Recommendation on Interfaces to be standardized

Author(s)	Martin Thomas (I-NAT-GST-CCS) , Betül Sögütlü , Patrick Marsch , HERBELET Guilhem (SNCF VOYAGEURS / DIRECTION DE L'INGENIERIE DU MATERIEL / CIM - ETF 3) , SPINDLER Markus , DOMANOWSKI Frank , Steffens, Sonja (SMO RI R&D F IL) , Traeger, Axel (SMO RI ML TC PLM) , MOTTOLA, Giuseppe Diodato , Klapka Štěpán RNDr.
Abstract	The System Pillar domain Computing Environment has the aim to perform an operational and system analysis related to computing environments in the further evolved rail operations system. In this first public deliverable of the domain, a recommendation on interfaces to be standardised is spoken out, based on an analysis of user stories related to computing environments, their expected benefit, and the assumed cost related to introduction of standardised interfaces needed to support the user stories.
Classification	EURAIL-internal
Status	In Progress
Version	0.8
Revision	184584
Last Change Date	14.03.2024
Copyright	The reproduction, distribution and utilisation of this document as well as the communication of its contents to others outside EUROPE's RAIL without express authorisation is prohibited. Offenders will be held liable for the payment of damages. All rights reserved by EUROPE'S RAIL in the event of the grant of a patent, utility model or design.

INFO: History table is not displayed, because this document is in status **doc_inprogress**.

RULE: History table is not displayed, in statuses: { draft doc_open doc_inprogress doc_contentApproval doc_contentDecision }

CONTACT: For more information contact Administrator

Content

1 Introduction 4
2 Terminology 5
3 User stories related to Computing Environment 9
4 Benefit assessment of the user stories 16
5 Computing Environment architecture and possibly standardized interfaces 21
5.1 Architecture 21
5.2 External Diagnostic, Logging, Orchestration and IT Security Interface(s) (I1) 22
5.3 Hardware Abstraction Interface (I2) 23
5.4 Virtualization Interface (I3) 23
5.5 Basic Integrity Platform Independence Interface (I4) 24
5.6 Safety Platform Independence Interface (I5) 24
5.7 Possible deployment options 25
6 Feasibility assessment and mapping of user stories to interfaces 27
7 Cost assessment of the interfaces 33
7.1 Methodology applied 33
7.1.1 Specification cost 34
7.1.2 Implementation cost 34
7.1.3 Testing Cost 34
7.1.4 Integration and authorization cost 34
7.1.5 Legal related cost 34
7.1.6 Operation & maintenance cost 35
7.1.7 Release Life Cycle Management cost 35
7.2 Cost assessment of Interface I1 35
7.3 Cost assessment of interface I2 38
7.4 Cost assessment of interface I3 40
7.5 Cost assessment of interface I4 44
7.6 Cost assessment of interface I5 48
8 Conclusion on the interfaces to be standardized 53
8.1 Proposed Next Steps 55
9 Open Points 56

References

1 - RCA/OCORA, Generic Safe Computing Platform - Specification of the PI API between Application and Platform

available under https://github.com/OCORA-Public/Publications/blob/master/09_OCORA%20Release%20R4/OCORA-TWS03-030_SCP_Specification_of_the_PI_API_between_Application_and_Platform.pdf

1 Introduction

The System Pillar domain Computing Environment has the aim to perform an operational and system analysis related to computing environments in the further evolved rail operations system, where the term Computing Environment refers to an environment on which secure functional applications up to SIL4 (relating to CCS applications or any other applications used in the context of rail operation) are run, comprised of hardware and software (i.e., the runtime environment), see also Section 3. It shall be noted that the domain covers both onboard and trackside computing environments, with a distinction in those cases where the respective requirements or business aspects strongly differ.

As one major additional outcome, the domain is also expected to provide a recommendation regarding which interfaces in the context of computing environments in the rail system should be standardized.

In this first public deliverable of the domain, exactly the aforementioned point is addressed, and a recommendation on interfaces to be standardized is spoken out, based on an analysis of user stories related to computing environments, their expected benefit, and the assumed cost related to introduction of standardized interfaces needed to support the user stories.

Note: all assessments made, and conclusions drawn in this document assume that proprietary Functional Systems will continue to exist indefinitely i.e., eventual conversion of vendor specific Functional Systems has not been considered. The standardized Computing Environment shall be able to host proprietary Functional Systems (with minimal change) as well as newly developed Functional Systems (fully based on the new standardized interfaces).

The document is structured as follows, reflecting exactly the analysis process conducted in the domain:
In Section 3, key terminology related to computing environments is defined, which is used through the document;
In Section 4, user stories related to the handling of computing environments in the rail operation context are introduced;
In Section 5, the user stories are assessed w.r.t. their expected benefit;
In Section 6, inspired by the identified user stories, a possible high-level architecture of a generic computing environment for the rail operations is introduced, along with the definition of multiple interfaces that would potentially be standardized;
In Section 7, the user stories are mapped to the identified possible interfaces, and assessed w.r.t. their feasibility;
In Section 8, the identified interfaces are assessed w.r.t. the expected cost impact of having these standardized (considering the whole life cycle from specification to operation and maintenance);
In Section 9, the document is concluded, with a recommendation on which interfaces should be standardized, based on the assessment of the user stories w.r.t. benefit and feasibility, and the assessment of the interfaces w.r.t. cost;
In Section 10, identified open points are listed.

2 Terminology

In the remainder of this document, the following terminology is used:

Title	Description
Application Execution Environment	The Application Execution Environment refers to the combination of Runtime Environment and Safety Environment.
Application Execution Environment	The Application Execution Environment refers to the combination of Runtime Environment and Safety Framework. Previous definition: *The Application Execution Environment refers to the combination of Runtime Environment and Safety Environment.*
Application Layer	The Application Layer contains Functional Applications that constitute Functional Systems.
Application Layer	The Application Layer refers to the functions provided by the Functional Systems. These functions are implemented as Tasks within Functional Applications. Previous definition: *The Application Layer contains Functional Applications that constitute Functional Systems.*
Basic Integrity Platform Independence Interface	The Basic Integrity Platform Independence Interface I4 (Interface 4) is used to perform a basic integrity platform independence with the applications. In other words, this API is an interface limited to non-safety functionalities between runtime environment and applications.
Compartment	A Compartment is a consistent, integrated entity comprising exactly one Runtime Environment Instance, Safety Environment Task Replicas of at most one Safety Environment, and Functional Application Task Replicas of its respective Functional Applications. It can be deployed on either a Physical or a Virtual Computing Element.
Compartment	A Compartment is a consistent, integrated entity comprising exactly one Runtime Environment Instance, at most one Safety Environment Replica, and Task Replicas of its respective Functional Applications. It can be deployed on either a Physical or a Virtual Computing Element. Previous definition: *A Compartment is a consistent, integrated entity comprising exactly one Runtime Environment Instance, Safety Environment Task Replicas of at most one Safety Environment, and Functional Application Task Replicas of its respective Functional Applications. It can be deployed on either a Physical or a Virtual Computing Element.*
Compartment Execution Environment	The Compartment Execution Environment refers to the combination of Physical Computing Element and Virtualization Environment.

Title	Description
Computing Element	The Computing Element provides physical or virtual compute resources.
External Diagnostic, Logging, Orchestration and IT Security Interface(s)	The External Diagnostic, Configuration & orchestration Interface I1 (Interface 1) comprises communication-based interfaces between rail systems and central infrastructure components as diagnostics, IT-security services and remote update.
Functional Application	A Functional Application is a comprehensive set of self-contained software functions, assumed to be provided as one product by a single vendor. Depending on its role in the overall function provided by the Functional System, it has a specific SIL (BIL up to SIL4) assigned (in-line with total FS SIL definition).
Functional Application Task	A Functional Application Task implements part of the functionality provided by a Functional Application. Depending on its role in the overall function provided by the Functional Application, it has a specific SIL assigned (in-line with total FA SIL definition). It may run replicated in multiple Compartments as FA Task Replicas.
Functional System	A Functional System is a comprehensive set of self-contained Compartments, assumed to be provided as one product by a single vendor. Depending on its overall function, it has a specific SIL assigned.
Functional System Deployment Rules	The Functional System Deployment Rules comprises all necessary information for deploying the respective Functional System onto specific approved Compartment Execution Environment(s). These deployment rules are compiled as part of the FS integration process and are part of each integrated, tested and qualified/approved Functional System along with its FS Compartments and all necessary approval documentation.
Hardware Abstraction Interface	The Hardware Abstraction Interface I2 (Interface 2) provides an abstraction of all technology layers above from the specific hardware used below, enabling easy replace ability of commercial of-the-shelf hardware procurable from a well-sized market of hardware vendors. Note: This is not really an interface, but rather a compatibility list of allowed hardware incl. CPU, memory, etc.
Hardware Layer	The Hardware Layer contains the actual Physical Computing Elements providing the compute resources to the platform.
Instance	An Instance is a specific realization of any entity.
M-out-of N	In an M-out-of-N configuration, a functional system consists of N redundant replicas, out of which at least M must be functionating correctly for the overall functional system to operate.

Title	Description
Management Compartment	The management compartment provides the deployment, configuration and monitoring services through the standardised interfaces. The details will be specified in the future documents.
Operational Interfaces	The I0 is the sum of all operational interfaces used from Functional Systems (as eg. an RBC) to communicate with other Functional Systems (as eg. an IXL). Expamles for these set of interfaces are the Eulynx Interfaces (SCI-xx) or interfaces like Euroradio or TSI-standardized interfaces.
Orchestration Interface	This interface is used to manage (monitor, control, diagnose, configure) the virtual computing environments. It only exists if a Virtualisation Interface is present. OI is part of I1.
Physical Computing Element	The Physical Computing Element refers to the physical device providing compute resources.
Replica	A Replica is a specific realization of any entity in a cluster of peers used for composite fail safety and/or availability. Replicas of the same entity always run in distinct Compartments deployed to distinct Computing Elements.
Runtime Environment	The Runtime Environment refers to the software needed to provide the services of the Runtime Layer in a single Compartment.
Runtime Layer	The Runtime Layer refers to the system services (e.g., application and computing resource orchestration, monitoring of the Functional Applications and the Application Execution Environment, tracing and logging, communication services that are not related to safety, security means incl. authentication, encryption, key storage, etc.) and the communication stack for information exchange between Functional Applications running on the same Computing Environment and with external entities. It may also include an operating system.
Safety Environment	The Safety Environment refers to all Safety Environment Tasks needed for a Functional System.
Safety Environment	The Safety Environment refers to the software needed to provide the services of the Safety Layer in one single Compartment. Previous definition: *The Safety Environment refers to all Safety Environment Tasks needed for a Functional System.*
Safety Environment Task	A Safety Environment Task implements part of the functionality provided by a Safety Environment. Depending on its role in the overall function provided, it has a specific SIL assigned (in-line with total SE SIL definition). It may run replicated in multiple Compartments as SE Task Replicas.

Title	Description
Safety Framework	The Safety Framework refers to all Safety Environment Replicas needed for a Functional System. Previous definition: *new term*
Safety Layer	The Safety Layer implements all the technical safety principles related to fulfilling the requirements of EN 50126, EN 50716 (formerly 50128), EN 50129, EN 50159 (e.g., composite fail safety, fault tolerance, voting mechanisms, redundancy mechanisms for availability, safety communication layers etc.) that are needed to enable the execution of Functional Applications up to SIL4.
Safety Platform Independence Interface	The aim of introducing Safety Platform Independence Interface I5 (Interface 5), is to be able to implement platform independent Safe Functional Applications (up to SIL4) i.e., applications, based on a generalized abstraction between the application logic and the system interfaces, will run unchanged on different platform implementations.
Task	A Task implements part of the functionality provided by a Functional Application. Depending on its role in the overall function provided by the Functional Application, it has a specific SIL assigned (in-line with total FA SIL definition). It may run replicated in multiple Compartments as Task Replicas. Previous definition: *A Functional Application Task implements part of the functionality provided by a Functional Application. Depending on its role in the overall function provided by the Functional Application, it has a specific SIL assigned (in-line with total FA SIL definition). It may run replicated in multiple Compartments as FA Task Replicas.*
Virtual Computing Element	The Virtual Computing Element refers to virtually provided compute resources with computing resource guarantees.
Virtualisation Environment	The Virtualisation Environment contains all software needed to provide (multiple) Virtual Computing Elements on a single Physical Computing Element.
Virtualisation Interface	The Virtualization Interface I3 (Interface 3) is used to provide a standardized interface above the virtualisation layer so that applications or higher platform layers are independent of a specific implementation of the computing hardware.
Virtualisation Layer	The Virtualisation Layer contains mechanisms that are able to provide Virtual Computing Elements needed to run multiple Compartments on a single physical hardware underneath.

