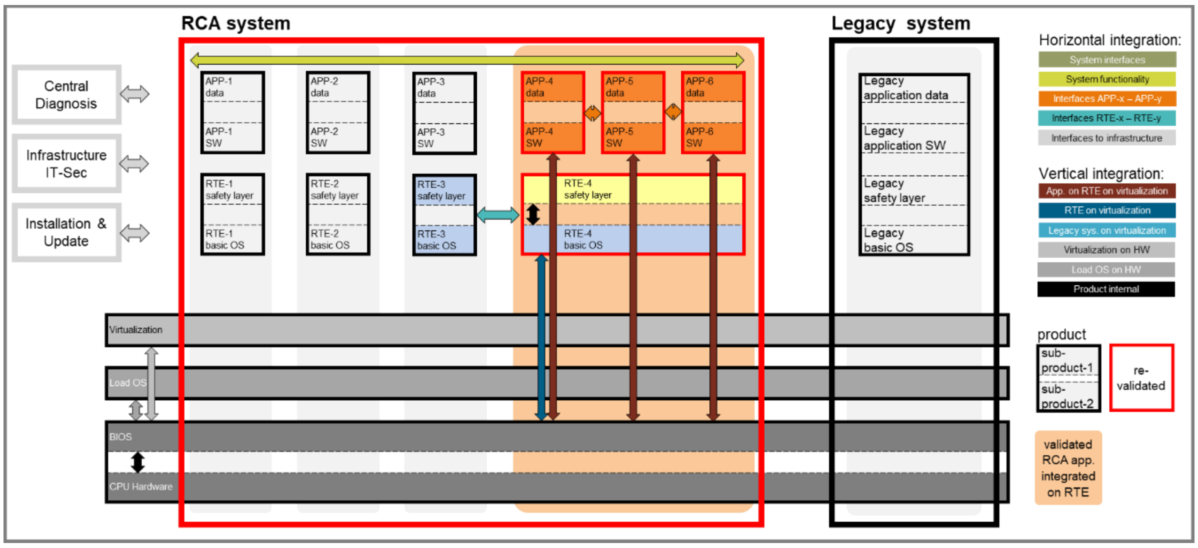


Figure 45 Integrations for scenario "new safety layer of RTE with application bundle"

5.3.8 Integrations for the scenario "new RCA application within application bundle"

A new version of an RCA application software leads to the creation of a new version of the application. The product-internal integration of the application software with the application data is the responsibility of the application vendor.
The new (re-validated) application version needs to be integrated with the RTE running on the virtual computing node (vertical integration) and with the neighboring applications running within the same bundle (horizontal integration) and the neighboring applications running in separated RTEs.
The need for integration with infrastructure components depends on the specific change in the new application software version.

Figure 46 below shows the required integrations when changing the RCA application APP-5 (highlighted yellow).

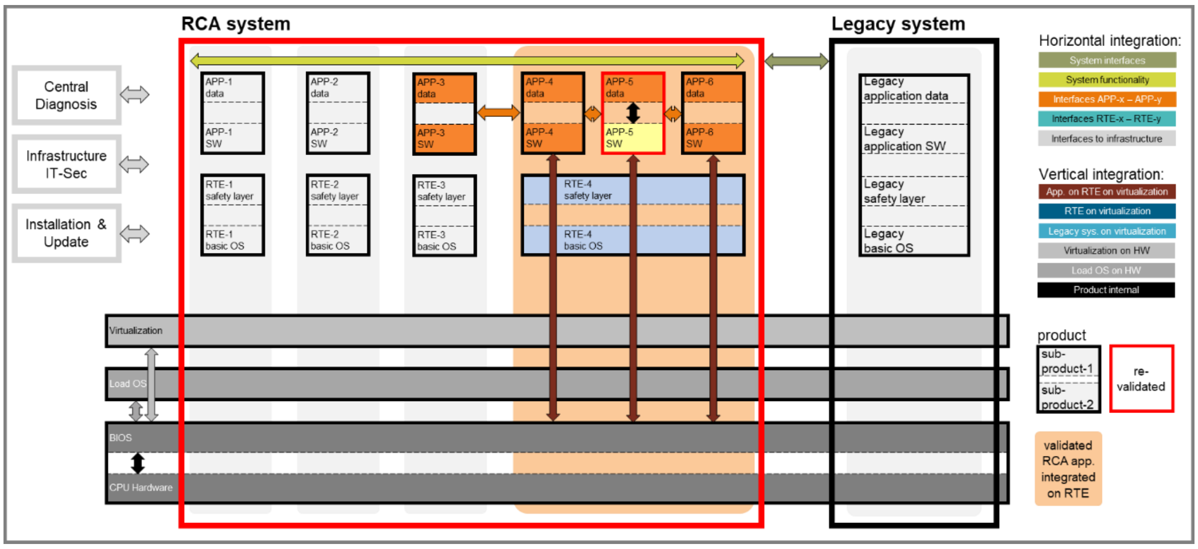


Figure 46 Integrations for scenario "new RCA application within application bundle"

5.3.9 Integrations for the scenario "new legacy safety layer"

A new legacy safety layer leads to the creation of a new version of the legacy system. The product-internal integration of the legacy safety layer with the other sub-products of the legacy system is the responsibility of the vendor of the legacy system.
The new (re-validated) legacy system version needs to be integrated with the virtual computing node (vertical integration).

Figure 47 below shows the required integrations when changing the legacy safety layer (highlighted yellow).

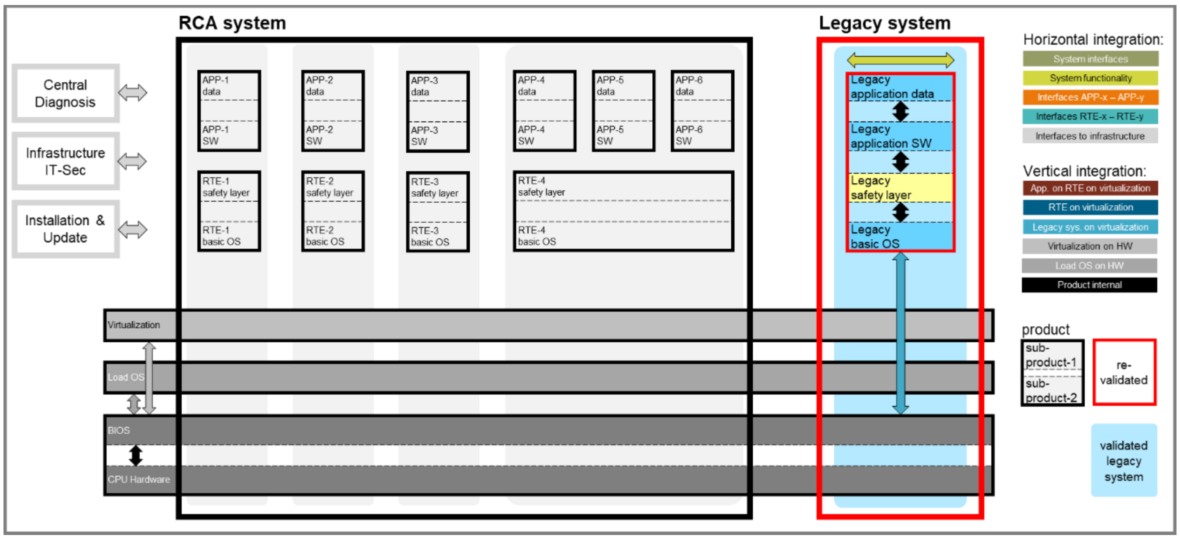


Figure 47 Integrations for scenario "new legacy safety layer"

5.3.10 Integration of infrastructure components (installation and update, IT security, diagnostics)

The dependencies between the infrastructure components and the individual parts of the running installations have to be considered but were not scope of this collaboration.

5.4 Security aspects of system changes

The integrity of a rail automation system is required to safeguard the correct operation of all safety functions. IEC 62443-4-2 [10] requires integrity and authenticity checks on software, configuration, and other information as well as reporting on integrity and authenticity violations.

This is typically achieved by digital signature for software and configurations. Checking on integrity and authenticity is typically realized by a secure boot chain, where each layer verifies the next layer before executing or using it.

The handling of signatures and verification keys must be considered in the design of the RTE, supporting multiple applications from potentially various different vendors.

Security verification and validation:

Software-based systems are particularly subject to vulnerabilities being discovered and potentially exploited. Additionally, the configuration of parts of the system can lead to security risks (unnecessary ports / services activated, unsecure configuration).

Therefore, security verification and validation steps are required before software can be put into operation.

Best practices for security verification and validation for rail automation systems are selected in the X2Rail-3 work package 9. Basis for these activities are the requirements of Practice 5 from IEC 62443-4-1 (Secure product development lifecycle requirements).

However, IEC 62443-4-1 provides only high-level requirements for security verification and validation. This encompasses test activities such as security requirements testing, threat mitigation testing, abuse case testing, static code analysis, attack surface analysis, known vulnerability scanning, software composition analysis and penetration testing, and the independence of testers.

Especially the scope and depth of these activities are not defined. Work package 9 of X2Rail-3 is defining the expected level of security testing and validation activities for rail automation domain.

A corresponding publication of the recommendations regarding security verification and validation is expected in Fall 2021.

5.5 Integration catalogue

Due to the large number of possible integration activities, central capturing of all the processed integration activities (vertical and horizontal) regarding all modular parts of the data center in a sort of "catalogue" is required.

The integration catalogue serves as a central basis for each maintenance activity in the context of software installation, software update, or hardware replacement. To have a clear view on which combinations of versions of the involved parts are integrated and therefore allowed to be used must be clearly defined. If for example a new COTS hardware version is to be used as a spare part in installations that are already running, the installations in which the new COTS hardware spare part can be used (because already integrated) or cannot be used (if not yet integrated) must be indicated for further reference.

This catalogue must be maintained by the infrastructure manager who is responsible for all the integration activities and associated validation/assessment processes.

The integration catalogue is a collection of the complete information on all performed integrations. The input data is provided by:

- **validation processes for safety-relevant integrations** (e.g., integration of App-X running on RTE-Y)
- **qualification** (of COTS hardware, load operating system, virtualization) **for non-safety-relevant integrations** (e.g., integration of the virtualization solution on the COTS hardware)

5.6 SMO experiences in the context of "integration”

The integration of individually assessed modular parts with a generic safety platform and generic applications to achieve a generic overall system is a process that has been established for decades at SMO and in a comparable way at other companies/vendors.

<u>SMO example:</u>
The same generic Simis ECC safety platform is used for several applications: interlocking for DB, RBC for DB, axle counter for DB, level crossing for DB. The same ECC works in similar applications (running on the same ECC API) in Switzerland, Austria, Netherlands, Belgium, Norway, Hungary, etc.

The required integration processes for this case (individually assessed modular parts for different use cases) have been improved continuously, driven by experiences regarding integration dependencies, synergies in tool usage (test system, simulations, engineering tool) and the certification process, to achieve a generic type approval for a generic system (e.g., interlocking system for DB).

The basic points for an efficient overall integration process are:

- The responsible roles require knowledge and competencies to be capable of acting responsibly.
- The scope of responsibility must be precisely defined to be capable of responsibility.
- All integration activities must be coordinated and parallelized in the best and most meaningful way to handle bug fixes as soon as possible (late repairs are cost- and time-intensive).
- Complexity (generic software) must be separated in the best way possible from the customer installation data; the major goal is a generic type approval for the complex generic system used in the same way for several installations with lean data testing for specific installations.
- The generic test environment "Testbahnhof" (test station) must be used as a basis for automated and complete (100% functionality and performance) system testing.

- A harmonized toolchain must be used as a common basis for testing (system test, data test).
- A harmonized toolchain must be used as a common basis for data preparation ("Testbahnhof", customer installations).

The aspects listed below are some refined or additional findings and determinations worth to point out for further clarification in the context of overall integration in a modular RCA approach:

1. The **safety layer** is a generic building set.
   For each application (= user of the safety layer), it is decided by configuration which parts of the safety layer building set are needed (e.g., communication protocols).
   Not every application uses the complete functionality, e.g., an interlocking does not need the safety communication protocols for communication to the train.

2. **Performance testing** by the safety layer test cases does not completely ensure the performance (scalability, throughput, reaction time, etc.) of each application in each application-related situation.
   Each **application must perform application-related performance tests,** especially for worst-case scenarios from the application's point of view in a specific configuration regarding the selected parts of the safety layer building set.

3. The scenarios with **inter-application communication** of different applications running together on the same safety layer need to be **tested by all involved applications as an integrated set of applications.** This also includes performance testing of the applications.

4. Safety layer developments of **performance relevant functionalities** (e.g., communication protocols) should **not be finalized before the integration with the respective application** and real communication partners (including performance testing of the application) has been done successfully.
   Experience has shown that it is a very complex task to define requirements for such a platform properly before integration, especially the non-functional ones.
   Parallel usage of the safety layer in application testing is highly recommended because it is very important for the "early" handling of findings (identified during application testing).

5. **Generic type approval** (e.g., for an interlocking system for DB) with complete re-use is the major goal to achieve lean engineering and engineering data testing for different installations.

6. A generic test environment for application tests ("**Testbahnhof**") is best practice for testing the **complete (100%) functionality** of a complex application in the system context and ensures re-use of the test environment with the **highest degree of automation**.
   Individual installations never use the complete application functionality; so in the context of individual installations, only partial application testing would be possible.
   These characteristics can even lead to more than one "Testbahnhof", depending on the complexity of the application and thus required topologies to activate all features in all possible configurations and relevant situations.
   Especially this environment shall not only support testing on the target hardware with a fully activated safety mechanism but also a lean simulation environment to accelerate numerous (functional) test cases (regression, etc.).

7. For the **"Testbahnhof**" environment**, the knowledge of application experts is necessary to define** the required details for 100% testing of the complete application (from a functional and performance point of view).

8. **Simulations** are used as connected systems to achieve an overall system close to reality (e.g., simulation of object controllers for interlocking testing, simulation of on-board units for RBC testing). It is not practical to define each input at each external interface by test cases itself.

9. The **test system** shall allow flexible configuration regarding the usage of real partner systems/usage of simulated partner systems and shall always support the usage of original configuration/engineering data in the system (or parts of the system) under test.

10. Usage of the original "Testbahnhof" **engineering tool** allows efficient data preparation and ensures data quality and data consistency as well as early engineering tool availability in the case of engineering tool enhancements.
    Development and extension of the engineering tool must therefore be performed in a parallel and coordinated way from the start of application development.

11. Usage of the same **test system** for software testing and data testing ensures a high level of re-use of test tools (e.g., simulations) and test cases for both test scopes.

12. **Data testing** is done by data testing experts (not application experts) with the same test tools and the original data of the customer installation, i.e., with a completely set-up system with all running applications used. This ensures the quality of the complete installation (not only part of it).
    Test cases for data testing are focused straight on the installation data (not on the complex application logic functionality itself)

13. **Fully automated processing of each test** allows very lean regression tests in each integration situation (even if e.g., only the COTS hardware version has changed without any impact).
    Automated repeating of all test cases is easier to do and to argument than deep diving into safety-related argumentation about "what tests need to be repeated (or not) for what reason".

14. The integration activities **shall be parallelized** in the best way possible.
    The best way possible means to start at the right time and in a defined sequence:
    - not too early to avoid immature deliveries
    - not too late to be able to react (repair) as early as possible in the case of any findings
    Late findings are very time-/cost-intensive to repair.

15. **Mutual waiting points** in parallel activities (e.g., parallel testing on the application side and the safety layer side) are, inside one company, never "end-to-start" but in most cases **"end-to-end"** relationships, i.e., to finish the activities.
    E.g., the application side cannot finish final integration of the application on the safety layer before the safety layer itself is provided as a final version.

16. **"End-to-start" waiting points** (the application cannot start because the safety layer is not provided) only occur at "official delivery interfaces", e.g., to other companies, because "parallel integration of draft versions" is not the way two companies are working together. Such "end-to-start" waiting points at official delivery interfaces between different companies impede optimized parallelization and have extremely high claiming potential between the companies involved.

17. **It is not enough to rely on "compatibility statements on paper".**
    It has happened several times that, e.g., new COTS deliveries are declared as being "compatible" on paper and, in reality, the time behavior has changed in a way that affects availability of the safe software running on it. Therefore, each delivery shall be integrated (even if the delivery is "compatible" on paper).
    As applications are becoming more and more complex, compatibility itself turns out to be a complex phenomenon, especially for non-functional aspects.

5.7 Example: cross-vendor integration in the EULYNX context

To illustrate the complete picture of modularized architectures, the Figure 48 below shows, as an example, the current architecture of the existing legacy systems for an interlocking.

Today, in the EULYNX context, only system interfaces are standardized. All system-internal integrations (vertical and horizontal) are the complete responsibility of the legacy vendor.

The standardized system interfaces between the central interlocking logic and the decentralized object controllers are based on the already established RaSTA safety protocol.

Cross-vendor integration is only to be performed for the horizontal interface between the systems (interlocking logic, object controller) for an established technical communication interface and two vendors involved.

From a technical architecture point of view, this integration seems to be quite easy, but has actually turned out to be challenging during realization due to a lack of clear responsibilities and processes. It is hence clear that an even more modular approach involving horizontal and vertical integration requires the definition of responsibilities and processes and suggests, as detailed in the following section, a standardized testing approach.

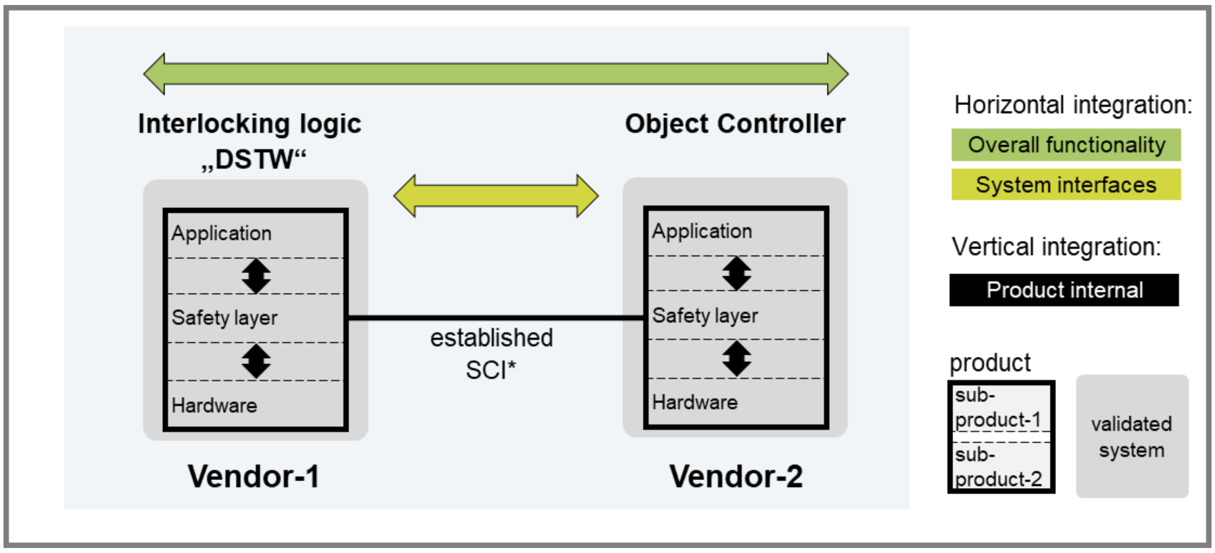


Figure 48 Cross-Vendor integration EULYNX

The EBA Bund Anlage 17 [5] deals specifically with cross-vendor integration and points out its complexity.

5.8 Consequences and proposal for an automated integration test approach

For the automated integration testing of a defined system variant with the modular RCA approach, the scope of the system variant needs to be defined precisely.

It is proposed to define individual system variants with different functional scopes according to the functionality of the individual installations, e.g., interlocking, RBC, interlocking + RBC.

The functional scope defines the needs regarding the environment as test data ("Testbahnhof"), integration test cases, simulations for object controllers, on-board units, etc.

Test automation shall cover overall integrations for the generic system variant(s) and data testing of the installations.

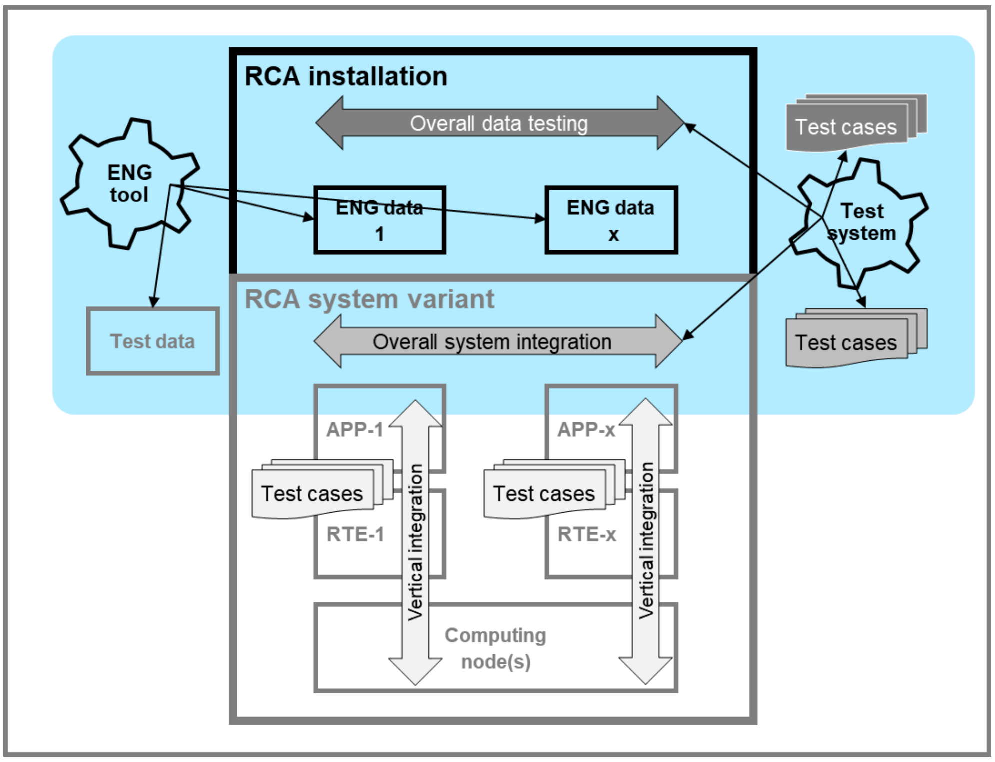


Figure 49 Standardization for automated overall integration

Vertical integration of the involved parts (e.g., RTEs, applications) shall be the responsibility of the individual vendors:

- vertical integration of the basic layers of the computing nodes (hardware, load operating system, virtualization) in the responsibility of the infrastructure manager
- vertical integration of the RTEs running on the virtual computing nodes in the responsibility of the RTE vendors
- vertical integration of the applications running on the RTEs on the virtual computing nodes in the responsibility of the application vendors

For the special "application bundle" constellation, horizontal integration of the bundled applications (APP-4 / APP-5 / APP-6 in Figure 50 below) needs to be considered. All application vendors need to be involved for that matter.