36 items found

3 User stories related to Computing Environment

In the following, identified key user stories related to computing environments are listed. All user story descriptions follow the same pattern, i.e.

- initially it is stated, who benefits from the user stories (e.g., rail infrastructure manager, railway undertaking and/or supplier);
- then the user story itself is formulated;
- and finally the rationale of the user story is provided.

Also, the derivations of the user stories from the Common Business Objectives for the future rail system are shown.

SPT2CE-18 - Minimize overall dependencies

As a supplier, rail infrastructure manager or railway undertaking, I would like to minimize dependencies among Functional Application, Runtime Environment and Hardware, in order to minimize obsolescence related risks and costs.

SPT2CE-18 has been derived from the following Common Business Objectives:

ID	Title
SPT1RS-198	automate lifecycle processes, independent lifecycle
SPT1RS-190	Changeability and upgradeability(1)
SPT1RS-188	Changeability and upgradeability(3), simpilified integration
SPT1RS-227	Flexible incorporation of innovations
SPT1RS-194	independent lifecycle, simple exchange
SPT1RS-218	modularity
SPT1RS-197	overall CAPEX/OPEX optimisation(1)
SPT1RS-220	overall CAPEX/OPEX optimisation(2)
SPT1RS-229	seamless and selective exchange of components under production
SPT1RS-221	standardized architecture(1)
SPT1RS-247	standardized architecture(2)

11 items found

SPT2CE-19 - Aggregate multiple Functional Applications on the same Instance of a Computing Platform

As a rail infrastructure manager or railway undertaking, I would like to aggregate multiple Functional Applications , possibly from different suppliers (which may also be railways), on the same Instance of a Computing Platform in order to minimize required space, energy dissipation, cost and/or latency and increase resilience w.r.t. supply chain issues.

SPT2CE-19 has been derived from the following Common Business Objectives:

ID	Title
SPT1RS-200	availability: less assets
SPT1RS-190	Changeability and upgradeability(1)
SPT1RS-211	efficient energy use(1)
SPT1RS-210	efficient energy use(2)
SPT1RS-197	overall CAPEX/OPEX optimisation(1)
SPT1RS-220	overall CAPEX/OPEX optimisation(2)
SPT1RS-226	systems: extensible capacity, scalability(2)

7 items found

SPT2CE-20 - Remotely add, modify, delete or configure functions

As a supplier, rail infrastructure manager or railway undertaking, I would like to be able to remotely add, modify, delete or configure functions (related to Functional Application or Computing Platform) via a harmonized approach (without or with minimal effort and lean process for new authorization), in order to reduce operational expenses, time to deployment, Functional Application Downtime and Service Unavailability .

SPT2CE-20 has been derived from the following Common Business Objectives:

ID	Title
SPT1RS-193	automate lifecycle processes
SPT1RS-190	Changeability and upgradeability(1)
SPT1RS-212	non invasive/noticeable cyber security
SPT1RS-197	overall CAPEX/OPEX optimisation(1)
SPT1RS-220	overall CAPEX/OPEX optimisation(2)

5 items found

SPT2CE-21 - Replace the Computing Platform without vendor lock-in

As a rail infrastructure manager or railway undertaking, I would like to be able to replace the Computing Platforms (Runtime Environment and Hardware, to the extent that this is possible) used for Functional Applications up to SIL4 without vendor lock-in toward the Functional Application supplier(s) and with minimal or no re-authorization effort, in order to minimize obsolescence related risks, leverage latest progress in the evolution of Computing Platforms and/or reduce cost of ownership / cost of operation.

SPT2CE-21 has been derived from the following Common Business Objectives:

ID	Title
SPT1RS-198	automate lifecycle processes, independent lifecycle
SPT1RS-190	Changeability and upgradeability(1)
SPT1RS-196	increase market size by standardisation

ID	Title
SPT1RS-194	independent lifecycle, simple exchange
SPT1RS-218	modularity
SPT1RS-197	overall CAPEX/OPEX optimisation(1)
SPT1RS-220	overall CAPEX/OPEX optimisation(2)
SPT1RS-247	standardized architecture(2)

8 items found

SPT2CE-22 - Deploy Functional Applications through a harmonized approach

As a rail infrastructure manager or railway undertaking, I would like to be able to deploy Functional Applications on Computing Platforms through a harmonized approach across parts of or my entire infrastructure or fleet, in order to minimize operational costs and risks related to handling multiple different solutions.

SPT2CE-22 has been derived from the following Common Business Objectives:

ID	Title
SPT1RS-193	automate lifecycle processes
SPT1RS-198	automate lifecycle processes, independent lifecycle
SPT1RS-190	Changeability and upgradeability(1)
SPT1RS-224	operational harmonisation, unique requirements
SPT1RS-197	overall CAPEX/OPEX optimisation(1)
SPT1RS-220	overall CAPEX/OPEX optimisation(2)
SPT1RS-221	standardized architecture(1)
SPT1RS-247	standardized architecture(2)

8 items found

SPT2CE-23 - Computing Platform suitable for Functional Applications up to SIL4

As a rail infrastructure manager or railway undertaking, I need a Computing Platform (Runtime Environment and Hardware) suitable for Functional Applications up to SIL4, in order to be able to integrate railway-in-house or third-party developed Functional Applications (e.g., Digital Register, CTMS).

SPT2CE-23 has been derived from the following Common Business Objectives:

ID	Title
SPT1RS-218	modularity
SPT1RS-197	overall CAPEX/OPEX optimisation(1)
SPT1RS-220	overall CAPEX/OPEX optimisation(2)

ID	Title
SPT1RS-221	standardized architecture(1)
SPT1RS-247	standardized architecture(2)

5 items found

SPT2CE-24 - Planned movement/relocation of Functional Applications from one instance of a Computing Platform to another

As a rail infrastructure manager, I would like to be able to move/relocate Functional Applications from one instance of a Computing Platform (Runtime Environment and Hardware) to another, based on lean pre-authorization, with a Functional Application Downtime maximally on the order of minutes, for instance in the context of scaling / consolidating Data Centers.

SPT2CE-24 has been derived from the following Common Business Objectives:

ID	Title
SPT1RS-198	automate lifecycle processes, independent lifecycle
SPT1RS-190	Changeability and upgradeability(1)
SPT1RS-211	efficient energy use(1)
SPT1RS-210	efficient energy use(2)
SPT1RS-170	increase capacity
SPT1RS-197	overall CAPEX/OPEX optimisation(1)
SPT1RS-220	overall CAPEX/OPEX optimisation(2)
SPT1RS-229	seamless and selective exchange of components under production
SPT1RS-221	standardized architecture(1)
SPT1RS-247	standardized architecture(2)
SPT1RS-226	systems: extensible capacity, scalability(2)

11 items found

SPT2CE-25 - Replace one Hardware by another with minimal or no re-authorisation effort

As a rail infrastructure manager or railway undertaking I would like to be able to replace one Hardware by another with minimal or no re-authorization effort and with a Functional Application Downtime maximally on the order of the minutes, for instance in response to hardware failures, or for upgrade/scaling/obsolescence reasons.

SPT2CE-25 has been derived from the following Common Business Objectives:

ID	Title
SPT1RS-193	automate lifecycle processes
SPT1RS-198	automate lifecycle processes, independent lifecycle

ID	Title
SPT1RS-154	availability, robustness, reliability
SPT1RS-190	Changeability and upgradeability(1)
SPT1RS-188	Changeability and upgradeability(3), simpilified integration
SPT1RS-211	efficient energy use(1)
SPT1RS-210	efficient energy use(2)
SPT1RS-170	increase capacity
SPT1RS-194	independent lifecycle, simple exchange
SPT1RS-218	modularity
SPT1RS-197	overall CAPEX/OPEX optimisation(1)
SPT1RS-220	overall CAPEX/OPEX optimisation(2)
SPT1RS-229	seamless and selective exchange of components under production
SPT1RS-221	standardized architecture(1)
SPT1RS-247	standardized architecture(2)
SPT1RS-226	systems: extensible capacity, scalability(2)

16 items found

SPT2CE-26 - Automatically instantiate and/or activate a Functional Application on new hardware in case all hardware on which the Functional Application runs fails (e.g., in case of a disaster affecting a whole data center)

As a rail infrastructure manager and railway undertaking, I would like that a Functional Application for which all hardware it is running on fails (e.g., in the event of a disaster) is automatically instantiated and/or activated on new hardware within seconds and without loss of defined data in order to avoid/minimize Functional Application Downtime and Service Unavailability.

Note: For cost and space efficiency, a "hot standby" is not considered a solution to address this user story

SPT2CE-26 has been derived from the following Common Business Objectives:

ID	Title
SPT1RS-193	automate lifecycle processes
SPT1RS-154	availability, robustness, reliability
SPT1RS-197	overall CAPEX/OPEX optimisation(1)
SPT1RS-220	overall CAPEX/OPEX optimisation(2)

ID	Title
SPT1RS-229	seamless and selective exchange of components under production
SPT1RS-247	standardized architecture(2)

6 items found

SPT2CE-27 - Add computing elements (e.g. CPUs, memory or storage) from another supplier

As a rail infrastructure manager or railway undertaking, I would like to add computing elements (e.g., CPUs, memory or storage) from another supplier with minimal or no re-authorization effort to an existing Computing Platform in order to increase its capacity, minimize obsolescence risks, avoid unnecessary replacements of larger sets of hardware, minimize the acquisition cost or increase the overall availability of the Computing Platform.

SPT2CE-27 has been derived from the following Common Business Objectives:

ID	Title
SPT1RS-198	automate lifecycle processes, independent lifecycle
SPT1RS-154	availability, robustness, reliability
SPT1RS-190	Changeability and upgradeability(1)
SPT1RS-188	Changeability and upgradeability(3), simpilified integration
SPT1RS-211	efficient energy use(1)
SPT1RS-210	efficient energy use(2)
SPT1RS-194	independent lifecycle, simple exchange
SPT1RS-218	modularity
SPT1RS-197	overall CAPEX/OPEX optimisation(1)
SPT1RS-220	overall CAPEX/OPEX optimisation(2)
SPT1RS-229	seamless and selective exchange of components under production
SPT1RS-247	standardized architecture(2)
SPT1RS-226	systems: extensible capacity, scalability(2)

13 items found

SPT2CE-28 - Interface a Computing Platform with existing systems

As a rail infrastructure manager or railway undertaking, I would like to interface a Computing Platform with existing systems through a communication network and/or discrete hardwired connections in order to minimize the acquisition and integration cost.

SPT2CE-28 has been derived from the following Common Business Objectives:

ID	Title
SPT1RS-190	Changeability and upgradeability(1)

ID	Title
SPT1RS-194	independent lifecycle, simple exchange
SPT1RS-197	overall CAPEX/OPEX optimisation(1)

3 items found

SPT2CE-29 - Add, modify, delete or configure functions related to IT/OT security and/or communication protocols

As a supplier, rail infrastructure manager or railway undertaking, I would like to be able to add, modify, delete or configure functions related to IT/OT security and/or communication protocols without or with minimal effort and lean process for new authorization, in order to reduce operational expenses, time to deployment, Functional Application Downtime and Service Unavailability, and respond quickly to security risks.

SPT2CE-29 has been derived from the following Common Business Objectives:

ID	Title
SPT1RS-193	automate lifecycle processes
SPT1RS-198	automate lifecycle processes, independent lifecycle
SPT1RS-190	Changeability and upgradeability(1)
SPT1RS-188	Changeability and upgradeability(3), simpilified integration
SPT1RS-196	increase market size by standardisation
SPT1RS-194	independent lifecycle, simple exchange
SPT1RS-218	modularity
SPT1RS-212	non invasive/noticeable cyber security
SPT1RS-197	overall CAPEX/OPEX optimisation(1)
SPT1RS-220	overall CAPEX/OPEX optimisation(2)
SPT1RS-229	seamless and selective exchange of components under production
SPT1RS-221	standardized architecture(1)
SPT1RS-247	standardized architecture(2)

13 items found

SPT2CE-30 - System operation and update deployment without or with minimal on-site presence

As a supplier, rail infrastructure manager or railway undertaking, I would like to be able to operate my entire infrastructure/fleet or perform any deployments of or updates on Functional Applications or Computing Platforms (including related parameters updates) without or with minimal on-site presence, in order to minimize operational costs, reduce the time for updates.

SPT2CE-30 has been derived from the following Common Business Objectives:

ID	Title

ID	Title
SPT1RS-193	automate lifecycle processes
SPT1RS-198	automate lifecycle processes, independent lifecycle
SPT1RS-190	Changeability and upgradeability(1)
SPT1RS-211	efficient energy use(1)
SPT1RS-210	efficient energy use(2)
SPT1RS-212	non invasive/noticeable cyber security
SPT1RS-197	overall CAPEX/OPEX optimisation(1)
SPT1RS-220	overall CAPEX/OPEX optimisation(2)
SPT1RS-229	seamless and selective exchange of components under production

9 items found

4 Benefit assessment of the user stories

In this section, the previously defined user stories are further elaborated on w.r.t. details on the expected benefits, and an overall assessment on the expected comparative benefit level is provided.

Please note that in this section the focus is only on expected general high-level benefits of the user stories (mainly from the perspective of the listed beneficiaries), while the cost for the introduction of standardized APIs needed to support the user stories is purposely excluded here. The latter is explicitly investigated in Section 7, which then allows to provide an overall conclusion in Section 8.

Note that a quantitative evaluation of the benefits of the user stories is not performed here, as this would obviously depend strongly on the legacy environment of the beneficiary (i.e., the railway infrastructure manager, railway undertaking and/or supplier), the scale of a particular deployment (e.g., number of interlockings, fleet size), etc. Instead, only a qualitative and comparative assessment of the expected benefit level is provided with the distinction between:

- High: Substantial CAPEX and OPEX benefits
- Modest: Substantial CAPEX **or** OPEX benefits, or modest benefits for both CAPEX and OPEX

Note that the stated expected benefit levels apply mainly to new application functions; benefits for legacy applications may be reduced.

User Story: ***SPT2CE-18 - Minimize overall dependencies***

Details on expected major benefits	Expected comparative benefit level

Details on expected major benefits	Expected comparative benefit level
• Benefits through being able to plan and conduct migration of application, RTE and HW more independently (as independent as possible by a pre-defined integrator) than today (CAPEX and OPEX benefits) • Better leverage of the latest state of the art of RTE and HW (CAPEX and OPEX benefits) • Reduction of obsolescence risks (reduction of OPEX and increased quality of rail operation)	High

User Story: SPT2CE-19 - Aggregate multiple Functional Applications on the same Instance of a Computing Platform

Details on expected major benefits	Expected comparative benefit level
• Trackside: reduction of data center space requirements, cooling, etc. (CAPEX savings in the order of more than 1 BEUR in Germany, also expected OPEX savings) • Onboard: space, energy dissipation and weight reduction yields increased space for passengers • Allows for simpler system extension, in particular Onboard, with new functionality, i.e. no hardware deployment leads to reduced CAPEX and OPEX cost. • Minimize the risk of failure, obsolescence and ease of maintenance leads to reduced OPEX.	High

User Story: SPT2CE-20 - Remotely add, modify, delete or configure functions

Details on expected major benefits	Expected comparative benefit level
• Substantial OPEX savings both trackside and onboard (at least double digit MEUR per year just in Germany) due to the fact that neither personnel needs to move to the object (loco or infrastructure) nor objects need to be moved to special locations to get an update. • Increased quality of rail operation through reduced application downtime • Increased security and safety by reducing possible human errors during a manual deployment/update process. • On-board : avoid the immobilization of trains in maintenance center. Therefore improves the overall availability of the fleet	High

User Story: SPT2CE-21 - Replace the Computing Platform without vendor lock-in

Details on expected major benefits	Expected comparative benefit level

Details on expected major benefits	Expected comparative benefit level
• In particular OPEX benefit through being able to replace RTE and HW to most advanced solutions (e.g., regarding georedundancy, leverage of latest developments in IT sector, etc.) • Possible entrance of new vendors for RTE and hardware, providing CAPEX and OPEX benefit to whole sector	High

User Story: SPT2CE-22 - Deploy Functional Applications through a harmonized approach

Details on expected major benefits	Expected comparative benefit level
• Substantial reduction of OPEX on both trackside and onboard • Substantial acceleration of technology rollout, allowing to leverage benefits of certain ETCS levels, etc., earlier	High

User Story: SPT2CE-23 - Computing Platform suitable for Functional Applications up to SIL4

Details on expected major benefits	Expected comparative benefit level
• Major economic benefit (at least double digit MEur in Germany) if a railway undertaking and vendor can buy/license RTE and hardware incl. safety layer for running in-house developed functions, instead of developing parts of the RTE itself	High

User Story: SPT2CE-24 - Planned movement/relocation of Functional Applications from one instance of a Computing Platform to another

Details on expected major benefits	Expected comparative benefit level
• Large CAPEX benefit (>1 bln EUR in Germany) if functions can be relocated among data centers and data centers constantly scaled up, instead of pre-dimensioning decades in advance • Additional CAPEX savings through avoiding local over-dimensioning of HW	High

User Story: SPT2CE-25 - Replace one Hardware by another with minimal or no re-authorisation effort

Details on expected major benefits	Expected comparative benefit level
• CAPEX savings if COTS HW with relaxed requirements can be used, as this can be replaced when required • CAPEX savings as HW can be replaced through more performant HW in the future (e.g., with savings in space, power dissipation, etc.) • CAPEX savings due to reduced time-to-market and no re-assessment needed when HW gets obsolete. • OPEX savings if HW failures can be addressed with any available HW Computing Element in stock	High

User Story: SPT2CE-26 - Automatically instantiate and/or activate a Functional Application on new hardware in case all hardware on which the Functional Application runs fails (e.g., in case of a disaster affecting a whole data center)

Details on expected major benefits	Expected comparative benefit level
• Substantial improvements in the quality of rail operation (e.g., less downtime) • CAPEX savings due to needing less backup HW in standby	Medium

User Story: SPT2CE-27 - Add computing elements (e.g. CPUs, memory or storage) from another supplier

Details on expected major benefits	Expected comparative benefit level
• CAPEX and OPEX benefits through being able to leverage latest advances in HW independent of the application or RTE vendor • Increased quality of rail operations through being able to replace HW faster due to vendor independence • CAPEX benefit through a "working" HW market - i.e. no HW vendor lock-in	High

User Story: SPT2CE-28 - Interface a Computing Platform with existing systems

Details on expected major benefits	Expected comparative benefit level

Details on expected major benefits	Expected comparative benefit level
• CAPEX benefit: integration with existing legacy systems i.e., no need to replace existing systems when introducing new systems based on the Computing Platform	Medium

User Story: SPT2CE-29 - Add, modify, delete or configure functions related to IT/OT security and/or communication protocols

Details on expected major benefits	Expected comparative benefit level
• CAPEX and OPEX benefits through being able to update parts of the system, possibly provided by 3rd party vendors, without being dependent on the application or RTE vendor • Improved level of cybersecurity etc. through being able to upgrade related functions more frequently without affecting safety. • Improved connectivity performance (and related OPEX benefits) through being able to update connectivity protocols to newest standards • Improved quality of rail operation through reduced downtime due to faster function update	High

User Story: SPT2CE-30 - System operation and update deployment without or with minimal on-site presence

Details on expected major benefits	Expected comparative benefit level
• Strong OPEX savings (at least double-digit MEur per year in Germany) through avoiding on-site visits • Increased security (safety?) by reducing possible human errors during an manual on-site deployment/update	High

5 Computing Environment architecture and possibly standardized interfaces

5.1 Architecture

In the course of the discussion on the user stories introduced in the past sections, a possible layered computing environment architecture with possibly standardized interfaces between the layers has been developed and is shown in Figure 5.1-4.

Note: Potential interfaces subsequently shown are to be understood in the way that it could be good to have a harmonized interface in the depicted position for the rail sector (subject to the investigations in the subsequent chapters). To which extent these interfaces would be based on existing, standardized interfaces (e.g., from other sectors), or dedicated to the rail sector, is for further investigation (of course, one should always strive to reuse existing, already standardized interfaces wherever possible).

Further note that the term "interface" is here used both w.r.t. interoperablity (i.e., usage of one layer from one supplier on another layer provided by a different supplier) and portability (i.e., portability of one layer from one underlying layer to a similar one from another supplier)

Layered Architecture and introduction of interfaces

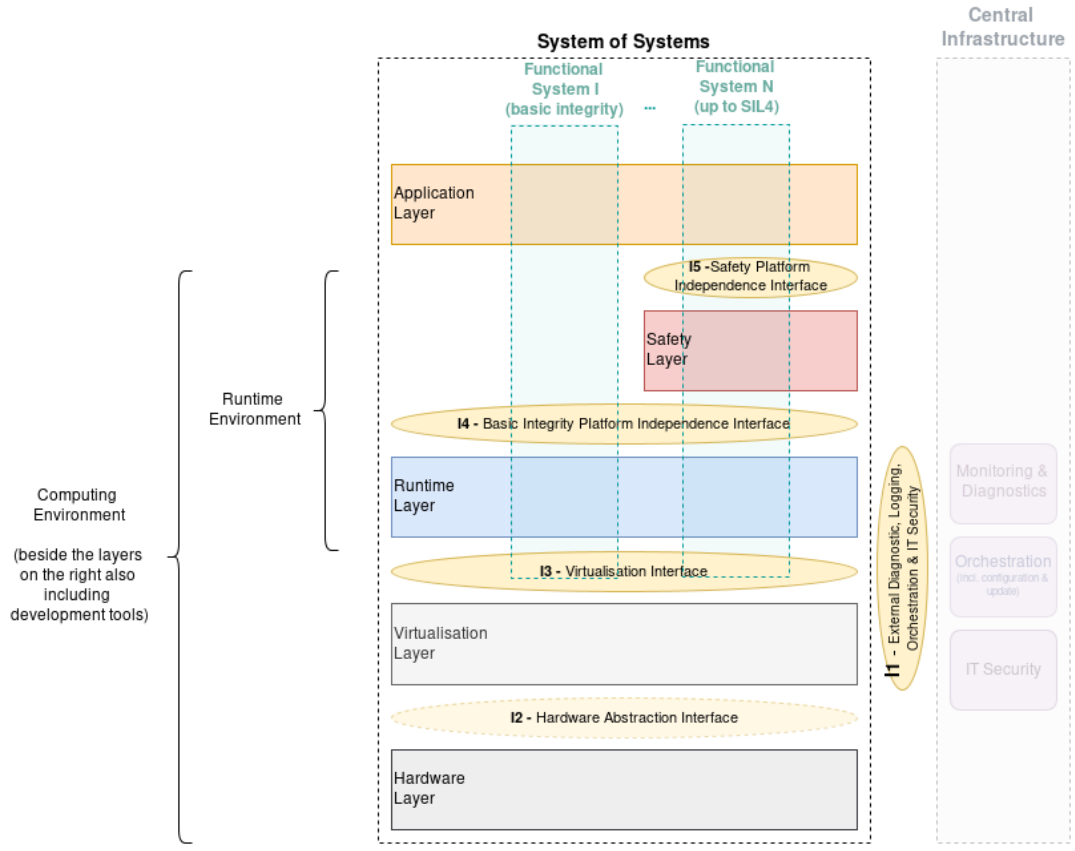


Figure 1 Layered architecture and introduction of interfaces

Note: Figure 1 obviously provides a highly abstracted view of a possible application and platform architecture and purposely refrains from any details that would already narrow down the realization of a specific interface. The main purpose is to identify on which level in the stack standardized interfaces could principally be relevant. It is clear that some aspects of the figure would change depending on specific interface realizations. For instance, if interface I3 is based on the notion of virtualization, operating systems resides above the interface. If, however, I3 involves the

notion of containerization, there would also be an operating system below the interface. Similarly, whether parts of interface I1 terminate in the runtime layer or in the virtualization layer, depends on the detailed interface realizations.