For automated system integration, the horizontal integrations need to be solved within a standardized (= vendor-independent) test environment.

To that end, the following aspects need to be standardized:

- **data interfaces** (CFG and ENG data) with data format and data values
  This data needs to be vendor-independent to create a generic “Testbahnhof” as the test environment.

- **test cases** for integration of the **application interfaces** from a functional and performance point of view
  These test cases must be as modular as the system variants, e.g., there must be test cases for the interlocking functionality and for the RBC functionality.

- **test cases** for integration of the **RTE-RTE-interfaces** from a functional and performance point of view
  These test cases are focused on the safety-related communication protocol for the exchange of application data between the RTEs.

- **test cases** for integration with the **infrastructure components**

- **test cases** for the **overall system functionality** (from a functional and performance point of view)
- **test system** to process the defined integration test cases
- **engineering tool** as a data preparation tool for creation and maintenance of the “Testbahnhof” test environment
- **simulations** of the external components (e.g., object controllers, on-board units) with defined interfaces for manipulation of the behavior by dedicated test cases
  E.g., to manipulate the behavior of a "point controller simulation" by a test case which tests the situation that the point position changes unexpectedly.

Note:
The RTE-related data depends very closely on the RTE solution, e.g., the application-related RTE configuration for cyclic running of the application replicas on the RTE in 2oo3 or 2x2oo2 or a different mode.

Due to this dependency on the RTE solution, however, the RTE data content cannot be standardized. This leads to vendor dependencies in the RTE-related parts of the test environment.

The amount of RTE-related data is very modest; an RTE utilizes approximately 10% of the data of the applications running on the RTE.

Figure 50 below shows the required overall integrations for a generic RCA system variant

For automated data testing of a specific RCA installation, the following aspects need to be additionally standardized:

- Engineering tool: For vendor-independent handling of the ENG data, the engineering tool needs to be standardized.
  The same engineering tool should be used for data preparation for the test environment ("Testbahnhof") and the real-life installations.
  Note:
  The ENG data of the individual applications is provided by the application vendors.
  The usage of different engineering tools would lead to a multiple effort for engineering tool development and an increased effort for the handover of ENG data between application vendors and data testers.
- Tool support by data preparation (engineering tool) for data testing (test system): The engineering tool shall support the automated creation of installation-related test cases for data testing.

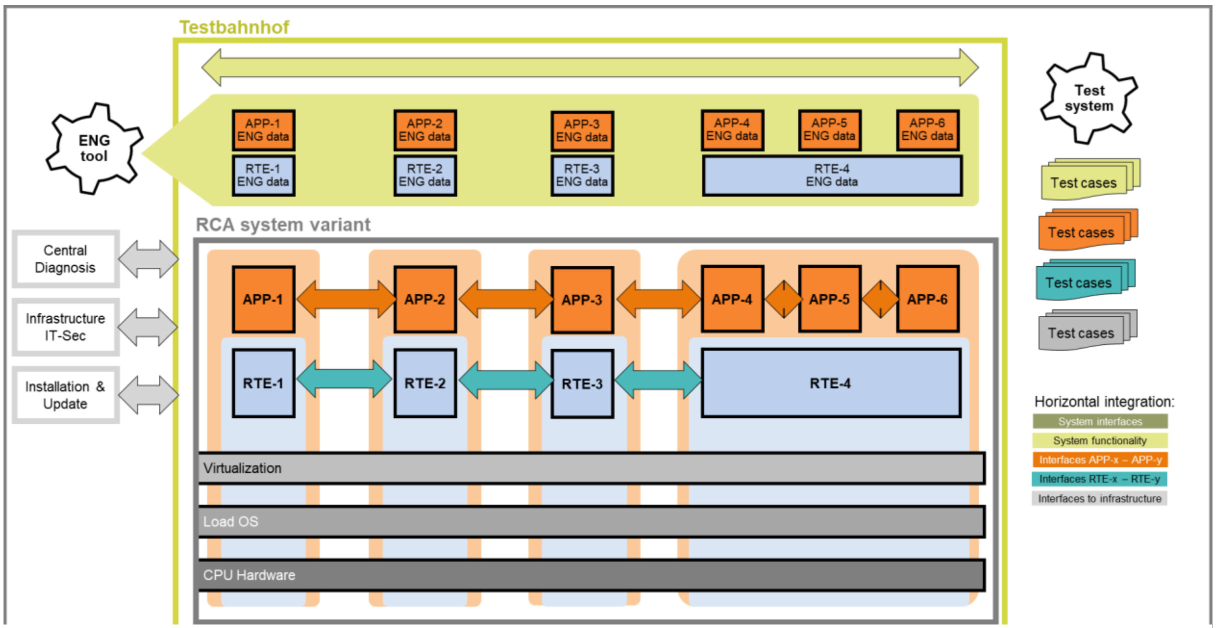


Figure 50 Overall integrations for an RCA system variant

Figure 51 below shows the required integration for data testing of a specific RCA installation.

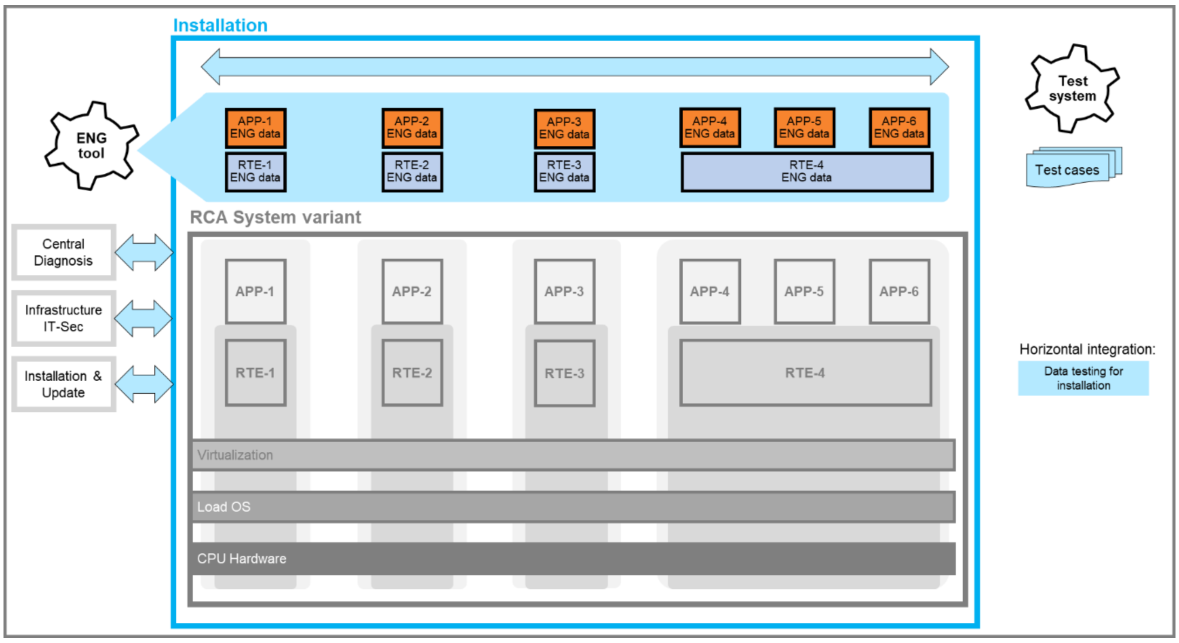


Figure 51 Data testing for RCA installation

6 Effects on operation

Compared to more classical approaches towards system architecture, it is proposed to account for a number of operational impacts as outlined in the following section. Particularly, the concept of "vendor multiplicity" also requires the implementation of additional measures.

6.1 Operational impacts

To assess the impacts on running operations of railway operators and vendors, it is important to consider the complete system lifecycle. Therefore, it is necessary to differentiate between the following lifecycle phases:

- concept/design
- development/setup
- testing/integration/migration/rollout
- homologation
- maintenance/operations
- update/upgrade/extensions

The End-of-life phase is omitted, as it is not the focus of our current considerations.

In addition to assigning operational impacts to lifecycle phases, functional clusters are formed to further distinguish the specific area of impact:

- platform architecture – design, development, integration, and verification of the safe computing platform and its applications
- multi-vendor approach – possibility to source safe computing platform components and applications from several vendors.
  This also includes new vendors coming from non-railway domains.
- obsolescence management – consequences that arise from the usage of COTS components, which in general have a shorter lifetime and shorter market cycles
- cybersecurity – efforts to protect the safe computing platform and the safe applications from security vulnerabilities
- geographical redundancy – distribution of the safe applications over geographically separated data centers to increase availability
- centralization – implications stemming from a large degree of centralization of applications in few data centers
- training and talent access – effort for and cost of hiring new technical experts and training staff on standardized, state-of-the-art technology

As a final step, the potential financial range of operational impacts is indicated, based on internal estimations:

positive potential financial impact:

- EUR 0 < value < EUR 50 million
- EUR 50 million < value < EUR 100 million
- value > EUR 100 million

negative potential financial impact:

- EUR 0 < value < EUR 50 million
- EUR 50 million < value < EUR 100 million
- value > EUR 100 million

This leads to the following combination of lifecycle overview, functional areas, and financial value:

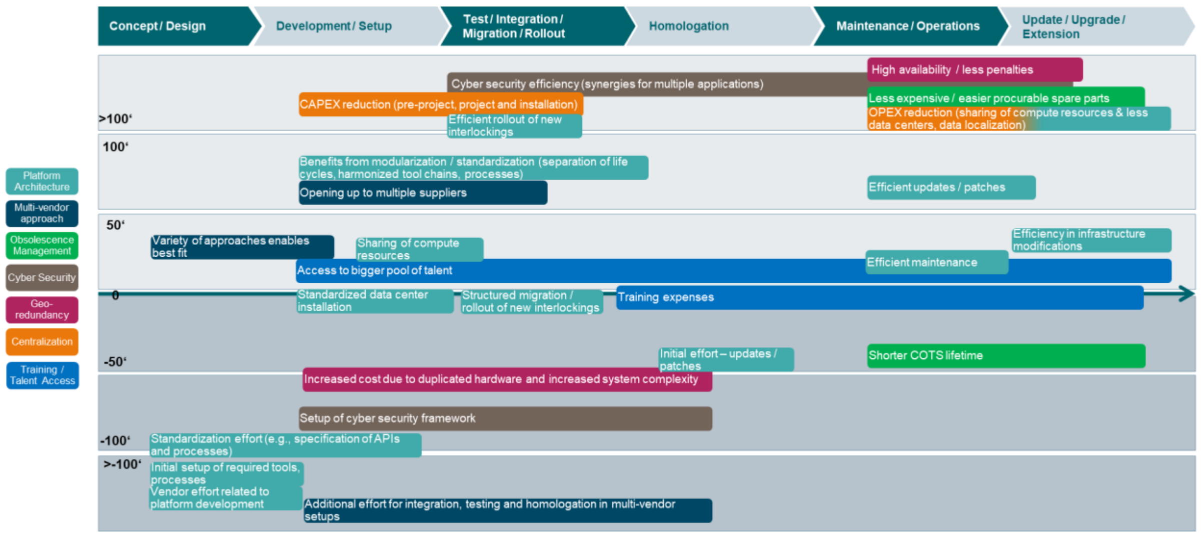

Concept / Design
Development / Setup
Test / Integration / Migration / Rollout
Homologation
Maintenance / Operations
Update / Upgrade / Extension
Platform Architecture
Multi-vendor approach
Obsolescence Management
Cyber Security
Geo-redundancy
Centralization
Training / Talent Access
>100‘
100‘
50‘
0
-50‘
-100‘
>-100‘
High availability / less penalties
Cyber security efficiency (synergies for multiple applications)
CAPEX reduction (pre-project, project and installation)
Less expensive / easier procurable spare parts
OPEX reduction (sharing of compute resources & less data centers, data localization)
Efficient rollout of new interlockings
Benefits from modularization / standardization (separation of life cycles, harmonized tool chains, processes)
Opening up to multiple suppliers
Efficient updates / patches
Efficiency in infrastructure modifications
Variety of approaches enables best fit
Sharing of compute resources
Efficient maintenance
Access to bigger pool of talent
Standardized data center installation
Structured migration / rollout of new interlockings
Training expenses
Initial effort – updates / patches
Shorter COTS lifetime
Increased cost due to duplicated hardware and increased system complexity
Setup of cyber security framework
Standardization effort (e.g., specification of APIs and processes)
Initial setup of required tools, processes
Vendor effort related to platform development
Additional effort for integration, testing and homologation in multi-vendor setups

Figure 52 Overview about lifecycles, functional areas, and financial values

When looking at the combined overview of operational changes, it becomes visible where and to which financial degree and impact will be most likely felt. A general trend that can be observed is that extensive investments are required at the beginning of the lifecycle of the new system approach. Here, for example, the setup of new developments, frameworks, tools, and processes comes to bear that is expected to be leveraged throughout the lifecycle. Hence, more significant potentials for savings can be seen in later phases when the operational ramp-up has been turned into production. Overall, there appears to be a clear positive business case of introducing a SIL4 Data Center including a standardized interface between application and RTE.