The identified layers are expected to comprise:

- **Application Layer**: Contains Functional Applications;
- **Safety Layer**: Contains all functions related to fault tolerance (e.g., composite fail safety, voting mechanisms, redundancy mechanisms for availability) needed to enable that Functional Applications up to SIL4 can run on the computing environment;

  Note: whether the stated functions can indeed be put into a separate layer by itself requires further investigation, for instance in the context of the further work on interfaces I4 and I5

- **Runtime Layer**: Contains the Runtime Environment, assumed to comprise system services (e.g., application and computing resource orchestration, monitoring of the Functional Applications and the Platform itself, tracing and logging, communication services that are not related to safety, security means incl. authentication, encryption, key storage, etc.) and the communication stack for information exchange between Functional Applications running on the same Platform and with external entities
- **Virtualisation Layer**: Contains mechanisms like a hypervisor to allow different systems with a safety and runtime layer to run concurrently on the same pool of Hardware underneath.
- **Hardware Layer**: Contains the actual Hardware, i.e. the physical and electronic parts of a computer or other piece of equipment.

  Note: It is generally assumed that the hardware layer should ideally consist of commercial-of-the-shelf hardware which is NOT safety-certified

The interfaces between Functional Systems must be defined as standardized communication based interfaces. For this it does not make a difference if the connected Functional System is running aggregated on the same computing elements (in own containers), running on separated computing elements or even running on specific hardware.

The possibly standardized interfaces related to the layered computing environment are explained in the following sections.

5.2 External Diagnostic, Logging, Orchestration and IT Security Interface(s) (I1)

The External Diagnostic, Configuration & orchestration Interface, hereafter referred to as **I1** (Interface 1), comprises communication-based interfaces between rail systems and central infrastructure components as diagnostics, IT-security services and remote update.
These interfaces and belonging processes shall be defined as standard for all rail systems: centralized in a data center as e.g. IXL-logic, decentralized in the field as e.g. object controllers and onboard systems.

I1 consists of the following (sub-)interfaces:

- IF-DIAGNOSTICS
- IF-LOGGING
- IF-ORCHESTRATION
- IF-IT-SEC

IF-DIAGNOSTICS is the interface to provide diagnosis data to the central diagnostics system.
In this interface the transport protocol (e.g. OPC-UA) shall be defined as standard, the diagnostics data itself cannot be standardized in detail (depends on concrete solution on side of the individual SW layers of the rail system).
The following diagnostics data is provided via this interface:

- Health status of the functional application, provided by the functional application
- Health status of the safety layer, provided by the safety layer
- Health status of the non-safe runtime layer, provided by the non-safe runtime layer
- Health status of the COTS hardware, provided by a dedicated diagnosis application
- Health status of the virtualization layer, provided by a dedicated diagnosis application

IF-LOGGING is the interface to provide logging data to the central logging service
In this interface the logging protocol (e.g. syslog over TLS) shall be defined as standard, the logging data itself cannot be standardized in detail (depends on concrete solution on side of the individual SW layers of the rail system).
The following logging data is provided via this interface:

- Security related logs (as required by IEC 62443)
- Continuous monitoring logs (as required by IEC 62443)
- Rail system specific logs – depending on the concrete solution of the rail system (e.g. need for juridical recording on SW-interfaces between SW-parts provided by different vendors)

IF-ORCHESTRATION is is the interface through which an external entity can update, start, stop or reconfigure functions (within application and platform), register and unregister hardware, update orchestration policies, etc.

IF-IT-SEC is the interfaces to central security services

- Interface to central time-service (system wide time synchronization)
- Interface to (local) registration authority LRA.
- Interface to the Identify and Access Management service IAM

5.3 Hardware Abstraction Interface (I2)

The Hardware Abstraction Interface (I2) provides an abstraction of all technology layers above from the specific hardware used below, enabling easy replace ability of commercial of-the-shelf hardware procurable from a well-sized market of hardware vendors.
Note: This is not really an interface, but rather a compatibility list of allowed hardware incl. CPU, memory, etc.

5.4 Virtualization Interface (I3)

The abstraction from a specific implementation of the Hardware is achieved using a virtualization layer. This layer includes important functions such as virtualization, separation and "containerization".
Virtualization Interface (Interface 3) is used to provide a standardized interface above this layer so that applications or higher platform layers are independent of a specific implementation of the computing hardware.

5.5 Basic Integrity Platform Independence Interface (I4)

The Basic Integrity Platform Independence Interface, hereafter mentioned as **I4** (Interface 4), is used to perform a basic integrity platform independence with the applications. In other words, this API is an interface limited to non-safety functionalities between runtime environment and applications. Despite the restricted scope of basic integrity services, on the one hand, **I4** provides applications portability among runtime environment from different suppliers or different versions of the same runtime environment (runtime implementation). On the other hand, this solution allows runtime environment to support application evolution.

As stated above, **I4** is a direct interface with the runtime environment and non-safety applications. However, this interface can also be an interface with runtime environment and a safety layer for safe applications up to SIL4. In this case, the figure 2 shows two different ways to obtain safety services.
The first option is the integration of three independent elements: a safety layer, Safety Platform Independence Interface (see definition below) and the application. This configuration is presented with functional systems 3 and 5. Here, **I4** is an interface with the runtime environment and a safety layer.
The second option is the integration of the previous elements (safety layer, Safety Platform Independence Interface and application) provided by a single supplier. This configuration is depicted with function systems 2.

The main characteristics of this **interface 4** are:
- provide an interface that facilitates the portability of applications to runtime environment as well as runtime implementations.
- provide messages services to enable communication among applications (flow, addressing, messages)
- provide timing functionalities (e.g. message timing, scheduling timing etc.)
- Provide process-control, memory management, performance, application life cycle etc.

Note: Interface I4 conceptually relates to the "extended PI API" introduced in Generic Safe Computing Platform - Specification of the PI API between Application and Platform

5.6 Safety Platform Independence Interface (I5)

The aim of introducing Safety Platform Independence Interface, hereafter mentioned as **I5** (Interface 5), is to be able to implement platform independent Safe Functional Applications (up to SIL4) i.e., applications, based on a generalized abstraction between the application logic and the system interfaces, will run unchanged on different platform implementations.
Applications programmed against **I5** are assumed to be at a minimum source code portable, or possibly even binary code portable, between different platform implementations. All safety-related functions not inherent in the application logic are implemented as part of the platform.
While this may not be the only approach to meet SIL requirements, it is assumed that the platform uses “composite fail-safety” according to EN 50129 for achieving fault tolerance. This means that a functional application is run multiple times (for M-out-of-N redundancy), and the results of the replicated instances are compared to provide one overall safe output.
Functional Applications consist of one or multiple Functional Actors exchanging messages. Furthermore, it is assumed that the functional actors consists of a processing logic and a state, and that some form of State Machine Replication (SMR) is used. The SMR requires that the functional actor has a defined initial state and is deterministic in the sense that with a given state and given message received, it always results in the same change of the state

and the same sequence of sent messages. It is not allowed that there is randomness in the execution or the sent messages, dependency on local or overall timing, or dependency on the local node state.
A message distribution logic within the platform ensures that an identical sequence of messages is delivered to the functional actor replicas. As the functional actor replicas all have the same initial state and are deterministic, they will produce an identical sequence of output messages and will go through the same sequence of states.
A voting logic within the platform compares the messages and the states of the different replicas and detects inconsistencies. In case of such, it provides an error indication which is processed further by additional monitoring mechanisms to ensure a safe reaction of the overall system.
It is expected that **I5** consists of at least the following:

- A minimal set of API functions for safety critical applications including:
  - Communication functions providing replication and location transparency i.e., a functional actor does neither know if it runs in multiple replicas or if the peer functional actors it communicates with, run in multiple replicas nor does it need to know if the peers it communicates with run locally (on the same platform instance) or remotely (on a different platform instance).
  - Functions via which a functional actor can determine or be informed about the status of other (platform-internal or external) functional actors.
  - Functions related to tracing and logging, security and access to persistent storage.
- Standardized exported safety related application conditions that a programmer of an application business logic must comply with in order to ensure that the full system is safety certifiable.

- A standardized configuration mechanism e.g., to define the MooN configuration of functional actors, to define computing resource requirements of functional actors, to define performance requirements of functional actors, etc.

Note: Interface I5 conceptually relates to the "PI API" introduced in Generic Safe Computing Platform - Specification of the PI API between Application and Platform

5.7 Possible deployment options

For illustration purposes, Figure 5.7-2 shows different deployment options utilizing the computing environment layers and interfaces introduced beforehand. Note that the figure neither implies that all depicted options would practically be rolled out, nor can the listed options be considered as exhaustive.
From left to right, the figure shows the following deployment options:

- **Legacy system or new system using proprietary RTE and Safety Layer**: In this option, a single vendor provides a specific Functional Application in conjunction with a proprietary Safety Layer (in case the application is safety relevant) and a proprietary Runtime Layer. This bundle provided by one supplier uses a standardized Virtualisation Interface and Hardware Abstraction Interface to allow the usage of a Virtualization Layer and Hardware Layer provided by different suppliers. Such deployment would for instance be relevant in the context of user stories:
  SPT2CE-19 - Aggregate multiple Functional Applications on the same Instance of a Computing Platform
  or
  SPT2CE-27 - Add computing elements (e.g. CPUs, memory or storage) from another supplier;

- **New System offering standardized Basic Integrity Platform Independence Interface (I4)**: In this option,

a single vendor provides a Functional Application in conjunction with a proprietary Safety Layer (if required by the application) and a proprietary Runtime Layer in such a way that the Basic Integrity Platform Independence Interface(I4) is implemented. This option would for instance support:

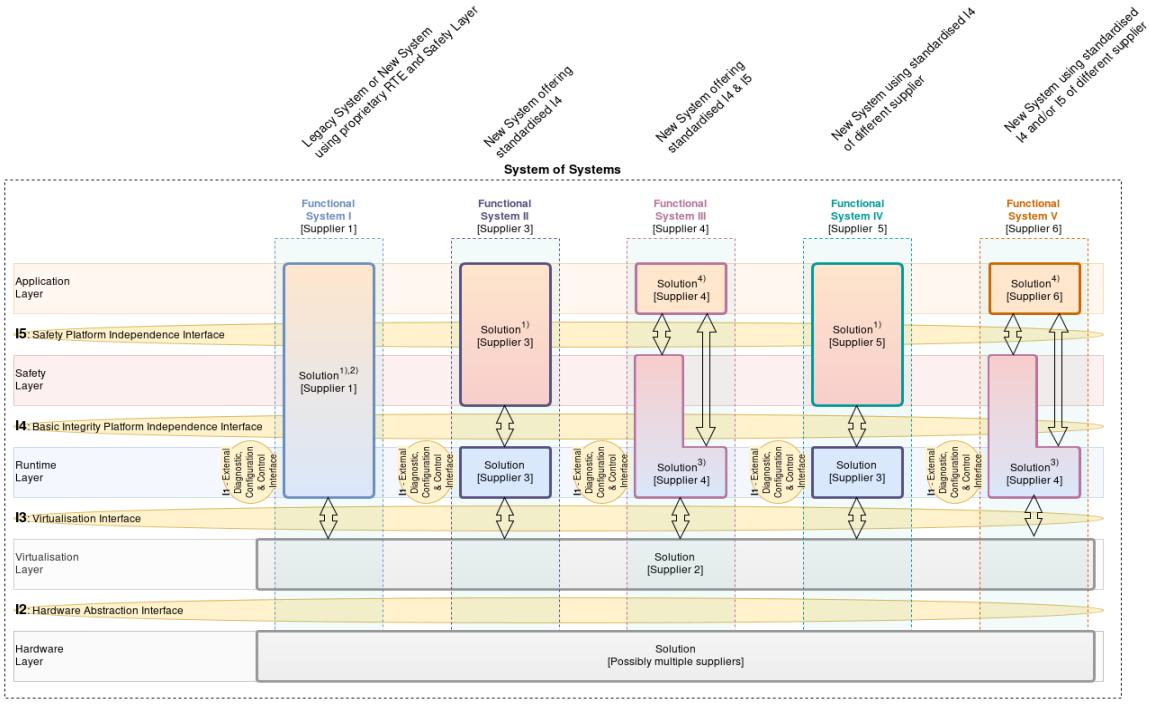
- SPT2CE-21 - Replace the Computing Platform without vendor lock-in,
- SPT2CE-24 - Planned movement/relocation of Functional Applications from one instance of a Computing Platform to another, or
- SPT2CE-26 - Automatically instantiate and/or activate a Functional Application on new hardware in case all hardware on which the Functional Application runs fails (e.g., in case of a disaster affecting a whole data center);

- **New System offering standardized Basic Integrity Platform Independence Interface (I4) and Safety Platform Independence Interface (I5)**: In this option, Functional Application, Safety Layer and Runtime Layer are still provided by the same vendor, but with the implementation of interfaces I4 and I5, in order to better facilitate the addition of further Functional Applications at a later point in time. This option would for instance support:
  - SPT2CE-19 - Aggregate multiple Functional Applications on the same Instance of a Computing Platform

- **New System using standardized Basic Integrity Platform Independence Interface (I4) of different supplier:** This option separates the application from the underlying runtime in a very common way, as used in many OS approaches. The application uses I4 to abstract all HW/OS tasks from the application. Safety aspects, as needed by the application, need to be dealt with directly in the application. This, of course, includes assessing suitability of underlying runtime for usage in safe context, as this is naturally not part of the runtime layer. The abstraction and API of I4 can be specified in a clear way so that it is technically possible to use solutions from different suppliers for the same, unchanged (source and binary) application. Of course, a safety assessment needs to be partly touched, depending on differences/commonalities of safety relevant architectural capabilities used in the layers below I4.
  - SPT2CE-19 - Aggregate multiple Functional Applications on the same Instance of a Computing Platform

- **New System using standardize Basic Integrity Platform Independence Interface (I4) and/or Safety Platform Independence Interface (I5) of different supplier:** This option is basically similar to the option above with the essential difference that the platform-related safety aspects are separated from the functional application through a standardized interface I5. This means, that potentially all aspects of functional and safety necessities can be fulfilled unchanged, even if I4+I5 (and the respective layers) are delivered from different vendors. Note that achieving this goal is challenging, but the benefits for application vendors (and IMs/RUs) would be huge. Achieving a safety assessment then means to prove fulfillment of a well-defined set of rules & SACs, no specific extra-assessment targeting layers beyond I5 is to be created from an application vendor. The independent suppliers, of course, need to deliver those assessments in an appropriate form and quality.
  - SPT2CE-23 - Computing Platform suitable for Functional Applications up to SIL4

Figure # Possible deployment options (for illustration purposes only)



1) In case the box implements a safe function, it includes a safety layer that may be proprietary.
2) This box could implement an existing legacy function or a new function.
3) This box implements a RTE and safety layer and exposes I4 and I5.
4) In case the box implements a safe function, it would use I5, otherwise it would use I4

Figure 2 Possible deployment options (for illustration purposes only)

6 Feasibility assessment and mapping of user stories to interfaces

In this section, the identified user stories related to computing environments are assessed w.r.t. their feasibility and mapped to the possibly standardized interfaces defined in the previous section.
Note that for each user story, a separate feasibility assessment is provided for the case that

1. All Functional Applications sharing the same hardware or other layers require only basic integrity according to CENELEC EN 5012x (column 1 in the table below);
2. At least some involved Functional Applications required SIL (column 2 in the table below);

On the right side of Table X the possibly standardized interface I1- I5 are listed, where

- **M** denotes that it is **mandatory** that an interface is standardized to render a user story feasible;
- **R** denotes that it is not strictly required that an interface is standardized for a user story, but standardization of the interface would be **recommended** to support the user story;
- **S** denotes that a standardized interface would be **supportive** of the user story (but it is not seen as mandatory or recommended for the user story, and could be solved with some work around)
- **A** denotes that an interface is neither recommended nor supportive to be standardized for a certain user story, but that the interface would be **affected** by the user story;

User Story: SPT2CE-18 - Minimize overall dependencies

Feasibility for Basic Integrity Applications	Feasibility for SIL Applications	Overall Feasibility	Interfaces				
			I1	I2	I3	I4	I5
Minimize dependencies among Application and RTE: Considered **feasible**, **requires I4**. **Minimize dependencies among RTE and HW**: Considered **feasible**, may work with off-the-shelf virtualization solutions. Therefore, it **requires I3**.	**Minimize dependencies among Application and RTE**: The suppliers in the domain jointly rate this as **difficult to achieve**. Any activities in the direction of related I5 should well consider/analyze how a multi-vendor constellation could be authorized, and how this would work from a legal perspective **Minimize dependencies among RTE and HW**: This would be **feasible** with **I3**.	**Largely feasible** (with exceptions on left)	R	M	M	M	S*

* if I3 is available, I5 provides additional benefits in terms of the minimization of dependencies. If I3 is not available, I5 would be mandatory to render the user story feasible (for SIL applications). Note that there is a large controversy on the feasibility and business case of I5, see Chapter 8 .

User Story: SPT2CE-19 - Aggregate multiple Functional Applications on the same Instance of a Computing Platform

Feasibility for Basic Integrity Applications	Feasibility for SIL Applications	Overall Feasibility	Interfaces				
			I1	I2	I3	I4	I5
Is considered **feasible** with commercial off-the-shelf virtualization solutions. It is **recommended** that **I1**, **I3** and **I4** are standardized.	Aggregating mixed critical applications on a same RTE is considered **complicated especially for SIL4** by the suppliers in the domain. Any activities in the direction of related **I5** should well consider/analyze how a multi-vendor constellation could be authorized, and how this would work from a legal perspective. However, it is considered **feasible** to aggregate multiple safe Applications (incl. RTE) on a common HW if for SIL applications, application and RTE are in one integration responsibility, and **I3** is **standardized**.	**Feasible** (with limitations on left)	R	M	M	R*	S*

* If I3 is not available, I4 and I5 would be mandatory to render the user story feasible (for basic integrity and SIL

applications, respectively). Note that there is a large controversy on the feasibility and business case of I5, see Chapter 8

User Story: SPT2CE-20 - Remotely add, modify, delete or configure functions

Feasibility for Basic Integrity Applications	Feasibility for SIL Applications	Overall Feasibility	Interfaces				
			I1	I2	I3	I4	I5
Considered **feasible**, in the understanding that • any update is done in conjunction with the system integration. • the granularity at which updates can be done are supplier specific (though the Interface through which updates are done is standardized) This requires specification of **I1.**	Considered **feasible**, in the understanding that • any update is done in conjunction with the system integrator • the granularity at which updates can be done are supplier specific (though the Interface through which updates are done is standardized) This requires specification of **I1.**	**Feasible**	M	-	-	-	-

User Story: SPT2CE-21 - Replace the Computing Platform without vendor lock-in

Feasibility for Basic Integrity Applications	Feasibility for SIL Applications	Overall Feasibility	Interfaces				
			I1	I2	I3	I4	I5
For basic integrity applications, this appears **feasible**. It is recommended that **I1**, **I3**, **I4** are specified.	From supplier perspective in the domain, it appears **difficult** to replace the RTE under an existing application. It would rather be recommended to port the Functional System (consisting of Functional Application and RTE) onto a new HW, based on a specified **I3**. Any activities in direction of an **I5** should well consider/analyze how a multi-vendor constellation could be authorized, and how this would work from a legal perspective.	**Feasible** (with limitations stated in column SIL Applications)	R	M	M	M	S*

* If I3 is not available, I5 would be mandatory to render the user story feasible (for SIL applications). Note that there is a large controversy on the feasibility and business case of I5, see Chapter 8

User Story: SPT2CE-22 - Deploy Functional Applications through a harmonized approach

Feasibility for Basic Integrity Applications	Feasibility for SIL Applications	Overall Feasibility	Interfaces				
			I1	I2	I3	I4	I5
Considered **feasible** if **I1 and I3** is specified, **I2** is recommended.	Considered **feasible** if **I1 and I3** is specified, **I2** is recommended.	**Feasible**	M	R	M	-	-

User Story: SPT2CE-23 - Computing Platform suitable for Functional Applications up to SIL4

Feasibility for Basic Integrity Applications	Feasibility for SIL Applications	Overall Feasibility	Interfaces				
			I1	I2	I3	I4	I5
Is considered **feasible** if **I4** is standardized.	Would require **I5** (if Application vendor wants to avoid using a proprietary Interface), though being questioned by suppliers in domain. An alternative solution could be to specify **I4** a nd design the overall platform architecture such that rail infrastructure managers could buy the "safety layer" as a separate product.	There is large controversy on the feasibility and business case related to **I5**, and consequently also on the **feasibility** of this user story. This is to be **furth er investigated**, a first step being the ongoing prototyping and planned investigations on authorization in the Innovation Pillar.	-	-	-	M	M

User Story: SPT2CE-24 - Planned movement/relocation of Functional Applications from one instance of a Computing Platform to another

Feasibility for Basic Integrity Applications	Feasibility for SIL Applications	Overall Feasibility	Interfaces				
			I1	I2	I3	I4	I5

Feasibility for Basic Integrity Applications	Feasibility for SIL Applications	Overall Feasibility	Interfaces				
			I1	I2	I3	I4	I5
Is considered **feasible**, possibly with Virtualization Layer solutions. It is recommended that **I1**, **I3** and **I4** are specified.	It is considered **difficult** by the suppliers in the domain to move/relocate Functional Applications from one supplier onto an RTE of another. Any activities in the direction of related **I5** should strongly consider authorization and legal aspects. It is considered **feasible** that a bundle of safe application and RTE from vendor are moved/relocated to the HW of a different vendor, if **I3** is specified.	**Feasible** (with limitations stated in column SIL Applications)	R	-	M	S*	S*

* If I3 is not available, I4 and I5 would be mandatory to render the user story feasible (for basic integrity and SIL applications, respectively). Note that there is a large controversy on the feasibility and business case of I5, see Chapter 8

User Story: SPT2CE-25 - Replace one Hardware by another with minimal or no re-authorisation effort

Feasibility for Basic Integrity Applications	Feasibility for SIL Applications	Overall Feasibility	Interfaces				
			I1	I2	I3	I4	I5
Considered **feasible** if **I2** and **I3** are specified	Considered **feasible** if **I2** and **I3** are specified	**Feasible**	-	M	M	-	-

User Story: SPT2CE-26 - Automatically instantiate and/or activate a Functional Application on new hardware in case all hardware on which the Functional Application runs fails (e.g., in case of a disaster affecting a whole data center)

Feasibility for Basic Integrity Applications	Feasibility for SIL Applications	Overall Feasibility	Interfaces				
			I1	I2	I3	I4	I5