6.2 Impacts on total cost of ownership

The following sections describe additional aspects regarding the overall cost situation, which are implied by the top-level objectives.

6.2.1 COTS-based data center

Cost saving factors of a COTS-based data center are:

- reduction of hardware by a high level of integration of several systems running on the same hardware
- reduced costs for hardware maintenance (hardware replacement)
- reduced costs for migration to new hardware (with a defined interface to the RTE)
- usage of non-domain-specific hardware
- reduced obsolescence risk
- reduced costs for spare parts logistics (e.g., storage costs)

It seems beneficial that, apart from standardizing key interfaces, processes and tools also undergo standardization.

If each vital application were to run on a different implementation of common functionalities (e.g., for virtualization, IT security, remote update, geographical redundancy, dynamic resource management), this would result in a complicated configuration and administration of the SIL4 Data Center and consequently increase the effort for integration, installation, and maintenance significantly.

The usage of generic hardware reduces the obsolescence risk and may have a positive cost effect on hardware purchases; also, spare parts logistics are expected to be more cost-efficient over the whole lifecycle. On the other hand, a qualification process for the respective hardware must be passed each time a single new element will be introduced into the existing system.

This implies that a completely new value chain must be defined for successor systems (or spare systems).

With the introduction of the SIL4 Data Center, for the first time there would be a standardized approach to computing infrastructures for safety-critical railway applications for the first time that, by design, facilitates centralization. Such a centralization is already possible today, but only through proprietary means (and application architectures like EULYNX already support this). In Figure 52, the costs and benefits of centralization are related to reducing the number of data centers covering a specific application functionality for all of Germany from a few dozen to a single-digit number. This centralization does not directly depend on the usage of a new RTE and COTS-based hardware.

6.2.2 Development effort for applications

Regarding the development effort for applications, one should differentiate the following cases:

porting of legacy applications to run on the same virtualization layer as other applications (but not yet supporting the target API between application and RTE)
This effort is considered to be quite modest, at least for vendors who today have an RTE solution that can generally run on COTS hardware. In this case, one must also consider that the existing applications (e.g., interlocking logic for DB) must be maintained and probably enhanced with new functionalities in the future to achieve a common data center with complete functionality even for legacy applications.

1. porting of legacy applications to support the target API between application and RTE
   It is assumed that this effort would be substantial. Since it is intended anyway (e.g., by RCA) to introduce a new application architecture for CCS systems, it is further assumed that there is no benefit in porting a legacy application as such to the target API between application and RTE. Instead, one would develop new applications according to the new application architecture and SIL4 Data Center architecture including the target API between application and RTE
2. effort to develop new applications in compliance with the SIL4 Data Center including the target API between application and RTE
   It is assumed that the effort to develop applications is the same with or without compliance with the SIL4 Data Center. Possibly, application development efforts may be slightly reduced through a standardized API between application and RTE, as additional efforts to define such interface by each vendor would be avoided. The main condition for this is a well-described requirement specification considering the new split of safety-relevant functions and allocation to the newly composed elements.

6.2.3 Development effort for RTEs and legacy systems

There will be a great deal of effort to develop a new RTE and to adapt legacy systems to run the legacy systems in the same way in a COTS-based SIL4 Data Center.

To that end, it is important to define the requirements regarding the safety concept of the safety layers (RTE/legacy solution) not too restrictively to enable a common solution for both scenarios for each legacy vendor.

It shall be possible to adapt a legacy solution in such a way that the legacy safety layer

- can be provided as an RTE (with generic API) and
- can continue to be as a legacy solution with legacy API between legacy safety layer and legacy application

Requirements that are too restrictive would lead to the situation, in which legacy vendors need to completely reimplement the RTE and vital application and possibly need to maintain two completely independent implementations (one for legacy systems, one for RCA) – which might conflict with human resource and budget constraints.

Therefore, the RTE design needs to keep a balance between the standardization of interfaces and the flexibility to port existing legacy applications onto that platform with an acceptable effort.

6.2.4 Integration effort

The major challenge is that the main work packages and needed processes in the context of cross-vendor integration have not been defined to date. The responsibility for integration must be clearly defined and clear processes must be introduced. Even for the current EULYNX architecture, in which two vendors are involved for the interlocking logic and the object controllers, this task has not been solved to date.

Due to the architectural modularity and higher vendor multiplicity of RCA-based systems (compared to EULYNX), resulting in many cross-dependencies, the integration will be much more complicated. This can be distinguished as:

- vertical integration of safe applications on an RTE
- horizontal integration of several safe applications running on same RTE

The large number of common interfaces and involved vendors will lead to additional work packages and new roles of responsibility for cross-vendor integration. Existing test cases, tool chains, and test environments that can involve different vendors must be developed and approved in advance.

Particular attention should be placed on the neutral re-usability because it can be assumed that the setup (composition) in each project (interlocking installation) will be different. This can be the case either from the very beginning or will evolve from maintenance procedures where different available devices will replace the original part.

Synergies with existing test cases are expected to be very low due to the disruptive technical approach.

Costs and/or benefits regarding integration, assessment, and homologation should be further investigated in detail to assess the business case over the complete lifetime. Additional focus should be on the definition of accompanying rules and processes.

6.2.5 Operational benefits

The new architecture will have a significant impact on operational aspects which have to be carefully evaluated.

OPEX reductions due to centralization (space, power consumption, maintenance) are expected. Nevertheless, it must be evaluated if these benefits will be reduced in brown-field environments, as the re-use or different use of existing interlocking rooms is not always possible.

CAPEX: Reduced costs for new buildings including studies, planning, installation are expected especially in green-field environments.

Maintenance, updates: A positive impact on maintenance due to increased efficiency and centralized access for corrective measures and patches (cybersecurity, etc.) is expected and must be further evaluated.

Human resources: There are several aspects to consider:

- Training effort is expected to be reduced due to a homogenous installed base. Nevertheless, it is not yet possible to evaluate a clearly positive or negative impact compared to the current situation or alternative technical solutions.
- Talent access: Better access to young talents is expected, as these talents tend to invest into state-of-the-art technologies rather than e.g., relay-technology.
- Operational efficiency. Efficiency is expected to increase due to centralization and operation through regional operation centers.

The discussed operational benefits for infrastructure owners are not only related to the SIL4 Data Center architecture and might as well be achieved through alternative system architectures that enable centralization of the SIL logic.

6.2.6 Project timelines

The cross-vendor integration of a huge number of interfaces will lead to increased project durations because “official” interfaces between different vendors cannot be managed and coordinated in such an efficient way compared to a single vendor in terms of integrated processes, commonly used toolchains, and parallelized workstreams between platform development, application development, and project engineering.

In addition, the cross-vendor alignment of time schedules and delivery interfaces for integration will also lead to increased project durations.

On the positive side, standardization of the interfaces and processes will reduce the integration effort in a multi-vendor scenario. Once the initial setup, processes, and integration responsibility are clearly defined, benefits for individual projects (cost and time savings) or an industrial rollout are expected.

These benefits are not directly related to the architecture but can (or should) be achieved through industrialization of project processes. A limiting factor in signaling projects is the installation of the field elements and the necessary cabling work. This effort is expected to remain unchanged. The portion of this part of the value chain should be estimated and put in relation to the achievable savings.

6.3 Maintenance strategy

To maintain the data center, a defined maintenance strategy is necessary to handle all the individual changes (with different vendors involved) within the overall lifecycle of the data center in the best way possible.

For this purpose, the following aspects need to be considered:

1. maintenance driven by an **unintended need for action**, as e.g., obsolescence issues regarding the COTS hardware or bug fixing within the software layers
2. maintenance driven by **operational business**, as e.g., enhancements and modifications in existing installations
3. maintenance to ensure an **efficient maintenance process** to keep the data center up to date, e.g., changing parts whenever it is possible to update all the parts to the newest available versions as efficiently as possible

For each part of the data center, it must be evaluated if regular updating in shorter time periods and small steps (e.g., steps from version 1.1 via 1.2, 1.3, 1.4 to 2.0) is better than updating in longer time periods with larger steps (e.g., step from version 1.1 to 2.0).

Note:
Compatibility statements of new versions of individual parts refer normally only to the direct predecessor version of the part.
For instance, the release notes only describe the changes from version x.y to version x.y+1 and do not provide information about compatibility between the versions before x.y.
This means that in the case of “skipping some intermediate versions“, all the compatibility statements of each individual intermediate version need to be taken into account.

The vendor multiplicity of all the parts of all installations is a basic factor for definition of the maintenance strategy.
Each change in the data center – for whatever reason – leads to the need for integrations with different vendors involved (involved at least for support).
So, it may happen that a change in a part of vendor 1 leads to support by vendor 2 being necessary for integration purposes.

For such a maintenance scenario with different vendors involved, time scheduling and financing of the integration efforts need to be clarified with all vendors involved.

6.3.1 Hardware obsolescence

During the lifetime of a data center, different COTS hardware devices will be used from different vendors and in different versions from each vendor.
Due to the fact that every COTS hardware needs to be integrated for all affected installations (which are to run on it), a defined maintenance strategy is necessary to efficiently handle different COTS hardware versions as spares for hardware obsolescence purposes.

- One strategy could be to replace hardware devices preferably 1:1 for each individual hardware computing node, i.e., providing spare devices in the exact same types/versions as actively used.
  For this strategy, only the relevant combinations of software and hardware need to be integrated, but it is necessary to have all hardware variants/versions as spares in stock.
- Another strategy could be to use the same hardware version as spares for all installations. To that end, it is necessary to integrate the new hardware version with every installation in advance to be ready to use the hardware version later in maintenance.

The preferred maintenance strategy from an integration costs and spare handling costs point of view will probably depend on the situation regarding automated integration of COTS hardware devices for a specific installation.

6.3.2 Virtualization lifecycle

During the lifetime of a data center, different versions of the virtualization solution will be used.

Each new version of the virtualization solution needs to be "hardened" for the specific usage within the SIL4 Data Center to ensure the required level of reliability and fulfillment of the safety-related requirements of the RTEs.

Due to the fact that every virtualization solution needs to be integrated for all affected installations (which are to run on it), a defined maintenance strategy is necessary to efficiently handle different virtualization versions on all the used hardware computing nodes.

Handling the new versions in the most efficient way will depend on the specific virtualization solution.