Feasibility for Basic Integrity Applications	Feasibility for SIL Applications	Overall Feasibility	Interfaces				
			I1	I2	I3	I4	I5
For Basic Integrity applications, it is assumed **feasible** (subject to networking aspects) that a platform-external entity monitors the health of a Computing Platform, and triggers the instantiation of a Functional Application instance on a new Computing Platform instance in the event of a failing Computing Platform instance. This requires specification of **I1** and **I3**.	For SIL Applications, it is assumed **feasible** (subject to authorization and networking aspects) that an RTE could detect a Computing Platform failure affecting a Functional Application and trigger the activation of a HW instance (e.g., container) through a specified **I3**. It is further assumed that authorization could be expected if new HW instances are always pre-configured and pre-qualified.	**Likely feasible** (subject to networking aspects, and for SIL applications additionally subject to authorization aspects, see columns Basic Integrity Applications and SIL Applications)	M	-	M	S*	S*

* If I3 is not available, I4 and I5 would be mandatory to render the user story feasible (for basic integrity and SIL applications, respectively). Note that there is a large controversy on the feasibility and business case of I5, see Chapter 8

User Story: SPT2CE-27 - Add computing elements (e.g. CPUs, memory or storage) from another supplier

Feasibility for Basic Integrity Applications	Feasibility for SIL Applications	Overall Feasibility	Interfaces				
			I1	I2	I3	I4	I5
Is considered **feasible** if **I2** is specified. **I1** and **I3** are recommended.	Is considered **feasible** if **I2** is specified. **I1** and **I3** are recommended.	**Feasible**	R	M	R	-	-

User Story: SPT2CE-28 - Interface a Computing Platform with existing systems

Feasibility for Basic Integrity Applications	Feasibility for SIL Applications	Overall Feasibility	Interfaces				
			I1	I2	I3	I4	I5
Generally feasible but needs to be dealt case-by-case with the respective existing system.	Generally feasible but needs to be dealt case-by-case with the respective existing system.	**Feasible** (from the technical point of view)	-	-	-	-	-

User Story: SPT2CE-29 - Add, modify, delete or configure functions related to IT/OT security and/or communication protocols

Feasibility for Basic Integrity Applications	Feasibility for SIL Applications	Overall Feasibility	Interfaces				
			I1	I2	I3	I4	I5
Feasible for BI applications	Current assumption is: **feasible**, but it needs to be validated by ARCH and PRAMSS	**Feasible** (from the technical point of view)	M	-	R	R	-

User Story: SPT2CE-30 - System operation and update deployment without or with minimal on-site presence

Feasibility for Basic Integrity Applications	Feasibility for SIL Applications	Overall Feasibility	Interfaces				
			I1	I2	I3	I4	I5
Considered **feasible**, requires that **I1** is specified.	Considered **feasible**, requires that **I1** is specified.	**Feasible**	M	-	-	-	-

7 Cost assessment of the interfaces

In this section, the previously identified interfaces are assessed w.r.t. the cost implications (on Functional Applications and Computing Platforms) if these would be standardized. Similar to the assessment of the benefit of the user stories, it is difficult to perform a detailed quantitative assessment of the costs incurred through the standardization of a particular interface, as this depends on whether or not a supplier already has existing proprietary interfaces that need to be further supported and evolved despite the introduction of standardized interfaces, or on how similar proprietary interfaces are to eventually standardized interfaces. Hence, only a relative and qualitative comparison of costs is performed, as detailed in the following sub-sections.

7.1 Methodology applied

The cost assessment is performed along a set of cost categories covering the whole life-cycle from the actual specification to operation and maintenance, as detailed in the following sections. In general, the following assumptions apply:

- The considered baseline against which the cost comparison is performed is a setup where vendors have their own platforms and proprietary APIs;
- Concerning technical as well as economical feasibility, there are obvious and large differences in designing new applications basing on the suggested standardized interfaces versus porting existing legacy applications to these interfaces. The analyses were performed for the case of newly implemented functional applications. It is expected that many benefits of the investigated interfaces may also apply to legacy applications, but various cost related to adaptation of applications, porting, etc. will be larger than for newly implemented functional applications.
    - All interface and Computing Platform related costs (excluding application costs) of having standardized interfaces are compared to the case that each vendor keeps on evolving its own proprietary interfaces;
    - Giving the consequence, costs include the costs for adapting proprietary platforms, but not the cost for adapting legacy applications;
- The costs always refer to the overall sector cost, no matter who has the major effort, or whether effort is shifted from one market player to another;
- The assumption is that in the compared cases with and without standardization the number of vendors in the

market is the same (i.e., a possible increase of the number of vendors through standardized interfaces is not considered).

Please note that the subsequent cost estimations are strongly sensitive to the aforementioned assumptions. If, for instance, the ecosystem would change towards more application or more platform vendors, the assessment of specification cost would strongly change.
Please note also that the applied viewpoint does not explicitly separate different roles like "vendor of the platform" or "application vendor using an existing platform". Cost assessments focusing on those viewpoints can and most likely will look completely different.

7.1.1 Specification cost

This cost category includes costs related to the identification of requirements, harmonization of the needs among the stakeholders, stakeholder management and coordination of the drafting or well-defined requirements incl. putting them into a tool and developing test strategies. Note that maintenance and compatibility management is to be considered as well.

7.1.2 Implementation cost

Implementation costs are the costs of upgrading especially existing platforms, applications are not focused on so far. Furthermore, quality management, configuration management and operational instructions of the platform and APIs, design specification and design implementation belong as well in this category. In addition, Safety Management such as maintenance instructions and safety case shall be also considered.

Reminder: based on current evaluation, upgrading the legacy application might arise additional costs

7.1.3 Testing Cost

Testing costs cover almost all the necessary testing environment to grant a solid functionality. It is also essential to have testing specifications so that e.g., tool chain tests, test stages within the life cycle and catalogue of the regression test are well defined. Further tests would be testing against the interface such as conformity tests or interoperability tests, backward compatibility tests.

7.1.4 Integration and authorization cost

Costs related to integration and authorization, considering, e.g.

- vendor multiplicity;
- the number of required authorization artifacts, certificates, etc.
- the fact that standardized interfaces may tend to be more expansive than proprietary interfaces

7.1.5 Legal related cost

These costs include the cost for juridical recording, started from specification to the actual conduction of recording and juridically hardened storage and further more.

7.1.6 Operation & maintenance cost

Operation and maintenance costs might derive from company to company in different forms and shapes, nevertheless considering following aspects might give direction of what expenses might need to be calculated. These are exemplary cost of implementing security patches, identification and fixing of bugs, functional upgrades and hardware replacement activities.

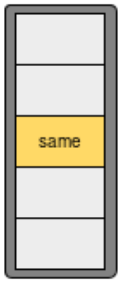
7.1.7 Release Life Cycle Management cost

Cost related to the life cycle management of applications and platforms, especially in the context of the coexistence of multiple (proprietary and standardized) interfaces, incl.

- Release management
- Compatibility management
- Migration planning

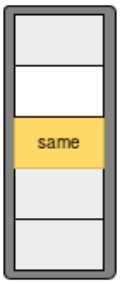
7.2 Cost assessment of Interface I1

Specification cost compared to using non-standardized proprietary interface:

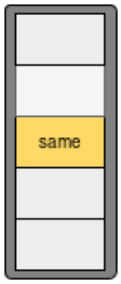
Cost reduction	Cost increase	Comments	The overall cost is considered
• if no proprietary **I1** interfaces exist, the effort is reduced by standard (avoid multiple proprietary specification)	• if proprietary interfaces exist the specification of a standard is additional effort	-	

Implementation cost compared to using non-standardized proprietary interface:

Cost reduction	Cost increase	Comments	The overall cost is considered

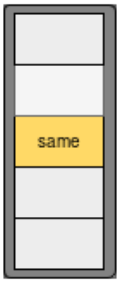
Cost reduction	Cost increase	Comments	The overall cost is considered
• digitalization will make these interfaces mandatory. Standard leads to reduced effort, without a standard a lot of solution-specific flavor would have the need to be adapted and maintained. • reference-implementation possible for solutions (tools, ..) on other side of the interface	• one-time effort for adaption of existing solutions to fulfill standard API • work-around or technical depths are not acceptable • competency acquisition from third party	• cost of implementation is related to complexity of the solution	

Testing cost compared to using non-standardized proprietary interface:

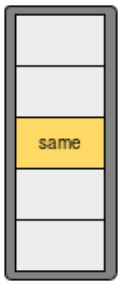
Cost reduction	Cost increase	Comments	The overall cost is considered
• efforts to design test set is reduced	• the broader span of users implies a broader testing scope • testing against standardized interface is higher effort • work-around or technical depths are not acceptable • increased effort for each version/baseline to demonstrate interoperability with all implementations of the other side	• cost of testing is related to complexity of the solution	

Integration and authorization cost compared to using non-standardized proprietary interface:

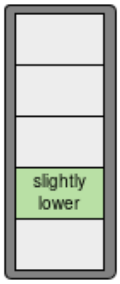
Cost reduction	Cost increase	Comments	The overall cost is considered

Cost reduction	Cost increase	Comments	The overall cost is considered
• standardization of ISA review will decrease authorization costs	• integration effort scales with the amount of communication partners (with different versions) • increased effort in case of multiple suppliers involved • increase in timeline in case of multiple suppliers involved	• cost of integration and authorization is related to complexity of the solution	

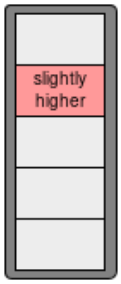
Legal cost compared to using non-standardized proprietary interface:

Cost reduction	Cost increase	Comments	The overall cost is considered
- usage of a standardized interface for juridical data	- implementation of a standardized interface to juridical data	• total effort considered as max same, as adding an juridical if and using it would compensate	

Operation & Maintenance cost compared to using non-standardized proprietary interface:

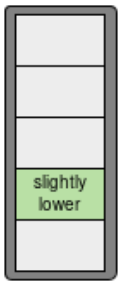
Cost reduction	Cost increase	Comments	The overall cost is considered
• only 1 standard needs to be maintained • standardized processes and tools • reduced amount of solutions on other side of the API • reduced training effort • errors in handling of tools / equipment should occur less often	• compatibility management within the standard • higher delivery time (inter suppliers)	-	

Release life-cycle management cost compared to using non-standardized proprietary interface:

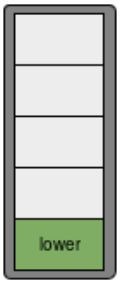
Cost reduction	Cost increase	Comments	The overall cost is considered
• harmonized approach for the orchestration of the software.	• compatibility management in case of changes in interface(s) • version management for interface(s)	• Interface(s) do not have direct impact onto lifecycle-management regarding the SW-layers in the rail systems.	

7.3 Cost assessment of interface I2

Specification cost compared to using non-standardized proprietary interface:

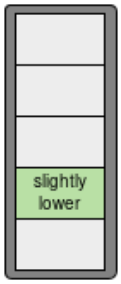
Cost reduction	Cost increase	Comments	The overall cost is considered
• only one type of COTS HW needs to be defined	• additional one-time effort to specify the API of the COTS HW	• interface needs to be stable for efficient COTS HW spares handling	

Implementation cost compared to using non-standardized proprietary interface:

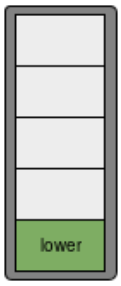
Cost reduction	Cost increase	Comments	The overall cost is considered
• no specific HW development, COTS HW available on the market	• additional effort for common (vendor independent) boot loader-SW as initial SW on the COTS HW spares	• assumption: usage of COTS HW is mandatory	

Testing cost compared to using non-standardized proprietary interface:

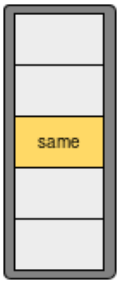
Cost reduction	Cost increase	Comments	The overall cost is considered

Cost reduction	Cost increase	Comments	The overall cost is considered
• basic testing with same effort on COTS HW as on specific HW, but no specific HW needed for test runs.	-	• testing is necessary mainly on side of the HW abstraction layer. Total effort depends on the amount of different solutions for this HW abstraction layer.	

Integration and authorization cost compared to using non-standardized proprietary interface:

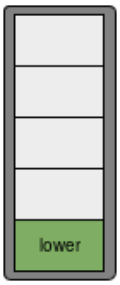
Cost reduction	Cost increase	Comments	The overall cost is considered
• no safety-authorization for COTS HW	-	• even with HW abstraction layer some basic regression tests will be necessary for each RTE-solution to qualify the COTS HW. • regression-tests per RTE-solution with standard interface is same effort as with proprietary interface	

Legal cost compared to using non-standardized proprietary interface:

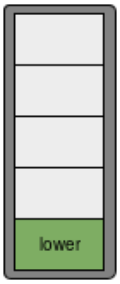
Cost reduction	Cost increase	Comments	The overall cost is considered
-	-	• no COTS HW related legal aspects. COTS HW considered as "non safety critical HW"	

Operation & Maintenance cost compared to using non-standardized proprietary interface:

Cost reduction	Cost increase	Comments	The overall cost is considered

Cost reduction	Cost increase	Comments	The overall cost is considered
• reduced amount of COTS HW needed • simplified obsolescence • harmonized processes and tools for SW-installation on COTS hardware • no vendor-specific HW re-development • no need for stocking of spare hardware or electronic pieces	• process of HW qualification to establish with different RTE-suppliers and COTS HW suppliers	-	

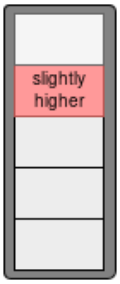
Release life-cycle management cost compared to using non-standardized proprietary interface:

Cost reduction	Cost increase	Comments	The overall cost is considered
• update strategy of HW usage solvable with reduced effort for handling of COTS HW	-	-	

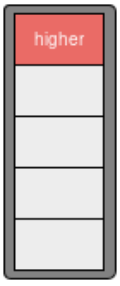
7.4 Cost assessment of interface I3

Specification cost compared to using non-standardized proprietary interface:

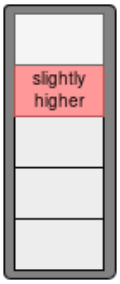
Cost reduction	Cost increase	Comments	The overall cost is considered

Cost reduction	Cost increase	Comments	The overall cost is considered
• if no proprietary I3 exists, effort is reduced by standard (avoid multiple proprietary specifications)	• if proprietary interface exists, the specification of a standard is additional effort • specifying interface that supports different requirements of the possible safety-architectures is very challenging. • compatibility handing in interface version increased effort considering broader span of users	• technical details of common SW layer for HW abstraction not yet clarified (if technically feasible for different safety-concepts in the different safety layers running above)	

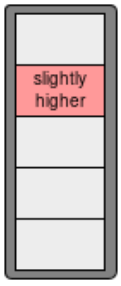
Implementation cost compared to using non-standardized proprietary interface:

Cost reduction	Cost increase	Comments	The overall cost is considered
• If a common solution is feasible then the effort is reduced (avoiding multiple proprietary solutions).	• if such an interface does not exist today it will need a lot of effort to realize it. • if such an interface exists as proprietary interface today it will need a lot of effort to adapt it. • the broader span of users implies a higher effort in development • competency acquisition from third party • work-around or technical depths are not acceptable	• It's not yet clarified if common solution with standardized **I3** i s feasible for different solutions of safety concepts. • cost of implementation is related to complexity of the solution • off-the-shelf solutions can be used anyway (independent from standardized interfaces)	

Testing cost compared to using non-standardized proprietary interface:

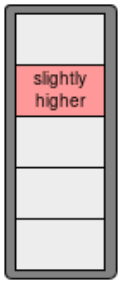
Cost reduction	Cost increase	Comments	The overall cost is considered
• generically having a standardized interface would mean to also use standardized test cases which would contribute to cost reduction.	• if a proprietary interface exists the existing SW needs to be adapted and/or 2 interfaces need to be implemented and maintained • testing against standardized interface is higher effort • work-around or technical depths are not acceptable • increased effort for each version/baseline to demonstrate interoperability with all implementations of SW running on I3	• cost of testing is related to complexity of the solution	

Integration and authorization cost compared to using non-standardized proprietary interface:

Cost reduction	Cost increase	Comments	The overall cost is considered
• standardization of ISA review will decrease authorization costs	• integration effort scales with the amount of existing artifacts (RTEs, application), scales with amount of existing versions • increased effort in case of multiple suppliers involved • increase in timeline in case of multiple suppliers involved	• cost of integration/authorization is related to complexity of the solution • Each new version of the virtualization layer need to be analyzed regarding impact to existing systems.	

Legal cost compared to using non-standardized proprietary interface:

Cost reduction	Cost increase	Comments	The overall cost is considered

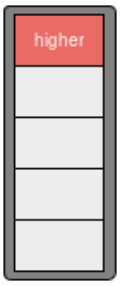
Cost reduction	Cost increase	Comments	The overall cost is considered
-	• effort for solution of juridical recording when different suppliers are involved • system responsibility in case "several suppliers involved"	• legal costs arise by situation "several vendors involved for different SW layers running on same COTS HW"	

Operation & Maintenance cost compared to using non-standardized proprietary interface:

Cost reduction	Cost increase	Comments	The overall cost is considered
• same solution on all HW nodes • reduced training costs	• increased maintenance effort due to short lifecycle for HW-abstraction with dependencies to COTS HW and different vendors involved for the SW layers • compatibility management with discussions between multiple parties • increased effort for error analysis with different vendors involved.	-	

Release life-cycle management cost compared to using non-standardized proprietary interface:

Cost reduction	Cost increase	Comments	The overall cost is considered

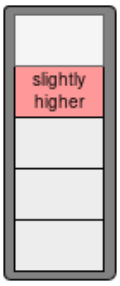
Cost reduction	Cost increase	Comments	The overall cost is considered
-	• compatibility management in case of changes in I3 • version management for I3 for used combinations • considering coexistence of proprietary interfaces • multiple artifacts provided by multiple vendors increases the amount of needed SW-updates in context of life-cycle management. • effort scales with amount of updates • amount of updates scales with amount of systems and amount of different HW versions used	• amount of updates driven by changes in SW-Layer to support new underlying COTS hardware. Even without touching I3 a SW-update will be necessary to use new COTS HW.	

7.5 Cost assessment of interface I4

Specification cost compared to using non-standardized proprietary interface:

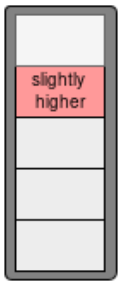
Cost reduction	Cost increase	Comments	The overall cost is considered
• if no proprietary I4 exists, effort is reduced by standard (avoid multiple proprietary specifications)	• specification is additional effort (proprietary interfaces already exist) • broader span of users (multiple suppliers) leads to broader interface • coexistence of both interfaces (standard and proprietary to avoid impact onto existing SW)	• On the short term costs might be higher but long term perspective would be lower.	

Implementation cost compared to using non-standardized proprietary interface:

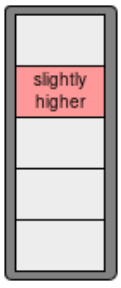
Cost reduction	Cost increase	Comments	The overall cost is considered
• Sticking to an already established standard (e.g., Open Source) would be a potential cost reduction.	• if a proprietary interface exists the existing SW needs to be adapted and 2 interfaces need to be implemented and maintained • the broader span of users implies a broader interface with higher effort in development • competency acquisition from third party • work-around or technical depths are not acceptable	• cost of implementation is related to complexity of the solution • off-the-shelf solutions can be used anyway (independent from standardized interfaces).	

Testing cost compared to using non-standardized proprietary interface:

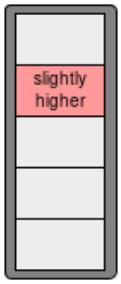
Cost reduction	Cost increase	Comments	The overall cost is considered

Cost reduction	Cost increase	Comments	The overall cost is considered
• in the case of application portability, the testing campaign could be limited to a limited number of regression tests. • generically having a standardized interface would mean to also use standardized test cases which would contribute to cost reduction. •	• if a proprietary interface exists the existing SW-tests needs to be adapted and/or two interfaces need to be tested • the broader span of users implies a broader interface with broader testing scope • work-around or technical depths are not acceptable, 100% test fulfillment necessary. • increased effort for each version/baseline to demonstrate interoperability with all implementations of the other side.	• cost of testing is related to complexity of the solution • cost of adaption depends on the detail of the application itself • subset of the test cases has to be aligned with the assessor by the integrator.	

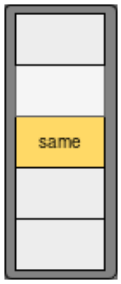
Integration and authorization cost compared to using non-standardized proprietary interface:

Cost reduction	Cost increase	Comments	The overall cost is considered
• standardization of ISA review will decrease authorization costs	• integration effort scales with the amount of existing artifacts (RTE, application) and versions • increased effort in case of multiple suppliers involved • increase in timeline in case of multiple suppliers involved	• cost of integration/authorization is related to complexity of the solution	

Legal cost compared to using non-standardized proprietary interface:

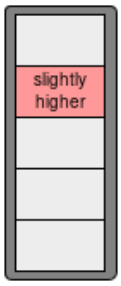
Cost reduction	Cost increase	Comments	The overall cost is considered
-	• effort for solution of juridical recording when different suppliers are involved • system responsibility in case "several suppliers involved"	• legal costs arise by situation "several vendors involved for different SW layers running on same COTS HW"	

Operation & Maintenance cost compared to using non-standardized proprietary interface:

Cost reduction	Cost increase	Comments	The overall cost is considered
• Sticking to an already established standard (e.g., Open Source) would be a potential cost reduction.	• increased effort for error analysis with different vendors involved. • increased effort for continuous vulnerability management with different vendors involved. • increased timeline for deliveries when several vendors involved	• I4 between SW-layers is not "visible" in context of operation and maintenance	

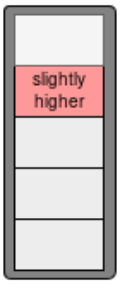
Release life-cycle management cost compared to using non-standardized proprietary interface:

Cost reduction	Cost increase	Comments	The overall cost is considered

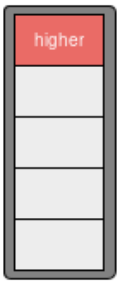
Cost reduction	Cost increase	Comments	The overall cost is considered
-	• compatibility management in case of changes in I4 • version management for I4 for used combinations • considering coexistence of proprietary interfaces • multiple artifacts provided by multiple vendors increases the amount of needed SW-updates in context of life-cycle management. • effort scales with amount of updates • amount of updates scales with amount of systems and amount of different SW versions us	• amount of updates driven by changes in SW-Layer in context of IT-security patches. Even without touching I4 a SW-update will be necessary to use newest SW.	