6.3.3 Maintenance of individual Installations

If an individual part of an installation needs to be changed (e.g., for safety or availability reasons), it must be clarified and decided how to handle this change.

Such a change in one part leads to consequential integration efforts/costs for other parts.

<u>Example:</u>
If the RTE needs to be changed (for pure RTE reasons), this results in efforts to integrate all affected applications which are running on this RTE.
If ten installations use this RTE, these integrations with all the affected applications need to be done for all ten installations (because an RTE within an installation cannot be replaced without any integration).

To handle such scenarios, different strategies are possible:

- updating the RTE (and as a consequence all integrations) for all ten installations without a deep dive analysis of the question “is it really necessary to replace the used RTE by a new RTE version for each individual installation 1, 2, 3, …?”
- performing a deep dive analysis to decide which installation is affected
  If the RTE change affects a specific RTE functionality, such as e.g., the bundling of applications, then installations without bundled applications do not need to be updated.

Such an analysis can only be done by application experts to decide, if an RTE update is required for the application or not.

Such an analysis is a safety-relevant analysis and needs be confirmed by safety experts for the application.

For such a scenario, the cost situation must be clarified, i.e., who will provide the budget for application analysis, integration, validation, etc. if it is caused by the RTE (from another vendor).

6.3.4 Data center equipment catalogue

For the data center, it is necessary to catalogue every detail of the complete equipment in a sort of equipment catalogue.

In this catalogue, all individual parts (COTS hardware, virtualization, RTEs, APPs, etc.) and their version numbers must be listed for every installation of the data center.
This catalogue is the basis for each decision about maintenance activities within the data center.

Example 1:
The catalogue is the basis for identifying the affected installations in case that a failure in one specific part (e.g., application x in version y) is detected.

Example 2:
The catalogue is the basis for identifying the affected installations in case that a specific version of the COTS hardware is no longer available and needs to be replaced – in the case of spares – by another COTS hardware version.

6.4 Vendor multiplicity

This section describes the situation regarding "vendor multiplicity" in the modular approach considering aspects of standardization and integration.

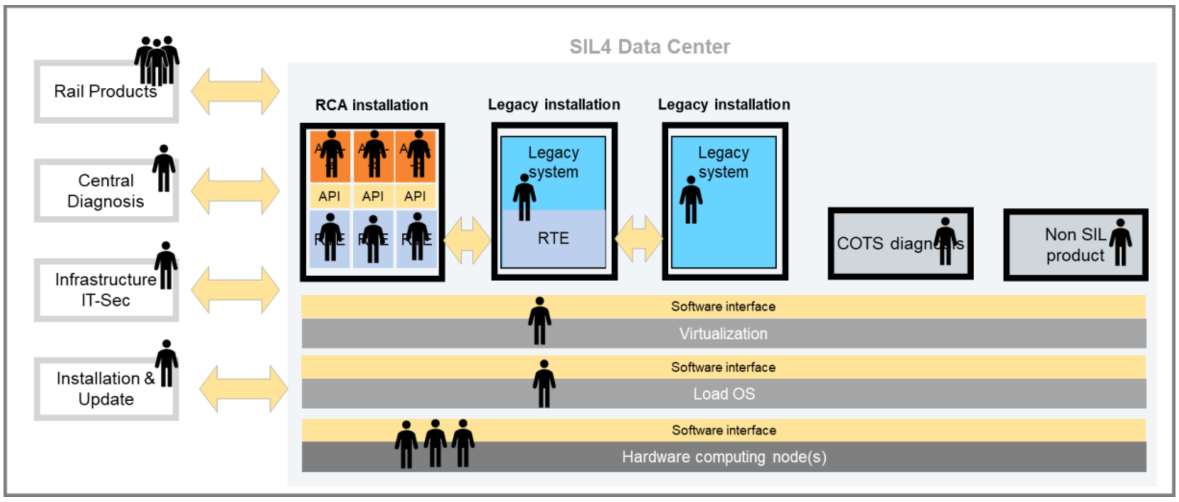


Figure 53 Vendor multiplicity

6.4.1 COTS hardware

From a technical point of view, the COTS hardware can be purchased directly from different vendors if the respective minimum requirements are fulfilled. Also, any change in the hardware (CPU hardware, CPU BIOS) needs to be qualified.

Responsibility for the COTS hardware needs to be defined regarding qualification and monitoring of the supplied hardware, as there is more to it than simply acquiring the hardware.

For hardware qualification, see section 5.3.1.

Note:
Once a hardware setup has been chosen, it cannot be fundamentally changed throughout system lifecycle. The hardware setup needs to be specified in detail from the start and evaluated in its safety concept to meet the homologation prerequisites.

6.4.2 Virtualization solution

For a common virtualization solution within the complete data center, the same solution shall be used for all hardware computing nodes.

The number of vendors that must be involved to realize this common solution depends on the detailed virtualization solution. But in general, it is recommended to keep the solution as simple as possible, because it must fulfill the set of requirements that are imposed by the totality of safety requirements for all RTEs. The virtualization layer should not be safety relevant. This implies that the RTE must detect faulty behavior of the virtualization layer.

Any change in the virtualization software needs to be qualified.

Responsibility for the virtualization software needs to be defined regarding qualification and monitoring of the supplied software, as there is more to it than simply acquiring the virtualization software.

For virtualization qualification, see section 5.3.3.

6.4.3 RTE vendors

From a technical point of view, it shall be achieved that RTEs of different vendors can run on the same COTS hardware, encapsulated by virtual machines.

This is possible by standardizing the interfaces and the behavior of the RTE. This would, in fact, take the vision from OCORA initiative [2] one step further in providing not only a standardized separation of application and platform, but also of RTE and computing nodes.

The basic requirements regarding the RTE behavior of the generic computing platform shall be defined as standard for all RTEs, even for legacy systems.

From a safety point of view, the basic condition is that each safety layer (within the RTE or the legacy systems) shall allow to run any software on the same COTS hardware (which will run separately on their own virtual computing nodes).

In principle, it would be possible to run two or more RTEs from different vendors in one installation (e.g., Frankfurt interlocking), but it must be thoroughly investigated if this is beneficial from the following points of view:

- economic (costs for sourcing two or more RTE implementations)
- development effort (standardized communication interface between the RTEs necessary)
- maintenance (operation of two of more RTE instances with different solutions for hardware usage e.g., as 2oo3 or 2x2oo2)
- For a mixture of different RTE-solutions within one installation, the following aspects currently remain unresolved:
- A standardized safety protocol for the communication between different RTE solutions is necessary.
- An integrity check for the installation is necessary (cannot be solved by the RTE because each RTE is only responsible for a part of the installation).
- Time synchronization between applications is not possible (because such a synchronization is only possible within the context of the same RTE solution).

Also, the application cannot be switched from on RTE to another without additional integration. This will be discussed further in section 4.5.

6.4.4 APS* vendors

Applications can be provided by different vendors if they follow the definition of the standard API.

Due to close dependencies at the interface between RTE and application from a safety point of view, there will be RTE solution-specific rules for application development (e.g., SRACS, SIL4-relevant development tools of category T3).

This means that each RTE solution allows to provide applications by different vendors, but the result of the provided application has a relation to the utilized (= integrated) RTE.

For bundled APS*, the inter-dependencies for the integration of several APS* running together within the same application replica need to be considered.

For details regarding the RTE solution-specific API, see section 4.5.

6.4.5 Legacy installation

By defining a standardized generic computing platform (including a standardized virtualization layer) for all installations, it will be possible to provide a legacy installation as a complete package by one legacy vendor.

The legacy installation needs to be installed on the virtual computing nodes to run together with other installations in the same data center on the same computing nodes.

6.4.6 Non-SIL COTS diagnostics

The non-SIL software for COTS diagnostics (running on the generic computing platform on each hardware computing element) shall be provided for the complete data center by one vendor.

The COTS diagnostic software is a non-SIL software and does not have any dependencies to the installed rail products.

COTS diagnostics are focused on the generic computing platform (virtualization, load operating system, COTS hardware).

6.4.7 Other non-SIL products

Each of the non-SIL products can be provided by a different vendor; it is possible to install several non-SIL products provided by different vendors.

6.4.8 Installation and update infrastructure

Installation and update is a common functionality which is needed for the complete data center, irrespective of the amount and type (RCA, legacy) of installed systems. This common functionality shall be implemented by one solution for the common infrastructure provided by one vendor.

Of course, it would be possible to separate this basic common functionality into individual parts with partial functionality. But to enable the involvement of several different vendors, all the interfaces and dependencies between these individual parts need to be defined as standard. This is not clarified in detail to date.

6.4.9 IT security infrastructure

IT security is a common functionality which is needed for the complete data center irrespective of the amount and type (RCA, legacy) of installed systems. This common functionality shall be implemented by one solution for the common infrastructure provided by one vendor.

Of course, it would be possible to separate this basic common functionality into individual parts with partial functionality. But to enable the involvement of several different vendors, all the interfaces and dependencies between these individual parts need to be based on an international standard (e.g., IEC 62443). Section 3.3 specifies the requirements for an international standard and section 4.12

specifies the proposed architecture for the common IT security infrastructure/shared IT security services.

6.4.10 Central diagnostics infrastructure

Central diagnostics is a common functionality which is needed for the complete data center, irrespective of the amount and type (RCA, legacy) of installed systems.

Central diagnostics should be solved by a common solution, even over several data centers. This common functionality shall be implemented by one solution for the common infrastructure provided by one vendor.

6.5 Development tools

The RTE shall give access to application vendors that do not develop their own RTE. In the context of "application development with different vendors involved", the development tools garners significant attention, in particularly the tools for development and data preparation.

6.5.1 Tools for application development

The API of the RTE shall be as generic as possible and shall not restrict flexibility in the usage of development tools on the application side, i.e., the RTE shall not require specifically defined development tools for the development of the application logic itself. A standardized tool set for application development is not defined to date.

It may depend on the individual needs of an application, if a tool set is required and what kind of tool set is the best solution.
A development tool could support the "style"-conformity of development of exactly this one kind of application when it is developed by different vendors.

Examples:
An interlocking logic application based on the geographical redundancy principle is developed with other tools than an interlocking logic application based on the route table logic principle.
An RBC application logic (with a high degree of mathematical calculations) is developed with yet another specialized development tool.

The benefit of standardized application development tools is mainly driven by the needs of the applications and not by the API of an RTE.

6.5.2 Tools for generation of executable applications to run on a specific RTE solution

Each RTE solution will provide an RTE-related tool set which needs to be used for generation of an executable application, e.g., for compiling, safety measure implantation, etc.

Such specific tools depend on the specific safety solution of the RTE and cannot be standardized across the RTE vendors.

6.5.3 Tools for development of an RTE

For RTE development, a standardization of development tools is not practical because RTE solutions will e.g., arise from already existing solutions of legacy safety layers, which would not support standardized tools across the legacy vendors.

Additionally, the methods and programming principles for RTE development will depend on the details of the RTE-related safety concept, which is not standardized across the RTE vendors.

6.5.4 Data preparation and transformation tool

6.5.4.1 RTE data

Each individual RTE solution will need some configuration and engineering data and such data needs to be:

- created by a tool in a readable format as a basis for data validation
- transformed by a T3 tool from the readable format into the RTE-specific binary format for the running installation

To the current day, neither the format for such RTE data nor the tools for data creation and data transformation have been standardized.

Due to the close relationship between the RTE-specific data definition and the safety concept of an RTE, standardization is only partially possible (e.g., data format).

6.5.4.2 Application data

Each individual application will need the engineering data for the specific installation and perhaps (depending on the application itself) configuration data.

Such application data needs to be:

- created by a tool in a readable format as a basis for data validation
- transformed by a T3 tool from the readable format into the specific binary format for the running installation
  To keep the RTE independent from the format of the application data, the binary format of the application data shall be defined by the application itself.
  Consequently, the data transformation tool (readable format → binary format) belongs to the application.

The data format and tools for data preparation and data transformation (from the readable format into the binary format) are not defined as of today.

The benefit of standardization depends on the complexity and amount of data of the individual applications.

Standardization would be possible:

- for a specific individual application (e.g., APSMOT) from a data content point of view, if the data of the application (e.g., APSMOT) is the same for each solution of this application (and can be provided to any vendor of this application)
- across the different applications with regards to data format and data transformation tools
- for both with regards to data preparation tools

7 Certification and homologation

7.1 Applicable standards

Homologation of a safety-related railway system is based on the standards of the EN 501xx series:

- EN 50126-1:2017
- EN 50126-2:2017
- EN 50129:2018
- EN 50128:2011
- EN 50159:2010

7.2 Homologation

The precondition for homologation is the successful assessment against the requirements in these standards. If the system consists of multiple, independently assessed components, they can be integrated based on their mutual safety-related application rules.

The homologation of applications within a SIL4 Data Center provides the following additional challenges:

- integration of applications on multiple RTEs
- use of commercial off-the-shelf hardware and software
- independently assessed applications sharing the underlying software layers and even hardware

Figure 54 below shows the different variants of applications and underlying layers with different SIL levels.

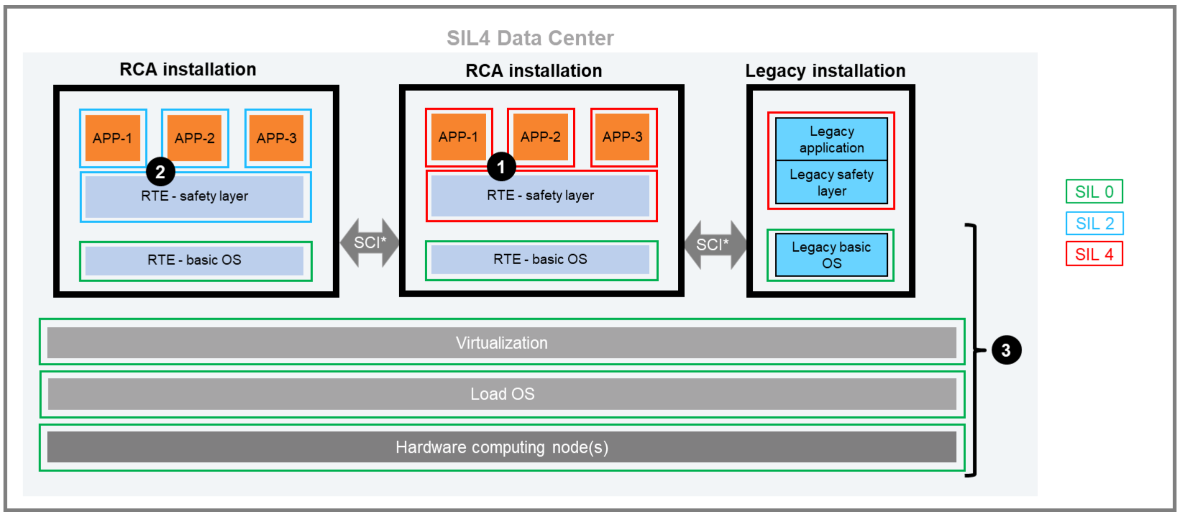


Figure 54 Basic architecture of the SIL4 Data Center

For the variants with the interfaces as marked above with (1), (2), and (3), the relevant standards are:

(1) + (2) EN 50128, Sections 7.3.4. and 7.2.4.9, and EN 50129, Section B3

(3) pre-existing items according to EN 50129/2018, Sections 6.2 and F2.11

7.2.1 Integration on multiple RTEs

The application should be able to run on a range of RTEs. This leads to the requirement to define a common application interface that is implemented by different RTEs.

If a safety platform is used to implement a safety-related application, the application must fulfill the application rules of the safety platform, particularly the safety-related application rules (SRACs). If an application is intended to run on multiple safety platforms, it must fulfill the SRACS of all of these platforms. This "set of possible SRACs" must be defined a priori to enable the parallel development of RTEs.

The challenge when defining the SRAC part of the application interface is to find the right balance between:

- strong rules for the application, hindering application development but allowing divergent and innovative RTEs
- a minimal set of rules for the application, based on given safety principles of the RTE, therefore hindering innovation and competition between RTEs

This is not a fundamental problem, but a question of optimization between conflicting targets that must be addressed in the definition phase of the interface.

It should be noted that the porting of applications to different RTEs may be facilitated by having (a large portion of) RTE-specific SRACs be inherently covered by the tooling (e.g., compiler) provided by the RTE vendor.

In general, an investigation should be made into the extent to which SRACs (or a "superset of SRACs") can be standardized without posing too many limitations on the application side and on the extent of RTE innovation.

7.2.2 Usage of COTS hardware and software components

The assumption is that the data center consists of computing hardware that has not been developed according to and assessed against any safety standards (COTS hardware).

Additionally, the exact hardware the system will run on is not known when the RTE is developed and assessed. Furthermore, it is expected that the type of hardware used in extension or replacement of existing installations will change over the lifetime of the system. Similar requirements must be met for the non-safety-related software stack below the RTE.

The EN 50129:2018 standard explicitly allows the option to include complex items consisting of hardware and/or software that are developed outside of the context of a safety-related development to be used within a safety-related system (Section 6.2). This provides a framework for proving the safety of a system that realizes functional safety on COTS components.

To use such unqualified components within a safety-related function, EN 50129 allows the following strategies:

1. retrograde safety demonstration
2. external negation of the hazardous failure mode
3. demonstration that the failure mode cannot credibly occur due to properties of the component

Option 1 is clearly not feasible, but options 2 and 3 can be used to create a valid SIL4 system based on generic hardware and software components.

To that end, the hazardous functional failure modes of the pre-existing components must be analyzed at the interface without detailed knowledge and assumptions of the COTS components (hardware and software) itself. The necessary measures to handle these failure modes must be implemented in the RTE. This might lead to significantly more failure scenarios than for the traditional use case of a completely known hardware and software structure. The RTE, as the layer that must handle these failures, needs to implement a robust detection and mitigation mechanism to handle these possible failures. The target is to develop an RTE that can control all failure states of the pre-existing software and hardware, relying on as few of these properties as possible.

The remaining assumptions about the COTS platform that are used in the safety argumentation must be such that:

- their violation is sufficiently unlikely by themselves (e.g., creation of codes of significant length by random bit errors) or
- they are provable within feasible installation and maintenance procedures

For the latter, the same consequence as in the previous challenge exists: These rules must be agreed upfront to allow standardized data center operation.

7.2.3 Sharing of hardware and software components between systems

The very idea of a data center is to share resources between independently homologated systems. They might run on the same computing node separated only by virtualization. The necessary proof of isolation could be mainly based on the same measures within the RTE that protect against hazardous influences of the non-safety-related pre-existing components.

Since not all assumptions, that can be made about COTS software, would hold against other safety-related railway systems, additional rules on maintenance and installation might arise.

8 Challenges unsolved today

This section summarizes the biggest obstacles and challenges that were identified, and the major open points which could not be covered within the collaboration. This provides an orientation for the next steps and further investigation in future research projects.

8.1 Cross-vendor integration processes, roles, and responsibilities

A new level of integration will be required for the following reasons:

- integration of RTEs and applications of different vendors
- higher level of modularity of the product scopes (modular RCA architecture)

One consequence will be that integration efforts currently performed inhouse by vendors will shift to the infrastructure manager (or a contracted integrator; new role). This will go hand-in-hand with a shift of responsibilities. Current experience (integration of EULYNX object controllers with interlockings) already shows) significant problems at a lower grade (less vendors, less products). EBA Bund Anlage 17 [5] defines rules but is also limited to EULYNX and does not consider these increased challenges.

With this in mind, an investigation will be required into how the increased efforts and thus costs in this area would relate to the expected benefits of a modular architecture with standardized interfaces and competition among (more) vendors.

8.2 Juridical recording of cross-vendor influences for clear responsibilities

Juridical recording is needed to analyze any system malfunction and to find out which part of the system has caused the error. To that end, it does not make a difference if the malfunction is related to safety or availability.

With a modular RCA approach based on a SIL4 cloud platform, the challenges increase as follows:

- higher level of modularity (increases the number of modules for which juridical recording needs to take place, and this will influence performance by logging of all internal messages, e.g., between all application instances and RTE voter instances on an external juridical recorder)
- mixed responsibility due to a higher number of vendors (who is ultimately responsible?)
- shared resources (virtualization, hardware) by applications of more than one vendor (again, who is ultimately responsible?)

Not only is there a quantitative increase in the need for juridical recording but there is also a new quality: where a vendor used virtualization as part of a complete product, that vendor was responsible for the product as a whole. There was no need to separately handle the virtualization (which is no SIL4 software). With virtualization being shared amongst products of different vendors, the liability of (the vendor of) the virtualization must be clarified.

8.3 Generic process part of API

Standardization of a generic safety solution will be challenging due to contradicting safety concepts of different RTE vendors. Each RTE solution with their own safety concept requires specific safety-related activities on the application side (e.g., usage of RTE-related specific T3 tools for code generation, SRAC argumentation on the application side, integration with RTE, etc.).

The details depend on the specific RTE safety concept solution and cannot be defined as standard.

Necessary delivery packages for application development (test kit, development kit) depend on the specific RTE solution and cannot be defined as standard for different RTE solutions by different vendors either.

These aspects will lead to safety-relevant dependencies between application and a specific RTE solution; for details, see section 4.5.

An "abstraction" of the application from the RTE solution-specific safety-related activities would be theoretically possible, but this would increase complexity in the shift of the application-related safety responsibility to another role.
For this "abstraction", see section 4.5.8.

8.4 Standardization of test and engineering environment for system variants

RCA defines a modular architecture with vendor multiplicity for the modular parts. This leads to different combinations with parts provided by different vendors.

For the handling of this flexibility in the different combinations, the overall activities to integrate the modular parts to form an assessed system need to be automated and standardized in the best possible way.

To achieve vendor-independent automation of integration testing, the complete test environment (test tools, test data, test cases) needs to be standardized. This standardization also affects the data interfaces (of applications and RTEs) and the engineering tool that provides vendor-independent test data as e.g., "Testbahnhof" for system integration.

For integration testing, see also section 5.

In addition to integration testing, the validation activities at the system level need to be solved for the different combinations:

- overall risk analysis
- overall RAMS calculation
- SRAC argumentation

8.5 RTE – bundling of applications within same application replica

As described in section 4.4.4, it will become a relevant consideration to bundle applications and thus will be a specific concern for RTE platform requirements.

The requirements for the application behavior within the bundle regarding application performance, response time, message throughput, and application scheduling need to be defined as basic input requirements for RTE requirements.

To that end, the scalability of the installation size needs to be defined, i.e., whether a huge installation consists of one application bundle for the complete control area of the huge installation or of multiple applications bundles for area parts. This needs to be decided to define the maximum limits of one application bundle.

8.6 RTE – standardized communication between different RTE solutions

RCA in its current phase has defined conceptual interfaces at the application level. It must be pointed out that a standardized safety protocol, carrying the payload of the application-level RCA interfaces, should be defined by RCA. This shall consider the aspects of safety, availability, flexibility (publish/subscribe), and mixed SIL for the communication participants.

<u>Note:</u>
This standardized safety protocol is necessary for the scenario of "applications of one installation are running on different RTE solutions". If all applications of one installation are running on the same RTE solution, then the RTE-internal communication mechanism is sufficient for exchanging data between the applications of the same installation.