7.6 Cost assessment of interface I5

Specification cost compared to using non-standardized proprietary interface:

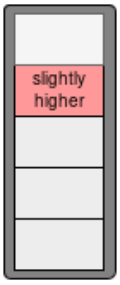
Cost reduction	Cost increase	Comments	The overall cost is considered
• if no proprietary I5 exists, effort is reduced by standard (avoid multiple proprietary specifications)	• high effort for safety-critical interface as standard with overall safety concept • additional effort to existing proprietary interfaces • coexistence of proprietary interfaces and standard interface • complicated compatibility management for safety-critical standard	• without standardized design rules and safety-architecture a generic I5 is not possible	

Implementation cost compared to using non-standardized proprietary interface:

Cost reduction	Cost increase	Comments	The overall cost is considered
• the number of RTEs implementing the interface could be reduced as certain suppliers might decide to use existing solutions.	• adaption of existing safety-layers to fulfill the standard interface • if a proprietary interface exists the existing SW needs to be adapted and/or two interfaces need to be implemented • a standardized spec. for safety-critical interface will be "fatter" than today and needs largely increased effort (as proprietary interface for exactly the own purpose). • work-around or technical depths are not acceptable • competency acquisition from third party	• reference test-bed for safety layer not feasible due to different solutions of safety concepts. Each safety-layer will need it's own test bed (even if the I5 would be standardized). • cost of implementation is related to complexity of the solution	

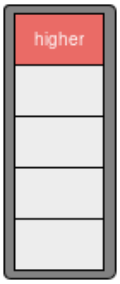
Testing cost compared to using non-standardized proprietary interface:

Cost reduction	Cost increase	Comments	The overall cost is considered

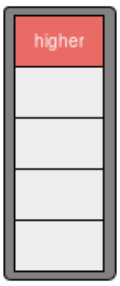
Cost reduction	Cost increase	Comments	The overall cost is considered
• generically having a standardized interface would mean to also use standardized test cases which would contribute to cost reduction.	• if a proprietary interface exists the existing SW-tests needs to be adapted and/or two interfaces need to be tested • safety-critical test effort increases with broader interface for broader span of users • testing against standardized safety critical interface higher as for proprietary internal interface • increased effort for handling of tests for different interface versions (all interface versions used in the field) and interface variants (proprietary & standard) • 100% fulfillment necessary, no workarounds allowed.	• cost of testing is related to complexity of the solution	

Integration and authorization cost compared to using non-standardized proprietary interface:

Cost reduction	Cost increase	Comments	The overall cost is considered

Cost reduction	Cost increase	Comments	The overall cost is considered
• standardization of ISA review will decrease authorization costs	• increased effort for compatibility and interoperability management for safety critical interface • safety-solution specific SRACS lead to increased effort • integration effort scales with the amount of existing artifacts (RTEs, application) and versions • highly increased effort in case of multiple suppliers involved • increase in timeline in case of multiple suppliers involved	• safety-concepts of safety-layers are hard to be standardized, which leads probably to solution specific safety-relevant aspects of I5 e.g., specific SRACS	

Legal cost compared to using non-standardized proprietary interface:

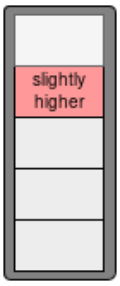
Cost reduction	Cost increase	Comments	The overall cost is considered
-	• assignment of responsibility hardly possible when different vendors involved • effort for solution of juridical recording (100% traces) when different suppliers are involved • more legal entities involved in the safety case (when different suppliers involved)	• 100% tracing of all SW touching points between different vendors not possible.	

Operation & maintenance cost compared to using non-standardized proprietary interface:

Cost reduction	Cost increase	Comments	The overall cost is considered
• standardized interface leads to standardized processes regarding operation for the users.	• increased effort for error analysis with different vendors involved. • several safety experts involved in case of multiple suppliers • increased effort for continuous vulnerability management with different vendors involved. • increased effort for maintenance releases with different suppliers involved. • increase in timeline for delivery of maintenance releases with different suppliers involved	• I5 between SW-layers is not “visible” in context of maintenance. • As mentioned above in the introduction section, there will not be many vendors. And the positive effects for the users will probably exceed the increase for the few vendors.	

Release life-cycle management cost compared to using non-standardized proprietary interface:

Cost reduction	Cost increase	Comments	The overall cost is considered

Cost reduction	Cost increase	Comments	The overall cost is considered
• standardized interface will lead to more efficient processes for the users	• compatibility management in case of changes in interface • interface version management • considering coexistence of proprietary interfaces • multiple safety-critical artifacts provided by multiple vendors increases the amount of needed SW-updates in context of life-cycle management. • effort scales with amount of updates • amount of updates scales with amount of systems and amount of different SW versions used	• amount of updates driven by changes in underlying SW-Layers • even without standardized I5 a SW-update will be necessary to use newest underlying SW (to support new COTS HW or IT-sec patches).	

8 Conclusion on the interfaces to be standardized

Note: *The conclusions presented in this section are still preliminary and not yet finally agreed among all domain members.*

In Figure 3, an overall evaluation of the five interfaces considered for possible standardization is provided, capturing both the estimation of the expected costs related to the standardization of the interfaces, and the expected benefit. It shall here be noted that the depicted average cost merely reflects the arithmetic (unweighted) average of the cost assessment in the individual cost categories, and can hence only be seen as a rough indication for the trend.

The comparative need shown in the figure is determined as follows:
For each user story recommending that a certain interface be standardized, 2 points are given if the benefit of the user story is considered medium (according to the analysis in Section ...), or 3 points if the user story is rated as highly beneficial
For each user story mandating that a certain interface be standardized, 4 points are given if the benefit of the user story is considered medium (according to the analysis in Section ...), or 6 points if the user story is rated as highly beneficial
After calculation of an individual score per interface candidate, the scores are then classified into groups "high", "medium" and "low"

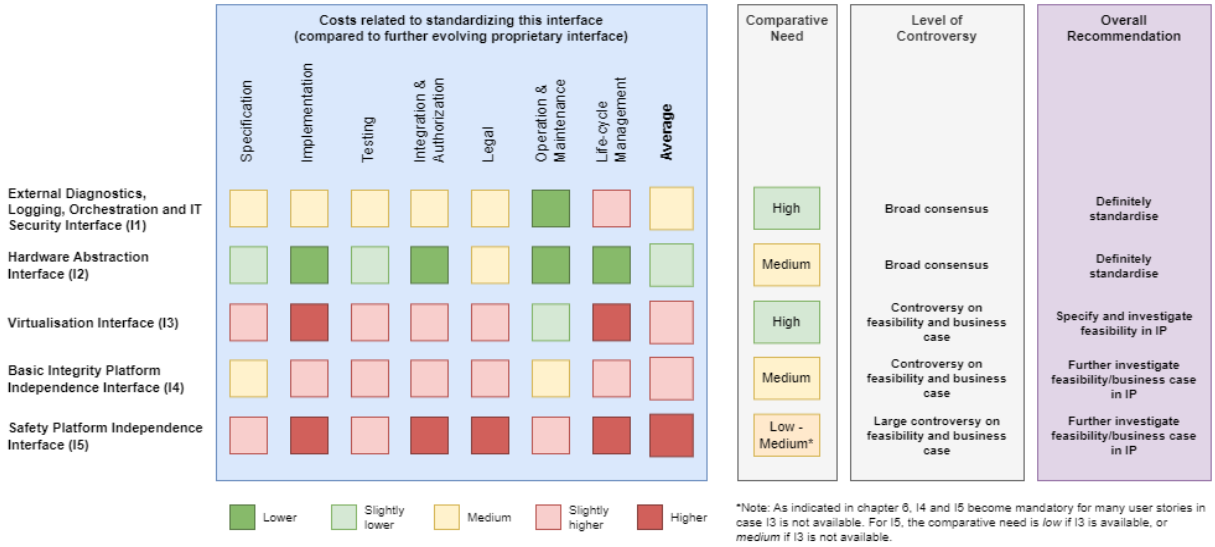


Figure 3 Overall assessment of the interfaces considered for standardization.

Overall, the following conclusion is drawn:

External Diagnostic, Logging, Orchestration and IT Security Interface(s) (I1): Here, the costs incurred by the standardization of this interface are considered similar to the case where it is not standardized, while the need to standardize this is seen as high, as many of the stated user stories require or at least recommend this interface to be standardized. As there is broad consensus on the aforementioned in the domain, this should definitely be standardized.

Hardware Abstraction Interface (I2): Here, the need is only medium, as it is not strictly mandatory for most of the user stories that this interface is standardized, but the cost related to the interface standardization are even considered lower than if the interface were not standardized. As there is broad consensus on the aforementioned in the domain, this should also definitely be standardize.

Virtualization Interface (I3): Here, the analysis indicates that the cost incurred by the standardization of this interface is slightly higher than if this were not standardized, but this aspect is outweighed by a high need, as many user stories strictly require this interface to be standardized. However, there is controversy on whether the standardization of this interface is feasible and has a positive business case (in particular for SIL applications). It is hence recommended, that a first study on the interface is conducted in the Innovation Pillar (in particular FA2 R2DATO, where the topic has already been taken up), to further assess feasibility, before a decision on standardization is taken.

Basic Integrity Platform Independence Interface (I4): Here, the cost evaluation has led to a similar result as for I3, indicating that the cost incurred through standardization is slightly higher than if (only) proprietary variants of this interface are further evolved. The need is a bit lower than for, e.g., I3, as less user stories strictly require that I4 is standardized, but still 65% of the user stories are enabled by interface I4. It would hence be beneficial to standardize this, but at slightly lower priority compared to the previous interfaces. In general, there is controversy in the domain on the feasibility and business case associated to the standardization of I4, hence it is recommended that this is further investigated and prototyped in FA2 R2DATO (as already ongoing), before a decision on

standardization is taken.

Safe Platform Independence Interface (I5): Here, the cost analysis has indicated that standardization would incur higher cost compared to the case where (only) proprietary variants of this interface are further pursued. Further, there are less user stories having a strong need for this interface to be standardized - if interface I3 can be standardized, which is not clear, as stated above. It should be stated that in general that there is a strong controversy on the feasibility and business case associated to the standardization of this interface. It is hence recommended that I5 is further investigated and prototyped in FA2 R2DATO (as already ongoing). In particular, it should be investigated how the specification of this interface can still allow for different fault tolerance approaches, and how identified challenges related to integration and authorization can be overcome.
It should be noted that as mentioned before, that all assessments of benefits of user stories and the cost incurred in standardizing interfaces is highly sensitive to various assumptions made. Also, the assessment is strongly dependent on the perspective taken (e.g., application vendor, platform vendor, railway, integrator perspective, etc.).
due to, e.g., dependencies among interfaces it is considered beneficial to well coordinate the specification of interfaces I1 - I3, which are jointly expected to unlock 9 out of 13 user stories.

The somewhat modest evaluation of the benefit of interfaces I4 and I5 is mainly based on the assumption that interface I3 is a easier and more cost-efficient solution to cover the needs of most user stories. If it should turn out that interface I3 cannot be realised (e.g., because of authorization issues or lack of vendor support of required solutions), interfaces I4 and I5 would become much more relevant.

8.1 Proposed Next Steps

For the interfaces investigated by the domain, the following actions are proposed:

- For the **External Diagnostic, Logging, Orchestration and IT Security Interface(s) (I1)**, the requirements on the interface(s) should be identified by the domain. Based on these, it should be clarified to which extent the Transversal CCS domain covers the required interfaces, or which gap remains. Where any gap would be specified is tbd;

- The **Virtualization Interface (I3)**, as stated above, should be drafted and further investigated in the Innovation Pillar (specifically in the FA2 R2DATO project, where this has already been taken up);

IP was informed, decision is pending...

- The **Hardware Abstraction Interface (I2)** should ideally be specified in the context of I3;

Handover to IP, together with I3 done.

- For the **Basic Integrity Platform Independence Interface (I4)** and **Safe Platform Independence Interface (I5)** the ongoing prototyping in the Innovation Pillar / FA2 R2DATO project should be continued, with an emphasis on assessing feasibility and how identified challenges related to integration and authorization can be overcome.

9 Open Points

The following open points have been identified that will require further investigation:

- The Computing Platform described here in this document is based on standard hardware, all interfaces to other systems are communication based interfaces. By this the aspect of "parallel I/O" is supported through external interfaces. Such hardware near I/O control mechanism has to be realized as separated Functional Systems based on specific hardware for I/O control.
    - Trackside this are e.g. Object Controllers connected via SCI interfaces (defined by EULYNX).
    - Onboard this are onboard controllers connected via "one common bus".
- Responsibility splits among suppliers and integrator in the case of the potentially standardized interfaces and different deployment options depicted in Figure 1
- There is a controversy on whether there is a need and to which extent interface I3 should be standardized. This is due to the fact that many state-of-the art virtualization solutions/standards already exists in the market and standardization impact on the business case (in particular for SIL applications) are not clear. It is therefore recommended to further investigate interface I3.
- As the realization and associated business cases for I3 are still not clear therefore scope and relevancy of st andardizing interface I4 and I5 need to be further investigated.
- The terminology used is partially inconsistent and needs alignment.