See also section 4.4.1.

8.7 Virtualization requirements for the standardized operation of RTEs

In order that all safety layers can run on the same virtualization solution, the safety aspects of the virtualization solution need to be standardized so that they fit all possible safety layer solutions.

The requirements for the virtualization solution must be defined considering all possible safety layer solutions (safety argumentation, time behavior).

The requirements shall consider the following:

1. the required mechanism from a safety point of view for each safety layer to check and ensure the correct distribution of the safety-relevant software parts to separate hardware computing nodes
   This requirement is safety-relevant, i.e., safety must be achieved via a combination of mechanisms provided by the non-safety-relevant virtualization and overlayed checks by the safety layers of the RTEs.
2. the time behavior, e.g., to ensure the time behavior of the individual software parts running in parallel and synchronously in separate instances to achieve a safety layer that runs stably
   This requirement is required for availability.
   If this is not fulfilled by the virtualization software, then there will be safe reactions by the safety layers which massively affect availability (including stopping an installation).
3. the reaction time, e.g., to ensure the reaction time of the individual software parts running within the virtualization
   This requirement is required for availability.
   If this is not fulfilled by the virtualization software, then there will be safe reactions by the safety layers which massively affect availability (including stopping an installation).
4. the safety-related rules regarding the dynamic resource management
   This requirement is safety-relevant; for details see section 8.8.

Today, these requirements are not defined, and it is not yet clarified which virtualization solution those requirements could fulfill.

8.8 System – dynamic CPU resource management

The "dynamic CPU resource management within one installation" aspect is not deeply analyzed today.

Dynamic CPU resource management means that the individual software parts of an installation (e.g., safe voting, safe clock, application replica) are not mapped statically to the individual hardware computing nodes, but dynamically, e.g., depending on the dynamic availability of the individual CPU resources.

Due to safety relevance, it is recommended to handle all changes in CPU resource management for every installation of a safety layer in the same way to the greatest extent possible.
This means resolving the issue for a mixture of installations (legacy/RCA) running in different configurations (2oo3, 2x2oo2).

Additionally, the following aspects need to be considered:

- The network components (switches, routers) need to be involved in any change in the overall configuration regarding the used hardware. For instance, if communication with a connected system as an object controller is no longer to be done by the safety layer on one hardware (before) but by the safety layer on another hardware (new).
- The update infrastructure for installation and update needs to be involved in any change in the overall configuration regarding the used hardware to ensure overall consistency and control about "what installation is running on which hardware computing elements".
  Which software is executed on which COTS hardware device must be defined at all times, i.e. the software must not disappear in the "jungle" of available COTS hardware devices.

Since the mapping to CPU hardware resources is a basic condition for the safety argumentation of each safety layer solution, this "dynamic" aspect will have a safety-related impact on the involved non-safety-related parts like virtualization and update infrastructure.

From a system availability point of view, such a dynamic resource management is not needed, because each system shall, of course, already provide the required availability with static resource management by local and even geographical redundancy. "Static" does not necessarily mean "inflexible", because changes are nevertheless possible but only under the responsibility of the configurator of the system and not the system itself.

For more information, see also section 3.1.13

8.9 “Non-safe RTE" for non-safety-related applications

Today, the technical solution for an RTE for "non-safety-related applications running within the same data center with same redundancy principles" is not defined.

Non-safety-related applications do not need safety mechanisms like voting and safety protocols, but redundancy is needed for the running system (local redundancy, geographical redundancy) and for communications.

Whether a kind of "non-safe RTE" is useful for such non-safety-related applications or if each non-safety-related application is developed completely separately, i.e., independent from any RTE, needs to be determined.

The needs of non-safety-related applications may differ completely from those of safe applications from the point of view of application processing time, data handling in running mode, data handling in communication.

9 Conclusion

The research collaboration between Siemens Mobility and Deutsche Bahn within the sector initiative “Digitale Schiene Deutschland” addressed and tackled many aspects for a future SIL4 Data Center over its total duration of one year.

Firstly, the RCA/OCORA White Paper [3] along with its high-level objectives were analyzed, and the implications identified. Not all the objectives can be fulfilled in combination, which necessarily leads to a prioritization that is described in this report. Furthermore, the new objective of geographical redundancy was identified within this process. This has been deemed necessary to keep the availability up in a more centralized data center structure, even in case of catastrophic events like earthquakes or flooding. Additionally, a review of existing requirements based on EULYNX concepts was performed. EULYNX is rather focused on external system communication interfaces that are used today for control, command and signaling systems. On that basis, it is recommended to extend EULYNX' scope by the aspects relevant for a SIL4 Data Center.

The architectural approach towards the SIL4 Data Center is similar but slightly different compared to the approach outlined by RCA/OCORA in the White Paper [3]. The layers for COTS hardware virtualization, RTE, and the idea of standardized interfaces are the same, but in addition, the SIL4 Data Center approach strongly suggests the implementation of a standardized interface between the virtualization layer and the RTE. The virtualization component is to be identical for all computing nodes of the SIL4 Data Center to enable the deployment and redeployment of safe applications according to the respective needs required by a variety of operational situations within the data center. By following this concept, the SIL4 Data Center would enable the operator to benefit from a far more flexible basis. Within this approach, the safe and non-safe parts of the architecture are strictly separated. The safety-relevant layers are reduced to the upper parts of the RTE, the specific applications and their configuration data. Therefore, the safe components can be designed accordingly and the impact of SRACS on non-safe layers can be minimized.

In other words, this modular approach supports the implementation of various products delivery by different vendors. It furthermore allows diverse lifecycles of different products. That means for instance that the installed COTS hardware can be replaced more frequently than a safe application. The report lays out several cases of how to replace components without interrupting safe operation of the overall system.

On the topic of safe communications between individual applications, the report describes the concept of protocol gateways, and outlines how these gateways can be used to implement safe communication protocols. For Deutsche Bahn and Siemens Mobility, it has been crucial to define that the safety-related part of the protocol is intended to belong to the RTE and will be configured only if needed by the application.

An especially important insight reached through the collaboration is that the SIL4 Data Center must include a migration concept for legacy applications. It is therefore recommended that the architectural approach enables running of legacy applications in parallel to the future RCA-compliant applications in the same SIL4 Data Center. This can be achieved, for instance, by implementing legacy applications directly on the virtualization layer, and apart from the standardized RTE. To support the migration concept outlined in this report, the interfaces and functionalities of the SIL4 Data Center (e.g., load tool, update mechanisms, and juridical recording) are to be standardized fundamentally and according to EULYNX, as legacy applications need to be adapted to these standardized interfaces eventually.

Since command, control and signaling systems have partially hard timing constrains, the timing behavior will be a defining system requirement. To achieve the necessary response times, the additional penalty of inter-RTE communication can be avoided by bundling intricately connected applications on one RTE as an application bundle. In such an application bundle, all applications together represent one application replica which is running on one single RTE. Yet this instrument comes at a cost as it complicates the integration of the system. Applications within the same application bundle cannot be updated without recertification of the complete application bundle, which is again likely to result in negative effects on project partitioning and sourcing.

The definition of the APIs was not part of the scope of this project, but as a result of the collaboration of Deutsche Bahn and Siemens Mobility, it became very clear that this is a challenging topic not only due to the complexity of the programming interface between RTE and application or RTE and virtualization layer, respectively. From today's perspective, it is even more complicated to standardize the interfaces between runtime components and administrative tools like, for instance, IT security, installation and update tools, central diagnostic tools and data, juridical recording, data interface of the RTE, and RTE-related tools that instrument the application code with safety mechanism according to the safety concept of the RTE.

Several concepts have been evaluated regarding the design of geo graphical redundancy. It has been found necessary to compare the number of required data center locations on the one hand and the availability of safety applications in catastrophic situations on the other. It is important to point out that the utilization of existing SIL4 Data Centers in combination with replication of the safe application state over the different sites instead of having local replication may represent a promising option for implementing geo graphical redundancy. In this regard, the so called 'split brain' issue (maintenance resulting in inconsistencies between individual but overlapping parts of the overall system) has been briefly discussed as well. It is necessary to resolve this issue either per individual application or generally for the whole SIL4 Data Center architecture, but it was decided to not investigate this problem further as part of this research collaboration.

To sufficiently consider security aspects of the SIL4 Data Center's architecture, the collaborating partners rely on the EU research project X2Rail-3, as there is an existing work package on cybersecurity. As it is based on IEC 62443, it is directly applicable to serve the requirements of the SIL4 Data Center.

Besides addressing architectural characteristics of the SIL4 Data Center, the complexity of integration and testing has been assessed thoroughly. This is one of the most important aspects of this report, since the complexity in cross-vendor integration and testing is expected to grow super-linearly according to the number of involved vendors. Therefore, this report lists several scenarios on which integration and testing must be executed.
It cannot be overstated that attention must be paid to the fact that there is a shift in responsibilities for integration and testing, as it will move from the supplier towards the infrastructure operator, who will have to take over the role of a system integrator. It is therefore necessary to maintain control over the overall system complexity, which can be achieved through a sound and standardized integration concept, including standardized processes, interfaces, tools, and highly automated test cases.

Additionally, the partners listed constraints from a certification and homologation perspective, as the SIL4 Data Center's modular approach towards architecture is expected to lead to significantly greater efforts to achieve certification and homologation. For instance, if one safe application is to run on several RTE implementations, it is to be certified for each individual combination. Moreover, application segregation needs to be ensured, so that applications do not have side effects on the operation of safe applications.
It is therefore of the upmost importance to the collaborating partners to find a middle ground between flexibility and effort to address this issue.

Finally, this report shows that the outlined standardization, modularization, and automation of the SIL4 Data Center approach represent a major opportunity for the future of rail operations. However, it comes at a significant price as serious investments need to be undertaken to move from the conceptual level to reality and to reap the expected benefits. In addition to that, the multi-vendor concept will create greater transparency regarding the cost structures of such large systems.

As outlined at the very beginning of this report, Deutsche Bahn and Siemens Mobility aimed to jointly investigate a key element of future rail operations. Though there may still be a long way ahead, based on the results portrayed in this research report, the partners are convinced that if designed and implemented the right way, the SIL4 Data Center is of great importance for enabling an increase in capacity, quality, and efficiency of rail operations. It is incredibly important to avoid the summarized pitfalls and effort drivers and further focus on feasibility and practicability of implementation. In this spirit, Deutsche Bahn and Siemens Mobility look forward to continuing to implement the SIL4 Data Center.

10 Annex

10.1 Terms and Abbreviations

Term / Abbreviation	Description
2oo3 principle	2-out-of-3 principle This is a fail-safe system in which two ***application replicas*** (including an ***RTE***) are running as a safe solution, each of the replicas on one ***hardware computing node***. If one of the instances fails, the system will stop. To increase availability (e.g., in case of hardware failures), an additional third RTE instance is running, leading to a 2oo3 system as the normal mode. If one of the three instances fails, the other two instances are running in 2oo2 mode which is still safe but has decreased availability.
2x2oo2 principle	2 times 2-out-of-2 principle This is a fail-safe system in which two ***application replicas*** (including an ***RTE***) are running as a safe solution, each of the replicas on one ***hardware computing node***. If one of the instances fails, the system will stop. To increase availability (e.g., in case of hardware failures), the same 2oo2 system exists a second time, leading to a 2x2oo2 system as the normal mode. If one of the instances fails, the containing 2oo2 fails, and this leads to 1x2oo2 mode which is still safe but has decreased availability.
API	Application Programming Interface of the ***RTE*** This is the interface of the RTE which includes all specifications and rules to be fulfilled by the application. It defines mechanisms for interactions between multiple applications, and services of the RTE, the kinds of calls or requests that can be made, how to make them, data formats that should be used, the conventions to follow, etc.
Application	Computing software that carries out the business logic, and runs on top of the ***RTE***.
Application bundle	Package of applications running together as one (1) ***application replica*** This means that the ***RTE*** voter does not see the bundle-internal data flow, only the output of the bundle is relevant for safe voting. For such an application bundle, an ***Application Manager*** is necessary.
Application Manager (AppMan)	Software part of the RTE which provides all needed services to run several applications together as bundle with time-critical inter-application communication.
Application replica	Instances of the same application All application replicas run synchronized in parallel. From an outside-world point of view, they act "as one": they receive one input and provide one output. Output of an individual application replica is not safe itself, but safety is achieved by voting of the outputs of several application replicas.

Term / Abbreviation	**Description**
APS	Advanced Protection System APS* is a placeholder for any of the APS components (example: SL = Safety Logic) APS is a SIL4 subsystem within an ***RCA*** based architecture.
APS-FOT	APS Fixed Object Transactor. This is an application within the ***RCA*** architecture.
APS-MOT	APS Mobile Object Transactor. This is an application within the ***RCA*** architecture.
APS-MT	APS Movement Authority Transactor. This is an application within the ***RCA*** architecture.
APS-OA	APS Object Aggregation. This is an application within the ***RCA*** architecture.
APS-SL	APS Safety Logic. This is an application within the ***RCA*** architecture.
APS-SM	APS Safety Manager. This is an application within the ***RCA*** architecture.
ATO	Automatic Train Operation
Basic OS	Basic operating system of the ***RTE*** and a subproduct of the RTE (e.g., Linux)
CCS	Command, Control and Signaling
CISO	Chief Information Security Officer This is the role of the overall manager responsible for IT security within an organization.
Clock	Safety relevant functionality of the ***RTE*** to provide a safe, strictly monotonous clock for any timing-relevant mechanism of the RTE, e.g., the cyclic triggering for the ***application replicas***
Computing node	Abstract term for a computing resource used by an ***application replica*** with its ***RTE*** It can be a hardware computing node (computer, CPU, CPU core) or a virtual computing node.
Configuration data CFG data	Each software layer may define for itself which kind of data can be configured to use the software layer in the required way. Such data is called "configuration data". Configuration data belongs close to the software and should not be confused with ***engineering data***, which is defined for a specific installation.

Term / Abbreviation	Description
CONNECTA	CONtributing to Shift2Rail's NExt generation of highly Capable and safe TCMS and brAkes https://cordis.europa.eu/project/id/730539
COTS	Commercial off-the-shelf
Engineering data ENG data	Data defined for a specific customer installation. This data depends on the specific topology and is defined specifically for each installation.
EULYNX	European initiative by infrastructure managers to standardize interfaces and elements of the signaling systems See https://www.eulynx.eu
FRMCS	Future Railway Mobile Communication System
Geographical redundancy	Redundancy where ***application replicas*** of an installation are distributed over sites at different geographical locations If one site fails (e.g., due to a blackout) the other site continues to run. This means, a failure of one site does not lead to a system failure.
Hardware computing node	Dedicated computer (using one to many CPUs) or dedicated CPU (using one to many cores) or dedicated CPU core It is up to the ***safety layer*** of the ***RTE*** to select which of the hardware computing nodes is provided/used to fulfill the safety case. All hardware computing nodes should be ***COTS*** components.
HIDS	Host-based Intrusion Detection System
Horizontal integration	Integration of different applications of an installation, each application running on an ***RTE***.
IAM	Identify Access Management
IXL	Interlocking
Legacy Application	A currently existing application, e.g., the interlocking logic for DB A legacy application is running on a legacy platform which includes a legacy ***safety layer***.
Legacy solution	A currently existing solution, e.g., the interlocking logic for DB A legacy solution is a legacy application running on a virtual computing node in the same SIL4 Data Center next to an RCA solution.

Term / Abbreviation	Description
Load OS	Extended bootloader which is preinstalled on the ***hardware computing nodes*** and is usually executed directly after (re)booting the computing platform The main functionality of the load OS is the secure bootstrapping (authentication and data integrity) of software components, typically the virtualization solution and the kernel.
NIDS	Network Intrusion Detection System
OCORA	Open CCS On-board Reference Architecture https://github.com/OCORA-Public/Publication
Product	An application package which is provided by a vendor. A product can consist of several ***subproducts***.
RaSTA	Rail Safe Transport Application RaSTA is a unified safety-related communication protocol for the standardized communication between different rail systems.
RBC	Radio Block Center
RCA	Reference CCS Architecture https://ertms.be/workgroups/ccs_architecture
RTE instance	Each ***RCA installation*** is implemented by several RTE instances running distributed on several ***hardware computing nodes.*** With each RTE instance (on separated hardware computing nodes), standalone ***application replicas*** are running. Example: In an RTE using the ***2oo3 principle***, the RTE is running in three instances on three hardware computing nodes, e.g., COTS CPUs) each with an application replica of the same application.
Runtime environment (RTE)	Application platform providing basic services to the application to support ***SIL4****, such as:* • implementation of the fail-safety principle (redundancy support, voting of output) • programming model • general services (like diagnostics) From a commercial point of view, it is at the same time a product which can be provided by different vendors.
Safety layer	Software layer which accomplishes and guarantees the safety integrity level of hosted applications It is expected to comprise functions that ensure that the computing platform meets the railway standards EN 50126, EN 50128, EN 50657 and EN 50129 in their application up to SIL4 by covering:

Term / Abbreviation	Description
	• integrity checking • fault tolerance mechanism (e.g., achieved through redundancy and voting, or through diagnostic functions) • synchronization and communication services related to safety (e.g., needed for fault tolerance) • hardware and software monitoring as needed in safety context The specific technical implementation of functional safety is up to the ***RTE*** vendor and legacy vendor and agnostic towards hosted applications. ***Legacy solutions*** also have their own safety layers, but within a legacy solution, the API between safety layer and application is defined specifically for the needs of the legacy application.
SCI*	Standardized Communication Interface in a ***EULYNX*** architecture SCI* is a placeholder for the different interfaces (examples: SCI-P, SCI-TDS).
SDI*	Standardized Diagnostic Interface in a ***EULYNX*** architecture SDI* is a placeholder for the different diagnostic interfaces (examples: SDI-P, SDI-TDS)
SIEM	Security, Information and Event Management
SIL SIL4	Safety Integrity Level Safety Integrity Level 4
SMI*	Standardized Maintenance Interface in a ***EULYNX*** architecture SMI* is a placeholder for the different maintenance interfaces (examples: SMI-P, SMI-TDS)
SRAC	Safety-Relevant Application Condition These are safety-relevant conditions (= rules) provided by a safety-relevant software which need to be fulfilled by the (human and technical) user of the safety-relevant software. The fulfillment of these SRACs must be evaluated by the user from a safety point of view and the fulfillment must be confirmed by validation and assessment.
Subproduct	Part of a ***product*** A subproduct is provided as part of the product by the same vendor (as the product). The division of a product into subproducts makes sense in case that the operational handling of individual parts of the product is done in different ways due to different lifecycles or different safety integrity levels. Example: The product ***RTE*** is divided into the subproducts • ***SIL4 safety layer*** • ***non-SIL basic OS***

Term / Abbreviation	**Description**
T3	Tool class T3 as defined in EN 50128
Virtualization	Virtualization solution to encapsulate individual installations running on the same ***hardware computing nodes*** in separated virtual machines It is part of the computing platform and a standalone product provided by the virtualization vendor.
VM	Virtual Machine
X2Rail-[1..3]	Shift2Rail https://shift2rail.org/, https://projects.shift2rail.org/s2r_ip2_n.aspx?p=X2RAIL-3

10.2 References

Reference ID	Document and Link
RCA Architecture [1]	RCA (Reference CCS architecture) https://ertms.be/workgroups/ccs_architecture
OCORA initiative [2]	Open CCS On-board reference architecture https://github.com/OCORA-Public/Publication
OCORA-40-004-Gamma-CP-Whitepaper [3]	Approach for a generic safe computing platform for railway applications https://github.com/OCORA-Public/Publication/blob/master/01_OCORA%20Gamma%20Release/40_Technical%20Documentation/OCORA-40-004-Gamma_Computing-Platform-Whitepaper.pdf
OCORA-40-013-Gamma-CP-Requirements [4]	Generic safe computing platform high-level requirements https://github.com/OCORA-Public/Publication/blob/master/01_OCORA%20Gamma%20Release/40_Technical%20Documentation/OCORA-40-013-Gamma_Computing-Platform-Requirements.pdf
EBA Bund Anlage 17 [5]	EBA Bund - Anlage 17: Integration Hersteller übergreifend https://www.eba.bund.de/SharedDocs/Downloads/DE/Infrastruktur/AllgemeineVorschriften/27_EIGV/224_27_EIGV_GIuV_Anlage_17_Integration_herstelleruebergreifend.pdf;jsessionid=9A88DA5A8AD1517B0F356FC77BD2A666.live11292?__blob=publicationFile&amp;v=3
NIS directive [6]	NIS Directive https://www.enisa.europa.eu/topics/nis-directive
Cybersecurity Act of the European Union [7]	Regulation (EU) 2019/881 of the European Parliament and of the Council of 17 April 2019 on ENISA (the European Union Agency for Cybersecurity) and on information and communications technology cybersecurity certification and repealing Regulation (EU) No 526/2013 (Cybersecurity Act) https://eur-lex.europa.eu/eli/reg/2019/881/oj
CSA Cloud Controls Matrix (CCM) [8]	CSA Cloud Controls Matrix (CCM) https://cloudsecurityalliance.org/research/cloud-controls-matrix/
RFC 3647 [9]	Internet X.509 Public Key Infrastructure - Certificate Policy and Certification Practices Framework https://datatracker.ietf.org/doc/html/rfc3647

Reference ID	Document and Link
RCA Architecture [1]	RCA (Reference CCS architecture) https://ertms.be/workgroups/ccs_architecture
OCORA initiative [2]	Open CCS On-board reference architecture https://github.com/OCORA-Public/Publication
OCORA-40-004-Gamma-CP-Whitepaper [3]	Approach for a generic safe computing platform for railway applications https://github.com/OCORA-Public/Publication/blob/master/01_OCORA%20Gamma%20Release/40_Technical%20Documentation/OCORA-40-004-Gamma_Computing-Platform-Whitepaper.pdf
OCORA-40-013-Gamma-CP-Requirements [4]	Generic safe computing platform high-level requirements https://github.com/OCORA-Public/Publication/blob/master/01_OCORA%20Gamma%20Release/40_Technical%20Documentation/OCORA-40-013-Gamma_Computing-Platform-Requirements.pdf
EBA Bund Anlage 17 [5]	EBA Bund - Anlage 17: Integration Hersteller übergreifend https://www.eba.bund.de/SharedDocs/Downloads/DE/Infrastruktur/AllgemeineVorschriften/27_EIGV/224_27_EIGV_GIuV_Anlage_17_Integration_herstelleruebergreifend.pdf;jsessionid=9A88DA5A8AD1517B0F356FC77BD2A666.live11292?__blob=publicationFile&amp;v=3
NIS directive [6]	NIS Directive https://www.enisa.europa.eu/topics/nis-directive
Cybersecurity Act of the European Union [7]	Regulation (EU) 2019/881 of the European Parliament and of the Council of 17 April 2019 on ENISA (the European Union Agency for Cybersecurity) and on information and communications technology cybersecurity certification and repealing Regulation (EU) No 526/2013 (Cybersecurity Act) https://eur-lex.europa.eu/eli/reg/2019/881/oj
IEC 62443 [10]	IEC 62443 Industrial communication networks – Network and system security

10.3 List of Figures